

Package: birddog (via r-universe)

September 30, 2026

Title Sniffing Emergence and Trajectories in Academic Papers and Patents

Version 2.0.0

Description Provides a unified set of methods to detect scientific emergence and technological trajectories in academic papers and patents. The package combines citation network analysis with community detection and attribute extraction, also applying natural language processing (NLP) and structural topic modeling (STM) to uncover the contents of research communities. It implements metrics and visualizations of community trajectories, including novelty indicators, citation cycle time, and main path analysis, allowing researchers to map and interpret the dynamics of emerging knowledge fields. Applications of the method include: Souza et al. (2022) [doi:10.1002/bbb.2441](https://doi.org/10.1002/bbb.2441), Souza et al. (2022) [doi:10.14211/ibjesb.e1742](https://doi.org/10.14211/ibjesb.e1742), Matos et al. (2023) [doi:10.1007/s43938-023-00036-3](https://doi.org/10.1007/s43938-023-00036-3), Maria et al. (2023) [doi:10.3390/su15020967](https://doi.org/10.3390/su15020967), Biazatti et al. (2024) [doi:10.1016/j.envdev.2024.101074](https://doi.org/10.1016/j.envdev.2024.101074), Felizardo et al. (2025) [doi:10.1007/s12649-025-03136-z](https://doi.org/10.1007/s12649-025-03136-z), and Miranda et al. (2025) [doi:10.1016/j.ijhydene.2025.01.089](https://doi.org/10.1016/j.ijhydene.2025.01.089).

License GPL-3

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports dplyr, ggraph, ggplot2, plotly, igraph, tidygraph, tidyr, tibble, Matrix, purrr, readr, rlang, glue, openalexR, RColorBrewer, scales, stringr

VignetteBuilder knitr

Suggests benchmarkme, knitr, rmarkdown, cli, ggHoriPlot, gganimate, ggrepel, ggthemes, janitor, gt, testthat (>= 3.0.0), tictoc, viridis, zoo, stm, tidytext, udpipe

Config/testthat/edition 3

Depends R (>= 4.1.0)

URL <https://roneyfraga.com/birddog/>,
<https://github.com/roneyfraga/birddog>

BugReports <https://github.com/roneyfraga/birddog/issues>

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libuv1-dev libxml2-dev libssl-dev libx11-dev

Repository <https://roneyfraga.r-universe.dev>

Date/Publication 2026-06-23 17:31:21 UTC

RemoteUrl <https://github.com/roneyfraga/come-to-telegram-rickluminari1--birddog>

RemoteRef HEAD

RemoteSha ed7b6b4f6ab38e4ce012e18adb9952c9327a281f

Contents

birddog-package	3
data_state_thresholds	5
fixed_state_thresholds	6
get_openalex_fields	6
is_flow	8
mixed_sort	9
plot_groups_influence_matrix	10
plot_groups_influence_network	11
plot_groups_lineage_2d	13
plot_groups_lineage_3d	15
plot_groups_map	17
plot_groups_map_animation	18
plot_groups_map_interactive	20
plot_groups_per_year	21
plot_trajectory_confluence	23
plot_trajectory_confluence_interactive	27
plot_trajectory_confluence_matrix	29
plot_trajectory_dag	31
plot_trajectory_dag_interactive	34
plot_trajectory_dispersion	36
plot_trajectory_dynamics	37
plot_trajectory_dynamics_interactive	39
plot_trajectory_formation	41
plot_trajectory_lines_2d	43
plot_trajectory_lines_3d	46
read_openalex	49
read_wos	50
sniff_cct	52
sniff_components	55

sniff_entropy	56
sniff_groups	59
sniff_groups_attributes	60
sniff_groups_cumulative	62
sniff_groups_cumulative_citations	64
sniff_groups_hubs	65
sniff_groups_influence	67
sniff_groups_keywords	69
sniff_groups_lineage	70
sniff_groups_stm_prepare	71
sniff_groups_stm_run	72
sniff_groups_terms	73
sniff_key_route	75
sniff_network	77
sniff_trajectory_braid	78
sniff_trajectory_cct	79
sniff_trajectory_channel	80
sniff_trajectory_coherence	81
sniff_trajectory_community	82
sniff_trajectory_comparison	83
sniff_trajectory_confluence	84
sniff_trajectory_contribution	85
sniff_trajectory_dag	86
sniff_trajectory_destination	87
sniff_trajectory_dynamics	88
sniff_trajectory_emergence_owners	90
sniff_trajectory_entropy	91
sniff_trajectory_formation	92
sniff_trajectory_hubs	93
sniff_trajectory_self_sufficiency	94
subset.birddog_flow	95
validate_flow	96

Index	97
--------------	-----------

birddog-package	<i>birddog: sniffing emergence and trajectories in academic papers and patents</i>
-----------------	--

Description

Tools to detect emergence and trace technological/scientific trajectories in papers and patents. It reads OpenAlex and Web of Science data, builds citation-based networks, identifies groups, and summarizes their dynamics.

