Visualizing relationships

(Advanced Data Visualizations with Stata: Part VIII)

Asjad Naqvi

Austrian Institute of Economic Research (WIFO)

Stata Germany Conference – Munich
19 June 2026

- We often work with relational data: trade networks, aid flows, FDI links, and other network-structured data.
- Existing relationship-mapping packages such as arcplot and sankey are useful, but large bi-directional flows can quickly become messy and hard to interpret.
- Older packages exist but are no longer actively maintained.
- Today we introduce ntwrk, a new Stata package for comprehensive network analysis, covering both measures and visualizations for directed and weighted networks.

https://github.com/asjadnaqvi/stata-ntwrk

Stata package: ntwrk

- Why ntwrk? The name network is reserved by StataCorp.
- Previous attempts, especially nwcommands (Thomas Grund), were ahead of their time, but key routines have since been deprecated (last updated 11 years ago).
- Several user packages focus on either analysis or visualization, but many are no longer updated or have limited options.
- This package has been about three years in the making: I almost presented it in 2023, then abandoned and restarted it several times.
- It started from porting over path-finding routines in nwcommands, especially Breadth-First Search (BFS), and has now expanded to Dijkstra for weighted networks (A* implementation coming soon).
- The package is currently being stress-tested to compare outputs of network measures and layouts with established libraries in R and Python.

The syntax is broken down into several blocks:

```
ntwrk value [if] [in], from(varname) to(varname)

[network measures] [parameters] [network layout] [node options]
[link options] [arc options] [save options]
```

The users can specify a list of measures using `measure(<measure list>)`

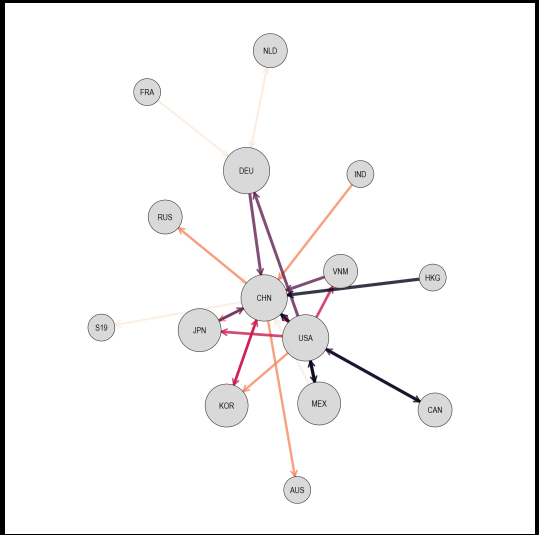
Measure	Description
`degree`	Total (undirected) degree: number of distinct neighbors regardless of direction.
`indegree`	Number of edges pointing **into** a node (in-degree). In undirected graphs equals `degree`.
`outdegree`	Number of edges pointing **out of** a node (out-degree). In undirected graphs equals `degree`.
`between`	**Betweenness centrality**: the fraction of all shortest paths in the network that pass through a node.
`closeness`	**Closeness centrality**: the reciprocal of the average shortest-path distance from a node to all reachable nodes.
`harmonic`	**Harmonic centrality**: sum of reciprocal distances to all other reachable nodes.
`clustering`	**Local clustering coefficient**: the fraction of a node's pairs of neighbours that are themselves connected.
`transitivity`	**Global transitivity (network-level)**: ratio of closed triangles to all connected triples in the graph.
`eccentricity`	**Eccentricity**: the maximum shortest-path distance from a node to any other reachable node.
`eigenval`	**Eigenvalue centrality score**: the dominant eigenvalue of the adjacency matrix, reported as a constant for the graph.
`eigenvec`	**Eigenvector centrality**: each node's component in the principal eigenvector of the adjacency matrix.
`katz`	**Katz centrality**: a damped version of eigenvector centrality that also credits nodes for *all* paths.
`pagerank`	**PageRank**: the stationary distribution of a random walker who follows edges with probability $(1 - d)$.
`hits`	**HITS (Hyperlink-Induced Topic Search)**: computes two scores per node: *hub* and *authority*.

The users can specify a layout using `layout(<option>)`

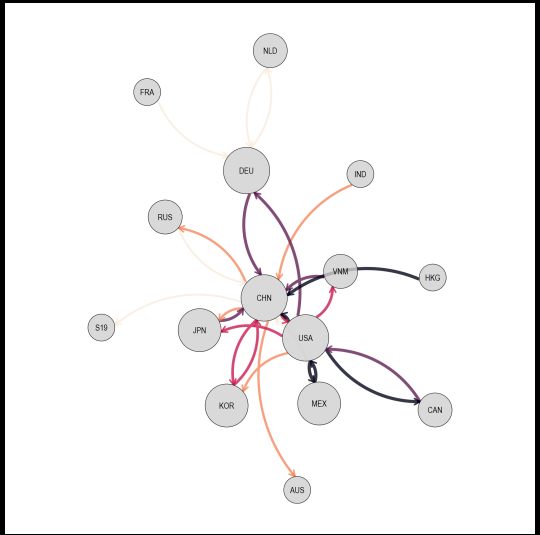
Layout	Description
`fr`	**Fruchterman–Reingold** (default): a force-directed algorithm that treats edges as springs and nodes as repelling charges.
`kk`	**Kamada–Kawai**: a stress-minimization layout that positions nodes so that their graph-theoretic distances match their Euclidean distances as closely as possible.
`spectral`	**Spectral layout**: uses the eigenvectors of the graph Laplacian to embed nodes into two dimensions.
`star`	Places one node at the center and distributes all remaining nodes evenly around a single circle.
`sphere`	Distributes all nodes uniformly on a circle (spherical shell in 2D).
`grid`	Arranges nodes on a regular rectangular grid.
`random`	Places nodes at uniformly random coordinates within the canvas.
`bipartite`	Two-column layout that places nodes on two vertical tracks based on their role (source vs. destination in the edge list).

