

# Package: tsn (via r-universe)

October 3, 2026

**Title** Time-Series Network Construction

**Version** 1.4.1

**Description** Constructs distance, visibility, and state-transition networks from univariate time series. The core constructor provides whole-series and sliding-window distance networks, natural and horizontal visibility graphs following Lacasa et al. (2008) <[doi:10.1073/pnas.0709247105](https://doi.org/10.1073/pnas.0709247105)> and Luque et al. (2009) <[doi:10.1103/PhysRevE.80.046103](https://doi.org/10.1103/PhysRevE.80.046103)>, and optional state aggregation. Companion verbs expose discretization, trend classification, and an optional bridge to the 'Nestimate' transition-network models. Core network construction uses only packages distributed with R.

**License** MIT + file LICENSE

**URL** <https://github.com/mohsaqr/tsn/>, <https://pak.dynasite.org/tsn/>

**BugReports** <https://github.com/mohsaqr/tsn/issues/>

**Depends** R (>= 4.1.0)

**Imports** graphics, grDevices, stats, utils

**Suggests** cograph, igraph, knitr, Nestimate, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr, rmarkdown

**LazyData** true

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Repository** <https://mohsaqr.r-universe.dev>

**Date/Publication** 2026-09-03 07:51:47 UTC

**RemoteUrl** <https://github.com/mohsaqr/tsn>

**RemoteRef** HEAD

**RemoteSha** 7e3b1f2966de3293494cd8b4d08e6f7891689dff

## Contents

|                                       |    |
|---------------------------------------|----|
| as.data.frame.ts_tna . . . . .        | 2  |
| as.data.frame.ts_tna_group . . . . .  | 3  |
| as.data.frame.tsn . . . . .           | 4  |
| as.matrix.ts_tna . . . . .            | 5  |
| as.matrix.tsn . . . . .               | 6  |
| discretize . . . . .                  | 6  |
| esm_srl . . . . .                     | 8  |
| plot.ts_tna . . . . .                 | 9  |
| plot.ts_tna_group . . . . .           | 11 |
| plot.tsn . . . . .                    | 12 |
| plot.tsn_series_networks . . . . .    | 13 |
| plot.tsn_states . . . . .             | 14 |
| plot.tsn_trend . . . . .              | 17 |
| print.tsn . . . . .                   | 18 |
| print.tsn_states . . . . .            | 19 |
| print.tsn_trend . . . . .             | 19 |
| series_networks . . . . .             | 20 |
| srl . . . . .                         | 21 |
| summary.tsn . . . . .                 | 22 |
| summary.tsn_series_networks . . . . . | 22 |
| summary.tsn_states . . . . .          | 23 |
| summary.tsn_trend . . . . .           | 23 |
| trend . . . . .                       | 24 |
| ts_tna . . . . .                      | 25 |
| tsn . . . . .                         | 30 |
| vg . . . . .                          | 34 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>36</b> |
|--------------|-----------|

---

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| as.data.frame.ts_tna | <i>Coerce a transition network to a tidy edge table</i> |
|----------------------|---------------------------------------------------------|

---

## Description

Returns one row per state pair, so a transition network can be read, sorted, or joined as data rather than reached into. `as.data.frame()` on the underlying Nestimate object errors, so this method supplies the tidy view.

## Usage

```
## S3 method for class 'ts_tna'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  what = c("edges", "series"),
```

```
    ...
  )
```

### Arguments

|           |                                                                                                                                                           |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| x         | A ts_tna result.                                                                                                                                          |
| row.names | Optional row names.                                                                                                                                       |
| optional  | Ignored.                                                                                                                                                  |
| what      | Which table to return: "edges" (one row per state pair, the default) or "series" (the tidy per-observation source table with id, time, value, and state). |
| ...       | Ignored.                                                                                                                                                  |

### Value

A base data frame.

### Examples

```
set.seed(1)
network <- ts_tna(cumsum(rnorm(60)), labels = c("low", "mid", "high"))
as.data.frame(network)
head(as.data.frame(network, what = "series"))
```

---

```
as.data.frame.ts_tna_group
```

*Coerce a grouped transition network to a tidy data frame*

---

### Description

A grouped model is a collection of networks, one per group. This method is the tidy view of it: every table it returns carries a group column, so a comparison across groups is a data frame you can read, sort, or join rather than a set of objects you have to reach into one at a time.

### Usage

```
## S3 method for class 'ts_tna_group'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  what = c("edges", "series", "groups"),
  ...
)
```

**Arguments**

|                        |                                                                                                                                                                                                                           |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>         | A <code>ts_tna_group</code> result from <code>ts_tna()</code> and friends, built with their group argument.                                                                                                               |
| <code>row.names</code> | Optional row names.                                                                                                                                                                                                       |
| <code>optional</code>  | Ignored.                                                                                                                                                                                                                  |
| <code>what</code>      | Which table to return: "edges" (one row per group and state pair, the default), "series" (the per-observation source table for every group), or "groups" (a one-row-per-group index of how much data backs each network). |
| <code>...</code>       | Ignored.                                                                                                                                                                                                                  |

**Value**

A base data frame whose first column is group.

**See Also**

`ts_tna()` for the group argument that builds these models.

**Examples**

```
data(esm_srl)
networks <- ts_tna(
  subset(esm_srl, !is.na(effort)),
  value = "effort", id = "name", time = "occasion",
  group = "day_type", labels = c("low", "mid", "high")
)
as.data.frame(networks, what = "groups")
head(as.data.frame(networks))
```

---

|                                |                                            |
|--------------------------------|--------------------------------------------|
| <code>as.data.frame.tsn</code> | <i>Coerce a TSN result to a data frame</i> |
|--------------------------------|--------------------------------------------|

---

**Description**

Coerce a TSN result to a data frame

**Usage**

```
## S3 method for class 'tsn'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  what = c("edges", "series"),
  connected = FALSE,
  ...
)
```

**Arguments**

|           |                                                                                                                                                                                                          |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x         | A tsn result.                                                                                                                                                                                            |
| row.names | Optional row names.                                                                                                                                                                                      |
| optional  | Ignored.                                                                                                                                                                                                 |
| what      | Which table to return: "edges" (the stable dyad schema, the default) or "series" (the tidy source time series, including the discretized state column when the network uses states).                     |
| connected | Return only the retained edges? The edge table lists every evaluated dyad; TRUE keeps the rows whose connected flag is set, i.e. the edges that survived the connect rule. Ignored when what = "series". |
| ...       | Ignored.                                                                                                                                                                                                 |

**Value**

A base data frame: the stable TSN dyad schema when what = "edges", or the source time series when what = "series".

---

|                  |                                                                   |
|------------------|-------------------------------------------------------------------|
| as.matrix.ts_tna | <i>Coerce a transition network to a weighted adjacency matrix</i> |
|------------------|-------------------------------------------------------------------|

---

**Description**

The state-to-state weight matrix behind a `ts_tna()`-family model: transition probabilities for `ts_tna()`, counts for `ts_ftna()`, and so on. Without this method `as.matrix()` falls through to the default, which coerces the object's component list rather than its network.

**Usage**

```
## S3 method for class 'ts_tna'
as.matrix(x, rownames.force = NA, ...)
```

**Arguments**

|                |                  |
|----------------|------------------|
| x              | A ts_tna result. |
| rownames.force | Ignored.         |
| ...            | Ignored.         |

**Value**

A named numeric matrix, states in row and column order.

**Examples**

```
set.seed(1)
network <- ts_tna(cumsum(rnorm(60)), labels = c("low", "mid", "high"))
as.matrix(network)
```

---

|               |                                                           |
|---------------|-----------------------------------------------------------|
| as.matrix.tsn | <i>Coerce a TSN result to a weighted adjacency matrix</i> |
|---------------|-----------------------------------------------------------|

---

### Description

Coerce a TSN result to a weighted adjacency matrix

### Usage

```
## S3 method for class 'tsn'
as.matrix(x, rownames.force = NA, ...)
```

### Arguments

|                |               |
|----------------|---------------|
| x              | A tsn result. |
| rownames.force | Ignored.      |
| ...            | Ignored.      |

### Value

A named numeric matrix.

