Network Data and Graph Workflows

Jared Edgerton

Why Networks Matter

Many social processes are fundamentally relational:

- Conflict and cooperation
- Information diffusion
- Influence and dependency
- Collective action

Networks make relations explicit.

What Is Network Data

Network data consist of:

- Nodes (actors, units)
- Edges (relations, interactions)
- Attributes on both

The unit of analysis is often the *tie*, not the node.

From Tables to Graphs

Network construction requires choices:

- What counts as a node?
- What counts as a tie?
- Directional or undirected?
- Weighted or binary?

Representation decisions shape inference.

Graph Construction (Conceptual)

```python
import networkx as nx

G = nx.from_pandas_edgelist(
    df,
    source='i',
    target='j',
    create_using=nx.DiGraph()
)
```

```r
library(igraph)

g <- graph_from_data_frame(
  df,
  directed = TRUE
)
```

Centrality Measures

Centrality captures different notions of importance:

- Degree
- Betweenness
- Eigenvector / PageRank

No single measure is universally correct.

Community Detection

Communities identify:

- Clusters of dense ties
- Functional groupings
- Latent structure

Algorithms impose assumptions about structure.

Statistical Network Models

Statistical models treat networks as outcomes:

- Ties are random variables
- Structure is explained, not assumed
- Dependence is explicit

ERGMs (Local Dependence)

```
# PSEUDOCODE: Exponential Random Graph Model (ERGM)

initialize theta
repeat until convergence:
    propose Y_prime
    delta_g = g(Y_prime) - g(Y_observed)
    accept Y_prime with prob min(1, exp(theta · delta_g))
    update theta

return theta
```

ERGMs explain why certain *local configurations* appear.

AME Models (Latent Structure)

```
# PSEUDOCODE: Additive-Multiplicative Effects (AME)

for each dyad (i, j):
    y_ij = X_ij beta + a_i + b_j + u_i' v_j + error

initialize beta, a, b, u, v
repeat until convergence:
    update beta
    update a_i and b_j
    update latent positions u_i, v_j

return beta, latent_positions
```

AMEs capture unobserved dependence with low-rank geometry.

Diffusion Models

```
# PSEUDOCODE: Network Diffusion / Event History

for time t:
    for actor i:
        exposure_i = sum_j w_ij * events_j(t-1)
        hazard_i = f(exposure_i, covariates_i, baseline_t)
        if random_draw < hazard_i:
            events_i(t) = 1
```

Diffusion models make time and exposure explicit.

Graph Neural Networks (GNNs)

```
# PSEUDOCODE: Graph Neural Network (Message Passing)

initialize h_i^0 = X_i

for layer l in 1:L:
    for node i:
        messages = aggregate(h_j^(l-1) for j in neighbors(i))
        h_i^l = update(h_i^(l-1), messages)

use h_i^L for prediction
```

GNNs learn representations optimized for prediction.

Model Comparison

ERGMs:
  Explain local structure
  Interpretable parameters
  Scale poorly

AMEs:
  Explain latent dependence
  Scalable, interpretable geometry

Diffusion models:
  Explain temporal contagion
  Causal intuition

GNNs:
  Optimize prediction
  Limited interpretability

Validation in Network Analysis

Validation challenges include:

- Dependence across observations
- Overfitting structure
- Temporal instability

Out-of-sample checks are essential.

What We Emphasize in Practice

- Start with clear representations
- Match models to questions
- Be explicit about dependence
- Validate aggressively

Discussion

- Which network choices feel most consequential?
- When do black-box models make sense?
- How should uncertainty be communicated?