Example: `layout(fr)` – Fruchterman–Reingold

Links

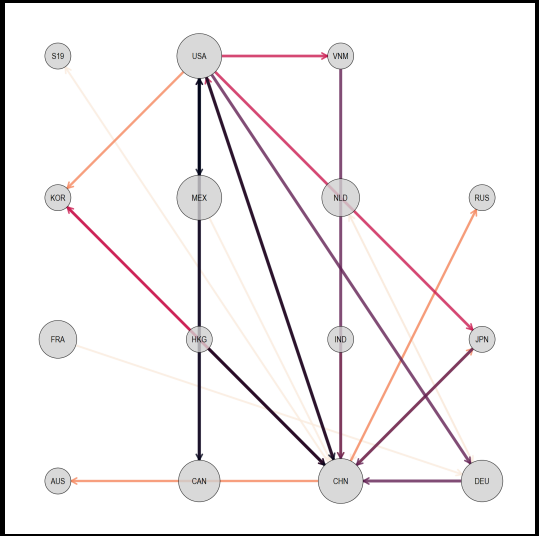


Arcs

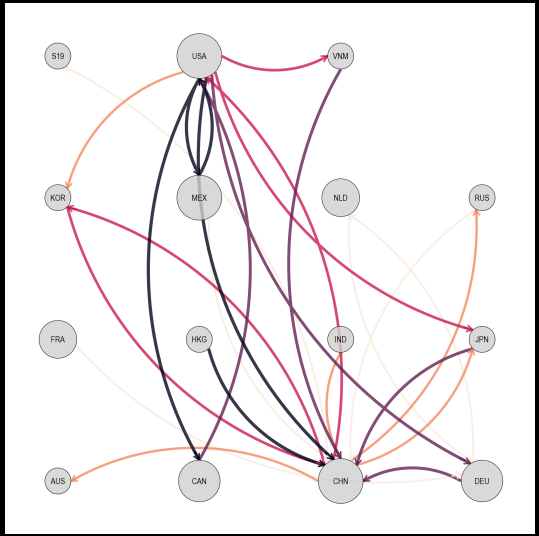


Example: `layout(grid)` – Grid

Links

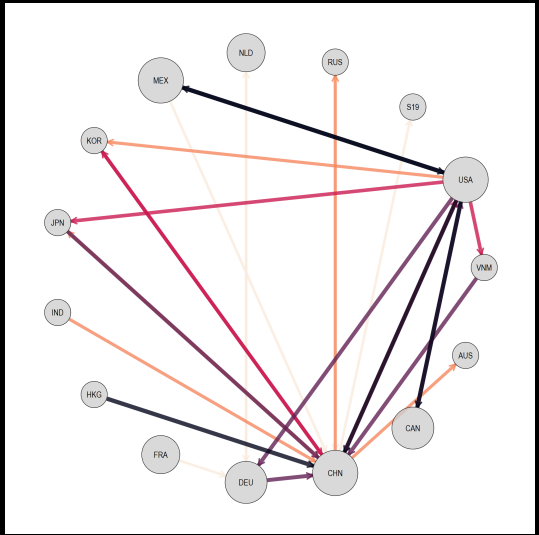


Arcs

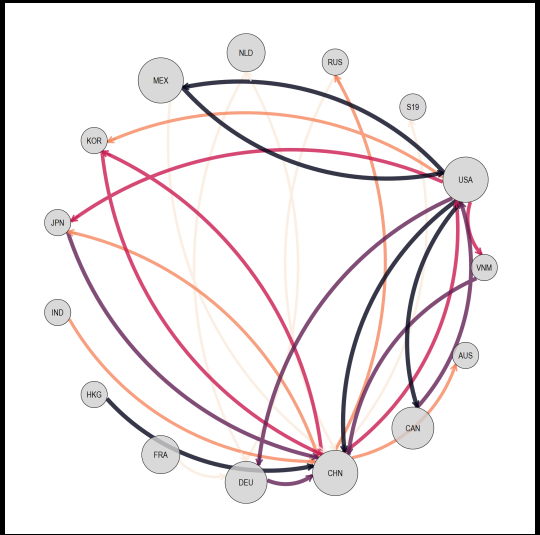


Example: `layout(star)` – Star

Links



Arcs



Other updates from this year

ntwrk is built on other background packages that have had major releases this year:

- graphfunctions core routines including generating arcs and circles in now integrated in this package.
- graphfunctions added function 'labrepel' for repelling labels on graphs and it is similar to the FR-algorithm.
- vcontrol launched that allows version control for Stata packages between SSC, and remote repositories. Automatically checks and updates from the latest possible source.
- Many more features will be announced over the summers.

Thank you!

More Stataviz:

- StataViz portfolio
- The Stata Guide on Medium
- The Stata Gallery on Medium

Connect with me:

- asjadnaqvi@gmail.com
- github.com/asjadnaqvi
- @AsjadNaqvi
- AsjadNaqvi