Package 'socialdrift'

May 9, 2026

Type Package

Title Temporal Auditing of Social Interaction Networks

Version 0.1.0

Description Tools for constructing, auditing, and visualizing temporal social interaction networks from event-log data. Supports graph construction from raw user-to-user interaction logs, longitudinal tracking of network structure, community dynamics, user role trajectories, and concentration of engagement over time. Designed for computational social science, platform analytics, and digital community health monitoring. Includes four longitudinal audit indices: the Network Drift Index ('NDI'), Community Fragmentation Index ('CFI'), Visibility Concentration Index ('VCI'), and Role Mobility Index ('RMI'). 'NDI', 'CFI', 'VCI', and 'RMI' are purpose-built composite scores for longitudinal platform auditing.

License GPL-3

URL https://github.com/causalfragility-lab/socialdrift

BugReports https://github.com/causalfragility-lab/socialdrift/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports dplyr (>= 1.1.0), tibble (>= 3.2.0), tidyr (>= 1.3.0), purrr (>= 1.0.0), ggplot2 (>= 3.4.0), igraph (>= 1.5.0), stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown, patchwork

RoxygenNote 7.3.3

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Subir Hait [aut, cre] (ORCID: <https://orcid.org/0009-0004-9871-9677>)

Maintainer Subir Hait <haitsubi@msu.edu>

Repository CRAN

Date/Publication 2026-04-21 19:52:45 UTC

Contents

`as_social_events` *Coerce a data frame into a standardized social event table*

Description

Validates and standardizes a data frame of user-to-user interaction events for use in temporal social network analysis with **socialdrift**.

Usage

```
as_social_events(
  data,
  actor = "actor_id",
  target = "target_id",
  time = "timestamp",
  event_type = NULL,
  weight = NULL,
  actor_group = NULL,
```

```
  target_group = NULL
)
```

Arguments

data	A data frame or tibble containing interaction events.
actor	Column name (character) for the source actor. Default "actor_id".
target	Column name (character) for the target actor. Default "target_id".
time	Column name (character) for the event timestamp. Must be POSIXct, POSIXt, or Date. Default "timestamp".
event_type	Optional column name (character) for interaction type (e.g., "follow", "reply"). If NULL, defaults to "interaction".
weight	Optional column name (character) for edge weight. If NULL, defaults to 1 for all events.
actor_group	Optional column name (character) for actor group membership (used in group disparity analyses).
target_group	Optional column name (character) for target group membership.

Value

A tibble of class social_events with standardized columns: actor_id, target_id, timestamp, event_type, weight, and optionally actor_group, target_group.

Examples

```
events <- data.frame(
  from   = c("u1", "u1", "u2", "u3"),
  to     = c("u2", "u3", "u3", "u4"),
  when   = as.POSIXct(c("2025-01-01", "2025-01-03",
                        "2025-01-04", "2025-01-06"))
)
ev <- as_social_events(events, actor = "from", target = "to", time = "when")
ev
```

---

audit_group_disparities

Audit network metrics by user group over time

Description

Computes per-group summaries of structural position (degree, betweenness) across all periods in a graph series. Useful for detecting systematic disparities in network access or visibility.

Usage

```
audit_group_disparities(
  data,
  graph_series,
  group_var = "actor_group",
  window = c("month", "week", "day", "quarter", "year")
)
```

Arguments

data	A standardized social event tibble with group membership columns.
graph_series	A social_graph_series.
group_var	Column name identifying actor groups. Default "actor_group".
window	Aggregation window. Default "month".

Value

A tibble with one row per period x group, including: mean_indegree, mean_outdegree, mean_betweenness, isolation_rate (proportion of isolated nodes), and n_users.

Examples

```
data(sim_social_events)
ev <- as_social_events(
  sim_social_events,
  actor_group  = "actor_group",
  target_group = "target_group"
)
gs <- build_graph_series(ev, window = "month")
audit_group_disparities(ev, gs)
```

---

build_graph_series *Build a time series of graph snapshots*

Description

Splits event data into non-overlapping time windows and builds one graph per window. Returns a named list of igraph objects.