---

|            |                                             |
|------------|---------------------------------------------|
| discretize | <i>Discretize a Time Series into States</i> |
|------------|---------------------------------------------|

---

### Description

discretize() exposes every internal state discretizer as a first-class verb. It accepts the same inputs as [tsn\(\)](#) (plus a tsn object) and returns a tidy one-row-per-observation table of discretized states. tsn() routes its own state aggregation through the same engine, so tsn(x, "<method>") and discretize(x, method = "<method>") produce identical state assignments. Scalar methods learn one shared state space from all selected values. Temporal methods ("ordinal", "adaptive\_magnitude", and "dtw") first compute patterns or windows within each series, never across an ID boundary, and then use a shared alphabet or clustering model.

### Usage

```
discretize(
  data,
  value = NULL,
  id = NULL,
  time = NULL,
  series = NULL,
  method = "quantile",
  n_states = 3L,
```

```

    breaks = NULL,
    labels = NULL,
    transform = "none",
    m = NULL,
    tau = NULL,
    seed = NULL
)

```

### Arguments

|           |                                                                                                                                                                                                                                |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data      | A numeric vector, ts, matrix, named list of numeric vectors, data frame, or a tsn object (its source series are used).                                                                                                         |
| value     | Optional value-column name for long data.                                                                                                                                                                                      |
| id        | Optional series-ID column name for long data.                                                                                                                                                                                  |
| time      | Optional time-column name for long data.                                                                                                                                                                                       |
| series    | Optional series IDs or wide-data column names to select.                                                                                                                                                                       |
| method    | Discretization method. One of "threshold", "width", "quantile", "kde", "kmeans", "gaussian", "hclust", "ordinal", "symbolic", "change_points", "entropy", "magnitude", "adaptive_magnitude", "percentile_magnitude", or "dtw". |
| n_states  | Number of states. Not consumed by "ordinal", whose state count follows the embedding arguments m and tau; supplying both is an error.                                                                                          |
| breaks    | Optional interior thresholds for method = "threshold". An error with any other method, which computes its own boundaries.                                                                                                      |
| labels    | Optional custom state labels. Length must equal the number of states produced. When NULL, states are numbered consecutively.                                                                                                   |
| transform | Pre-discretization transform: "none" (default), "log" (uses $\log_{10}(\text{abs}(x))$ ), or "zscore" (standardized values).                                                                                                   |
| m         | Embedding dimension for method = "ordinal" (default 3).                                                                                                                                                                        |
| tau       | Embedding lag for method = "ordinal" (default 1).                                                                                                                                                                              |
| seed      | Optional seed for the stochastic discretizers ("kmeans" and "gaussian"). An error with the deterministic methods.                                                                                                              |

### Value

A tidy data frame of class `tsn_states` with columns `id`, `time`, `value`, `state` (a factor), and `probability`. The discretization model and any bin boundaries are stored in the `model` and `breaks` attributes.

### References

Bandt, C., & Pompe, B. (2002). Permutation entropy: A natural complexity measure for time series. *Physical Review Letters*, 88, 174102. doi:[10.1103/PhysRevLett.88.174102](https://doi.org/10.1103/PhysRevLett.88.174102)

### Examples

```
discretize(c(1, 5, 2, 9, 3, 7, 4, 8), method = "quantile", n_states = 3)

discretize(
  c(1, 5, 2, 9, 3, 7, 4, 8),
  method = "quantile",
  n_states = 3,
  labels = c("low", "mid", "high")
)

discretize(c(3, 1, 4, 1, 5, 9, 2, 6), method = "ordinal", m = 3)

discretize(c(10, 12, 8, 40, 44, 9), method = "width", transform = "log")
```

---

esm\_srl

---

*Momentary Self-Regulated Learning Experience-Sampling Data*


---

### Description

A fully anonymized intensive longitudinal data set in which 41 students rated their momentary self-regulation, motivation, and anxiety several times per day (47 to 79 occasions each, one or more per day). Every indicator is on a 0-100 scale. The participant identifiers are fictional names and the calendar dates have been shifted by a constant offset — within-person spacing is preserved while no real dates or identities remain. There are 41 missing indicator values in total.

### Usage

```
esm_srl
```

### Format

A data frame with 2,820 rows and 13 variables:

**name** Fictional participant identifier (41 unique students).  
**occasion** Within-person occasion index, ordered in time.  
**date** Anonymized (constant-shifted) assessment date.  
**day\_type** Factor with levels "weekday" and "weekend", derived from date.  
**efficacy** Momentary self-efficacy (motivation), 0-100.  
**value** Momentary task value (motivation), 0-100.  
**planning** Momentary planning (self-regulation), 0-100.  
**monitoring** Momentary monitoring (self-regulation), 0-100.  
**effort** Momentary effort regulation (self-regulation), 0-100.  
**regulation** Momentary strategy regulation (self-regulation), 0-100.  
**motivated** Momentary felt motivation (motivation), 0-100.  
**enjoyment** Momentary enjoyment (motivation), 0-100.  
**anxiety** Momentary anxiety, 0-100.

## Details

The same data set ships with the authors' 'idiographic' package; the copy here additionally carries the `day_type` column, derived from `date`, so grouped models (see the `group` argument of `ts_tna()`) need no caller-side preparation.

## Source

Package authors' own experience-sampling study, collected by the authors and anonymized as described above (fictional identifiers, constant-shifted dates). Distributed with the package under its license.

---

|             |                                              |
|-------------|----------------------------------------------|
| plot.ts_tna | <i>Plot a Time-Series Transition Network</i> |
|-------------|----------------------------------------------|

---

## Description

Combined views of a `ts_tna()`-family result. `type = "combined"` (default) places each source series — with optional state shading and/or a classification ribbon — beside a transition network on the same row. With `network = "per_series"` (default) every row pairs a series with **its own** network (rebuilt via `series_networks()`, so all networks share the same node set and are directly comparable); `network = "summary"` draws the pooled model instead, spanning all series rows and titled "Summary". One state colour mapping ties the shading, ribbons, and network nodes together. `type = "network"` draws only the network(s) — a row of per-series networks or the single summary; `type = "series"` only the series panels. Every combination of `overlay` (shaded or not) and `ribbon` is available.

## Usage

```
## S3 method for class 'ts_tna'
plot(
  x,
  type = c("combined", "network", "series"),
  series = NULL,
  max_series = 3L,
  network = c("per_series", "summary"),
  overlay = c("horizontal", "vertical", "none"),
  ribbon = FALSE,
  points = FALSE,
  node_size = c("instrength", "outstrength", "strength"),
  node_scale = 1,
  network_width = 0.85,
  show_weights = TRUE,
  alpha = 0.28,
  line_color = "#3B4252",
  line_width = 1.5,
  point_size = 0.9,
  strip_height = 0.09,
```

```

palette = NULL,
legend = TRUE,
xlab = "Time",
ylab = "Value",
cex = 1,
grid = TRUE,
background = "#FFFFFF",
...
)