Links

- Website: <http://roneyfraga.com/birddog/>
- GitHub: <https://github.com/roneyfraga/birddog>
- Issues: <https://github.com/roneyfraga/birddog/issues>

Pipeline

`read_openalex()/read_wos() -> sniff_network() -> sniff_components() -> sniff_groups_cumulative() -> sniff_groups_lineage() -> sniff_trajectory_dag() -> sniff_trajectory_braid() -> analysis(sniff_trajectory_confluence(), _formation(), _destination(), _contribution(), _self_sufficiency(), _cct(), _entropy(), _hubs(), _dynamics())`, each paired with a `plot_*` function. Focus an analysis with `subset(flow, ...)`. "Flow" names the object kind, not an algorithm (stock vs flow: groups are stock, trajectories are flow): every detector returns a flow, and alternative detectors (`sniff_trajectory_<algo>()`) return the same contract, checked by `validate_flow()`.

Label grammar

`cNgN` is a group (component N, group N); `y<YYYY><cNgN>` is a group-year node; `tr: :<cNgN>` is a central trajectory (reaches the last year, one per final group); `tr1..trN` are absorbed trajectories. This grammar is part of the public API and only changes at major versions.

Theoretical background

Trajectories are detected as system-level objects (Dosi, 1982): disjoint chains in the temporal DAG of cumulative clusterings. A trajectory that stops being detected loses its identity, not its papers; tracking its terminal cohort to the last year classifies the outcome as convergence (one destination), divergence (many), or dormancy (none) — with emergence indicators on the living chains (Rotolo et al., 2015; Carley et al., 2017).

Author(s)

Maintainer: Roney Fraga Souza <roneyfraga@gmail.com> ([ORCID](#)) [copyright holder]

Other contributors:

- Luis Felipe de Souza Rodrigues <lfsouza25@gmail.com> [contributor]

See Also

Useful links:

- Report bugs at <https://github.com/roneyfraga/birddog/issues>

data_state_thresholds *Data-driven, outlier-robust dynamic-state thresholds*

Description

Derives a `fixed_state_thresholds()`-shaped list from an observed `sniff_trajectory_dynamics()` table, so the cut points adapt to the dataset instead of fixed constants. The growth cuts are placed k robust deviations (median absolute deviation) from the median, which resists outliers such as a single fast-growing young core; the bounded metrics use their median as the neutral split, and dormancy_share keeps the absolute majority rule (its distribution is usually too concentrated at zero to standardize).

Usage

```
data_state_thresholds(dyn, k = 1)
```

Arguments

<code>dyn</code>	A <code>sniff_trajectory_dynamics()</code> tibble; only its metric columns are read, so the classification it currently carries is irrelevant.
<code>k</code>	Number of robust deviations (scaled MAD) above/below the median for the growth cuts (default 1): <code>emergence_growth</code> = median + $k \times \text{MAD}$ and <code>decline_growth</code> = median - $k \times \text{MAD}$ over the living cores' <code>growth_rate</code> . Larger k widens the maturity band (fewer emergence / dormancy calls).

Details

Use it as a drop-in for the thresholds argument; the metrics are threshold-independent, so one pass is enough to derive and re-classify: `dyn <- sniff_trajectory_dynamics(flow)`, `th <- data_state_thresholds(dyn)`, `dyn <- sniff_trajectory_dynamics(flow, thresholds = th)`.

Value

A list shaped like `fixed_state_thresholds()`: `emergence_growth`, `decline_growth` (robust, from the centrals' growth), `emergence_novelty` (median novelty of the centrals), `convergence_entropy` (median destination entropy of the absorbed), and `dormancy_share` (the default majority rule). Any quantity that cannot be derived (too few points, zero spread) falls back to its `fixed_state_thresholds()` value.

See Also

`fixed_state_thresholds()`, `sniff_trajectory_dynamics()`, `plot_trajectory_dynamics()`

fixed_state_thresholds

Fixed-constant thresholds for dynamic-state classification

Description

The fixed knobs for `sniff_trajectory_dynamics()`. These are **not** the applied default: by default the cut points are derived from the data via `data_state_thresholds()`, and these constants serve as the explicit fixed-constant option (pass this list as `thresholds`) and as the fallback for any quantity that cannot be derived. The classification follows two axes (see `sniff_trajectory_dynamics()`): a life-cycle **phase** for living (central) trajectories and a terminal **fate** for declining (absorbed) ones.

Usage

```
fixed_state_thresholds()
```

Value

A list: `emergence_growth` (min annualized `growth_rate` for a central to count as emergence; default 0.15), `emergence_novelty` (min novelty for emergence; default 0.30), `decline_growth` (a central at or below this `growth_rate` is dormant by stall; default -0.05), `convergence_entropy` (an absorbed trajectory whose normalized destination entropy is below this converged, else diverged; default 0.5), `dormancy_share` (an absorbed trajectory whose terminal cohort drops at least this share is dormant by death; default 0.5).

get_openalex_fields *Get Fields from OpenAlex for Work IDs*

Description

Retrieves specified fields for OpenAlex work IDs using the OpenAlex API. Processes data in batches to avoid API rate limits.

Usage

```
get_openalex_fields(
  openalex_ids,
  variables = "publication_year",
  batch_size = 50,
  save_dir = NULL
)
```

Arguments

openalex_ids	Character vector of OpenAlex work IDs (format: "W1234567890") or a data frame/tibble containing a column named "CR" with OpenAlex IDs. IDs can be semicolon-separated strings which will be split automatically.
variables	Character vector of variable names to fetch from OpenAlex. Options include: "publication_year", "doi", "type", "source_display_name", or any valid OpenAlex work field. Default is "publication_year".
batch_size	Number of IDs to process per API call (default: 50). Smaller batches help avoid API rate limits.
save_dir	Optional path to directory where intermediate results should be saved as RDS files. If NULL (default), no saving occurs. Directory will be created if it doesn't exist.