Usage

```
build_graph_series(
  data,
  window = c("day", "week", "month", "quarter", "year"),
  directed = TRUE,
  weighted = TRUE,
  remove_self_loops = TRUE
)
```

Arguments

`data`	A standardized social event tibble (output of `as_social_events()`).
`window`	Aggregation window: one of `"day"`, `"week"`, `"month"`, `"quarter"`, or `"year"`. Default `"month"`.
`directed`	Logical; if `TRUE` (default) graphs are directed.
`weighted`	Logical; if `TRUE` (default) edges carry aggregated weights.
`remove_self_loops`	Logical; if `TRUE` (default) self-loops are removed.

Value

A named list of igraph objects of class `social_graph_series`. Names are ISO date strings representing the start of each period.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
length(gs)
names(gs)
```

`build_graph_snapshot` *Build a weighted graph snapshot from event data*

Description

Aggregates event-level interactions into a directed or undirected igraph object for a specified time window.

Usage

```
build_graph_snapshot(
  data,
  start = NULL,
  end = NULL,
  directed = TRUE,
  weighted = TRUE,
  remove_self_loops = TRUE
)
```

Arguments

data	A standardized social event tibble (output of as_social_events()).
start	Optional start date/time (inclusive). Events before this are excluded.
end	Optional end date/time (exclusive). Events from this point are excluded.
directed	Logical; if TRUE (default) the graph is directed.
weighted	Logical; if TRUE (default) edge weights reflect total event count between each pair.
remove_self_loops	Logical; if TRUE (default) self-loops are removed.

Value

An igraph object. Returns an empty graph if no events fall in the specified window.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
g  <- build_graph_snapshot(ev)
igraph::vcount(g)
igraph::ecount(g)
```

---

classify_user_roles *Classify structural roles of users in a graph*

Description

Assigns each node a structural role based on its in-degree, out-degree, and betweenness centrality relative to the rest of the network.

Usage

```
classify_user_roles(graph)
```

Arguments

graph	An igraph object.

Details

Role classification rules (applied in order):

1. **isolated** — degree = 0 in both directions.
2. **bridge** — betweenness in top quartile.
3. **core** — both in- and out-degree in top quartile.

4. **popular** — in-degree in top quartile, out-degree below.
5. **broadcaster** — out-degree in top quartile, in-degree below.
6. **peripheral** — all other nodes.

Value

A tibble with columns:

node Node name.

indegree In-degree.

outdegree Out-degree.

betweenness Betweenness centrality.

role Assigned structural role.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
g  <- build_graph_snapshot(ev)
classify_user_roles(g)
```

---

clustering_ts *Compute global clustering coefficient over time*

Description

The global clustering coefficient (transitivity) measures the tendency of nodes to form tightly connected triangles. High values indicate clique-like structure; low values indicate sparse or tree-like networks.

Usage

```
clustering_ts(graph_series)
```

Arguments

graph_series	A social_graph_series (output of build_graph_series()).

Value

A tibble with columns period and clustering. Returns NA for periods with fewer than 3 nodes.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
clustering_ts(gs)
```

community_drift *Compute community drift between adjacent time periods*

Description

Derives period-over-period changes in community structure metrics. Acts as a first-order approximation of how communities are evolving.

Usage

```
community_drift(community_tbl)
```

Arguments

community_tbl	Output of detect_communities_ts().

Value

The input tibble augmented with columns:

delta_n_communities Change in number of communities.

delta_modularity Change in modularity.

delta_largest_community Change in size of largest community.

Examples

```
data(sim_social_events)
ev  <- as_social_events(sim_social_events)
gs  <- build_graph_series(ev, window = "month")
ct  <- detect_communities_ts(gs)
community_drift(ct)
```

community_fragmentation_index *Community Fragmentation Index (CFI)*

Description

Computes a composite Community Fragmentation Index for each period, combining modularity, number of communities, and singleton prevalence into a single bounded score between 0 and 1. Higher values indicate a more fragmented, siloed network.

Usage

```
community_fragmentation_index(community_tbl)
```

Arguments

`community_tbl` Output of `detect_communities_ts()`.