```

## Arguments

|                                                 |                                                                                                                                                                                                                                             |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>                                  | A <code>ts_tna</code> result from <code>ts_tna()</code> , <code>ts_ftna()</code> , <code>ts_cna()</code> , or <code>ts_atna()</code> .                                                                                                      |
| <code>type</code>                               | "combined", "network", or "series".                                                                                                                                                                                                         |
| <code>series</code>                             | Optional series IDs to draw (default: up to <code>max_series</code> ).                                                                                                                                                                      |
| <code>max_series</code>                         | Maximum number of series panels (default 3).                                                                                                                                                                                                |
| <code>network</code>                            | Which network(s) to draw: "per_series" (default) — one network per series, each built from that series' own transitions — or "summary" — the single pooled model across all series.                                                         |
| <code>overlay</code>                            | Series shading: "horizontal" (default), "vertical", or "none".                                                                                                                                                                              |
| <code>ribbon</code>                             | Whether to run the state strip under each series panel.                                                                                                                                                                                     |
| <code>points</code>                             | Whether to draw state-coloured observation points.                                                                                                                                                                                          |
| <code>node_size</code>                          | Node sizing rule: "instrength" (default), "outstrength", or "strength" — passed to <code>cograph</code> as <code>scale_nodes_by</code> (self-loops excluded from the centrality).                                                           |
| <code>node_scale</code>                         | Multiplier for <code>cograph</code> 's node size range ( <code>node_size_range = c(7, 14) * node_scale</code> ).                                                                                                                            |
| <code>network_width</code>                      | Width of the network column relative to the series column (default 0.85); combined view only.                                                                                                                                               |
| <code>show_weights</code>                       | Whether to print edge weights on the network.                                                                                                                                                                                               |
| <code>alpha</code>                              | Shading opacity.                                                                                                                                                                                                                            |
| <code>line_color, line_width, point_size</code> | Series styling.                                                                                                                                                                                                                             |
| <code>strip_height</code>                       | Ribbon strip height as a fraction of each panel.                                                                                                                                                                                            |
| <code>palette</code>                            | State colours (preset name, named vector, or vector).                                                                                                                                                                                       |
| <code>legend</code>                             | Whether to draw the state legend row.                                                                                                                                                                                                       |
| <code>xlab, ylab</code>                         | Series axis titles.                                                                                                                                                                                                                         |
| <code>cex</code>                                | Global text size multiplier.                                                                                                                                                                                                                |
| <code>grid</code>                               | Whether to draw the background grid.                                                                                                                                                                                                        |
| <code>background</code>                         | Panel background colour.                                                                                                                                                                                                                    |
| <code>...</code>                                | Forwarded to <code>cograph::splot()</code> for the network panel (e.g. <code>scale_nodes_by</code> , <code>node_size_range</code> , <code>scale_nodes_scale</code> , <code>edge_color</code> ); user values override the defaults set here. |

## Details

Network panels are rendered exclusively by `cograph::splot()` — the full TNA style with self-loops and probability labels — with node fills matching the state shading and nodes sized by `cograph`'s native centrality scaling (`scale_nodes_by = "instrength"` by default, with `loops = FALSE`), so heavily-entered states read as larger and self-transitions do not drown the between-state structure; pass `scale_nodes_by = list("instrength", loops = TRUE)` through `...` to include self-transitions. `"outstrength"` and `"strength"` (total) are alternatives, and `...` is forwarded to `cograph::splot()` for full control (anything you set there overrides these defaults). `show_weights` controls `cograph`'s edge labels.

## Value

`x`, invisibly.

## Examples

```
set.seed(1)
series <- list(a = cumsum(rnorm(80)), b = cumsum(rnorm(80)))
network <- ts_tna(series, n_states = 3, labels = c("low", "mid", "high"))
plot(network)
plot(network, network = "summary")
plot(network, ribbon = TRUE, overlay = "none")
plot(network, "network")
```

---

plot.ts\_tna\_group

*Plot One Network of a Grouped Transition Model*

---

## Description

A grouped model holds one complete `ts_tna` network per group, and there is no single picture of all of them, so one group is drawn at a time. The selected network is rendered by `plot.ts_tna()` with its full argument surface — combined series-plus-network views, state shading, ribbons, and every `cograph::splot()` option.

## Usage

```
## S3 method for class 'ts_tna_group'
plot(x, y = NULL, group = NULL, ...)
```

## Arguments

|                    |                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------|
| <code>x</code>     | A <code>ts_tna_group</code> result from <code>ts_tna()</code> and friends, built with their group argument. |
| <code>y</code>     | Ignored.                                                                                                    |
| <code>group</code> | Group label to draw. It may be omitted when the model holds exactly one group.                              |
| <code>...</code>   | Passed to <code>plot.ts_tna()</code> .                                                                      |

**Value**

`x`, invisibly.

**See Also**

`ts_tna()` for the `group` argument; `as.data.frame.ts_tna_group()` for the tidy views of the collection.

**Examples**

```
data(esm_srl)
networks <- ts_tna(
  subset(esm_srl, !is.na(effort)),
  value = "effort", id = "name", time = "occasion",
  group = "day_type", labels = c("low", "mid", "high")
)
plot(networks, group = "weekday")
plot(networks, group = "weekend", type = "network")
```

---

plot.tsn

---

*Plot a TSN Result*


---

**Description**

By default, `plot()` renders the constructed network through `cograph::splot()`. The diagnostic source-series view (with optional state shading) remains available via `type = "series"` and does not require `cograph`.

**Usage**

```
## S3 method for class 'tsn'
plot(x, type = c("network", "series"), ...)
```

**Arguments**

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <code>x</code>    | A <code>tsn</code> result.                                          |
| <code>type</code> | What to draw: "network" (default) or "series".                      |
| <code>...</code>  | Arguments forwarded to the network or series renderer. See details. |

**Details**

Network rendering belongs exclusively to `cograph`. `tsn` supplies restrained defaults that keep edges visible and avoid meaningless per-node rainbow fills; named arguments in `...` override any of them. `tsn` results already implement the `cograph_network` contract. Source-series views remain `tsn`-specific because they visualize the original observations rather than a generic network.

For `type = "network"`, use `cograph` arguments such as `layout`, `labels`, `scale_nodes_by`, `node_size_range`, and `edge_label_style`. For `type = "series"`, `...` accepts `series`, `overlay`, `points`, `trend`, `columns`, `max_series`, `scales`, and `palette`.

**Value**

x, invisibly.

**Examples**

```
network <- tsn(c(3, 1, 4, 2, 5, 3, 6, 2, 7), "hvg")
if (requireNamespace("cograph", quietly = TRUE)) {
  plot(network)
  plot(network, layout = "spring")
}

data(srl)
states <- tsn(
  srl,
  value = "effort",
  id = "name",
  time = "day",
  series = "Erik",
  unit = "state",
  discretization = "quantile"
)
plot(states, "series", overlay = "vertical")
```

---

plot.tsn\_series\_networks

*Plot One Per-Series Transition Network*


---

**Description**

Plot One Per-Series Transition Network

**Usage**

```
## S3 method for class 'tsn_series_networks'
plot(x, y = NULL, series = NULL, ...)
```

**Arguments**

|        |                                                                                   |
|--------|-----------------------------------------------------------------------------------|
| x      | A tsn_series_networks result from <a href="#">series_networks()</a> .             |
| y      | Ignored.                                                                          |
| series | Series ID to draw. It may be omitted when the collection holds exactly one model. |
| ...    | Passed to <a href="#">plot.ts_tna()</a> .                                         |

**Value**

x, invisibly.

### Examples

```
states <- discretize(c(3, 1, 4, 2, 5, 3, 6, 2, 7))
networks <- series_networks(ts_tna(states))
plot(networks, type = "network")
```

---

|                 |                                    |
|-----------------|------------------------------------|
| plot.tsn_states | <i>Plot a state discretization</i> |
|-----------------|------------------------------------|

---

### Description

Four views of a discretized series, selected with type. "overlay" (default) draws each series with translucent state shading behind it; "ribbon" keeps the series clean and runs a state-classification strip underneath it instead; "heatmap" draws one row per series with one coloured tile per observation, which scales to many series where the faceted views do not; "stack" places an aligned companion series above the discretized series.

