Looking at the complexities. Network Analysis in Psychological Research. A focus on Bayesian Networks

Matteo Orsoni
University of Bologna

In contemporary psychological research, network analysis has emerged as a powerful methodological paradigm, providing sophisticated tools for modeling complex relationships among measurable psychological constructs (Briganti et al., 2024). This framework represents a departure from conventional reductionist paradigms, embracing instead a comprehensive perspective that acknowledges the inherently complex and interdependent structure of psychological phenomena.
This chapter starts by introducing the various network typologies and methodological applications currently used in psychological science and psychometrics (22.1, 22.2, 22.3). The focus then shifts specifically to Bayesian Networks (BNs), detailing their theoretical utility (23) and the algorithms used to learn them from empirical data (25). In Section 26, we demonstrate these concepts through a practical R implementation. Finally, we conclude with perspectives on the future of BNs in psychological research (27) and provide a comprehensive guide to software tools for structural and parameter learning (28).

Introduction to Network Analysis in Psychological Research

Core Network Modeling Frameworks

Network analytic methodologies can be broadly organized into three principal domains: approaches for static (cross-sectional) observations, techniques for temporal (longitudinal) data structures, and specialized methods addressing latent constructs or intervention effects, and into two major architectural families: Pairwise Markov Random Fields (PMRFs) and Directed Acyclic Graphs (DAGs). The following paragraphs would serve as a guide for the reader in determining the most appropriate analytical framework based on their research design and data structure. In depth details about each of these frameworks can be retrieved in the foundamental work of Briganti and colleagues (inserisci citazione)

Pairwise Markov Random Fields (PMRFs). PMRFs employ undirected edges to characterize bidirectional conditional dependencies among variables. This framework dominates contemporary applications of undirected network estimation in cross-sectional research contexts. The specific implementation of PMRFs varies according to data characteristics:

- **Gaussian Graphical Model (GGM):** Applied to normally distributed continuous data.
- **Ising Model:** Designed for dichotomous (binary) variables.
- **Mixed Graphical Model (MGM):** Handles heterogeneous data structures incorporating categorical, continuous, and count-based variables.

Directed Acyclic Graphs (DAGs) and Bayesian Networks (BNs). These frameworks utilize directed edges and are explicitly constructed for causal modeling and probabilistic inference. Bayesian Networks integrate probability distributions with DAG structures, wherein directed edges encode conditional dependencies that may be interpreted as candidate causal pathways.

Temporal and Dynamic Network Frameworks

These methodological approaches are developed for characterizing how psychological phenomena evolve temporally, moving forward the limitations of cross-sectional observation by capturing dynamic processes and their evolution over time.

Autoregressive (VAR) Network Models Advanced implementations, including the multilevel graphical vector-autoregressive network (GVAR) and the panel-GVAR, can handle intensive longitudinal and panel data structures by modeling multivariate time series relationships.

These frameworks generate three distinct network representations: (1) **Temporal networks** feature directed edges capturing predictive relationships wherein deviations in one node at time t predict another node's state at time $t + k$, representing the autoregressive lagged associations that reveal how variables influence each other across time intervals. (2) **Contemporaneous networks** display undirected edges representing bidirectional associations among nodes within identical measurement intervals, capturing within-timepoint correlations after controlling for temporal dependencies. (3) **Between-person networks** consist of undirected edges characterizing bidirectional relationships among individuals' stable baseline levels (trait-level constructs) aggregated across temporal observations, reflecting individual differences in mean levels around which temporal dynamics unfold.

Time-Varying Network Models These frameworks are essential for detecting systematic alterations in network edges or model parameters across time, enabling investigation of phenomena such as treatment response trajectories or transitions

toward pathological states. Time-varying models account for nonstationarity in psychological processes, revealing how associations among variables strengthen, weaken, or reverse direction over weeks or months. Parameter evolution may be modeled as continuous (smooth trajectories using spline-based or kernel methods) or discrete (employing change-point detection methods to identify abrupt shifts in network structure).

Rate-of-Change Network Models These models provide unique insights into symptom dynamics by examining how symptoms co-evolve temporally, employing derivative measures (velocity) to characterize conditional dependencies among evolving symptoms. Rather than modeling levels of variables, this approach captures the rate at which symptoms intensify or diminish, revealing nonlinear dynamic relationships such as self-reinforcing feedback loops or dampening effects that may not be apparent in standard level-based models.

Idiographic Network Comparison Methodologies such as the Idiographic Network Intervention Test (INIT) enable rigorous statistical comparison of person-specific network architectures, facilitating tests of inter-individual heterogeneity and treatment-induced changes in network structure. These approaches employ bootstrapping procedures and permutation tests to determine whether differences in edge weights between time periods or conditions exceed what would be expected by sampling variability alone, providing a framework for evaluating whether interventions causally alter the temporal dynamics of psychological systems

Dynamic Bayesian Networks DBNs are specialized graphical models employed when temporal dependencies exist in the data, making them necessary when observations are not independent, such as when a single subject is measured over multiple occasions. DBNs extend the traditional Bayesian Network framework to handle time series data, often modeling the structure as a form of a Structural Vector AutoRegressive (SVAR) model (Vowels et al., 2022). They work by unrolling the graph over time, allowing the model to handle how a previous node influences itself at a future time point. Consequently, DBNs are capable of modeling both present effects (relationships occurring within the same measurement slice) and time-lagged effects (relationships between variables across different time points). For instance, algorithms such as DYNOTEARS seek to learn the SVAR model and utilize separate adjacency matrices to represent these intra-slice and inter-slice graph edges (Vowels et al., 2022).

Network Psychometrics and Experimental Integration

Network methodologies have evolved specialized techniques that bridge traditional psychometric theory with experimental design principles, providing researchers

with tools to address theoretical and methodological challenges in psychological measurement.

Latent Network Modeling (LNM) Addresses challenges including measurement error and semantic item overlap by estimating network structures from the implied latent covariance matrix, thereby permitting modeling of conditional independence structures among unobserved latent constructs rather than observed indicators (Epskamp et al., 2017). This approach enables researchers to identify unique patterns of dependencies between latent variables (e.g., depression, anxiety) while accounting for measurement imprecision in observed items. The Residual Network Model (RNM) extends LNM by characterizing residual interactions remaining after accounting for latent variable influence, allowing for identifiable models in which local independence assumptions are structurally violated. RNM captures item-specific dependencies that persist beyond what common factors explain, such as semantic overlap or method effects (Briganti et al., 2024; Epskamp et al., 2017).

Confirmatory Network Modeling Enables researchers to evaluate the fit of theoretically-specified network structures using confirmatory testing procedures, analogous to Confirmatory Factor Analysis (CFA). This framework facilitates network replication testing across independent samples and formal model comparisons between network and factor-analytic models using standard fit indices and likelihood ratio tests. Researchers can specify hypothesized edge structures based on theory and test whether empirical data support these configurations, advancing theory-driven network research (Briganti et al., 2024; Epskamp et al., 2017).

Network Intervention Analysis (NIA) Synthesizes network analysis with experimental methodologies (e.g., Randomized Controlled Trials) by incorporating treatment allocation as a network node (Briganti et al., 2024). NIA enables decomposition of treatment effects into direct symptom-specific impacts (effects on targeted symptoms) versus indirect propagated effects (cascading changes through network connections to non-targeted symptoms). This approach provides mechanistic insights into how interventions alter psychological systems, distinguishing between symptoms that respond directly to treatment and those that improve secondarily through network propagation (Briganti et al., 2024).

Meta-analytic Network Modeling Approaches such as Meta-analytic Gaussian Network Aggregation (MAGNA) estimate unified network structures aggregated across multiple independent investigations, pooling correlation matrices or raw data from diverse samples to increase statistical power and identify replicable network features. These methods account for between-study heterogeneity while

extracting consensus network structures, enabling researchers to distinguish robust, generalizable edges from study-specific artifacts. (Wang, 2021).

In Table 26 are briefly summarized the algorithm presented divided for category, structural architecture and primary application.