Details

CFI is computed as:

$$CFI = 0.5 \cdot \hat{Q} + 0.3 \cdot \hat{C} + 0.2 \cdot \hat{S}$$

where $\hat{Q}$ is min-max scaled modularity, $\hat{C}$ is scaled community count, and $\hat{S}$ is singleton proportion.

Value

A tibble with columns `period` and `cfi`.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
ct <- detect_communities_ts(gs)
community_fragmentation_index(ct)
```

`creator_concentration` *Measure concentration of incoming attention over time*

Description

Computes how concentrated incoming interactions (in-degree) are across the network. High concentration indicates a few nodes receive most attention.

Usage

```
creator_concentration(graph_series, p = 0.1)
```

Arguments

`graph_series`	A `social_graph_series` (output of `build_graph_series()`).
`p`	Proportion of top actors to include in concentration measures. Default `0.10` (top 10%).

Value

A tibble with columns:

period Time period.

indegree_gini Gini coefficient of in-degree (0 = equal, 1 = monopoly).

top_p_share Share of total in-degree held by top-p actors.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
creator_concentration(gs)
```

`degree_inequality_ts` *Compute degree inequality over time*

Description

Uses the Gini coefficient of node degree as a measure of how unequally connections are distributed across the network. A Gini of 0 means perfect equality; 1 means one node holds all connections.

Usage

```
degree_inequality_ts(graph_series, mode = c("all", "in", "out"))
```

Arguments

`graph_series`	A `social_graph_series` (output of `build_graph_series()`).
`mode`	Degree mode: `"all"` (default), `"in"`, or `"out"`.

Value

A tibble with columns `period`, `degree_gini`, and `degree_mean`.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
degree_inequality_ts(gs)
```

detect_communities_ts *Detect communities over time using the Louvain method*

Description

Applies the Louvain community detection algorithm to each snapshot in a graph series. Directed graphs are coerced to undirected for community detection (standard practice for modularity-based methods).

Usage

```
detect_communities_ts(graph_series)
```

Arguments

graph_series	A social_graph_series (output of build_graph_series()).

Value

A tibble with columns:

period Time period (character).

n_communities Number of communities detected.

modularity Modularity score (higher = more modular community structure).

largest_community_size Size of the largest community.

singleton_count Number of isolated single-node communities.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
detect_communities_ts(gs)
```

engagement_gap *Compute engagement gap between user groups over time*

Description

Compares average in-degree (received attention) and out-degree (initiated interactions) between two user groups defined in the original event data.

Usage

```
engagement_gap(
  data,
  graph_series,
  group_var = "actor_group",
  window = c("month", "week", "day", "quarter", "year")
)
```

Arguments

data	A standardized social event tibble with actor_group and target_group columns (set via as_social_events()).
graph_series	A social_graph_series (output of build_graph_series()).
group_var	Column in data identifying actor groups. Default "actor_group".
window	Aggregation window matching the one used in graph_series. Default "month".

Value

A tibble with columns period, group, mean_indegree, mean_outdegree, and engagement_ratio (indegree / outdegree).

Examples

```
data(sim_social_events)
ev <- as_social_events(
  sim_social_events,
  actor_group  = "actor_group",
  target_group = "target_group"
)
gs  <- build_graph_series(ev, window = "month")
engagement_gap(ev, gs, group_var = "actor_group", window = "month")
```

---

network_density_ts *Compute network density over time*

Description

Network density is the proportion of possible edges that are present. High density indicates a highly connected network; low density indicates sparse interaction.

Usage

```
network_density_ts(graph_series)
```

Arguments

graph_series	A social_graph_series (output of build_graph_series()).

Value

A tibble with columns:

period Time period (character, ISO date).

n_nodes Number of active nodes.

n_edges Number of edges.

density Network density (0–1).