### Usage

```
## S3 method for class 'tsn_states'
plot(
  x,
  type = c("overlay", "ribbon", "heatmap", "stack"),
  series = NULL,
  overlay = c("horizontal", "vertical", "none"),
  lines = "none",
  min_run = 1L,
  points = TRUE,
  max_series = NULL,
  columns = NULL,
  palette = NULL,
  alpha = NULL,
  line_color = "#3B4252",
  line_width = 1.6,
  point_size = 1,
  strip_height = 0.1,
  sort = "none",
  border = FALSE,
  with = NULL,
  with_label = "Series",
  shade = TRUE,
  color_line = FALSE,
  ribbons = NULL,
  ribbon_label = "States",
  legend = TRUE,
  xlab = "Time",
  ylab = "Value",
```

```

    cex = 1,
    grid = TRUE,
    background = "#FFFFFF",
    ...
)

```

## Arguments

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x            | A tsn_states result.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| type         | The view: "overlay", "ribbon", "heatmap", or "stack". The stack view draws a companion series (with) above the discretized series with BOTH panels vertically shaded by the discretized states — e.g. the original time series read against the states of its rolling complexity.                                                                                                                                                                                                                                                                                                                                                                                                    |
| series       | Optional character vector of series IDs to display.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| overlay      | Colour shading for the overlay view: "horizontal" (default; value bands), "vertical" (time runs), or "none".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| lines        | Dashed guide lines complementing the shading: "horizontal" draws grey lines at the state boundaries (the discretization breaks when available, empirical boundaries otherwise). Transformed and magnitude-space breaks are converted back to the raw signed value axis; adaptive local boundaries are omitted. "vertical" draws a line at each state transition (filtered by min_run) <b>in the colour of the state that starts there</b> — a green dashed line means the green state begins at that point and runs until the next line; "both", or "none" (default). Combine one shading direction with dashed lines in the other, e.g. overlay = "horizontal", lines = "vertical". |
| min_run      | For lines = "vertical": only transitions into a state persisting at least min_run observations get a line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| points       | Whether to draw observations over each line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| max_series   | Maximum number of series to draw (25 for the heatmap, 10 otherwise).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| columns      | Number of panel columns.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| palette      | State colours: NULL for the default palette, a preset name ("default", "okabe", "viridis", "cool", "warm", "pastel", "dark"), a named colour vector keyed by state, or a plain colour vector.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| alpha        | Shading opacity for the overlay view.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| line_color   | Series line colour.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| line_width   | Series line width.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| point_size   | Observation point size.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| strip_height | Height of the ribbon strip as a fraction of the panel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| sort         | Heatmap row order: "none", "mean" (by series mean value), or "state" (group rows by modal state).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| border       | Whether heatmap tiles carry a thin separator border.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| with         | For type = "stack": a finite numeric vector aligned with the rows of x giving the companion series (e.g. the original values when x discretizes a derived series).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| with_label   | Y-axis title of the companion panel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| shade        | For type = "stack": whether to draw the vertical state shading on both panels.                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| color_line   | For type = "stack": colour each line segment by the state at its left endpoint instead of drawing a neutral line.                                                                                                                                                                                                                                                                                                                                                                                                        |
| ribbons      | For type = "ribbon": an optional NAMED list of additional classifications to stack below the series — each element a <code>tsn_states</code> , a <code>trend()</code> result, or a character/factor vector aligned with the selected series. The object's own states form the first strip (labelled <code>ribbon_label</code> ); one series at a time. This is the resilience-style multi-ribbon view: the series read against several classifications at once, each strip labelled on the axis with its own legend row. |
| ribbon_label | Strip label for the object's own states in the multi-ribbon view.                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| legend       | Whether to draw the legend row.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| xlab, ylab   | Axis titles.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| cex          | Global text size multiplier.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| grid         | Whether to draw the background grid.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| background   | Panel background colour.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| ...          | Reserved for future options.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

### Value

x, invisibly.

### Examples

```
data(srl)
complete <- subset(srl, !is.na(effort))
states <- discretize(
  complete,
  value = "effort", id = "name", time = "day",
  method = "quantile", n_states = 3
)
plot(states, series = "Erik")
plot(states, series = "Erik", overlay = "vertical", lines = "horizontal")
plot(states, "ribbon", series = "Erik", points = FALSE)
plot(states, "heatmap", max_series = 12, sort = "mean")
plot(states, "heatmap", palette = "viridis", max_series = 12)

# Stack: original series read against the states of its rolling volatility.
one <- subset(complete, name == "Erik")
volatility <- abs(c(0, diff(one$effort)))
vol_states <- discretize(
  data.frame(name = one$name, day = one$day, vol = volatility),
  value = "vol", id = "name", time = "day",
  method = "quantile", n_states = 3
)
plot(vol_states, "stack", with = one$effort, with_label = "Effort",
  ylab = "Volatility")
```

---

|                |                                    |
|----------------|------------------------------------|
| plot.tsn_trend | <i>Plot a trend classification</i> |
|----------------|------------------------------------|

---

### Description

Four views of a trend classification, selected with `type`. "points" (default) draws each series with observations coloured by trend state; "ribbon" runs the trend classification as a strip underneath the clean series line; "heatmap" draws one row per series with one coloured tile per observation, comparing trend timing across many series; "panels" stacks each series above its rolling trend metric (the slope or correlation `trend()` classified on) so a state change can be read against the metric that produced it.

### Usage

```
## S3 method for class 'tsn_trend'
plot(
  x,
  type = c("points", "ribbon", "heatmap", "panels"),
  series = NULL,
  max_series = NULL,
  columns = NULL,
  palette = NULL,
  line_color = "#9AA3AD",
  line_width = 1.3,
  point_size = 1.05,
  strip_height = 0.1,
  sort = "none",
  border = FALSE,
  flat_band = TRUE,
  legend = TRUE,
  xlab = "Time",
  ylab = "Value",
  cex = 1,
  grid = TRUE,
  background = "#FFFFFF",
  ...
)
```

### Arguments

|                         |                                                                                            |
|-------------------------|--------------------------------------------------------------------------------------------|
| <code>x</code>          | A <code>tsn_trend</code> result.                                                           |
| <code>type</code>       | The view: "points", "ribbon", "heatmap", or "panels".                                      |
| <code>series</code>     | Optional character vector of series IDs to display.                                        |
| <code>max_series</code> | Maximum number of series to draw (25 for the heatmap, 4 for the panel view, 10 otherwise). |
| <code>columns</code>    | Number of panel columns.                                                                   |

|              |                                                                                                                  |
|--------------|------------------------------------------------------------------------------------------------------------------|
| palette      | Optional named colours overriding individual trend states (e.g. c(Ascending = "forestgreen")).                   |
| line_color   | Series line colour.                                                                                              |
| line_width   | Series line width.                                                                                               |
| point_size   | Observation point size.                                                                                          |
| strip_height | Ribbon strip height as a fraction of the panel.                                                                  |
| sort         | Heatmap row order: "none", "mean", or "state".                                                                   |
| border       | Whether heatmap tiles carry a thin separator border.                                                             |
| flat_band    | For type = "panels", shade the epsilon flat band on the metric panel so the classification threshold is visible. |
| legend       | Whether to draw the legend row.                                                                                  |
| xlab, ylab   | Axis titles.                                                                                                     |
| cex          | Global text size multiplier.                                                                                     |
| grid         | Whether to draw the background grid.                                                                             |
| background   | Panel background colour.                                                                                         |
| ...          | Reserved for future options.                                                                                     |

**Value**

x, invisibly.

**Examples**

```
data(srl)
complete <- subset(srl, !is.na(effort))
classified <- trend(
  complete,
  value = "effort", id = "name", time = "day", window = 7
)
plot(classified, series = "Erik")
plot(classified, "ribbon", series = "Erik")
plot(classified, "panels", series = "Erik")
plot(classified, "heatmap", max_series = 12)
```

---

print.tsn

---

*Print a TSN network*


---

**Description**

Print a TSN network

**Usage**

```
## S3 method for class 'tsn'
print(x, ...)
```

**Arguments**

x                    A tsn result.  
 ...                  Additional arguments passed to print.data.frame().

**Value**

x, invisibly.