Methodology | Category	Structural Architecture	Primary Application
PMRFs (e.g., GGM, Ising)	Undirected, Cross-sectional	Characterizing conditional dependencies among variables at a single temporal snapshot.
Bayesian Networks (BNs)	Directed Acyclic Graphs	Causal inference and probabilistic reasoning through directed conditional dependency modeling.
VAR/GVAR/panelGVAR	Temporal, Contemporaneous	Decomposing within-person temporal dynamics (lagged and contemporaneous) from stable between-person trait-level associations.
Latent Network Modeling	Undirected/Latent Constructs	Characterizing relationships among latent variables while controlling for measurement error.
Network Intervention Analysis	Experimental manipulation	Distinguishing symptom-specific direct versus indirect treatment effects in experimental frameworks.

26. Summary of Principal Network Analysis Methodologies

Why choosing Bayesian Networks (BNs) in psychological and psychometric practice

BNs are especially attractive in psychological and psychometric applications because they represent variables in a DAG, allowing researchers to model directional probabilistic relations and formulate explicit causal hypotheses about how symptoms, items, or cognitive skills influence one another, rather than merely co-occur (Culbertson, 2016). Unlike undirected models such as Gaussian Graphical Models, which only reveal symmetric conditional associations, BNs can distinguish between potential causes, effects, and colliders, making them useful for identifying plausible intervention targets, refining theoretical models of mental disorders, and clarifying whether a central symptom is more likely to be a driver or a downstream consequence (Briganti et al., 2024). Their factorized structure makes them computationally efficient even in high-dimensional assessment settings, and they integrate naturally with other psychometric models to represent latent skills in educational and cognitive testing, enabling modular model building and hypothesis-

driven extensions of existing theories (Almond et al., 2015; Culbertson, 2016; Orsoni, Benassi, & Scutari, 2025; Vowels, 2025). At the same time, this causal interpretability hinges on strong structural assumptions, most importantly that the data arise from a DAG, that major confounders and latent causes have been adequately measured or modeled, and that observed dependencies faithfully reflect the underlying causal structure, so in practice BNs are most valuable when used as a transparent framework for combining substantive theory, prior knowledge, and empirical data, rather than as a purely automatic discovery tool (Orsoni, Benassi, & Scutari, 2025; Vowels, 2025).

Even if this is not the topic of the present chapter, the reader should take in mind that there are at least three core assumptions for rigorous BN causal inference: (1) **DAG data-generating structure**. The underlying causal system can be represented as a DAG, so there are no true feedback cycles in the variables included in the model. (2) **Causal sufficiency**. All common causes of the modeled variables are either measured or appropriately accounted for (no important unobserved confounders or selection mechanisms that induce spurious dependencies). (3) **Causal faithfulness**. This assumption asserts that every statistical independence (or dependence) found in the observed data is a direct result of the underlying causal network structure, rather than a coincidence.

Bayesian Networks: Foundations and Core Principles

As probabilistic graphical models grounded in causal inference theory, BNs are characterized by two fundamental elements: a structural representation that is the Directed Acyclic Graph (DAG), and an associated Joint Probability Distribution.

The Directed Acyclic Graph (DAG): Structural Foundation

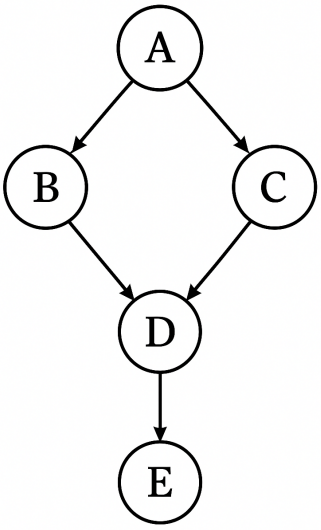
The architectural backbone of a BN is formally represented as a graph $G = (V, A)$, comprising a set of nodes V and a collection of directed edges A.

Nodes and Directed Edges

- **Nodes (V)** Each node, denoted $v_1, v_2, \ldots, v_N$, corresponds to a specific variable of interest within the research domain, such as individual symptom indicators or questionnaire items.
- **Directed Edges (A)** The connections linking nodes are represented as directed edges (also termed arcs), which encode conditional probabilistic dependencies. Importantly, these directional connections can be interpreted as potential causal pathways within the data structure. This directional specification fundamentally distinguishes BNs from conventional Pairwise Markov Random Fields (PMRFs), where undirected edges preclude straightforward causal interpretation.
- **Parent-Child Relationships** When a directed edge connects two nodes as $v_i \rightarrow v_j$, the starting node v_i is designated as the *parent* node, while the outgoing

node v_j is termed the *child* node. This parent-child terminology captures the directional flow of influence or information within the network.

An example of DAG with nodes and arches is represented in Figure 15.



15. Example of a DAG structure.

The Acyclicity Requirement A fundamental structural constraint of BNs is the requirement that the graph remain *acyclic*. This constraint prohibits the existence of any cyclical paths (sequences of directed edges that originate and terminate at the same node). This acyclicity requirement is not merely a technical convenience but a critical theoretical prerequisite that enables the mathematical decomposition and efficient factorization of the joint probability distribution underlying the network.

Conditional Independence Structure The DAG's primary theoretical function is to encode the complete set of conditional independence relationships that exist among the variables in the system.

d-Separation: This graphical criterion provides a formal algorithmic procedure for determining whether two nodes are statistically independent or conditionally independent given a third set of nodes. Specifically, if two variables (X_i and X_j) are d-separated by a conditioning set S, then the absence of a directed path connecting them encodes their conditional independence: $X_i \perp X_j \mid S$. This graphical property directly translates to probabilistic independence, establishing the bridge between network structure and statistical relationships.

The Joint Probability Distribution: Quantifying Relationships

Complementing the structural DAG, the joint probability distribution constitutes the second essential component of a BN, providing the quantitative specification of relationships among variables, and allows the BNs to be termed as "generative" or "world" models.

Factorization via the Local Markov Property The DAG structure enables a powerful simplification: the complex joint probability distribution $p(X)$ over all variables can be decomposed into a product of simpler, localized conditional probability distributions. This decomposition principle is termed the *local Markov property*.

The global joint distribution factors as a product of local conditional distributions, with each variable X_i conditioned exclusively on its parent nodes $\text{Pa}(X_i)$ in the DAG:

$$p(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Pa}(X_i)) \tag{6}$$

This factorization offers substantial analytical and computational advantages: rather than estimating a single high-dimensional joint distribution, we estimate multiple lower-dimensional local distributions, each characterizing how a single variable depends on its immediate network neighbors.

Parametric Specifications for Different Data Types The specific parametric form of the probability distributions varies according to the measurement scale and distributional characteristics of the observed data: (1) **Gaussian Bayesian Networks**. When variables are continuous (or treated as approximately continuous, as commonly assumed for Likert-scale data in psychopathology research), Gaussian BNs are employed. These models assume the data follow a multivariate normal distribution: $X \sim \mathcal{N}(\mu, \Sigma)$, where μ represents the mean vector and Σ the covariance matrix. (2) **Discrete and Ordinal Bayesian Networks**. For categorical variables, the local conditional distributions are specified using conditional probability tables (CPTs), which enumerate the probability of each possible value of a child node given each possible configuration of its parent nodes' values. While this approach accommodates the discrete nature of many psychological measurement instruments (e.g., Likert scales), treating ordinal variables as purely categorical may sacrifice information about the inherent ordering of response categories. Although specialized methodologies for deriving BNs from ordinal data are under development (Grzegorczyk, 2024; Orsoni, Benassi, & Scutari, 2025), a detailed exploration of this evolving research area is beyond our current scope. (3) **Conditional Linear Gaussian Networks**. These hybrid models accommodate heterogeneous data structures containing both discrete and continuous variables. Discrete

nodes are parameterized using conditional probability tables, while continuous nodes are modeled via linear regression equations. An important structural limitation of this framework is that continuous nodes cannot serve as parents to discrete nodes within the DAG (the causal flow must proceed from discrete to continuous variables).

Key Structural Configurations in Bayesian Networks

Several characteristic structural patterns are essential for understanding how BNs encode probabilistic and causal relationships.

Chains A chain involves a sequence of directed edges, such as $A \rightarrow B \rightarrow C$. In a chain, A and C are statistically dependent. However, they become conditionally independent if conditioned on the intermediate variable B (i.e., given knowledge of B; $A \perp C \mid B$).

