Network Estimation and Analysis with Nestimate

Mohammed Saqr, University of Eastern Finland
Sonsoles López-Pernas, University of Eastern Finland
Kamila Misiejuk, FernUniversität in Hagen, Germany

25 November 2026

Contents

Part I: Transition Network Analysis **2**

Theoretical Grounding . . . 2
Data . . . 2
Building Networks . . . 3
Validation . . . 7
Clustering . . . 8
Post-hoc Covariate Analysis . . . 9

Part II: Psychological Network Analysis **10**

Theoretical Grounding . . . 10
Data . . . 11
Regularized Network (EBICglasso) . . . 11

How to cite this article **11**

References **12**

`Nestimate` (Saqr et al., 2026) is a unified framework for estimating, validating, and comparing networks from sequential and cross-sectional data. It implements two complementary paradigms: **Transition Network Analysis (TNA)**, which models the relational dynamics of temporal processes as weighted directed networks using stochastic Markov models (Saqr et al., 2025c); and **Psychological Network Analysis (PNA)**, which estimates the conditional dependency structure among variables using regularized partial correlations and graphical models (Saqr et al., 2024). Both paradigms share the same `build_network()` interface, the same validation engine (bootstrap, permutation, centrality stability), and the same output format — enabling researchers to apply a consistent analytic workflow across fundamentally different data types.

This vignette demonstrates both paradigms, covering network estimation, statistical validation, data-driven clustering, and group comparison.

Part I: Transition Network Analysis

Theoretical Grounding

TNA uses stochastic process modeling to capture the dynamics of temporal processes via Markov models. Markov models align with the view that a temporal process is an outcome of a stochastic data generating process that produces various network configurations or patterns based on rules, constraints, or guiding principles. The transitions are governed by a stochastic process: the specific ways in which the system changes or evolves is rather random and therefore cannot be strictly determined. That is, the transitions are probabilistically dependent on preceding states — a method that assumes events are probabilistically dependent on the preceding ones. Modeling the temporal order of learning events has been used to study the temporal patterns of students' interactions and self-regulation (Saqr et al., 2019) and to map students' self-regulated learning tactics (Peeters et al., 2020). In network science, temporal networks keep the timing and order of interactions as part of the network (Holme & Saramäki, 2012; Holme, 2015).

The main principle of TNA is representing the transition matrix between events as a graph to take full advantage of graph theory potentials and the wealth of network analysis (Wasserman & Faust, 1994; Newman, 2018). TNA brings network measures at the node, edge, and graph level; pattern mining through dyads, triads, and communities; clustering of sub-networks into typical behavioral strategies; and rigorous statistical validation at each edge through bootstrapping, permutation, and case-dropping techniques. Such statistical rigor that brings validation and hypothesis testing at each step of the analysis offers a method for researchers to build, verify, and advance existing theories on the basis of a robust scientific approach.

For a comprehensive introduction to TNA in learning analytics, see Saqr et al. (2025a, 2025c). For detailed tutorials on TNA clustering and heterogeneity analysis, see López-Pernas et al. (2025). TNA is also implemented in the R package tna (Tikka et al., 2025).

Data

The `human_long` dataset contains 10,796 coded human interaction turns from 429 human-AI pair programming sessions across 34 projects. Each row is a single turn, with `code` recording the interaction type, `session_id` identifying the session, and `timestamp` providing temporal ordering. Vibe coding, programming by conversation with an AI assistant (Sarkar & Drosos, 2025), has been studied with transition networks and sequence analysis (Saqr, Misiejuk, & López-Pernas, 2026; Misiejuk et al., 2026).

```
library(Nestimate)
head(human_long)
#>   message_id   project   session_id  timestamp session_date      code
#> 1       3439 Project_7 0086cabebd15 1772661600   2026-03-05   Specify
#> 2       3439 Project_7 0086cabebd15 1772661600   2026-03-05   Command
#> 3       3439 Project_7 0086cabebd15 1772661600   2026-03-05   Specify
```

```
#> 4       3440 Project_7 0086cabebd15 1772661600   2026-03-05 Interrupt
#> 5       3442 Project_7 0086cabebd15 1772661600   2026-03-05    Verify
#> 6       3444 Project_7 0086cabebd15 1772661600   2026-03-05   Specify
#>         cluster code_order order_in_session
#> 1     Directive          1                1
#> 2     Directive          2                2
#> 3     Directive          3                3
#> 4 Metacognitive          1                4
#> 5    Evaluative          1                7
#> 6     Directive          1               10
```

The dataset is in long format: `code` records what happened, `session_id` who did it, and `timestamp` when. Additional columns like `project` and `cluster` are automatically preserved as metadata for downstream covariate analysis.