---

|                  |                                     |
|------------------|-------------------------------------|
| print.tsn_states | <i>Print a state discretization</i> |
|------------------|-------------------------------------|

---

**Description**

Print a state discretization

**Usage**

```
## S3 method for class 'tsn_states'
print(x, ...)
```

**Arguments**

x                    A tsn\_states result.  
 ...                  Passed to print.data.frame().

**Value**

x, invisibly.

---

|                 |                                     |
|-----------------|-------------------------------------|
| print.tsn_trend | <i>Print a trend classification</i> |
|-----------------|-------------------------------------|

---

**Description**

Print a trend classification

**Usage**

```
## S3 method for class 'tsn_trend'
print(x, ...)
```

**Arguments**

x                    A tsn\_trend result.  
 ...                  Passed to print.data.frame().

**Value**

`x`, invisibly.

---

series\_networks

*Per-Series Transition Networks*


---

**Description**

Split a pooled `ts_tna()`-family model into one full model per source series. Each element is rebuilt with the same network type and the shared state alphabet (`params = list(alphabet = ...)`), so every network has the same node set in the same order — directly comparable with each other and with the pooled summary model. Each element is a complete `ts_tna` netobject: it keeps its own series data, plots with `plot.ts_tna()`, and works with compatible Nestimate verbs. Statistical procedures still require enough independent sequences for their sampling design; a one-series split is descriptive rather than bootstrap-ready.

**Usage**

```
series_networks(x, series = NULL)
```

**Arguments**

|                     |                                                                                                                                        |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>      | A <code>ts_tna</code> result from <code>ts_tna()</code> , <code>ts_ftna()</code> , <code>ts_cna()</code> , or <code>ts_atna()</code> . |
| <code>series</code> | Optional series IDs to keep (default: all).                                                                                            |

**Value**

A named `tsn_series_networks` collection containing one `ts_tna` object per series. Its `print`, `summary`, and data-frame methods return a tidy one-row-per-series index; `plot()` draws the sole model or a model selected with its `series` argument.

**Examples**

```
set.seed(1)
series <- list(a = cumsum(rnorm(80)), b = cumsum(rnorm(80)))
pooled <- ts_tna(series, n_states = 3, labels = c("low", "mid", "high"))
networks <- series_networks(pooled)
plot(networks, series = "a")
```

---

srl*Daily Self-Regulated Learning Panel*

---

**Description**

A balanced intensive longitudinal panel in which 36 students reported nine self-regulated-learning indicators once per study occasion for 156 occasions. Every indicator is on a 0-100 scale, and day is a within-person occasion index, so the panel feeds `tsn()`, `discretize()`, and the `ts_tna()` family directly as long data. There are 131 missing indicator values in total.

**Usage**

srl

**Format**

A data frame with 5,616 rows and 11 variables:

**name** Student identifier (36 unique students).

**day** Within-person occasion index, 1 through 156.

**efficacy** Self-efficacy, 0-100.

**value** Task value, 0-100.

**planning** Planning, 0-100.

**monitoring** Monitoring, 0-100.

**effort** Effort regulation, 0-100.

**control** Control of learning, 0-100.

**help** Help seeking, 0-100.

**social** Social support, 0-100.

**organizing** Organizing, 0-100.

**Details**

The same data set ships with the authors' 'idiographic' package; the copy here is identical.

**Source**

Companion data of Saqr, M., & López-Pernas, S. (Eds.), *Advanced Learning Analytics Methods: AI, Precision and Complexity* (<https://github.com/lamethods/data2/tree/main/srl>), created and owned by the package authors and distributed under CC BY-NC-SA 4.0 (<https://creativecommons.org/licenses/by-nc-sa/4.0/>). Rows are ordered by name and day; values are unchanged. See the package COPYRIGHTS file for full attribution.

---

|             |                                |
|-------------|--------------------------------|
| summary.tsn | <i>Summarize a TSN network</i> |
|-------------|--------------------------------|

---

**Description**

Summarize a TSN network

**Usage**

```
## S3 method for class 'tsn'
summary(object, ...)
```

**Arguments**

|        |               |
|--------|---------------|
| object | A tsn result. |
| ...    | Ignored.      |

**Value**

A one-row data frame with columns method, unit, nodes, dyads, edges, density, minimum\_weight, maximum\_weight, and directed. density divides the connected edges by the number of possible edges for the network family: state networks (unit = "state") always count self-loops among the possible edges, because a state can follow itself, while series, window, and time-point networks never do, because their units cannot pair with themselves. The denominator therefore depends only on the network type, never on which edges happen to be observed, so densities are comparable across models of the same family.

---

|                             |                                                 |
|-----------------------------|-------------------------------------------------|
| summary.tsn_series_networks | <i>Summarize Per-Series Transition Networks</i> |
|-----------------------------|-------------------------------------------------|

---

**Description**

Summarize Per-Series Transition Networks

**Usage**

```
## S3 method for class 'tsn_series_networks'
summary(object, ...)

## S3 method for class 'tsn_series_networks'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'tsn_series_networks'
print(x, ...)
```

**Arguments**

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| object              | A tsn_series_networks result from <a href="#">series_networks()</a> . |
| ...                 | Ignored.                                                              |
| x                   | A tsn_series_networks result.                                         |
| row.names, optional | Ignored.                                                              |

**Value**

A tidy data frame with one row per series.

---

|                    |                                         |
|--------------------|-----------------------------------------|
| summary.tsn_states | <i>Summarize a state discretization</i> |
|--------------------|-----------------------------------------|

---

**Description**

Summarize a state discretization

**Usage**

```
## S3 method for class 'tsn_states'
summary(object, ...)
```

**Arguments**

|        |                      |
|--------|----------------------|
| object | A tsn_states result. |
| ...    | Ignored.             |

**Value**

A tidy data frame with one row per state giving the count, proportion, and mean value of observations in that state.

---

|                   |                                         |
|-------------------|-----------------------------------------|
| summary.tsn_trend | <i>Summarize a trend classification</i> |
|-------------------|-----------------------------------------|

---

**Description**

Summarize a trend classification

**Usage**

```
## S3 method for class 'tsn_trend'
summary(object, ...)
```

Arguments

|        |                     |
|--------|---------------------|
| object | A tsn_trend result. |
| ...    | Ignored.            |

Value

A tidy data frame with one row per observed trend state, giving the count and proportion of observations in that state.

---

|       |                                                 |
|-------|-------------------------------------------------|
| trend | <i>Classify Rolling Trends in a Time Series</i> |
|-------|-------------------------------------------------|

---

Description

trend() computes a rolling trend metric for each observation and classifies every point as "Ascending", "Descending", "Flat", "Turbulent", "Missing Data", or "Initial". It is a faithful base-R port of the upstream tsn::trend() behaviour: rolling slope (OLS, Theil-Sen, Spearman, or Kendall) or growth-factor metrics, an epsilon flat band, and a volatility override that reclassifies noisy segments as turbulent.