Forks (Common Cause) A fork, also known as *common cause* is a structure where a single node acts as a common cause for two other nodes, $A \leftarrow B \rightarrow C$. Similar to a chain, A and C are generally dependent, but they become conditionally independent if conditioned on the common cause B $A \perp C \mid B$.

V-Structures (Colliders) A v-structure, alternatively termed a *collider*, represents a fundamental triadic configuration within a DAG, characterized by the pattern $A \rightarrow C \leftarrow B$, where nodes A and B share no direct connection.

This configuration have a specific causal pattern with distinctive probabilistic behavior: two causes (A and B) independently influence a common effect (C). Critically, this structural pattern is statistically identifiable from observational data and forms a cornerstone for inferring causal structure from probabilistic information. The v-structure exhibits a counterintuitive probabilistic property. While the causes (A and B) are unconditionally independent, they become statistically dependent when conditioning on their shared effect (C). This phenomenon, termed *collider bias* or *Berkson's paradox*, has important implications for statistical inference (Vowels et al., 2022). Indeed, analyzing associations while conditioning on the effect node yields systematically different results than analyzing the causes independently. This behavior underlies many selection bias phenomena in observational research.

Markov Blankets: Local Neighborhoods The Markov blanket of a target node v_i constitutes the minimal set of nodes S that, when conditioned upon, renders v_i conditionally independent of all other nodes in the network. Formally, the Markov blanket comprises three types of nodes: the target node's parents (direct causes), its children (direct effects), and its *spouses* (the other parents of its children, that are nodes that share common effects with the target). This concept provides

substantial analytical utility. For example, when conducting inference about a specific target node, researchers can focus exclusively on its Markov blanket, effectively screening off the influence of the remainder of the network (Orsoni, Benassi, & Scutari, 2025). This localization principle enables efficient computation and targeted analysis. In predictive modeling and classification contexts, the Markov blanket represents the minimal feature set that preserves maximal predictive accuracy; no additional variables beyond the Markov blanket improve prediction of the target node (Orsoni, Benassi, & Scutari, 2025).

Equivalence Classes: Structural Ambiguity An important theoretical consideration is that a given joint probability distribution does not uniquely determine a single DAG structure. Multiple distinct DAG configurations can encode identical sets of conditional independence relationships, and therefore represent the same probability distribution.

For example, the three directed structures $A \rightarrow B \rightarrow C$, $A \leftarrow B \leftarrow C$, and $A \leftarrow B \rightarrow C$ all encode the same conditional independence structure (where A and C are conditionally independent given B), and thus cannot be distinguished based solely on observational probability distributions. All DAGs that encode identical probability distributions constitute an *equivalence class*. This class is uniquely characterized by two features: the underlying undirected skeleton (the graph structure with edge directions removed) and the set of v-structures present in the graph. These two elements together define the boundaries of what can be learned about causal structure from purely observational data without additional assumptions (Briganti et al., 2023).

Learning Bayesian Networks from Empirical Data

Having established the theoretical foundations of Bayesian Networks we now turn to the practical challenge of constructing BNs from observed data. This learning process bridges the gap between the conceptual framework presented previously and its empirical application in psychology and psychometric research. The construction of a BN from data conventionally proceeds through two sequential and complementary stages: **structure learning** (identifying the network architecture) and **parameter learning** (quantifying the relationships within that architecture). These stages can be conceptualized as first drawing the map of relationships, then populating that map with precise statistical quantities.

Structure Learning: Discovering the Network Architecture

Structure learning constitutes the initial and often most challenging phase of BN construction. The objective is to identify the optimal DAG configuration that faithfully represents the conditional independence structure inherent in the observed data. This automated discovery process can be guided by incorporating

domain expertise through mechanisms such as *blacklisting* (prohibiting specific edges known to be implausible) or *whitelisting* (requiring specific edges known to exist), thereby constraining the search space to theoretically plausible candidate models (Vowels et al., 2022). Structure learning algorithms are conventionally classified into three methodological families: constraint-based approaches (CBL), score-based approaches (SBL), and hybrid methods that synthesize elements of both.

Constraint-Based Learning (CBL) CBL algorithms operate by identifying patterns of conditional independence relationships, the statistical "constraints", through hypothesis testing procedures, subsequently connecting nodes that exhibit conditional dependence.

This brunch of algorithms implement a systematic sequence of statistical independence tests applied to the data. The testing procedure typically follows a hierarchical strategy, starting with simple bivariate (marginal) independence tests and progressively incorporating additional conditioning variables to evaluate higher-order conditional independence relationships (Briganti et al., 2023, 2024). The goal is to recover the complete set of conditional independence constraints that characterize the data-generating process, from which the DAG structure can be inferred.

This family includes different algorithms derived from the foundational Inductive Causation (IC) algorithm (Pearl, 2009). Notable examples include the Peter-Clark (PC) algorithm (Spirtes & Glymour, 1991), the Grow-Shrink (GS) algorithm (Margaritis, 2003), and the Incremental Association Markov Blanket (IAMB) (Tsamardinos et al., 2003) family of algorithms.

Score-Based Learning (SBL) Score-based algorithms evaluate and compare candidate DAG structures by computing a global goodness-of-fit metric (a "score") that quantifies how well each network structure explains the observed data. The optimization objective is to identify the DAG that maximizes this scoring function. These algorithms conduct a search through the space of possible DAG configurations, seeking the structure that most accurately captures the dependency patterns present in the data. The search is guided by a scoring criterion that balances model fit against complexity (to avoid overfitting). The Hill-Climbing (HC) (Scutari et al., 2019) algorithm exemplifies this approach through a greedy local search strategy. HC initializes with a starting configuration (commonly the empty graph containing no edges), computes its score, and then iteratively explores local modifications, specifically, adding an edge, removing an edge, or reversing an edge's direction. Each modification that yields an improved score is accepted, with the process continuing until no further improvement is possible (reaching a hopefully local optimum). Other score-based methods include Tabu search (Scutari et al., 2019), which maintains a memory of recently explored structures to avoid cycling

and facilitate escape from local optima. Tabu search demonstrates particular advantages in comparative analyses and when handling large sample sizes.

Hybrid Algorithms: Combining Complementary Strengths Hybrid algorithms strategically combine the computational efficiency of constraint-based methods with the optimization power of score-based approaches. The typical hybrid strategy employs constraint-based independence tests to rapidly eliminate implausible edges and reduce the enormous space of candidate networks to a manageable skeleton structure. Subsequently, a score-based search refines this reduced space by optimizing edge orientations and evaluating remaining structural decisions (Briganti et al., 2024; Vowels et al., 2022).

Achieving Structural Stability Through Resampling A critical practical consideration in structure learning is that the recovered network structure may exhibit variability across repeated estimations, particularly when working with cross-sectional data of limited sample size. Following the guidelines outlined by Briganti, Scutari, and McNally (2023), researchers often use bootstrapping-based consensus methods to mitigate this instability and improve the robustness of he inferred structure.

1. **Resampled Estimation:** The selected structure learning algorithm is applied repeatedly (typically 1,000 or more iterations) to bootstrap resamples drawn with replacement from the original dataset. This generates a distribution of candidate network structures rather than a single point estimate.
2. **Edge Strength Thresholding:** Only edges that appear with sufficient frequency across the bootstrap distribution, exceeding a predetermined strength threshold (e.g., present in at least 85% of bootstrap samples), are retained in the final consensus network. This criterion ensures that only robust, replicable edges are included.
3. **Direction Confidence:** Among retained edges, the directional orientation must also demonstrate consistency, appearing in a specified direction above a minimum threshold proportion (e.g., oriented in the same direction in at least 50% of instances where the edge appears). This guards against including edges with ambiguous or unstable directionality.

This bootstrapping procedure yields a final averaged or *consensus BN* that represents the most stable and reproducible structural features present in the data, filtering out spurious edges that may arise from sampling variability.