Details

This function:

1. Accepts either a character vector of IDs or a data frame with a "CR" column
2. Splits semicolon-separated ID strings into individual IDs
3. Validates IDs against the pattern "^W\d+\$"
4. Fetches specified variables from OpenAlex API in batches
5. Optionally saves each batch to disk as it's processed
6. Handles API errors gracefully with informative messages
7. Includes delays between batches to respect API rate limits

Value

A tibble with the following columns:

- id: The OpenAlex work ID
- One column for each requested variable (e.g., "publication_year", "doi", "type")

Rows without valid OpenAlex IDs or where API calls fail will have NA values.

Note

The OpenAlex API has rate limits. This function implements:

- Batch processing to reduce number of API calls
- 0.5 second delays between batches
- Error handling for failed API requests
- Progress messages to track execution
- Optional disk saving for data persistence

If you encounter rate limiting errors, consider reducing batch_size or implementing longer delays.

Examples

```
## Not run:
# From a character vector
ids <- c("W2261389918", "W1548650423", "W1504492735")
result <- get_openalex_fields(ids)

# Fetch multiple variables
result <- get_openalex_fields(
  ids,
  variables = c("publication_year", "doi", "type")
)

# From a data frame with CR column
oa_data <- data.frame(CR = c("W123;W456", "W789"))
result <- get_openalex_fields(oa_data)

# Save intermediate results while downloading
result <- get_openalex_fields(
  ids,
  variables = c("publication_year", "source_display_name"),
  save_dir = tempdir()
)

## End(Not run)
```

is_flow

Is this a trajectory flow object?

Description

"Flow" names the object kind, not the algorithm (stock vs flow: groups are stock, trajectories are flow). Every trajectory detector returns a flow – the temporal decomposition of the DAG into trajectories plus the confluence tree – so this predicate accepts the result of [sniff_trajectory_braid\(\)](#) and of any future `sniff_trajectory_<algo>()` detector. `is_flow()` is the cheap predicate; [validate_flow\(\)](#) is the contract's authoritative checker.

Usage

```
is_flow(x)
```

Arguments

x Any object.

Value

A length-one logical.

See Also

[sniff_trajectory_braid\(\)](#), [validate_flow\(\)](#)

Other trajectory detection: [sniff_trajectory_braid\(\)](#), [sniff_trajectory_channel\(\)](#), [sniff_trajectory_dag\(\)](#), [subset.birddog_flow\(\)](#), [validate_flow\(\)](#)

mixed_sort

Natural sort for alphanumeric strings

Description

Sort character vectors containing embedded numbers in natural numeric order rather than lexicographic order. For example, "c1g2" comes before "c1g10", which standard [sort\(\)](#) would not guarantee.

Usage

```
mixed_sort(x)
```

Arguments

x Character vector to sort.

Details

The function extracts all numeric segments from each string using [stringr::str_extract_all\(\)](#) and sorts by them sequentially (first number, then second). Designed for two-segment identifiers such as "c1g1", "c2g10", etc.

Value

A character vector sorted in natural numeric order.

See Also

[sort\(\)](#) for standard lexicographic sorting.

Examples

```
mixed_sort(c("c1g10", "c1g2", "c1g1", "c1g9"))
```

```
mixed_sort(c("group12", "group1", "group3"))
```

plot_groups_influence_matrix

Plot the directed group-influence matrix (heatmap)

Description

Experimental. The directed counterpart of `plot_trajectory_confluence_matrix()`: a heatmap of the cross-citation channels from a `sniff_groups_influence()` object. **Rows are the citing group** ("is influenced by") and **columns the cited group** ("the influencer"), so a cell and its mirror across the diagonal are the two directions of one pair. Confluence is symmetric; influence is not, and that broken symmetry is the bidirectional relationship.

Usage

```
plot_groups_influence_matrix(
  influence,
  fill = c("surprise", "raw", "debt", "audience", "salton"),
  diagonal = c("mute", "show", "drop"),
  show_values = TRUE,
  label_size = 4,
  order_by = c("group", "balance"),
  axis_text_size = NULL,
  axis_title_size = NULL,
  x_angle = 0,
  legend_position = "right",
  bg = "white",
  title = NULL
)
```

Arguments

influence	A <code>sniff_groups_influence()</code> object.
fill	Which value to colour by: "surprise" (default, the flow against the size null, on a diverging scale centred at 1), "raw" (citation counts), "debt", "audience" or "salton".
diagonal	How to treat the intra-group cohesion cells: "mute" (default, drawn grey, off the colour scale, so the between-group channels keep the contrast), "show" (on the scale like any cell) or "drop" (not drawn).
show_values	Print the value inside each cell (default TRUE).
label_size	Font size of the in-cell values (default 4).
order_by	Group order on both axes: "group" (default, <code>mixed_sort()</code>) or "balance" (sources first, sinks last).
axis_text_size, axis_title_size	Font sizes of the tick labels and axis titles; NULL (default) keeps the theme defaults.

x_angle	Rotation in degrees of the column (cited-group) labels on the top axis. Default 0 (horizontal); 90 stacks the cNgN ids vertically so they stay legible on a small figure with many groups.
legend_position	Legend placement: "right" (default), "left", "top", "bottom" or "none".
bg	Panel/plot background colour (default "white").
title	Plot title; NULL (default) removes it.