Usage

```
trend(  
  data,  
  value = NULL,  
  id = NULL,  
  time = NULL,  
  series = NULL,  
  window = NULL,  
  method = "slope",  
  slope = "robust",  
  epsilon = 0.05,  
  turbulence_threshold = 5,  
  flat_to_turbulent_factor = 1.5,  
  align = "center"  
)
```

Arguments

|        |                                                                                                                        |
|--------|------------------------------------------------------------------------------------------------------------------------|
| data   | A numeric vector, ts, matrix, named list of numeric vectors, data frame, or a tsn object (its source series are used). |
| value  | Optional value-column name for long data.                                                                              |
| id     | Optional series-ID column name for long data.                                                                          |
| time   | Optional time-column name for long data.                                                                               |
| series | Optional series IDs or wide-data column names to select.                                                               |

|                          |                                                                                                                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| window                   | Rolling-window width. When NULL, the adaptive default $\max(3, \min(n, \text{round}(n / 10)))$ is used, where $n$ is the shortest series length. Must satisfy $2 < \text{window} < n$ . |
| method                   | Trend metric: "slope" (default) or "growth_factor".                                                                                                                                     |
| slope                    | Slope estimator when method = "slope": "robust" (Theil-Sen, the default), "ols", "spearman", or "kendall".                                                                              |
| epsilon                  | Flat-band half-width. Default 0.05.                                                                                                                                                     |
| turbulence_threshold     | Baseline volatility threshold for the turbulent override. Default 5.                                                                                                                    |
| flat_to_turbulent_factor | Multiplier applied to turbulence_threshold for points already classified as flat. Default 1.5.                                                                                          |
| align                    | Window alignment: "center" (default), "right", or "left". The metric is assigned to the centre, rightmost, or leftmost point of the window respectively.                                |

### Details

The metric is first thresholded with epsilon: values above +epsilon (or above  $1 + \text{epsilon}$  for growth factors) are ascending, values below -epsilon (or  $1 - \text{epsilon}$ ) are descending, and the remainder are flat. A rolling volatility measure (coefficient of variation plus half the range factor of the metric) then overrides these labels with "Turbulent" when it exceeds turbulence\_threshold. Segments already labelled "Flat" use the higher threshold  $\text{turbulence\_threshold} * \text{flat\_to\_turbulent\_factor}$ , making them more resistant to noise-driven reclassification.

### Value

A tidy data frame of class `tsn_trend` with one row per observation and columns `id`, `time`, `value`, `metric`, and `state` (a factor over the six trend classes). `Print`, `summary`, and `plot` methods are provided.

### Examples

```
set.seed(123)
walk <- cumsum(rnorm(120))
trend(walk, window = 15, slope = "ols", epsilon = 0.1)

trend(c(1, 2, 3, 4, 5, 4, 3, 2, 1), window = 3, method = "growth_factor")
```

---

ts\_tna

---

*Transition Network Analysis of Time Series*


---

### Description

`ts_tna()`, `ts_ftna()`, `ts_cna()`, and `ts_atna()` bridge `tsn`'s discretization engine to the `Nestimate` package: a numeric time series (or several) is discretized into states with any of `tsn`'s discretizers, each series becomes one state sequence, and `Nestimate` builds the transition network — row-normalized probabilities (`ts_tna`), raw transition counts (`ts_ftna`), co-occurrence counts (`ts_cna`), or attention-weighted transitions (`ts_atna`).

**Usage**

```
ts_tna(  
  data,  
  value = NULL,  
  id = NULL,  
  time = NULL,  
  series = NULL,  
  discretization = "quantile",  
  n_states = 3L,  
  breaks = NULL,  
  labels = NULL,  
  transform = "none",  
  m = NULL,  
  tau = NULL,  
  segment = NULL,  
  overlap = FALSE,  
  group = NULL,  
  seed = NULL,  
  ...  
)
```

```
ts_ftna(  
  data,  
  value = NULL,  
  id = NULL,  
  time = NULL,  
  series = NULL,  
  discretization = "quantile",  
  n_states = 3L,  
  breaks = NULL,  
  labels = NULL,  
  transform = "none",  
  m = NULL,  
  tau = NULL,  
  segment = NULL,  
  overlap = FALSE,  
  group = NULL,  
  seed = NULL,  
  ...  
)
```

```
ts_cna(  
  data,  
  value = NULL,  
  id = NULL,  
  time = NULL,  
  series = NULL,  
  discretization = "quantile",
```

```

    n_states = 3L,
    breaks = NULL,
    labels = NULL,
    transform = "none",
    m = NULL,
    tau = NULL,
    segment = NULL,
    overlap = FALSE,
    group = NULL,
    seed = NULL,
    ...
)

ts_atna(
  data,
  value = NULL,
  id = NULL,
  time = NULL,
  series = NULL,
  discretization = "quantile",
  n_states = 3L,
  breaks = NULL,
  labels = NULL,
  transform = "none",
  m = NULL,
  tau = NULL,
  segment = NULL,
  overlap = FALSE,
  group = NULL,
  seed = NULL,
  ...
)

```

### Arguments

|                |                                                                                                                                                                                                                                                                                                                           |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data           | A numeric vector, ts, matrix, named list of numeric vectors, data frame, or a tsn_states result from <code>discretize()</code> .                                                                                                                                                                                          |
| value          | Optional value-column name for long data.                                                                                                                                                                                                                                                                                 |
| id             | Optional series-ID column name for long data.                                                                                                                                                                                                                                                                             |
| time           | Optional time-column name for long data.                                                                                                                                                                                                                                                                                  |
| series         | Optional series IDs or wide-data column names to select.                                                                                                                                                                                                                                                                  |
| discretization | Discretization method passed to <code>discretize()</code> (default "quantile"). When data is already a tsn_states, supplying this or any other discretization option (value, id, time, n_states, breaks, labels, transform, m, tau, seed) is an error: the states are already fixed and the option could not take effect. |
| n_states       | Number of states (default 3). Not consumed by discretization = "ordinal", whose state count follows m and tau; supplying both is an error.                                                                                                                                                                                |

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| breaks    | Optional interior thresholds for discretization = "threshold". An error with any other discretizer, which computes its own boundaries.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| labels    | Optional state labels; these become the network's node names (e.g. <code>c("low", "mid", "high")</code> ).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| transform | Pre-discretization transform: "none", "log", or "zscore".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| m, tau    | Embedding arguments for discretization = "ordinal".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| segment   | Optional block width, in observations, used to cut each series into several shorter sequences. Sequence-based inference resamples whole sequences, so a single long series offers nothing to resample; segmenting supplies the units. Blocks never span an ID boundary, and segmentation is applied <i>after</i> discretization, so the state alphabet is still learned from the whole series. At least 2.                                                                                                                                                                                                                                                                                                                                                                                      |
| overlap   | When segmenting, slide the block one observation at a time instead of partitioning (default FALSE). Partitioning loses the transition at every cut, roughly one per block. Sliding keeps every transition — at segment = 2 the blocks are the consecutive lag-1 pairs and the network is identical to the unsegmented one — but the blocks share observations, so they are not independent and intervals computed from them run narrow. Partition for conservative intervals that tolerate dependence beyond one lag; slide to preserve the estimate exactly.                                                                                                                                                                                                                                   |
| group     | Optional name of a column of data holding each observation's group, e.g. a condition, a cohort, or a context. Supplying it returns one network per group instead of a single pooled network. States are discretized from the pooled series first, so every group is cut on one common scale and their networks share a node set and are directly comparable. A sequence never spans a group boundary: where a series stays in one group it contributes a single sequence (unless segment splits it further), and where the group column alternates within a series each contiguous run becomes its own sequence, so no transition is ever counted between observations that were not adjacent in time. The transition across a group change is dropped from both groups: it belongs to neither. |
| seed      | Optional seed for the stochastic discretizers ("kmeans" and "gaussian"). An error with the deterministic discretizers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| ...       | Passed on to the corresponding Nestimate builder ( <code>Nestimate::build_tna()</code> and friends), e.g. <code>start</code> , <code>end</code> , <code>scaling</code> , <code>threshold</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

## Details

The result is a full Nestimate netobject (also a `cograph_network`), so compatible Nestimate descriptive and inferential verbs apply and `cograph::splot()` renders it directly. Inference still requires the sample its method assumes: for example, sequence bootstrap is degenerate for a single sequence. The object keeps its data: `$data` holds the wide state sequences Nestimate built from, `$ts_source` the tidy per-observation table (`id`, `time`, `value`, `state`), and `$meta$tsn` the discretization and builder settings, so the network remains traceable back to the raw series.

A single series supports every descriptive verb but no sequence-based test, because a bootstrap resamples sequences and one series is one sequence. The `segment` argument cuts a long series into blocks so that those tests have units to work with; see its documentation for the trade-off between partitioned and sliding blocks.

The `group` argument splits the model instead of pooling it: one network per condition, cohort, or context, all cut from one shared state alphabet so they can be compared. The result is a Nestimate `netobject_group`, so the grouped Nestimate verbs that suit a transition model apply to it directly. Verbs needing a precision matrix or a clustering attribute do not, and there is no `grouped plot()` method.