Building Networks

Building networks in Nestimate is a single step: `build_network()` is the universal entry point for all network estimation. It accepts long-format event data directly with three key parameters:

- **`action`**: the column containing state labels
- **`actor`**: the column identifying sequences (one sequence per actor)
- **`time`**: the column providing temporal ordering

Under the hood, `build_network()` automatically converts the long-format event log into wide-format sequences, handling chronological ordering, session detection, and metadata preservation. You can also call `prepare()` directly to inspect or reuse the processed data before passing it to `build_network()`.

Transition Network (TNA)

The standard TNA method estimates a first-order Markov model from sequence data. Given a sequence of events, the transition probability $P(v_j|v_i)$ is estimated as the ratio of observed transitions from state v_i to state v_j to the total number of outgoing transitions from v_i. These probabilities are assembled into a **transition matrix** T, where each element T_{ij} represents the estimated probability of transitioning from v_i to v_j (Saqr et al., 2025a).

```
net_tna <- build_network(human_long, method = "tna",
                         action = "code", actor = "session_id",
                         time = "timestamp")
print(net_tna)
#> Transition Network (relative probabilities) [directed]
#>   Weights: [0.018, 0.620]  |  mean: 0.111
#>
#>   Strongest edges (10 of 81):
#>    from      to      weight
#>    Request   Specify 0.620
#>    Refine    Specify 0.457
#>    Command   Specify 0.280
#>    Correct   Specify 0.278
#>    Specify   Command 0.263
#>    Interrupt Command 0.259
```

```
#>    Command   Command 0.235
#>    Verify    Command 0.224
#>    Inquire   Command 0.196
#>    Inquire   Inquire 0.188
#>
#>  extract_edges(include_self = TRUE) lists every edge; extract_transition_matrix() gives the weight matrix;
#>   extract_initial_probs() the initial probabilities.
```

Frequency Network (FTNA)

The frequency method preserves raw transition counts rather than normalizing to conditional probabilities (Saqr et al., 2025b). This is useful when absolute frequencies matter — a transition occurring 500 times from a common state may be more practically important than one occurring 5 times from a rare state, even if the latter has a higher conditional probability.

```
net_ftna <- build_network(human_long, method = "ftna",
                          action = "code", actor = "session_id",
                          time = "timestamp")
print(net_ftna)
#> Transition Network (frequency counts) [directed]
#>   Weights: [14.000, 732.000]  |  mean: 126.790
#>
#>   Strongest edges (10 of 81):
#>    from    to        weight
#>    Specify Command   732
#>    Request Specify   629
#>    Command Specify   548
#>    Specify Specify   479
#>    Specify Interrupt 463
#>    Command Command   459
#>    Refine  Specify   357
#>    Command Request   303
#>    Specify Frustrate 226
#>    Correct Specify   218
#>
#>  extract_edges(include_self = TRUE) lists every edge; extract_transition_matrix() gives the weight matrix;
#>   extract_initial_probs() the initial probabilities.
```

Attention Network (ATNA)

The attention method applies temporal decay weighting, giving more importance to recent transitions within each sequence. The decay rate `lambda` (default `1`, passed through `params = list(lambda = ...)`) controls how fast the weight falls off: higher values produce faster decay. This captures the idea that later events in a process may be more indicative of the underlying dynamics than early ones.

```
net_atna <- build_network(human_long, method = "atna",
                          action = "code", actor = "session_id",
                          time = "timestamp")
print(net_atna)
#> Attention Network (decay-weighted transitions) [directed]
#>   Weights: [11.018, 357.662]  |  mean: 71.690
```

```
#>
#>   Strongest edges (10 of 81):
#>    from    to        weight
#>    Specify Command   357.662
#>    Specify Specify   335.302
#>    Command Specify   297.131
#>    Request Specify   272.476
#>    Command Command   254.753
#>    Specify Interrupt 206.063
#>    Refine  Specify   167.226
#>    Command Request   145.985
#>    Specify Request   141.240
#>    Specify Frustrate 134.432
#>
#>  extract_edges(include_self = TRUE) lists every edge; extract_transition_matrix() gives the weight matrix;
#>   extract_initial_probs() the initial probabilities.
```