Parameter Learning: Quantifying Network Relationships

Once the DAG structure has been established through structure learning, the second stage focuses on precisely quantifying the strength and functional form of the relationships encoded by that structure. This parameter estimation phase translates the qualitative structural map into a fully specified probabilistic model. As

established previously, the DAG structure enables the factorization of the global joint probability distribution $P(X)$ over all variables into a product of simpler local conditional distributions: $P(X_i \mid \text{Pa}(X_i))$, where $\text{Pa}(X_i)$ denotes the parent node set of variable X_i in the DAG. Parameter learning estimates the specific parameters governing each of these local distributions. Each local distribution characterizes how a single target variable depends on its immediate parent variables in the network. These localized models are considerably simpler to estimate than the full joint distribution, representing one of the key computational advantages afforded by the DAG structure. The specific parameters to be estimated depend fundamentally on the measurement characteristics of the variables:

- *Discrete Bayesian Networks:* When variables are categorical (such as Likert-scale items treated as ordered categories), the local conditional distributions are represented as conditional probability tables (CPTs). Each CPT enumerates the probability distribution over the possible values of a child node for every possible configuration of its parent nodes' values. These tables are typically estimated using maximum likelihood estimation from observed frequency counts.
- *Gaussian Bayesian Networks:* When variables are continuous or approximately continuous (the common assumption for interval-scaled data), local distributions are specified as linear regression models. Each node is regressed on its parent nodes, with the regression coefficients quantifying the conditional dependencies. The parameters to be estimated include regression coefficients and residual variances for each local model.

The parameter learning, integrated with the learned DAG structure, yields a complete probabilistic model (a fully specified Bayesian Network). This operational model supports automated probabilistic reasoning through *belief updating* or *BN inference*, enabling researchers to answer complex queries about conditional probabilities, predict unobserved variables, evaluate hypothetical interventions, and trace the propagation of evidence through the network structure. This inferential capacity represents the practical payoff of the BN framework, transforming empirical data into a tool for understanding and prediction within complex psychological systems.

An Example of Implementation of BNs using R

Here we present an example of how to apply BNs using the `bnlearn` package in R (Scutari, 2010). This example follows the procedures and results reported in the article *Bayesian networks to evaluate and test the Raven's Colored Progressive Matrices* (Orsoni, Spinoso, et al., 2025). The full reproducible code is available in the corresponding OSF repository: https://osf.io/2qvun.

The objective is to present to the reader the structure and parameter learning

stages in BNs implementation. The analysis begins by setting up the computational environment and preparing the data for Bayesian network analysis. The first step involves installing and loading the required R packages, including **bnlearn** for Bayesian network learning and inference.

Environment Setup and Data Preparation

```r
# Required packages
packages <- c(
  "bnlearn", "readxl", "ggplot2", "dplyr", "parallel"
)

# Check and Install missing packages
install_if_missing <- function(pkg) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    install.packages(pkg, dependencies = TRUE)
  }
}

# Install BiocManager if needed
if (!requireNamespace("BiocManager", quietly = TRUE)) {
  install.packages("BiocManager")
}

# Install packages
invisible(lapply(packages, install_if_missing))

# Load packages
lapply(packages, library, character.only = TRUE)
```

Listing 4: R Package Installation and Loading

The *check_levels* function is used to identify columns in a dataset that contain only a single level. We use this function to detect factor levels and exclude variables with fewer than two levels. This information was used to construct Table 4. The function *generate_blacklist* creates a data frame of all pairwise combinations of item names where the first item's number is greater than the second's. We used this function to define a directional exclusion list (a "blacklist") between items implementing a Sequential Data-Driven Model. The *test_generalizability_loocv* function performs Leave-One-Out Cross-Validation (LOOCV) to evaluate the generalizability of a Bayesian network. For each data point, it fits the network on the remaining data and computes a loss (e.g., log-likelihood or prediction error) on the left-out observation. It returns loss statistics and handles errors and insufficient data.

```r

### Some Functions

check_levels <- function(x) {
```

```
  if (is.factor(x)) {
    return(nlevels(x))
  } else if (is.numeric(x)) {
    return(length(unique(x)))
  } else {
    return(NA)
  }
}


generate_blacklist <- function(column_names) {
  n_items <- length(column_names)
  blacklist <- expand.grid(from = column_names, to = column_
    names)

  # Extract numbers from column names
  from_nums <- as.numeric(gsub("Item", "", blacklist$from))
  to_nums <- as.numeric(gsub("Item", "", blacklist$to))

  # Keep only rows where 'from' number is greater than 'to'
    number
  blacklist <- blacklist[from_nums > to_nums, ]

  return(blacklist)
}


# LOOCV for generalizability testing
test_generalizability_loocv <- function(network, data, loss_
    type = "logl") {

  n <- nrow(data)
  individual_losses <- numeric(n)
  failed_folds <- 0

  cat("Running LOOCV with", n, "folds...\n")

  for (i in 1:n) {
    tryCatch({
      # Leave one out
      train_data <- data[-i, ]
      test_data <- data[i, , drop = FALSE]

      # Check if we have enough training data
      if (nrow(train_data) < ncol(data)) {
        individual_losses[i] <- NA
        failed_folds <- failed_folds + 1
        next
      }

      # Fit the network structure on training data
```

```
      fitted_net <- bn.fit(network, train_data)

      # Calculate loss on test observation
      if (loss_type == "logl") {
        individual_losses[i] <- -logLik(fitted_net, test_data)
      } else if (loss_type == "pred") {
        # Predictive accuracy - need to implement for discrete data
        individual_losses[i] <- compute_prediction_error(fitted_net, test_data)
      }

      if (i %% 5 == 0) cat("Completed fold", i, "/", n, "\n")

    }, error = function(e) {
      individual_losses[i] <- NA
      failed_folds <- failed_folds + 1
      if (failed_folds <= 5) {  # Only show first few errors
        cat("Error in fold", i, ":", e$message, "\n")
      }
    })
  }

  # Analyze results
  valid_losses <- individual_losses[!is.na(individual_losses) & is.finite(individual_losses)]

  if (length(valid_losses) == 0) {
    cat("LOOCV failed completely - no valid predictions\n")
    return(NULL)
  }

  success_rate <- length(valid_losses) / n

  cat("\n=== LOOCV Generalizability Results ===\n")
  cat("Successful predictions:", length(valid_losses), "/", n,
      "(", round(success_rate * 100, 1), "%)\n")
  cat("Mean loss:", round(mean(valid_losses), 4), "\n")
  cat("Standard deviation:", round(sd(valid_losses), 4), "\n")
  cat("Min loss:", round(min(valid_losses), 4), "\n")
  cat("Max loss:", round(max(valid_losses), 4), "\n")

  return(list(
    individual_losses = individual_losses,
    valid_losses = valid_losses,
    mean_loss = mean(valid_losses),
    sd_loss = sd(valid_losses),
    success_rate = success_rate
  ))
}
```

Listing 5: Utility Functions

Once the packages are loaded, the data are imported from an Excel file containing responses to the Raven's Coloured Progressive Matrices (CPMs) from 255 participants aged 4 to 11 years. The dataset is then subdivided into seven subgroups based on academic year, ranging from second-year kindergarten (age 6) to fifth-year primary school (age 11). For each subgroup, a global score is calculated by summing responses across all 36 items, and participants with zero global scores are removed to ensure data quality.

```
## Import Data
rm(list=setdiff(ls(), "..."))

path <- "Your own path"
setwd(path)

data <- read_excel("data.xlsx")

## Kindergarten Children

data_6 = data[data$`ACADEMIC YEAR` == 6, ]
data_7 = data[data$`ACADEMIC YEAR` == 7, ]

## Primary Children

data_1 = data[data$`ACADEMIC YEAR` == 1, ]
data_2 = data[data$`ACADEMIC YEAR` == 2, ]
data_3 = data[data$`ACADEMIC YEAR` == 3, ]
data_4 = data[data$`ACADEMIC YEAR` == 4, ]
data_5 = data[data$`ACADEMIC YEAR` == 5, ]

# Calculating global scores and filtering zero scores
data_1$global_score <- rowSums(data_1[, 5:40])
data_1 <- data_1[data_1$global_score != 0, ]

data_2$global_score <- rowSums(data_2[, 5:40])
data_2 <- data_2[data_2$global_score != 0, ]

data_3$global_score <- rowSums(data_3[, 5:40])
data_3 <- data_3[data_3$global_score != 0, ]

data_4$global_score <- rowSums(data_4[, 5:40])
data_4 <- data_4[data_4$global_score != 0, ]

data_5$global_score <- rowSums(data_5[, 5:40])
data_5 <- data_5[data_5$global_score != 0, ]

data_6$global_score <- rowSums(data_6[, 5:40])
data_6 <- data_6[data_6$global_score != 0,]

data_7$global_score <- rowSums(data_7[, 5:40])
data_7 <- data_7[data_7$global_score != 0,]
```

```
43
44 comb_data = list(data_6, data_7, data_1, data_2, data_3, data_
       4, data_5)
45 data_names <- c("Data 6", "Data 7", "Data 1", "Data 2", "Data 3
       ", "Data 4", "Data 5")
```