Multiple series are supported through every tsn input form (named list, matrix, wide or long data frame). Scalar discretizers learn from pooled values. Temporal discretizers compute patterns or windows separately within each series before assigning one shared state alphabet. Each series contributes one sequence. Passing an existing `discretize()` result skips discretization and uses its states as-is.

## Value

A Nestimate `netobject` of class `c("ts_tna", "netobject", "cograph_network")` with the additional fields `$ts_source` (tidy id/time/value/state table) and `$meta$tsn` (discretization settings). With `group`, a `c("ts_tna_group", "netobject_group")` collection holding one such network per group, which Nestimate's compatible grouped verbs (`net_prune()`, `state_distribution()`, `net_centrality()`, `compare_model()`, `permutation()`) accept directly and `as.data.frame()` renders as a tidy table.

## Examples

```
set.seed(1)
series <- list(
  a = cumsum(rnorm(60)),
  b = cumsum(rnorm(60)),
  c = cumsum(rnorm(60))
)
network <- ts_tna(series, n_states = 3, labels = c("low", "mid", "high"))
network$weights

# Frequency counts instead of probabilities:
counts <- ts_ftna(series, n_states = 3)

# Reuse an existing discretization:
states <- discretize(series, method = "kmeans", n_states = 3)
ts_tna(states)

# Cut one long series into blocks so sequence-based tests have units:
long <- cumsum(rnorm(300))
ts_tna(long, segment = 10, labels = c("low", "mid", "high"))

# Sliding lag-1 pairs keep every transition and the exact estimate:
ts_tna(long, segment = 2, overlap = TRUE, labels = c("low", "mid", "high"))

# One network per day type, on a shared alphabet:
data(esm_srl)
by_day <- ts_tna(
  subset(esm_srl, !is.na(effort)),
  value = "effort", id = "name", time = "occasion",
  group = "day_type", labels = c("low", "mid", "high")
)
```

```
)
as.data.frame(by_day, what = "groups")
```

tsn

*Build a Time-Series Network*

## Description

`tsn()` is the core entry point for constructing networks from time-series geometry. It builds distance networks between complete series or sliding windows, and natural or horizontal visibility networks between time points or discretized states.

## Usage

```
tsn(
  data,
  method = "visibility",
  value = NULL,
  id = NULL,
  time = NULL,
  series = NULL,
  unit = NULL,
  distance = "euclidean",
  connect = "full",
  window = NULL,
  step = 1L,
  neighbors = NULL,
  threshold = NULL,
  percentile = NULL,
  bandwidth = NULL,
  p = 2,
  bins = NULL,
  lag = NULL,
  tolerance = NULL,
  similarity = NULL,
  visibility = "natural",
  state = NULL,
  discretization = "gaussian",
  n_states = 3L,
  breaks = NULL,
  m = NULL,
  tau = NULL,
  directed = FALSE,
  limit = NULL,
  penetrable = 0L,
  decay = 0,
```

```

    aggregation = "sum",
    chain = FALSE,
    normalize = FALSE,
    seed = NULL
)

```

## Arguments

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data       | A numeric vector, ts, numeric matrix, named list of numeric vectors, or data frame.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| method     | Network selector. The base families are "distance" and "visibility". Convenience shortcuts resolve to a family with sensible defaults: "nvg"/"natural" (natural visibility graph), "hvg"/"horizontal" (horizontal visibility graph), and any discretizer name ("ordinal", "quantile", "symbolic", ...) which builds a visibility network on states from that discretizer. Shortcuts only set defaults; the granular arguments (unit, visibility, discretization) still apply.                                                                                                                                    |
| value      | Optional value-column name for long data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| id         | Optional series-ID column name for long data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| time       | Optional time-column name for long data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| series     | Optional series IDs for long data or numeric column names for wide data. Selection happens inside <code>tsn()</code> , and the requested order is authoritative for every input form: long rows are reordered to it exactly as wide, matrix, and list input are, so chained or directed edges do not depend on the input container.                                                                                                                                                                                                                                                                              |
| unit       | Node unit. Distance networks use "series", "window", or "time" (one node per time point); visibility networks use "time" or "state". When NULL, the natural unit for the selected method is inferred.                                                                                                                                                                                                                                                                                                                                                                                                            |
| distance   | Distance measure for distance networks. One of "euclidean", "manhattan", "maximum", "canberra", "minkowski", "binary", "cosine", "correlation", "spearman", "dtw", "ccf" (one minus the maximum absolute cross-correlation across lags), "nmi" (one minus normalized mutual information after separately quantile-binning each series), "voi" (variation of information after the same marginal binning), "event_sync" (one minus the Quiroga event-synchronization index; units are interpreted as event times), or "van_rossum" (exact van Rossum spike-train distance; units are interpreted as event times). |
| connect    | Distance-to-network rule.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| window     | Sliding-window width when unit = "window".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| step       | Sliding-window step.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| neighbors  | Number of neighbours when connect = "nearest".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| threshold  | Maximum distance when connect = "threshold".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| percentile | Proportion of shortest distances retained when connect = "percentile".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| bandwidth  | Positive kernel scale. Used by connect = "gaussian" and by the "negative_exp" and "gaussian" similarity kernels; defaults to the median positive distance.                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| p          | Minkowski power.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

|                |                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bins           | Number of marginal quantile bins for distance = "nmi" and distance = "voi".                                                                                                                                                                                                                                                                                       |
| lag            | Maximum cross-correlation lag for distance = "ccf". NULL uses the stats::ccf() default.                                                                                                                                                                                                                                                                           |
| tolerance      | Event-time scale for the event-based distances: an optional upper bound on the adaptive coincidence window for distance = "event_sync", or the kernel time constant for distance = "van_rossum". For event synchronization, NULL uses the uncapped adaptive local window. For van Rossum, NULL uses the median positive inter-event interval of each pooled pair. |
| similarity     | Optional similarity kernel mapping edge distances to weights: "inverse" (the default weight rule, $1 / (1 + d)$ ), "normalized_inverse" ( $1 - d / \max(d)$ ), "negative_exp" ( $\exp(-d / \text{bandwidth})$ ), or "gaussian" ( $\exp(-d^2 / (2 * \text{bandwidth}^2))$ ).                                                                                       |
| visibility     | Visibility rule: "natural" or "horizontal".                                                                                                                                                                                                                                                                                                                       |
| state          | Optional state-column name or state vector.                                                                                                                                                                                                                                                                                                                       |
| discretization | Internal state-discretization method. One of "threshold", "width", "quantile", "kde", "kmeans", "gaussian", "hclust", "ordinal", "symbolic", "change_points", "entropy", "magnitude", "adaptive_magnitude", "percentile_magnitude", or "dtw".                                                                                                                     |
| n_states       | Number of states. Not consumed by discretization = "ordinal", whose state count follows the embedding arguments m and tau; supplying both is an error.                                                                                                                                                                                                            |
| breaks         | Optional internal thresholds when discretization = "threshold". An error with any other discretizer, which computes its own boundaries.                                                                                                                                                                                                                           |
| m              | Embedding dimension for discretization = "ordinal" (default 3). Only valid with the ordinal discretizer.                                                                                                                                                                                                                                                          |
| tau            | Embedding lag for discretization = "ordinal" (default 1). Only valid with the ordinal discretizer.                                                                                                                                                                                                                                                                |
| directed       | Whether edges follow their ordered direction. Visibility edges then run forward in time; distance edges follow the evaluated unit order. With FALSE, reciprocal distance relations are represented by one undirected edge.                                                                                                                                        |
| limit          | Optional maximum visibility distance in the units of time (or observation steps when no numeric/date-time axis is supplied).                                                                                                                                                                                                                                      |
| penetrable     | Number of intermediate points allowed to block visibility.                                                                                                                                                                                                                                                                                                        |
| decay          | Non-negative exponential edge-decay rate per unit of time (or per observation step when no numeric/date-time axis is supplied).                                                                                                                                                                                                                                   |
| aggregation    | State-edge aggregation rule.                                                                                                                                                                                                                                                                                                                                      |
| chain          | When TRUE, distance networks connect only consecutive series or windows (a transition chain) instead of all pairs.                                                                                                                                                                                                                                                |
| normalize      | Distance rescaling applied before the connection rule. FALSE (default) leaves distances unchanged; TRUE or "max" divides by the maximum distance; "minmax" rescales to $[0, 1]$ ; "quantile" rescales by the 5th-95th percentile range, clamped to $[0, 1]$ .                                                                                                     |
| seed           | Optional seed for the stochastic discretizers ("kmeans" and "gaussian"). An error with the deterministic discretizers.                                                                                                                                                                                                                                            |