Value

A ggplot object.

See Also

`sniff_groups_influence()`, `plot_groups_influence_network()`, `plot_trajectory_confluence_matrix()`

Other visualization: `plot_groups_influence_network()`, `plot_groups_lineage_2d()`, `plot_groups_lineage_3d()`, `plot_groups_map()`, `plot_groups_map_animation()`, `plot_groups_map_interactive()`, `plot_groups_per_year()`, `plot_trajectory_confluence()`, `plot_trajectory_confluence_interactive()`, `plot_trajectory_confluence_map()`, `plot_trajectory_dag()`, `plot_trajectory_dag_interactive()`, `plot_trajectory_dispersion()`, `plot_trajectory_dynamics()`, `plot_trajectory_dynamics_interactive()`, `plot_trajectory_formation()`, `plot_trajectory_lines_2d()`, `plot_trajectory_lines_3d()`

Examples

```
## Not run:
infl <- sniff_groups_influence(groups)
plot_groups_influence_matrix(infl)
plot_groups_influence_matrix(infl, fill = "debt", diagonal = "drop")

## End(Not run)
```

`plot_groups_influence_network`

Plot the group-influence network (the net-influence spine)

Description

Experimental. Draws the directed influence backbone from a `sniff_groups_influence()` object as a node-link graph: an arrow runs from the **source** (the cited, more foundational group) to the **recipient** (the citing group). By default the edges are the net flow $\nu_{ij} = C_{ij} - C_{ji}$, the "who, on balance, leads whom" spine; nodes are coloured by role (source / broker / sink) and sized by citation activity.

Usage

```
plot_groups_influence_network(
  influence,
  weight = c("net", "gross", "surprise"),
  min_weight = 0,
  node_size = c("equal", "io", "balance"),
  colour_role = TRUE,
  edge_labels = TRUE,
  edge_digits = NULL,
  labels = TRUE,
  label_size = NULL,
  layout = "sugiyama",
  title = NULL
)
```

Arguments

influence	A sniff_groups_influence() object.
weight	Which edges and weights to draw: "net" (default, the net-flow spine \$net, one arrow per connected pair), "gross" (every directed channel, weighted by citation count), or "surprise" (every directed channel, weighted by the size-null surprise).
min_weight	Drop edges whose weight is below this (default 0).
node_size	How to size the group nodes: "equal" (default), "io" (citations made plus received) or "balance" (the absolute balance $ \beta $).
colour_role	Colour nodes by their source / broker / sink role (default TRUE).
edge_labels	Print the weight on each edge (default TRUE).
edge_digits	Decimal places for the edge weight labels; NULL (default) shows integers for "net"/"gross" and two decimals for the continuous "surprise" weight.
labels	Print the group label on each node (default TRUE).
label_size	Font size of the node labels; NULL (default) auto-sizes the nodes and the text so even the smallest node holds its group id.
layout	A ggraph/igraph layout name (default "sugiyama", the layered DAG layout).
title	Plot title; NULL (default) removes it.

Value

A ggplot object.

See Also

[sniff_groups_influence\(\)](#), [plot_groups_influence_matrix\(\)](#)

Other visualization: [plot_groups_influence_matrix\(\)](#), [plot_groups_lineage_2d\(\)](#), [plot_groups_lineage_3d\(\)](#), [plot_groups_map\(\)](#), [plot_groups_map_animation\(\)](#), [plot_groups_map_interactive\(\)](#), [plot_groups_per_year\(\)](#), [plot_trajectory_confluence\(\)](#), [plot_trajectory_confluence_interactive\(\)](#), [plot_trajectory_confluence_ma](#)

```
plot_trajectory_dag(), plot_trajectory_dag_interactive(), plot_trajectory_dispersion(),
plot_trajectory_dynamics(), plot_trajectory_dynamics_interactive(), plot_trajectory_formation(),
plot_trajectory_lines_2d(), plot_trajectory_lines_3d()
```

Examples

```
## Not run:
infl <- sniff_groups_influence(groups)
plot_groups_influence_network(infl)
plot_groups_influence_network(infl, weight = "gross", min_weight = 50)

## End(Not run)
```

plot_groups_lineage_2d

Visualize 2D Technological Trajectories from Group Evolution

Description

Creates a 2D visualization of technological trajectories based on group similarity metrics, showing the evolution of research groups over time with node size representing group importance and color representing publication-year deviation.

Usage

```
plot_groups_lineage_2d(
  groups_lineage,
  group = "c1g1",
  jaccard_similarity = 0.01,
  prop_tracked_intra_group_treshold = 0.2,
  label_type = "size",
  label_vertical_position = 0,
  label_horizontal_position = 0,
  label_angle = 0,
  time_span = NA,
  show_legend = TRUE,
  color_by = c("py_deviation", "final_group_share")
)
```

Arguments

groups_lineage A list with components `groups_similarity` and `groups_attributes`, typically produced by `sniff_groups_lineage()`. The `groups_similarity` element must be a named list of edge tables (one per group) with at least `from`, `to`, and `weight`; the `groups_attributes` element must be a named list of node tables containing, among others, `network_until`, `quantity_papers`, `prop_tracked_intra_group`, `tracked_documents`, and `PY.sd`.