Listing 6: Data Import and Subgroup Creation

Lastly, the data preparation converts the numeric item responses to factor variables, which is necessary for discrete Bayesian network analysis. Variable names are standardized to follow the pattern "Item_1" through "Item_36", and the code includes a diagnostic step to identify and remove any variables with fewer than two levels, as these cannot contribute to meaningful network relationships.

```
1
2 for (i in 1:length(comb_data)) {
3
4   # Identify numeric columns in the specified range (columns 5
      to 40)
5   numeric_cols <- sapply(comb_data[[i]][, 5:40], is.numeric)
6
7   # Convert numeric columns to factors
8   comb_data[[i]][, 5:40][, numeric_cols] <- lapply(comb_data[[i
      ]][, 5:40][, numeric_cols], as.factor)
9
10  # Rename columns as Item_1 to Item_36 (since 5:40 represents
      36 columns)
11  colnames(comb_data[[i]][, 5:40]) <- paste0("Item_", 1:36)
12
13  # Assign the modified subset to a new variable (e.g., data_1,
       data_2, etc.)
14  assign(paste0("data_", i), comb_data[[i]][, 5:40])
15
16  # Print the structure of the modified data subset
17  str(comb_data[[i]][, 5:40])
18 }
```

Listing 7: Renaming and converting the dataset colums in the appropriate format

Theoretical Network Models

The provided code is intended to replicate the analysis for the second-year Kindergarten population. To replicate the analysis for a different population, you should replace the correct dataset name wherever necessary throughout the code. E.g. for the first year primary school dataset use: comb_data[[1]], for the second comb_data[[2]], and so on.

Four theory-driven network structures are constructed to represent different hypotheses about item dependencies in the CPM test. The **Complete Independence Model** (Intransitive Dependency Model) assumes no relationships between items, represented by an empty graph with no edges. This serves as the baseline

null hypothesis for comparison. The **Fully Dependent Model** (Transitive Dependency Model) represents the opposite extreme, where each item depends on all previous items in the sequence. This creates a fully connected DAG with edges from every earlier item to every later item. The **Sequential Dependency Model** (Transitive Independency Model) implements a simpler chain structure where each item depends only on the immediately preceding item.

```
1
2  data_6 <- as.data.frame(comb_data[[6]][, 5:40])
3  str(data_6)
4
5  # Apply the function to all columns
6  level_counts <- sapply(data_6, check_levels)
7
8  # Print the number of levels for each variable
9  print("Number of levels in each variable:")
10 print(level_counts)
11
12 # Identify variables with less than two levels
13 problem_vars <- names(level_counts[level_counts < 2])
14 print("Variables with less than two levels:")
15 print(problem_vars)
16
17 # Remove variables with less than two levels
18 data_6 <- data_6[, level_counts >= 2]
19
20 ### Complete Indipendent Graph or Intransitive Dependency Model
21
22 ind_net_6 <- empty.graph(nodes = names(data_6))
23
24 ### Fully dependent graph or Transitive Dependency Model
25
26 net_6 <- empty.graph(nodes = names(data_6))
27
28 # Add edges in a recursive order
29 for (i in 2:ncol(data_6)) {
30   for (j in 1:(i-1)) {
31     net_6 <- set.arc(net_6, colnames(data_6)[j], to = colnames(
     data_6)[i])
32   }
33 }
34
35 ### Fully dependent on the previous item response or Transitive
       Independency Model
36
37
38 net_dep_6 <- empty.graph(nodes = names(data_6))
39
40 # Add edges to create a chain structure
41 for (i in 1:(ncol(data_6) - 1)) {
```

```
  net_dep_6 <- set.arc(net_dep_6, from = colnames(data_6)[i],
    to = colnames(data_6)[i + 1])
}
```

Listing 8: Theoretical Models

Data-Driven Network Learning via Bootstrap Analysis

The data-driven approach employs bootstrap stability analysis combined with the Tabu search algorithm to learn optimal network structures from the data. The Tabu search is a score-based structure learning algorithm that efficiently explores the space of possible network structures while avoiding local optima through an adaptive memory mechanism.

A critical aspect of the analysis is determining the optimal Imaginary Sample Size (ISS) parameter and threshold for arc inclusion. The code systematically tests ISS values of 1, 2, 5, 10, and 20 across 500 bootstrap samples. For each ISS value, the algorithm runs parallel computations using the Tabu search with specific parameters: 15 random restarts, a perturbation parameter of 75, and a maximum of 1500 iterations.

The bootstrap analysis produces arc strength estimates, which represent the proportion of bootstrap samples in which each potential arc appears. The analysis examines multiple threshold criteria, including the data-driven threshold (automatically estimated by bnlearn), the 85th percentile, 90th percentile, and median of arc strengths. Results are summarized showing the number of arcs retained at each threshold and runtime performance metrics.