## Details

Every supplied argument must be consumed by the resolved configuration. An option that the selected method, unit, distance, connect, or discretization cannot use — for example chain on a visibility network, limit on a distance network, p without distance = "minkowski", or breaks without discretization = "threshold" — is rejected with an error rather than silently ignored, so a typo or a misunderstood configuration cannot produce a plausible but unintended model. Shortcut methods ("nvg", "hvg", discretizer names) likewise reject granular arguments that contradict what the shortcut implies.

For state networks, caller-supplied state values fix the node order: factors keep their declared level order (an unused level becomes a zero-degree node), and plain character states are ordered by first appearance. Discretized states follow the discretizer's numeric state order. The state column of the stored source series preserves this factor.

## Value

A list-backed network object of class `c("tsn", "netobject", "cograph_network")`. Its `$table` component is the tidy dyad table; use `as.data.frame()` for that table, `as.data.frame(x, what = "series")` for the canonical source observations, and `as.matrix()` for the weighted adjacency matrix.

## References

- Lacasa, L., Luque, B., Ballesteros, F., Luque, J., & Nuño, J. C. (2008). From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, 105(13), 4972-4975. doi:[10.1073/pnas.0709247105](https://doi.org/10.1073/pnas.0709247105)
- Luque, B., Lacasa, L., Ballesteros, F., & Luque, J. (2009). Horizontal visibility graphs: Exact results for random time series. *Physical Review E*, 80, 046103. doi:[10.1103/PhysRevE.80.046103](https://doi.org/10.1103/PhysRevE.80.046103)
- Quian Quiroga, R., Kreuz, T., & Grassberger, P. (2002). Event synchronization: A simple and fast method to measure synchronicity and time delay patterns. *Physical Review E*, 66, 041904. doi:[10.1103/PhysRevE.66.041904](https://doi.org/10.1103/PhysRevE.66.041904)

## Examples

```
series <- list(
  first = c(1, 2, 3, 2, 1),
  second = c(1, 1, 2, 3, 5),
  third = c(5, 4, 3, 2, 1)
)

tsn(
  data = series,
  method = "distance",
  unit = "series",
  distance = "euclidean",
  connect = "full"
)

# One data argument, one method string.
tsn(c(3, 1, 4, 2, 5), "hvg")
```

```
tsn(c(3, 1, 4, 2, 5, 3, 6, 2, 7), "ordinal")
tsn(c(3, 1, 4, 2, 5, 3, 6, 2, 7), "distance")

data(srl)
tsn(
  srl,
  value = "effort",
  id = "name",
  time = "day",
  series = "Erik",
  unit = "state",
  discretization = "gaussian"
)
```

vg

*Build a Visibility Graph*

## Description

`vg()` is a discoverable verb for the package's flagship construction, the visibility graph. It is a thin wrapper around `tsn()` with `method = "visibility"`: `vg(x)` is equivalent to `tsn(x, "nvg")` and `vg(x, "horizontal")` is equivalent to `tsn(x, "hvg")`. Every input shape and every visibility-relevant argument that `tsn()` accepts is forwarded through `...`, and the result is the same dual-class `tsn/netobject/` `cograph_network` object.

## Usage

```
vg(data, type = c("natural", "horizontal"), ...)
```

## Arguments

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code> | A numeric vector, ts, numeric matrix, named list of numeric vectors, or data frame (the same inputs <code>tsn()</code> accepts).                                                                                                                                                                                                                                                                                                                      |
| <code>type</code> | Visibility rule: "natural" (the natural visibility graph, the default) or "horizontal" (the horizontal visibility graph).                                                                                                                                                                                                                                                                                                                             |
| <code>...</code>  | Any other <code>tsn()</code> argument relevant to visibility networks, for example <code>value</code> , <code>id</code> , <code>time</code> , <code>series</code> , <code>unit</code> , <code>state</code> , <code>discretization</code> , <code>n_states</code> , <code>m</code> , <code>tau</code> , <code>directed</code> , <code>limit</code> , <code>penetrable</code> , <code>decay</code> , <code>aggregation</code> , and <code>seed</code> . |

## Value

A tidy `tsn` network object (also a `netobject` and a `cograph_network`), equivalent to the corresponding `tsn()` call. The stored provenance call records `vg()` rather than `tsn()`.

## See Also

`tsn()` for the full argument set and the distance-network family.

**Examples**

```
vg(c(3, 1, 4, 2, 5, 3, 6, 2, 7))
vg(c(3, 1, 4, 2, 5, 3, 6, 2, 7), "horizontal")

network <- vg(c(3, 1, 4, 2, 5, 3, 6, 2, 7), "horizontal", directed = TRUE)
if (requireNamespace("cograph", quietly = TRUE)) {
  plot(network)
}

data(srl)
states <- vg(
  srl,
  value = "effort",
  id = "name",
  time = "day",
  series = "Erik",
  unit = "state",
  discretization = "quantile"
)
plot(states, "series", overlay = "vertical")
```

# Index

- \* **datasets**
  - esm\_srl, [8](#)
  - srl, [21](#)
- as.data.frame(), [29](#)
- as.data.frame.ts\_tna, [2](#)
- as.data.frame.ts\_tna\_group, [3](#)
- as.data.frame.ts\_tna\_group(), [12](#)
- as.data.frame.tsn, [4](#)
- as.data.frame.tsn\_series\_networks  
    (summary.tsn\_series\_networks),  
    [22](#)
- as.matrix.ts\_tna, [5](#)
- as.matrix.tsn, [6](#)
- cograph::splot(), [12](#)
- discretize, [6](#)
- discretize(), [21](#), [27](#), [29](#)
- esm\_srl, [8](#)
- plot.ts\_tna, [9](#)
- plot.ts\_tna(), [11](#), [13](#), [20](#)
- plot.ts\_tna\_group, [11](#)
- plot.tsn, [12](#)
- plot.tsn\_series\_networks, [13](#)
- plot.tsn\_states, [14](#)
- plot.tsn\_trend, [17](#)
- print.tsn, [18](#)
- print.tsn\_series\_networks  
    (summary.tsn\_series\_networks),  
    [22](#)
- print.tsn\_states, [19](#)
- print.tsn\_trend, [19](#)
- series\_networks, [20](#)
- series\_networks(), [9](#), [13](#), [23](#)
- srl, [21](#)
- summary.tsn, [22](#)
- summary.tsn\_series\_networks, [22](#)
- summary.tsn\_states, [23](#)
- summary.tsn\_trend, [23](#)
- trend, [24](#)
- trend(), [16](#)
- ts\_atna(ts\_tna), [25](#)
- ts\_atna(), [10](#), [20](#)
- ts\_cna(ts\_tna), [25](#)
- ts\_cna(), [10](#), [20](#)
- ts\_ftna(ts\_tna), [25](#)
- ts\_ftna(), [10](#), [20](#)
- ts\_tna, [25](#)
- ts\_tna(), [4](#), [5](#), [9–12](#), [20](#), [21](#)
- tsn, [30](#)
- tsn(), [6](#), [21](#), [34](#)
- vg, [34](#)