<code>group</code>	The specific group to visualize (default: "c1g1").
<code>jaccard_similarity</code>	Minimum Jaccard similarity threshold for connections (default: 0.1).
<code>prop_tracked_intra_group_treshold</code>	Minimum proportion of tracked intra-group documents for nodes to be included (default: 0.2).
<code>label_type</code>	Type of labels to display on nodes ("size" for weighted size or "id" for group IDs).
<code>label_vertical_position</code>	Kept for backward compatibility; no longer applied, as labels are auto-positioned (repelled) with a white halo so they stay readable over dark nodes and do not overlap.
<code>label_horizontal_position</code>	Kept for backward compatibility; not applied (see <code>label_vertical_position</code>).
<code>label_angle</code>	Kept for backward compatibility; not applied (labels are auto-positioned).
<code>time_span</code>	Optional vector of years to display; if NA, shows all (default: NA).
<code>show_legend</code>	Logical indicating whether to show the color legend (default: TRUE).
<code>color_by</code>	What the node color encodes: "py_deviation" (default) colors by the average publication-year deviation (PY. sd); "final_group_share" colors each node by the share of its papers that belong to group's last-year cluster (a plasma palette, 0-100\ showing how much of each earlier cluster funnels into the final group. The latter requires <code>docs_per_group</code> in <code>groups_lineage</code>).

Details

A descriptive stock view of one group's lineage network. A group's lineage can share year communities with other groups (soft membership), so this view makes no trajectory claims – for detection see [sniff_trajectory_braid\(\)](#).

Value

A `ggplot2` object visualizing the technological trajectories.

See Also

Other visualization: [plot_groups_influence_matrix\(\)](#), [plot_groups_influence_network\(\)](#), [plot_groups_lineage_3d\(\)](#), [plot_groups_map\(\)](#), [plot_groups_map_animation\(\)](#), [plot_groups_map_interactive\(\)](#), [plot_groups_per_year\(\)](#), [plot_trajectory_confluence\(\)](#), [plot_trajectory_confluence_interactive\(\)](#), [plot_trajectory_confluence_matrix\(\)](#), [plot_trajectory_dag\(\)](#), [plot_trajectory_dag_interactive\(\)](#), [plot_trajectory_dispersion\(\)](#), [plot_trajectory_dynamics\(\)](#), [plot_trajectory_dynamics_interactive\(\)](#), [plot_trajectory_formation\(\)](#), [plot_trajectory_lines_2d\(\)](#), [plot_trajectory_lines_3d\(\)](#)

Examples

```
## Not run:
# Compute trajectories first
traj_data <- sniff_groups_lineage(groups_cumulative)
```

```
# Visualize a specific group (pass the whole object; the function extracts what it needs internally)
plot_groups_lineage_2d(
  groups_lineage = traj_data,
  group = "c1g5",
  jaccard_similarity = 0.3
)

## End(Not run)
```

plot_groups_lineage_3d

Visualize 3D Technological Trajectories from Group Evolution

Description

Creates an interactive 3D visualization of technological trajectories showing the evolution of research groups over time with node size representing group importance and color representing publication year deviation.

Usage

```
plot_groups_lineage_3d(
  groups_lineage,
  group = "component1_g01",
  jaccard_similarity = 0.1,
  prop_tracked_intra_group_threshold = 0.2,
  label_type = "size",
  label_vertical_position = 0,
  label_horizontal_position = 0,
  label_angle = 0,
  time_span = NA,
  show_legend = TRUE,
  last_year_keywords = NULL,
  log_scale = FALSE
)
```

Arguments

groups_lineage A list containing two components:

- **groups_similarity**: Similarity data between groups
- **groups_attributes**: Attribute data for each group

group The specific group to visualize (default: "component1_g01")

jaccard_similarity Minimum Jaccard similarity threshold for connections (default: 0.1)

prop_tracked_intra_group_treshold	Minimum proportion of tracked intra-group documents for nodes to be included (default: 0.2)
label_type	Type of labels to display on nodes ("size" for weighted size or "id" for group IDs)
label_vertical_position	Vertical adjustment for node labels (default: 0)
label_horizontal_position	Horizontal adjustment for node labels (default: 0)
label_angle	Angle for node labels (default: 0)
time_span	Optional vector specifying the time span to display (default: NA shows all years)
show_legend	Logical indicating whether to show the color legend (default: TRUE)
last_year_keywords	Optional keywords description for the last year (default: NULL)
log_scale	Whether to apply log transformation to the z-axis size values (default: FALSE). Uses <code>log1p()</code> (i.e., $\log(1 + x)$) to compress large differences between node sizes.

Details

A descriptive stock view of one group's lineage network. A group's lineage can share year communities with other groups (soft membership), so this view makes no trajectory claims – for detection see [sniff_trajectory_braid\(\)](#).