Co-occurrence Network from Binary Data

When the data is binary (0/1) — as is common in learning analytics where activities are coded as present or absent within time windows — `format = "onehot"` declares it, because columns of 0 and 1 could also be positions. The resulting undirected network captures which events tend to co-occur.

```
data(learning_activities)
net <- build_network(learning_activities, method = "cna", actor = "student",
                     format = "onehot")
print(net)
#> Co-occurrence Network [undirected]
#>   Weights: [2681.000, 6262.000]  |  mean: 3867.381
#>
#>   Strongest edges (10 of 21):
#>    from    to      weight
#>    Video   Video   6262
#>    Review  Review  6186
#>    Reading Reading 6100
#>    Coding  Coding  5886
#>    Forum   Forum   5692
#>    Quiz    Quiz    5379
#>    Video   Review  3290
#>    Video   Coding  3278
#>    Reading Forum   3211
#>    Coding  Review  3183
#>
#>  extract_edges(include_self = TRUE) lists every edge; extract_transition_matrix() gives the weight matrix.
```

Window-based TNA (WTNA)

The `wtna()` function computes networks from one-hot encoded (binary) data using temporal windowing. It supports three modes:

- **`"transition"`**: directed transitions between consecutive windows

- **"cooccurrence"**: undirected co-occurrence within windows
- **"both"**: a mixed network combining transitions and co-occurrences

```
net_wtna <- wtna(learning_activities, actor = "student",
                 method = "transition", type = "frequency")
print(net_wtna)
#> Network (method: wtna_transition) [directed]
#>   Weights: [2493.000, 4314.000]  |  mean: 3025.056
#>
#>   Strongest edges (10 of 36):
#>    from    to      weight
#>    Video   Video   4314
#>    Reading Reading 4297
#>    Review  Review  4206
#>    Coding  Coding  4005
#>    Forum   Forum   3881
#>    Quiz    Quiz    3695
#>    Review  Video   3113
#>    Coding  Video   3094
#>    Video   Review  3018
#>    Reading Review  3008
#>
#>  extract_edges(include_self = TRUE) lists every edge; extract_transition_matrix() gives the weight matrix;
#>   extract_initial_probs() the initial probabilities.
```

```
net_wtna_rel <- wtna(learning_activities, method = "transition", type = "relative")
print(net_wtna_rel)
#> Network (method: wtna_transition) [directed]
#>   Weights: [0.136, 0.225]  |  mean: 0.167
#>
#>   Strongest edges (10 of 36):
#>    from    to      weight
#>    Video   Video   0.225
#>    Reading Reading 0.225
#>    Review  Review  0.224
#>    Coding  Coding  0.220
#>    Quiz    Quiz    0.213
#>    Forum   Forum   0.210
#>    Coding  Video   0.169
#>    Review  Video   0.167
#>    Coding  Review  0.162
#>    Forum   Review  0.162
#>
#>  extract_edges(include_self = TRUE) lists every edge; extract_transition_matrix() gives the weight matrix;
#>   extract_initial_probs() the initial probabilities.
```

Mixed Network (Transitions + Co-occurrences)

Since states can co-occur within the same window *and* follow each other across windows, a mixed network captures both relationships simultaneously — modeling co-activity and temporal succession in a single structure.

```r
net_wtna_mixed <- wtna(learning_activities, method = "both", type = "relative")
print(net_wtna_mixed)
#> Mixed Window TNA (transition + co-occurrence)
#> -- Transition (directed) --
#>   Nodes: 6  |  Edges: 36
#> -- Co-occurrence (row-normalised, directed) --
#>   Nodes: 6  |  Edges: 36
```

Validation

Most research on networks or process mining uses descriptive methods. The validation or the statistical significance of such models is almost absent in the literature. Having validated models allows us to assess the robustness and reproducibility of our models to ensure that the insights we get are not merely a product of chance and are therefore generalizable.