Two variants of the data-driven approach are implemented: a Fully Data-Driven Model that allows any arc direction, and a Sequential Data-Driven Model that constrains the search using a blacklist to prevent arcs from later items to earlier items. The blacklist ensures that the learned structure respects the sequential ordering of test items while still allowing the algorithm to discover complex dependencies among items.

```

iss_values <- c(1, 2, 5, 10, 20)
samples <- 500

# Initialize storage for results
bootstrap_results <- list()
bootstrap_summary <- data.frame(
  iss = iss_values,
  estimated_threshold = NA,
  total_arcs_learned = NA,
  avg_net_arcs = NA,
  max_strength = NA,
  mean_strength = NA,
  runtime_mins = NA,
  success = FALSE,
```

```
  stringsAsFactors = FALSE
)

cat("=== BOOTSTRAP SENSITIVITY ANALYSIS ===\n")
cat(sprintf("Testing ISS values: %s\n", paste(iss_values,
    collapse = ", ")))
cat(sprintf("Bootstrap samples: %d\n", samples))
cat("Using data-driven threshold estimation\n\n")

total_start_time <- Sys.time()

for (i in seq_along(iss_values)) {
  iss <- iss_values[i]
  cat(sprintf("--- Running Bootstrap with ISS = %s (%d/%d) ---\
    n",
              iss, i, length(iss_values)))

  iss_start_time <- Sys.time()

  tryCatch({

    cl <- makeCluster(4)

    clusterSetRNGStream(cl, iseed = 42)
    set.seed(NULL)

    # Run bootstrap with current ISS
    bootstrap_results[[paste0("ISS_", iss)]] <- boot.strength(
      data = data_6,
      algorithm = "tabu",
      algorithm.args = list(
        score = "bde",
        iss = iss,
        restart = 15,
        perturb = 75,
        max.iter = 1500
      ),
      R = samples,
      cluster = cl
    )

    stopCluster(cl)
    iss_end_time <- Sys.time()
    runtime_mins <- as.numeric(difftime(iss_end_time, iss_start
    _time, units = "mins"))

    # Get current bootstrap results
    current_result <- bootstrap_results[[paste0("ISS_", iss)]]

    # Extract the data-driven threshold
    data_threshold <- attr(current_result, "threshold")
```

```
64
65    # Create averaged network with estimated threshold
66    avg_net <- averaged.network(current_result, threshold =
      data_threshold)
67
68    # Calculate arc strength statistics
69    arc_strengths <- current_result$strength[current_result$
      strength > 0]
70
71    # Store summary statistics
72    bootstrap_summary$estimated_threshold[i] <- data_threshold
73    bootstrap_summary$total_arcs_learned[i] <- nrow(current_
      result)
74    bootstrap_summary$avg_net_arcs[i] <- narcs(avg_net)
75    bootstrap_summary$max_strength[i] <- max(current_result$
      strength)
76    bootstrap_summary$mean_strength[i] <- ifelse(length(arc_
      strengths) > 0, mean(arc_strengths), 0)
77    bootstrap_summary$runtime_mins[i] <- runtime_mins
78    bootstrap_summary$success[i] <- TRUE
79
80    # Print summary for current ISS
81    cat(sprintf("    Success! Runtime: %.2f minutes\n", runtime
      _mins))
82    cat(sprintf("    Total potential arcs examined: %d\n", nrow
      (current_result)))
83    cat(sprintf("    Arcs in averaged network (threshold=%.3f):
       %d\n",
84                data_threshold, narcs(avg_net)))
85    cat(sprintf("    Max arc strength: %.3f\n", max(current_
      result$strength)))
86    if (length(arc_strengths) > 0) {
87      cat(sprintf("    Mean arc strength (>0): %.3f\n", mean(
      arc_strengths)))
88    } else {
89      cat("    No arcs with positive strength found\n")
90    }
91
92    # Print top 5 strongest arcs if any exist
93    if (length(arc_strengths) > 0) {
94      top_arcs <- current_result[order(current_result$strength,
       decreasing = TRUE), ][1:min(5, sum(current_result$strength
       > 0)), ]
95      cat("    Top 5 strongest arcs:\n")
96      for (j in 1:nrow(top_arcs)) {
97        cat(sprintf("    %s -> %s (%.3f)\n",
98                    top_arcs$from[j], top_arcs$to[j], top_arcs$
      strength[j]))
99      }
100   }
101   cat("\n")
```

```
102
103   }, error = function(e) {
104     iss_end_time <- Sys.time()
105     runtime_mins <- as.numeric(difftime(iss_end_time, iss_start
      _time, units = "mins"))
106
107     bootstrap_summary$runtime_mins[i] <- runtime_mins
108     bootstrap_summary$success[i] <- FALSE
109
110     cat(sprintf("    ERROR for ISS = %s: %s\n", iss, e$message)
      )
111     cat(sprintf("    Runtime before failure: %.2f minutes\n\n",
       runtime_mins))
112   })
113 }
114
115 total_end_time <- Sys.time()
116 total_runtime <- as.numeric(difftime(total_end_time, total_
      start_time, units = "mins"))
117
118 # Print final summary
119 cat("=== BOOTSTRAP SENSITIVITY SUMMARY ===\n")
120 cat(sprintf("Total runtime: %.2f minutes\n", total_runtime))
121 cat(sprintf("Successful runs: %d/%d\n\n", sum(bootstrap_summary
      $success), length(iss_values)))
122
123 print(bootstrap_summary)
124
125 # Calcola threshold data-driven per ogni ISS
126 data_driven_analysis <- data.frame(
127   iss = iss_values,
128   fixed_threshold = bootstrap_summary$estimated_threshold,
129   percentile_85 = NA,
130   percentile_90 = NA,
131   median_threshold = NA,
132   arcs_with_85_perc = NA,
133   arcs_with_90_perc = NA,
134   arcs_with_median = NA
135 )
136
137 for (i in seq_along(iss_values)) {
138   iss <- iss_values[i]
139   result <- bootstrap_results[[paste0("ISS_", iss)]]
140   strengths <- result$strength[result$strength > 0]
141
142   # Calcola threshold alternativi
143   perc_85 <- quantile(strengths, 0.85)
144   perc_90 <- quantile(strengths, 0.90)
145   median_thresh <- median(strengths)
146
147   data_driven_analysis$percentile_85[i] <- perc_85
```

```
  data_driven_analysis$percentile_90[i] <- perc_90
  data_driven_analysis$median_threshold[i] <- median_thresh

  # Calcola numero di archi con threshold alternativi
  data_driven_analysis$arcs_with_85_perc[i] <- narcs(averaged.
    network(result, threshold = perc_85))
  data_driven_analysis$arcs_with_90_perc[i] <- narcs(averaged.
    network(result, threshold = perc_90))
  data_driven_analysis$arcs_with_median[i] <- narcs(averaged.
    network(result, threshold = median_thresh))
}

print(data_driven_analysis)

# Optimal network based on sensitivity analysis
optimal_iss <- 20
optimal_threshold <- 0.762

# Create averaged network with optimal parameters
avg.dag_boot_tabu_6 <- averaged.network(bootstrap_results$ISS_
    20, threshold = optimal_threshold)

# Verify the results match our analysis
cat("=== OPTIMAL NETWORK SUMMARY ===\n")
cat(sprintf("ISS used: %d\n", optimal_iss))
cat(sprintf("Threshold used: %.3f (90th percentile)\n", optimal
    _threshold))
cat(sprintf("Number of arcs in final network: %d\n", narcs(avg.
    dag_boot_tabu_6)))

# Plot the network
graphviz.plot(avg.dag_boot_tabu_6, shape = "ellipse",
              main = "Optimal Bayesian Network (ISS=20,
    threshold=0.762)")

undirected.arcs(avg.dag_boot_tabu_6)
```

Listing 9: Bootstrap sensitivity analysis for different ISS values for the Fully Data-Driven Model

```

column_names <- colnames(data_6)
print(column_names)
# Generate the blacklist
bl <- generate_blacklist(column_names)

# Print the first few rows of the blacklist to verify
print(head(bl))

# Bootstrap sensitivity analysis for different ISS values
iss_values <- c(1, 2, 5, 10, 20)
```

```r
samples <- 500

# Initialize storage for results
bootstrap_results <- list()
bootstrap_summary <- data.frame(
  iss = iss_values,
  estimated_threshold = NA,
  total_arcs_learned = NA,
  avg_net_arcs = NA,
  max_strength = NA,
  mean_strength = NA,
  runtime_mins = NA,
  success = FALSE,
  stringsAsFactors = FALSE
)

cat("=== BOOTSTRAP SENSITIVITY ANALYSIS ===\n")
cat(sprintf("Testing ISS values: %s\n", paste(iss_values,
    collapse = ", ")))
cat(sprintf("Bootstrap samples: %d\n", samples))
cat("Using data-driven threshold estimation\n\n")

total_start_time <- Sys.time()

for (i in seq_along(iss_values)) {
  iss <- iss_values[i]
  cat(sprintf("--- Running Bootstrap with ISS = %s (%d/%d) ---\
    n",
              iss, i, length(iss_values)))

  iss_start_time <- Sys.time()

  tryCatch({

    cl <- makeCluster(4)

    clusterSetRNGStream(cl, iseed = 42)
    set.seed(NULL)

    # Run bootstrap with current ISS
    bootstrap_results[[paste0("ISS_", iss)]] <- boot.strength(
      data = data_6,
      algorithm = "tabu",
      algorithm.args = list(
        score = "bde",
        iss = iss,
        restart = 15,
        perturb = 75,
        max.iter = 1500,
        blacklist = bl
      ),
```

```
      R = samples,
      cluster = cl
    )

    iss_end_time <- Sys.time()
    runtime_mins <- as.numeric(difftime(iss_end_time, iss_start
    _time, units = "mins"))

    # Get current bootstrap results
    current_result <- bootstrap_results[[paste0("ISS_", iss)]]

    # Extract the data-driven threshold
    data_threshold <- attr(current_result, "threshold")

    # Create averaged network with estimated threshold
    avg_net <- averaged.network(current_result, threshold =
    data_threshold)

    # Calculate arc strength statistics
    arc_strengths <- current_result$strength[current_result$
    strength > 0]

    # Store summary statistics
    bootstrap_summary$estimated_threshold[i] <- data_threshold
    bootstrap_summary$total_arcs_learned[i] <- nrow(current_
    result)
    bootstrap_summary$avg_net_arcs[i] <- narcs(avg_net)
    bootstrap_summary$max_strength[i] <- max(current_result$
    strength)
    bootstrap_summary$mean_strength[i] <- ifelse(length(arc_
    strengths) > 0, mean(arc_strengths), 0)
    bootstrap_summary$runtime_mins[i] <- runtime_mins
    bootstrap_summary$success[i] <- TRUE

    # Print summary for current ISS
    cat(sprintf("    Success! Runtime: %.2f minutes\n", runtime
    _mins))
    cat(sprintf("    Total potential arcs examined: %d\n", nrow
    (current_result)))
    cat(sprintf("    Arcs in averaged network (threshold=%.3f):
     %d\n",
                data_threshold, narcs(avg_net)))
    cat(sprintf("    Max arc strength: %.3f\n", max(current_
    result$strength)))
    if (length(arc_strengths) > 0) {
      cat(sprintf("    Mean arc strength (>0): %.3f\n", mean(
    arc_strengths)))
    } else {
      cat("    No arcs with positive strength found\n")
    }