Value

A plotly 3D visualization object

See Also

Other visualization: [plot_groups_influence_matrix\(\)](#), [plot_groups_influence_network\(\)](#), [plot_groups_lineage_2d\(\)](#), [plot_groups_map\(\)](#), [plot_groups_map_animation\(\)](#), [plot_groups_map_interactive\(\)](#), [plot_groups_per_year\(\)](#), [plot_trajectory_confluence\(\)](#), [plot_trajectory_confluence_interactive\(\)](#), [plot_trajectory_confluence_matrix\(\)](#), [plot_trajectory_dag\(\)](#), [plot_trajectory_dag_interactive\(\)](#), [plot_trajectory_dispersion\(\)](#), [plot_trajectory_dynamics\(\)](#), [plot_trajectory_dynamics_interactive\(\)](#), [plot_trajectory_formation\(\)](#), [plot_trajectory_lines_2d\(\)](#), [plot_trajectory_lines_3d\(\)](#)

Examples

```
## Not run:
# First get trajectory data
traj_data <- sniff_groups_lineage(groups_cumulative)

# Visualize a specific group in 3D
plot_groups_lineage_3d(
  groups_lineage = traj_data,
  group = "component1_g05",
  jaccard_similarity = 0.2
```

```
)
## End(Not run)
```

plot_groups_map	<i>Plot the stock map: the complete citation network coloured by final group</i>
-----------------	--

Description

The birddog stock map: every paper in the cumulative network is a point, laid out by a force-directed layout of the citation network and coloured by its final group, with a text label at each group's centroid. It is the stock snapshot – all final groups, the whole network at once – positioned by citation structure rather than text embeddings.

Usage

```
plot_groups_map(
  groups_cumulative,
  labels_text = NULL,
  network_until = NULL,
  layout = "drl",
  seed = 888L,
  point_size = 0.9,
  point_alpha = 0.7,
  show_unassigned = TRUE,
  label = TRUE,
  label_size = 3.2,
  title = "Group map",
  title_size = 14,
  normalize = FALSE,
  normalize_q = 0.99
)
```

Arguments

groups_cumulative	A stock object from <code>sniff_groups_cumulative()</code> ; each <code>network_until_<year></code> element must carry <code>\$network</code> (an <code>igraph/tbl_graph</code> whose vertices have a <code>name</code> attribute) and <code>\$documents</code> (with columns <code>name</code> , <code>group</code>).
labels_text	Optional <code>data.frame</code> with columns <code>id</code> (matching the group ids, e.g. "c1g1") and <code>text</code> (the human description). <code>NULL</code> (default) uses the bare group id as the label. If no <code>id</code> matches any group, a warning is issued and the group ids are used.
network_until	Snapshot year to map. <code>NULL</code> (default) uses the last element (the final groups).

layout	Force-directed layout passed to <code>ggraph::create_layout()</code> (default "dr1", the OpenOrd layout, which separates communities into legible territories on dense citation networks; "fr" and "kk" tend to collapse a large network into a single hairball).
seed	Integer seed for the stochastic layout (default 888L).
point_size	Node point size (default 0.9).
point_alpha	Node point alpha (default 0.7).
show_unassigned	Draw nodes with no group (NA) as grey background (default TRUE); FALSE drops them.
label	Draw the group text labels at centroids (default TRUE).
label_size	Font size of the centroid labels (default 3.2).
title	Plot title: the default "Group map", any string, or NULL for no title.
title_size	Title font size (default 14).
normalize	Rescale the layout for cross-year comparison (default FALSE). When TRUE, coordinates are centred on the median and divided by the robust radius (the <code>normalize_q</code> quantile of the distance to the centre), then framed with a fixed unit window, so the dense core fills the plot identically across snapshots. This equalises scale only, not orientation: the force-directed layout assigns each graph an arbitrary rotation, so node positions are not correspondable across years.
normalize_q	Quantile of the radial distance kept inside the normalised window (default 0.99); only used when <code>normalize = TRUE</code> .

Value

A ggplot object.

See Also

`sniff_groups_cumulative()`, `plot_trajectory_dag()`

Other visualization: `plot_groups_influence_matrix()`, `plot_groups_influence_network()`, `plot_groups_lineage_2d()`, `plot_groups_lineage_3d()`, `plot_groups_map_animation()`, `plot_groups_map_interactive()`, `plot_groups_per_year()`, `plot_trajectory_confluence()`, `plot_trajectory_confluence_interactive()`, `plot_trajectory_confluence_matrix()`, `plot_trajectory_dag()`, `plot_trajectory_dag_interactive()`, `plot_trajectory_dispersion()`, `plot_trajectory_dynamics()`, `plot_trajectory_dynamics_interactive()`, `plot_trajectory_formation()`, `plot_trajectory_lines_2d()`, `plot_trajectory_lines_3d()`

`plot_groups_map_animation`

Animate the stock map: the citation network forming year by year

Description

One layout of the final network, with every paper revealed at the year it first enters the cumulative network and held at its final position, so the cloud grows without nodes moving. Returns a `gganim` object; render it to a GIF/MP4 with `gganimate::animate()` and `gganimate::anim_save()`. Reveal and colour reuse the static `plot_groups_map()` engine.