Reliability

Split-half reliability assesses whether the network structure is stable when the data is randomly divided into two halves. High reliability means the network structure is a consistent property of the data, not driven by a small number of idiosyncratic sequences.

```r
network_reliability(net_tna)
#> Split-Half Reliability (1000 iterations, split = 50%)
#>   Mean Abs. Diff.     mean = 0.0169  sd = 0.0017
#>   Median Abs. Diff.   mean = 0.0129  sd = 0.0018
#>   Pearson             mean = 0.9707  sd = 0.0063
#>   Max Abs. Diff.      mean = 0.0741  sd = 0.0176
```

Bootstrap Analysis

Bootstrapping repeatedly draws samples from the original dataset with replacement to estimate the model for each sample (Efron & Tibshirani, 1994). When edges consistently appear across the majority of the estimated models, they are considered stable and significant. The bootstrap also provides confidence intervals and p-values for each edge weight, offering a quantifiable measure of uncertainty and robustness for each transition in the network.

```r
set.seed(42)
boot <- bootstrap_network(net_tna, iter = 100)
boot
#>   Edge                  Mean     95% CI          p
#>   ------------------------------------------------
#>   Request → Specify     0.619  [0.584, 0.655]  **
#>   Refine → Specify      0.455  [0.413, 0.502]  **
#>   Command → Specify     0.281  [0.256, 0.301]  **
#>   Correct → Specify     0.280  [0.249, 0.309]  **
#>   Specify → Command     0.263  [0.246, 0.279]  **
#>   ... and 40 more significant edges
#>
#> Bootstrap Network  [Transition Network (relative) | directed]
#>   Iterations : 100  |  Nodes : 9
```

```
#>   Edges      : 45 significant / 81 total
#>   CI         : 95%  |  Inference: stability  |  CR [0.75, 1.25]
```

Centrality Stability

Centrality measures quantify the role or importance of each state in the process (Freeman, 1978; Opsahl et al., 2010). Centrality stability analysis quantifies how robust centrality rankings are to case-dropping: the CS-coefficient is the maximum proportion of cases that can be dropped while maintaining a correlation of at least 0.7 with the original centrality values in at least 95% of the resamples. A CS-coefficient above 0.5 indicates stable rankings; below 0.25 indicates instability (Epskamp et al., 2018).

```
centrality_stability(net_tna, iter = 100)
#> Centrality Stability (100 iterations, threshold = 0.7)
#>   Drop proportions: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
#>
#>   CS-coefficients:
#>     InStrength       0.90
#>     OutStrength      0.80
#>     Betweenness      0.90
```

Clustering

Clusters represent typical transition networks that recur across different instances. Unlike communities, clusters involve the entire network where groups of sequences are similarly interconnected and each exhibits a distinct transition pattern with its own set of transition probabilities. Identifying clusters captures the dynamics, revealing typical behavioral strategies that learners frequently adopt. `build_clusters()` computes pairwise dissimilarities between sequences and partitions them into `k` groups; passing the clustering to `build_network()` then builds a separate network for each cluster (López-Pernas et al., 2025).

```
Cls <- build_clusters(net_tna, k = 3)
Clusters <- build_network(Cls, method = "tna")
Clusters
#> Group Networks (3 clusters via pam / hamming)
#>
#>   Group      Nodes  Edges  Weights          N
#>   Cluster 1  9      81     [0.006, 0.538]  145 (27.6%)
#>   Cluster 2  9      81     [0.003, 0.694]  299 (56.8%)
#>   Cluster 3  9      81     [0.018, 0.588]  82 (15.6%)
```

Permutation Test for Clusters

Permutation testing is particularly important for data-driven clusters: because clustering algorithms partition sequences to maximize between-group separation, some degree of apparent difference is guaranteed by construction. The permutation test provides the necessary corrective — by randomly reassigning sequences to groups while preserving internal sequential structure, it constructs null distributions for edge-level differences. Only differences that exceed this null distribution constitute evidence of genuine structural divergence rather than algorithmic artifacts. Given the grouped network, `permutation()` tests every pair of clusters in one call.

```
perm <- permutation(Clusters, iter = 100)
perm
#> Grouped Permutation Test
#> Groups: Cluster 1 vs Cluster 2, Cluster 1 vs Cluster 3, Cluster 2 vs Cluster 3
#>
#> -- Cluster 1 vs Cluster 2 --
#> Permutation Test: Transition Network (relative probabilities) [directed]
#>   Iterations: 100  |  Alpha: 0.05
#>   Nodes: 9  |  Edges tested: 81  |  Significant: 9
#>   Global test (networks differ overall?): M = 2.587 (p = 0.0099)  |  S = 0.244 (p = 0.0099)
#>
#> -- Cluster 1 vs Cluster 3 --
#> Permutation Test: Transition Network (relative probabilities) [directed]
#>   Iterations: 100  |  Alpha: 0.05
#>   Nodes: 9  |  Edges tested: 81  |  Significant: 13
#>   Global test (networks differ overall?): M = 2.336 (p = 0.0099)  |  S = 0.116 (p = 0.198)
#>
#> -- Cluster 2 vs Cluster 3 --
#> Permutation Test: Transition Network (relative probabilities) [directed]
#>   Iterations: 100  |  Alpha: 0.05
#>   Nodes: 9  |  Edges tested: 81  |  Significant: 15
#>   Global test (networks differ overall?): M = 3.012 (p = 0.0099)  |  S = 0.181 (p = 0.0594)
```