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
network_density_ts(gs)
```

---

network_drift *Compute the Network Drift Index (NDI) over time*

Description

The Network Drift Index quantifies how much a network's structure changes between consecutive time periods. It combines changes in edge turnover, degree distribution, community structure, and centralization into a single composite score.

Usage

```
network_drift(graph_series, w1 = 0.3, w2 = 0.25, w3 = 0.25, w4 = 0.2)
```

Arguments

graph_series	A social_graph_series (output of build_graph_series()).
w1	Weight for edge turnover. Default 0.30.
w2	Weight for degree distribution shift. Default 0.25.
w3	Weight for community structure change. Default 0.25.
w4	Weight for centralization change. Default 0.20.

Details

$$NDI_t = w_1\Delta_{edges} + w_2\Delta_{degree} + w_3\Delta_{community} + w_4\Delta_{centrality}$$

Default weights: w1 = 0.30, w2 = 0.25, w3 = 0.25, w4 = 0.20.

Component definitions:

edge_turnover Jaccard distance between edge sets of adjacent snapshots.

degree_shift Jensen-Shannon-like divergence of degree distributions.

modularity_change Absolute change in community modularity.

centralization_change Absolute change in degree centralization.

Value

A tibble with columns `period`, `edge_turnover`, `degree_shift`, `modularity_change`, `centralization_change`, and `ndi`.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
network_drift(gs)
```

---

network_drift_index *Compute the Network Drift Index (alias for network_drift)*

Description

Compute the Network Drift Index (alias for network_drift)

Usage

```
network_drift_index(graph_series, w1 = 0.3, w2 = 0.25, w3 = 0.25, w4 = 0.2)
```

Arguments

`graph_series`	A `social_graph_series` (output of `build_graph_series()`).
`w1`	Weight for edge turnover. Default `0.30`.
`w2`	Weight for degree distribution shift. Default `0.25`.
`w3`	Weight for community structure change. Default `0.25`.
`w4`	Weight for centralization change. Default `0.20`.

Value

See `network_drift()`.

plot_network_drift *Plot the Network Drift Index over time*

Description

Visualises the composite NDI and its four component scores across periods.

Usage

```
plot_network_drift(drift_tbl, show_components = TRUE)
```

Arguments

drift_tbl	Output of network_drift().
show_components	Logical; if TRUE (default), component scores are shown as a stacked area chart beneath the NDI line.

Value

A ggplot object.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
dt <- network_drift(gs)
plot_network_drift(dt)
```

plot_network_metrics *Plot one or more time-varying network metrics*

Description

Produces a line chart of a metric (or faceted chart for multiple metrics) from the output of metric functions such as network_density_ts(), reciprocity_ts(), or summarize_network_series().

Usage

```
plot_network_metrics(
  data,
  metric = NULL,
  title = "Network Metrics Over Time",
  colour = "#2c7bb6"
)
```

Arguments

data	A tibble with a period column and at least one numeric metric.
metric	Character vector of column name(s) to plot. If more than one, a faceted chart is produced. If NULL, all numeric columns except period are plotted.
title	Plot title. Default "Network Metrics Over Time".
colour	Line colour. Default "#2c7bb6".

Value

A ggplot object.

Examples

```
data(sim_social_events)
ev  <- as_social_events(sim_social_events)
gs  <- build_graph_series(ev, window = "month")
tbl <- network_density_ts(gs)
plot_network_metrics(tbl, metric = "density")
```

---

plot_role_trajectories

Plot role composition over time

Description

Shows how the proportion of each structural role changes across periods as a stacked bar chart.

Usage

```
plot_role_trajectories(role_tbl, type = c("stacked", "line"))
```

Arguments

role_tbl	Output of role_trajectories().
type	"stacked" (default) for a stacked proportional bar chart, or "line" for a line chart of role proportions.

Value

A ggplot object.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
rt <- role_trajectories(gs)
plot_role_trajectories(rt)
```

`reciprocity_ts` *Compute reciprocity over time*

Description

Reciprocity is the proportion of edges that have a mutual counterpart. Applies to directed graphs only. High reciprocity suggests mutual engagement; low reciprocity suggests broadcast-like interaction.

Usage

```
reciprocity_ts(graph_series)
```

Arguments

`graph_series`	A `social_graph_series` (output of `build_graph_series()`).