```

```
    # Print top 5 strongest arcs if any exist
    if (length(arc_strengths) > 0) {
      top_arcs <- current_result[order(current_result$strength,
     decreasing = TRUE), ][1:min(5, sum(current_result$strength
     > 0)), ]
      cat("    Top 5 strongest arcs:\n")
      for (j in 1:nrow(top_arcs)) {
        cat(sprintf("    %s -> %s (%.3f)\n",
                    top_arcs$from[j], top_arcs$to[j], top_arcs$
    strength[j]))
      }
    }
    cat("\n")

  }, error = function(e) {
    iss_end_time <- Sys.time()
    runtime_mins <- as.numeric(difftime(iss_end_time, iss_start
    _time, units = "mins"))

    bootstrap_summary$runtime_mins[i] <- runtime_mins
    bootstrap_summary$success[i] <- FALSE

    cat(sprintf("    ERROR for ISS = %s: %s\n", iss, e$message)
    )
    cat(sprintf("    Runtime before failure: %.2f minutes\n\n",
     runtime_mins))
  })
}

total_end_time <- Sys.time()
total_runtime <- as.numeric(difftime(total_end_time, total_
    start_time, units = "mins"))

# Print final summary
cat("=== BOOTSTRAP SENSITIVITY SUMMARY ===\n")
cat(sprintf("Total runtime: %.2f minutes\n", total_runtime))
cat(sprintf("Successful runs: %d/%d\n\n", sum(bootstrap_summary
    $success), length(iss_values)))

print(bootstrap_summary)

# Calcola threshold data-driven per ogni ISS
data_driven_analysis <- data.frame(
  iss = iss_values,
  fixed_threshold = bootstrap_summary$estimated_threshold,
  percentile_85 = NA,
  percentile_90 = NA,
  median_threshold = NA,
  arcs_with_85_perc = NA,
  arcs_with_90_perc = NA,
  arcs_with_median = NA
```

```
)

for (i in seq_along(iss_values)) {
  iss <- iss_values[i]
  result <- bootstrap_results[[paste0("ISS_", iss)]]
  strengths <- result$strength[result$strength > 0]

  # Calcola threshold alternativi
  perc_85 <- quantile(strengths, 0.85)
  perc_90 <- quantile(strengths, 0.90)
  median_thresh <- median(strengths)

  data_driven_analysis$percentile_85[i] <- perc_85
  data_driven_analysis$percentile_90[i] <- perc_90
  data_driven_analysis$median_threshold[i] <- median_thresh

  # Calcola numero di archi con threshold alternativi
  data_driven_analysis$arcs_with_85_perc[i] <- narcs(averaged.
    network(result, threshold = perc_85))
  data_driven_analysis$arcs_with_90_perc[i] <- narcs(averaged.
    network(result, threshold = perc_90))
  data_driven_analysis$arcs_with_median[i] <- narcs(averaged.
    network(result, threshold = median_thresh))
}

print(data_driven_analysis)

# Optimal network based on sensitivity analysis
optimal_iss <- 20
optimal_threshold <- 0.758

# Create averaged network with optimal parameters
avg.dag_boot_tabu_6_bl <- averaged.network(bootstrap_results$
    ISS_20, threshold = optimal_threshold)

# Verify the results match our analysis
cat("=== OPTIMAL NETWORK SUMMARY ===\n")
cat(sprintf("ISS used: %d\n", optimal_iss))
cat(sprintf("Threshold used: %.3f (90th percentile)\n", optimal
    _threshold))
cat(sprintf("Number of arcs in final network: %d\n", narcs(avg.
    dag_boot_tabu_6_bl)))

# Plot the network
graphviz.plot(avg.dag_boot_tabu_6_bl, shape = "ellipse",
              main = "Average Sequential Network for Second
    Year Kindergarten")
```

Listing 10: Bootstrap sensitivity analysis for different ISS values for the Sequential Data-Driven Model

Model Comparison Using Bayes Factors

After learning all candidate network structures, the models are compared using Bayes Factors (BF), which provide a principled approach to hypothesis testing in the Bayesian framework. The analysis computes pairwise comparisons between all five models: the Complete Independence Model, Fully Dependent Model, Sequential Dependency Model, Fully Data-Driven Model, and Sequential Data-Driven Model. Bayes Factors are calculated on the log scale using the BF() function from bnlearn, where positive values indicate evidence favoring the numerator model and negative values favor the denominator model. The magnitude of the log BF indicates the strength of evidence, with larger absolute values representing stronger support for one model over the other.

```
1
2  ## Model A over Model B
3  BF(ind_net_6, net_6, data_6, log = TRUE)
4  ## Model A over Model C
5  BF(ind_net_6, net_dep_6, data_6, log = TRUE)
6  ## Model A over Model D
7  BF(ind_net_6, avg.dag_boot_tabu_6, data_6, log = TRUE)
8  ## Model A over Model E
9  BF(ind_net_6, avg.dag_boot_tabu_6_bl, data_6, log = TRUE)
10
11 ## Model B over Model C
12 BF(net_6, net_dep_6, data_6, log = TRUE)
13 ## Model B over Model D
14 BF(net_6, avg.dag_boot_tabu_6, data_6, log = TRUE)
15 ## Model B over Model E
16 BF(net_6, avg.dag_boot_tabu_6_bl, data_6, log = TRUE)
17
18 ## Model C over Model D
19 BF(net_dep_6, avg.dag_boot_tabu_6, data_6, log = TRUE)
20 ## Model C over Model E
21 BF(net_dep_6, avg.dag_boot_tabu_6_bl, data_6, log = TRUE)
22
23 ## Model D over Model E
24 BF(avg.dag_boot_tabu_6, avg.dag_boot_tabu_6_bl, data_6, log =
       TRUE)
```

Listing 11: Computing the Bayes Factor for Hypothesis Testing

Testing Generalizability by using LOOCV function

To evaluate the generalizability and predictive performance of the learned networks, Leave-One-Out Cross-Validation (LOOCV) is implemented. The `test_generalizability_loocv()` function systematically removes each observation from the dataset, fits the network structure on the remaining data, and computes the log-likelihood loss on the held-out observation. This procedure is repeated for all observations, providing individual loss estimates for each data point. The

function includes error handling to manage cases where training data are insufficient or model fitting fails. Summary statistics are computed including mean loss, standard deviation, minimum and maximum loss values, and the success rate (proportion of observations for which predictions could be generated). LOOCV results provide crucial information about model stability and predictive validity across different developmental stages. Models with lower average loss and higher success rates demonstrate better generalizability to unseen data.

```
1
2 set.seed(42)
3 loocv_results_6 <- test_generalizability_loocv(avg.dag_boot_
    tabu_6_bl, data_6, loss_type = "logl")
```

Listing 12: Generalizability by using LOOCV

Perspectives on Bayesian Networks applications in Psychological Research

As the reader can notice, the use of BNs in psychology gives researchers a set of powerful tools for improving theoretical work and strengthening empirical investigation. By modeling directed and plausible causal relationships, BNs move beyond traditional correlational approaches and open new possibilities for understanding psychological constructs.