All three pairs of clusters differ overall: the global statistic M (the sum of the absolute edge differences) has $p = 0.0099$, the smallest value 100 permutations can give, and 9 to 15 of the 81 transitions differ per pair. With `iter = 100` the p-values are coarse; a reported analysis would use the default 1,000.

Post-hoc Covariate Analysis

`build_clusters()` supports post-hoc covariate analysis: covariates do not influence the clustering but are analyzed after the fact to characterize who ends up in which cluster. This is the appropriate approach when the clustering should reflect behavioral patterns alone, and the researcher then asks whether those patterns are associated with external variables.

The `group_regulation_long` dataset contains self-regulated learning sequences annotated with an `Achiever` covariate distinguishing high and low achievers. It was synthetically generated from the `group_regulation` dataset of the tna package (Tikka et al., 2025).

```
data("group_regulation_long")
net_GR <- build_network(group_regulation_long, method = "tna",
                        action = "Action", actor = "Actor",
                        time = "Time")
```

```
# Default estimator = "auto" inspects the cluster x covariate cross-tab
# and picks firth (brglm2, bias-reduced) only when rare cells signal
# separation risk; otherwise it uses the much faster nnet::multinom.
# With Achiever balanced 50/50, multinom is selected.
Post <- build_clusters(net_GR, k = 2, covariates = c("Achiever"))
summary(Post)
#> Sequence Clustering Summary
#>   Method:        pam
#>   Dissimilarity: hamming
#>   Silhouette:    0.1839
```

```
#>
#> Per-cluster statistics:
#>  cluster size mean_within_dist
#>        1  982         10.69340
#>        2 1018         18.59498
#>
#> Post-hoc Covariate Analysis (does not influence cluster membership)
#>
#> Cluster Profiles (categorical):
#>  Cluster N    Achiever=High N(%) Achiever=Low N(%)
#>  1        982 504 (51%)          478 (49%)
#>  2       1018 496 (49%)          522 (51%)
#>
#> Predictors of Membership (estimator = multinom, reference: Cluster 1):
#>  Cluster Variable    OR   95% CI       p     Sig
#>  2       AchieverLow 1.11 [0.93, 1.32] 0.245
#>
#> Model: AIC = 2774.6 | BIC = 2785.8 | McFadden R-squared = 0.00
#>
#> Note: Covariates are post-hoc and do not influence cluster assignments.
```

```
Postgr <- build_network(Post)
Postgr
#> Group Networks (2 clusters via pam / hamming)
#>
#>   Group      Nodes  Edges  Weights         N
#>   Cluster 1  9      78     [0.001, 0.506]  982 (49.1%)
#>   Cluster 2  9      78     [0.001, 0.494]  1018 (50.9%)
```

Part II: Psychological Network Analysis

Theoretical Grounding

Psychological network analysis estimates the conditional dependency structure among a set of variables (Borsboom et al., 2021; Epskamp & Fried, 2018). Variables (e.g., symptoms, traits, behaviors) are represented as nodes, and edges represent partial correlations — the association between two variables after controlling for all others. This approach reveals which variables are directly connected versus those whose association is mediated through other variables (Saqr et al., 2024).