Value

A tibble with columns `period` and `reciprocity`. Returns `NA` for undirected graphs or periods with no edges.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
reciprocity_ts(gs)
```

`role_mobility_index` *Role Mobility Index (RMI)*

Description

Quantifies how much users move between structural roles across time periods. A high RMI indicates a dynamic network where role transitions are frequent; a low RMI indicates structural stability.

Usage

```
role_mobility_index(role_tbl)
```

Arguments

`role_tbl`	Output of `role_trajectories()`.

Details

For each user observed in at least two consecutive periods, the RMI counts the proportion of adjacent-period pairs where the role changed. The overall RMI is the mean across all such users.

Value

A tibble with columns:

rmi_global Overall Role Mobility Index (0–1).

n_users_tracked Number of users with >= 2 observed periods.

mean_transitions Mean number of role transitions per user.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
rt <- role_trajectories(gs)
role_mobility_index(rt)
```

role_trajectories *Track user structural roles over time*

Description

Applies `classify_user_roles()` to every snapshot in a graph series and returns a longitudinal tibble of role assignments.

Usage

```
role_trajectories(graph_series)
```

Arguments

`graph_series`	A `social_graph_series` (output of `build_graph_series()`).

Value

A tibble with columns `period`, `node`, `indegree`, `outdegree`, `betweenness`, and `role`.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
role_trajectories(gs)
```

sim_social_events *Simulated social interaction event log*

Description

A synthetic dataset of 591 user-to-user interaction events spanning January through June 2025. Designed for demonstrating the **socialdrift** workflow. Includes group membership columns for disparity analyses.

Format

A data frame with 591 rows and 7 variables:

actor_id Source user ID (character, `"u1"` to `"u60"`).

target_id Target user ID (character).

timestamp Event timestamp (POSIXct, UTC).

event_type Type of interaction: `"follow"`, `"reply"`, `"mention"`, `"like"`, or `"repost"`.

weight Interaction weight (integer, always 1 in this dataset).

actor_group Group membership of the actor: `"A"`, `"B"`, or `"C"`.

target_group Group membership of the target: `"A"`, `"B"`, or `"C"`.

Details

The dataset was generated with preferential attachment: earlier-indexed users have slightly higher probability of being selected as targets, creating realistic degree inequality. Event types are sampled with probabilities approximating typical social platform distributions.

Source

Simulated data. See `data-raw/sim_social_events.R` for the generation script.

Examples

```
data(sim_social_events)
head(sim_social_events)
table(sim_social_events$event_type)
```

---

summarize_network_series

Summarize structural metrics across all periods in a graph series

---

Description

A convenience wrapper that computes density, reciprocity, clustering, and degree inequality for every period and returns a single wide tibble.

Usage

```
summarize_network_series(graph_series)
```

Arguments

`graph_series`	A `social_graph_series` (output of `build_graph_series()`).

Value

A tibble with one row per period and columns for all key metrics.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
summarize_network_series(gs)
```

validate_social_events

Validate a standardized social event table

Description

Checks that a social events tibble contains required columns with valid values. Called automatically by `as_social_events()`.

Usage

```
validate_social_events(data)
```

Arguments

`data`	A standardized social event tibble (output of `as_social_events()`).

Value

The validated tibble, invisibly.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
validate_social_events(ev)
```

visibility_concentration_index

Visibility Concentration Index (VCI)

Description

A composite index measuring whether visibility (incoming attention) is concentrated in a small fraction of users. Combines the Gini coefficient and top-share into a single interpretable score.

Usage

```
visibility_concentration_index(graph_series)
```

Arguments

`graph_series`	A `social_graph_series` (output of `build_graph_series()`).

Details

$$VCI = 0.6 \cdot Gini + 0.4 \cdot top\text{-}10\%\text{-}share$$

Values near 1 indicate extreme concentration; values near 0 indicate roughly equal attention distribution.

Value

A tibble with columns `period` and `vci`.

Examples

```
data(sim_social_events)
ev <- as_social_events(sim_social_events)
gs <- build_graph_series(ev, window = "month")
visibility_concentration_index(gs)
```

Index