Usage

```
plot_groups_map_animation(
  groups_cumulative,
  labels_text = NULL,
  layout = "dr1",
  seed = 888L,
  show_unassigned = TRUE,
  label = TRUE,
  label_size = 3.2,
  title_size = 14,
  point_size = 0.9
)
```

Arguments

<code>groups_cumulative</code>	A stock object from <code>sniff_groups_cumulative()</code> .
<code>labels_text</code>	Optional <code>data.frame</code> with columns <code>id</code> and <code>text</code> ; <code>NULL</code> (default) uses the bare group <code>id</code> . If no <code>id</code> matches any group, a warning is issued and the group <code>ids</code> are used.
<code>layout</code>	Force-directed layout passed to <code>ggraph::create_layout()</code> (default <code>"dr1"</code>).
<code>seed</code>	Integer seed for the stochastic layout (default <code>888L</code>).
<code>show_unassigned</code>	Reveal nodes with no group (<code>NA</code>) as grey (default <code>TRUE</code>).
<code>label</code>	Draw persistent group labels at the final centroids (default <code>TRUE</code>).
<code>label_size</code>	Font size of the centroid labels (default <code>3.2</code>).
<code>title_size</code>	Font size of the per-frame <code>"Year: ..."</code> title (default <code>14</code>).
<code>point_size</code>	Node point size (default <code>0.9</code>).

Details

Rendering: the transition is one frame per year. Render with `duration` (not `fps`), which works with either renderer: `gganimate::animate(anim, duration = 8)` gives `nframes/duration` `fps` (with the `magick` renderer that ratio must divide 100, e.g. 32 years / 8 = 4). Save the rendered object, not the `gganim`: `a <- gganimate::animate(anim, duration = 8); gganimate::anim_save("x.gif", a)`. Installing `gifski` removes the `fps` constraint.

Value

A `gganim` object (render with `gganimate::animate()`).

See Also

`plot_groups_map()`, `plot_groups_map_interactive()`

Other visualization: `plot_groups_influence_matrix()`, `plot_groups_influence_network()`, `plot_groups_lineage_2d()`, `plot_groups_lineage_3d()`, `plot_groups_map()`, `plot_groups_map_interactive()`, `plot_groups_per_year()`, `plot_trajectory_confluence()`, `plot_trajectory_confluence_interactive()`, `plot_trajectory_confluence_matrix()`, `plot_trajectory_dag()`, `plot_trajectory_dag_interactive()`, `plot_trajectory_dispersion()`, `plot_trajectory_dynamics()`, `plot_trajectory_dynamics_interactive()`, `plot_trajectory_formation()`, `plot_trajectory_lines_2d()`, `plot_trajectory_lines_3d()`

`plot_groups_map_interactive`

Interactive stock map: the citation network as a plotly HTML widget

Description

The interactive twin of `plot_groups_map()`. Every paper is a scattergl point laid out by a force-directed layout of the citation network and coloured by its final group; hovering a node shows its title, year, authors, keywords, and final group. Returns a plotly htmlwidget – it renders in the RStudio Viewer, embeds in Quarto/Shiny, or saves to a standalone file with `htmlwidgets::saveWidget()`.

Usage

```
plot_groups_map_interactive(
  groups_cumulative,
  labels_text = NULL,
  network_until = NULL,
  layout = "drl",
  seed = 888L,
  show_unassigned = TRUE,
  label = TRUE,
  label_size = 12,
  title = "Group map",
  title_size = 16,
  normalize = FALSE,
  normalize_q = 0.99,
  marker_size = 5
)
```

Arguments

<code>groups_cumulative</code>	A stock object from <code>sniff_groups_cumulative()</code> .
<code>labels_text</code>	Optional data.frame with columns id and text; NULL (default) uses the bare group id. If no id matches any group, a warning is issued and the group ids are used.
<code>network_until</code>	Snapshot year. NULL (default) uses the last element.

layout	Force-directed layout passed to <code>ggraph::create_layout()</code> (default "dr1").
seed	Integer seed for the stochastic layout (default 888L).
show_unassigned	Draw nodes with no group (NA) as grey background (default TRUE).
label	Draw group labels at centroids as annotations (default TRUE).
label_size	Font size (px) of the centroid label annotations (default 12).
title	Plot title: the default "Group map", any string, or NULL for no title.
title_size	Title font size in px (default 16).
normalize	Rescale to a fixed unit window for cross-year comparison (default FALSE); see <code>plot_groups_map()</code> .
normalize_q	Quantile of the radial distance kept inside the normalised window (default 0.99).
marker_size	Node marker size (default 5).

Value

A plotly htmlwidget.

See Also

`plot_groups_map()`, `sniff_groups_cumulative()`

Other visualization: `plot_groups_influence_matrix()`, `plot_groups_influence_network()`, `plot_groups_lineage_2d()`, `plot_groups_lineage_3d()`, `plot_groups_map()`, `plot_groups_map_animation()`, `plot_groups_per_year()`, `plot_trajectory_confluence()`, `plot_trajectory_confluence_interactive()`, `plot_trajectory_confluence_matrix()`, `plot_trajectory_dag()`, `plot_trajectory_dag_interactive()`, `plot_trajectory_dispersion()`, `plot_trajectory_dynamics()`, `plot_trajectory_dynamics_interactive()`, `plot_trajectory_formation()`, `plot_trajectory_lines_2d()`, `plot_trajectory_lines_3d()`

`plot_groups_per_year` *Plot groups per year: the clustering fan converging to the final groups*

Description

The stock-side bridge view. One column per year; within a column the cluster-year nodes are ranked by paper count with the largest at the baseline, dot area encodes paper count, and (optionally) the number of groups that year is printed atop the column. No succession edges, no birth rings – it shows the annual + cumulative clustering fan settling onto the final-year groups. It is the stock-side counterpart of the flow-side `plot_trajectory_dag()`, and reads the `sniff_trajectory_dag()` object directly.