`Nestimate` supports three estimation methods for continuous psychological data, all accessed through the same `build_network()` interface (binary data use `method = "ising"`, mixed continuous and categorical data `method = "mgm"`):

- **Correlation networks** (`method = "cor"`) estimate pairwise Pearson correlations and produce fully connected undirected networks. While informative as a starting point, they do not distinguish direct from indirect associations.
- **Partial correlation networks** (`method = "pcor"`) control for all other variables, revealing only direct associations. They provide a more accurate picture of the dependency structure, though results can be noisy in small samples.
- **Regularized networks via EBICglasso** (`method = "glasso"`) apply L1 regularization to the precision matrix, shrinking weak or unreliable edges to exactly zero. This is the recommended approach

for psychological network analysis, as it balances model fit against complexity and produces sparse, interpretable, and replicable structures.

Data

The `chatgpt_srl` dataset contains scale scores on five self-regulated learning (SRL) constructs — Comprehension and Study Understanding (CSU), Intrinsic Value (IV), Self-Efficacy (SE), Self-Regulation (SR), and Task Avoidance (TA) — for 1,000 responses generated by ChatGPT to a validated SRL questionnaire (Vogelsmeier et al., 2025).

```
data(chatgpt_srl)
head(chatgpt_srl)
#>        CSU       IV       SE       SR   TA
#> 1 4.538462 5.222222 4.000000 3.222222 5.00
#> 2 4.153846 4.333333 4.888889 4.333333 4.50
#> 3 4.846154 5.111111 3.777778 4.666667 4.25
#> 4 4.307692 4.222222 4.777778 3.777778 4.00
#> 5 4.307692 4.777778 4.333333 4.111111 5.75
#> 6 4.692308 3.777778 5.222222 3.888889 5.00
```

Regularized Network (EBICglasso)

The graphical lasso applies L1 regularization to the precision matrix (the inverse of the covariance matrix), producing a sparse network where weak or unreliable edges are shrunk to exactly zero (Friedman et al., 2008). The `gamma` parameter controls sparsity through EBIC model selection (Foygel & Drton, 2010) — higher values yield sparser networks.

```
net_glasso <- build_network(chatgpt_srl, method = "glasso",
                            params = list(gamma = 0.5))
net_glasso
#> Partial Correlation Network (EBICglasso) [undirected]
#>   Sample size: 1000
#>
#>   No edges: every weight is zero.
#>
#>  extract_edges(include_self = TRUE) lists every edge; extract_transition_matrix() gives the weight matrix.
#>
#>   Predictability (R²):
#>   CSU           0.000
#>   IV            0.000
#>   SE            0.000
#>   SR            0.000
#>   TA            0.000
#>
#>   Gamma: 0.50  |  Lambda: 0.0517
```

How to cite this article

Saqr, M., López-Pernas, S., & Misiejuk, K. (2026). *Network estimation and analysis with Nestimate*. Nestimate. https://doi.org/10.5281/zenodo.24528510 · HTML · PDF

References

Borsboom, D., Deserno, M. K., Rhemtulla, M., Epskamp, S., Fried, E. I., McNally, R. J., Robinaugh, D. J., Perugini, M., Dalege, J., Costantini, G., Isvoranu, A.-M., Wysocki, A. C., van Borkulo, C. D., van Bork, R., & Waldorp, L. J. (2021). Network analysis of multivariate data in psychological science. *Nature Reviews Methods Primers*, 1, 58. https://doi.org/10.1038/s43586-021-00055-w

Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap*. Chapman and Hall/CRC. https://doi.org/10.1201/9780429246593

Epskamp, S., & Fried, E. I. (2018). A tutorial on regularized partial correlation networks. *Psychological Methods*, 23(4), 617–634. https://doi.org/10.1037/met0000167

Epskamp, S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy: A tutorial paper. *Behavior Research Methods*, 50(1), 195–212. https://doi.org/10.3758/s13428-017-0862-1

Foygel, R., & Drton, M. (2010). Extended Bayesian information criteria for Gaussian graphical models. *Advances in Neural Information Processing Systems*, 23, 604–612.

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social Networks*, 1(3), 215–239. https://doi.org/10.1016/0378-8733(78)90021-7

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), 432–441. https://doi.org/10.1093/biostatistics/kxm045

Holme, P. (2015). Modern temporal network theory: A colloquium. *The European Physical Journal B*, 88(9), 234. https://doi.org/10.1140/epjb/e2015-60657-4

Holme, P., & Saramäki, J. (2012). Temporal networks. *Physics Reports*, 519(3), 97–125. https://doi.org/10.1016/j.physrep.2012.03.001