Recent literature, focused on BNs approach to support theory formulation and formalization by giving researchers a framework for developing clearer and more rigorous theoretical models (Orsoni, Benassi, & Scutari, 2025; Vowels, 2025). This approach could transform vague or informal ideas into precise DAGs, forcing explicit statements about which variables influence others and reducing ambiguity in theoretical claims. Because BNs allow the inclusion of expert constraints, such as specifying which connections should or should not exist, they enable researchers to incorporate prior knowledge directly into the modeling process, resulting in a more refined and data-informed prototype of the theory (Vowels, 2025). In addition, BNs encourage researchers to conceptualize psychological constructs as systems of causally interacting components rather than as reflections of hidden latent traits, allowing the exploration of complex and non-linear relationships among variables (Orsoni, Spinoso, et al., 2025). This systems perspective can reveal insights that traditional methods might overlook; for example, BN analyses of empathy have shown that emotional components may play a primary causal role relative to cognitive ones (Briganti et al., 2021). By highlighting plausible causal directions among items, BNs also serve as tools for hypothesis generation, suggesting new relationships and mechanisms that may not be visible through correlational approaches such as factor analysis or undirected network models (Orsoni, Spinoso, et al., 2025). Moreover, BNs help determine whether a central symp-

tom is primarily driving other symptoms or is largely driven by them, addressing a major limitation of traditional network models (Briganti et al., 2023).

Beyond symptoms, BNs reveal detailed item-level dependencies within psychometric measures, providing evidence about where the assumption of local independence fails. For instance, research using Raven's Colored Progressive Matrices shows that BN models capture complex, developmentally sensitive interactions between items more accurately than theory-driven models (Orsoni, Spinoso, et al., 2025). Because DAGs directly encode theoretical assumptions, BNs also help close the "theory-to-model gap," ensuring that statistical results are causally meaningful and remain consistent with the theory they are meant to represent.

Lastly, BNs also enhance causal inference and prediction by giving researchers tools to reason about cause and effect even when working solely with observational data (Vowels, 2025). They support automated causal reasoning through belief updating and systematic inference, allowing the network to function as a working model of the psychological system being studied. This enables researchers to explore theoretical implications through simulations rather than relying exclusively on empirical trials (Vowels, 2025). BNs also make it possible to simulate interventions, such as modeling the effect of a treatment by altering the network structure, for example, removing the causes of a symptom and assigning it a new distribution, to examine "what-if" scenarios that cannot be experimentally tested (Pearl, 2009). Importantly, the graphical structure clarifies which variables act as confounders and therefore need to be controlled to estimate causal effects accurately (Pearl, 2009; Vowels, 2025). This transparency strengthens causal conclusions and helps researchers design more rigorous and interpretable studies.

A Guide to BNs Software Packages

We would conclude this chapter by presenting to the reader a list of the most relevant packages and softwares can be used for structural and parameter learning stages of BNs; updated to 2025. These information have been retrieved from the work of Gaudillo and colleagues (2025).

Package - Software	Authors	Learning Type	License	Tool
gCastle	Huawei Noah's Ark Lab	Structure	Free	Python
bnlearn (R)	Marco Scutari	Both	Free	R
pgmpy	Ankur Ankan et al.	Both	Free	Python
Tetrad	CMU-CLeaR Group	Structure	Free	Java
Causal Command	Center for Causal Discovery	Structure	Free	Java (CLI)
causal-learn	CMU-CLeaR Group	Structure	Free	Python
pcalg	Markus Kalisch et al.	Structure	Free	R
LiNGAM	T. Ikeuchi et al.	Structure	Free	Python
CDT	D. Kalainathan, O. Goudet	Structure	Free	Python
pyAgrum	aGrUM Team	Both	Free	Python
Bnlearn (Python)	Erdogan Taskesen	Both	Free	Python
OpenMarkov	Research Centre for Intelligent DSS	Both	Free	Java
pomegranate	Jacob Schreiber	Both	Free	Python
BayesFusion	BayesFusion LLC	Both	Commercial	Multi-lang.
BayesiaLab	L. Jouffe, P. Munteanu	Both	Commercial	Standalone
Bayes Server	Bayes Server Ltd.	Both	Commercial	Multi-lang.

27. Summary of BN and causal discovery software. Learning Type could be Structural, Parameter, or Both.

References

Almond, R. G., Mislevy, R. J., Steinberg, L. S., Yan, D., & Williamson, D. M. (2015). *Bayesian networks in educational assessment*. Springer. https://doi.org/10.1007/978-1-4939-2125-6

Briganti, G., Scutari, M., Epskamp, S., Borsboom, D., Hoekstra, R. H. A., Golino, H. F., Christensen, A. P., Morvan, Y., Ebrahimi, O. V., Costantini, G., Heeren, A., de Ron, J., Bringmann, L. F., Huth, K., Haslbeck, J. M. B., Isvoranu, A.-M., Marsman, M., Blanken, T., Gilbert, A., et al. (2024). Network analysis: An overview for mental health research. *International Journal of Methods in Psychiatric Research*, *33*(4), e2034. https://doi.org/10.1002/mpr.2034

Briganti, G., Scutari, M., & McNally, R. J. (2023). A tutorial on bayesian networks for psychopathology researchers. *Psychological Methods*, *28*(4), 947–961. https://doi.org/10.1037/met0000479

Briganti, G., Scutari, M., & Linkowski, P. (2021). Network structures of symptoms from the zung depression scale. *Psychological Reports*, *124*(4), 1897–1911. https://doi.org/10.1177/0033294120942116

Culbertson, M. J. (2016). Bayesian networks in educational assessment: The state of the field. *Applied Psychological Measurement*, *40*(1), 3–21. https://doi.org/10.1177/0146621615590401

Epskamp, S., Rhemtulla, M., & Borsboom, D. (2017). Generalized network psychometrics: Combining network and latent variable models. *Psychometrika*, *82*(4), 904–927. https://doi.org/10.1007/s11336-017-9557-x

Gaudillo, J., Astrologo, N., Stella, F., Acerbi, E., & Canonaco, F. (2025). A guide to bayesian networks software packages for structure and parameter learning – 2025 edition. https://doi.org/10.48550/arXiv.2503.17025

Grzegorczyk, M. (2024). Being bayesian about learning bayesian networks from ordinal data. *International Journal of Approximate Reasoning*, *170*, 109205. https://doi.org/10.1016/j.ijar.2024.109205

Margaritis, D. (2003). *Learning bayesian network model structure from data* (tech. rep. No. CMU-CS-03-153). School of Computer Science, Carnegie Mellon University. https://apps.dtic.mil/sti/citations/ADA461103

Orsoni, M., Spinoso, M., Garofalo, S., Mazzoni, N., Giovagnoli, S., de Chiusole, D., Anselmi, P., Bacherini, A., Pierluigi, I., Stefanutti, L., Balboni, G., & Benassi, M. (2025). Bayesian networks to evaluate and test the raven's colored progressive matrices. *Intelligence*, *113*, 101964. https://doi.org/10.1016/j.intell.2025.101964

Orsoni, M., Benassi, M., & Scutari, M. (2025). Information theory, machine learning, and bayesian networks in the analysis of dichotomous and likert responses for questionnaire psychometric validation [Advance online publication]. *Psychological Methods*. https://doi.org/10.1037/met0000713

Pearl, J. (2009). *Causality: Models, reasoning, and inference* (2nd ed.). Cambridge University Press.

Scutari, M. (2010). Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, *35*(3), 1–22. https://doi.org/10.18637/jss.v035.i03

Scutari, M., Vitolo, C., & Tucker, A. (2019). Learning bayesian networks from big data with greedy search: Computational complexity and efficient implementation. *Statistics and Computing*, *29*(5), 1095–1108. https://doi.org/10.1007/s11222-019-09857-1

Spirtes, P., & Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, *9*(1), 62–72. https://doi.org/10.1177/089443939100900106

Tsamardinos, I., Aliferis, C. F., & Statnikov, A. R. (2003). Algorithms for large scale markov blanket discovery. *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2003)*, 376–380. https://aaai.org/papers/flairs-2003-073/

Vowels, M. J., Camgoz, N. C., & Bowden, R. (2022). D'ya like dags? a survey on structure learning and causal discovery. *ACM Computing Surveys*, *55*(4), Article 82. https://doi.org/10.1145/3527154

Vowels, M. J. (2025). A causal research pipeline and tutorial for psychologists and social scientists [Advance online publication]. *Psychological Methods*. https://doi.org/10.1037/met0000673

Wang, S. (2021). Recent integrations of latent variable network modeling with psychometric models. *Frontiers in Psychology*, *12*, 773289. https://doi.org/10.3389/fpsyg.2021.773289