López-Pernas, S., Tikka, S., & Saqr, M. (2025). Mining patterns and clusters with transition network analysis: A heterogeneity approach. In M. Saqr & S. López-Pernas (Eds.), *Advanced learning analytics methods: AI, precision and complexity* (pp. 447–468). Springer. https://doi.org/10.1007/978-3-031-95365-1_17

Misiejuk, K., López-Pernas, S., Oliveira, E., Rinja, D., & Saqr, M. (2026). It takes two to vibe code: Profiling student-AI programming interactions processes and their high order structure. In *Proceedings of the 26th Koli Calling International Conference on Computing Education Research* (pp. 1–11). ACM. https://doi.org/10.1145/3856208.3856254

Newman, M. (2018). *Networks* (2nd ed.). Oxford University Press. https://doi.org/10.1093/oso/9780198805090.001.0001

Opsahl, T., Agneessens, F., & Skvoretz, J. (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3), 245–251. https://doi.org/10.1016/j.socnet.2010.03.006

Peeters, W., Saqr, M., & Viberg, O. (2020). Applying learning analytics to map students' self-regulated learning tactics in an academic writing course. In *Proceedings of the 28th International Conference on Computers in Education (ICCE 2020)* (pp. 245–254).

Saqr, M., Beck, E., & López-Pernas, S. (2024). Psychological networks: A modern approach to analysis of learning and complex learning processes. In M. Saqr & S. López-Pernas (Eds.), *Learning analytics methods and tutorials: A practical guide using R* (pp. 639–671). Springer. https://doi.org/10.1007/978-3-031-54464-4_19

Saqr, M., López-Pernas, S., & Misiejuk, K. (2026). Nestimate: An R package for dynamic, probabilistic, and high order network analysis. *Applied Psychological Measurement*. https://doi.org/10.1177/01466216261472019

Saqr, M., López-Pernas, S., & Tikka, S. (2025a). Mapping relational dynamics with transition network analysis: A primer and tutorial. In M. Saqr & S. López-Pernas (Eds.), *Advanced learning analytics methods: AI, precision and complexity* (pp. 371–411). Springer. https://doi.org/10.1007/978-3-031-95365-1_15

Saqr, M., López-Pernas, S., & Tikka, S. (2025b). Capturing the breadth and dynamics of the temporal processes with frequency transition network analysis: A primer and tutorial. In M. Saqr & S. López-Pernas (Eds.), *Advanced learning analytics methods: AI, precision and complexity* (pp. 413–446). Springer. https://doi.org/10.1007/978-3-031-95365-1_16

Saqr, M., López-Pernas, S., Törmänen, T., Kaliisa, R., Misiejuk, K., & Tikka, S. (2025c). Transition network analysis: A novel framework for modeling, visualizing, and identifying the temporal patterns of learners and learning processes. In *Proceedings of the 15th International Learning Analytics and Knowledge Conference (LAK25)* (pp. 351–361). ACM. https://doi.org/10.1145/3706468.3706513

Saqr, M., Misiejuk, K., & López-Pernas, S. (2026). Human-AI collaboration or obedient and often clueless AI in instruct, serve, repeat dynamics? *The Internet and Higher Education*, 70, 101087. https://doi.org/10.1016/j.iheduc.2026.101087

Saqr, M., Nouri, J., & Fors, U. (2019). Time to focus on the temporal dimension of learning: A learning analytics study of the temporal patterns of students' interactions and self-regulation. *International Journal of Technology Enhanced Learning*, 11(4), 398–412. https://doi.org/10.1504/IJTEL.2019.102549

Sarkar, A., & Drosos, I. (2025). Vibe coding: Programming through conversation with artificial intelligence. In *Proceedings of the 36th Annual Conference of the Psychology of Programming Interest Group (PPIG 2025)*. https://doi.org/10.48550/arXiv.2506.23253

Tikka, S., López-Pernas, S., & Saqr, M. (2025). tna: An R package for transition network analysis. *Applied Psychological Measurement*, 49(6), 326–328. https://doi.org/10.1177/01466216251348840

Vogelsmeier, L. V. D. E., Oliveira, E., Misiejuk, K., López-Pernas, S., & Saqr, M. (2025). Delving into the psychology of machines: Exploring the structure of self-regulated learning via LLM-generated survey responses. *Computers in Human Behavior*, 173, 108769. https://doi.org/10.1016/j.chb.2025.108769

Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications*. Cambridge University Press. https://doi.org/10.1017/CBO9780511815478