Usage

```
plot_groups_per_year(
  dag,
  color_terminal = FALSE,
  point_size = 6,
  show_count = TRUE,
  count_size = 3.2,
  count_colour = "white",
  count_fill = "grey45",
  label_terminal = TRUE,
  label_size = 3,
  axis_text_size = NULL,
  title = "Groups per year"
)
```

Arguments

dag	A sniff_trajectory_dag() object.
color_terminal	Colour the dots by terminal_group to foreshadow the braid (default FALSE, a single neutral grey that reads as stock). The colour legend is suppressed either way.
point_size	Upper bound of the dot-size range; paper count is mapped to [1, point_size] (default 6).
show_count	Print the number of groups atop each year column, on a filled label for emphasis (default TRUE).
count_size	Font size of the per-year group-count text (default 3.2).
count_colour	Font colour of the count text (default "white").
count_fill	Background colour of the count label (default "grey45", a grey chip that makes the count stand out); use "white" for a plain box.
label_terminal	Label the final-year nodes with their group id (c1gN) to the right of the last column, with a leader line (default TRUE).
label_size	Font size of the final-node labels (default 3).
axis_text_size	Font size of the x-axis (year) tick labels; NULL (default) keeps the theme default.
title	Plot title (default "Groups per year"); NULL for none.

Details

The visual encoding:

- **One column per year** along the x-axis; the y-axis **has no intrinsic meaning** (nodes are stacked by rank).
- Within a column the cluster-year nodes are **ranked by paper count** with the largest at the baseline, and **dot area encodes the paper count**.
- With show_count = TRUE the **number of groups that year** is printed atop the column on a filled label (count_fill / count_colour).

- There are **no succession edges and no births**: this is the stock-side view of how many groups exist each year and how large they are, settling onto the final-year groups. For the flow (edges, trajectories) use `plot_trajectory_dag()`.
- With `label_terminal = TRUE` the final-year nodes are labelled c1gN to the right of the last column.

Value

A ggplot object.

See Also

`sniff_trajectory_dag()`, `plot_trajectory_dag()`, `plot_trajectory_dag_interactive()`

Other visualization: `plot_groups_influence_matrix()`, `plot_groups_influence_network()`, `plot_groups_lineage_2d()`, `plot_groups_lineage_3d()`, `plot_groups_map()`, `plot_groups_map_animation()`, `plot_groups_map_interactive()`, `plot_trajectory_confluence()`, `plot_trajectory_confluence_interactive()`, `plot_trajectory_confluence_matrix()`, `plot_trajectory_dag()`, `plot_trajectory_dag_interactive()`, `plot_trajectory_dispersion()`, `plot_trajectory_dynamics()`, `plot_trajectory_dynamics_interactive()`, `plot_trajectory_formation()`, `plot_trajectory_lines_2d()`, `plot_trajectory_lines_3d()`

Examples

```
## Not run:
# docs_per_group comes from sniff_groups_lineage()
dag <- sniff_trajectory_dag(docs_per_group)

plot_groups_per_year(dag)

# colour the dots by destination, bigger count chips and year labels
plot_groups_per_year(dag, color_terminal = TRUE, count_size = 5, axis_text_size = 14)

## End(Not run)
```

plot_trajectory_confluence

Plot the trajectory confluence (how the central trajectories form)

Description

A braided river system over `sniff_trajectory_confluence()`: every trajectory is a stream flowing left to right along the publication year, merging into its absorber where it hands off, until the central trajectories (`tr : : cNgN`) reach the final year. Each tributary is drawn **grey** until the first paper it transfers is published, then in its colour; at the merge a band of width equal to the **transferred** papers (`n` of the tributary's `cohort_size`) bends into the river, which widens by that amount. The central spine is centered on its band with tributaries packed above and below. With `target = NULL` the whole system is drawn (one colour per destination central); with `target = "tr : : c1g1"` only that central's direct feeders are drawn (one colour per tributary).

Usage

```
plot_trajectory_confluence(
  conf,
  target = "all",
  depth = NULL,
  min_n = 5,
  min_prop = 0.05,
  min_total_size = 0,
  min_duration_years = 0,
  width_range = c(0, 0.42),
  label_terminal = c("id_count", "id"),
  label_intermediary = c("id_count", "id"),
  label_terminal_size = 4.5,
  label_intermediary_size = 4,
  labels_text = NULL,
  legend_ncol = NULL,
  palette = NULL,
  axis_text_size = NULL,
  year_range = NULL,
  show_secondary = TRUE,
  secondary_min_prop = 0.2,
  smooth = TRUE,
  title = NULL
)
```

Arguments

conf	A sniff_trajectory_confluence() object.
target	What to draw. "all" (default, same as NULL) draws the full confluence (the whole nested forest). "finals" draws only the central backbones (the in-flow spines, tributaries hidden); "intermediary" draws only the tributary streams (the ribbons, central spines hidden). A single central traj_id (e.g. "tr::c1g1") draws only that central's tributaries; a vector of central ids (e.g. c("tr::c1g1", "tr::c1g5")) draws the selected centrals stacked. With a single central the streams are coloured per tributary; otherwise per destination central. A single intermediate trajectory id (e.g. "tr3") switches to a focus view : its feeders merge in on the left (who fed it) and the finals its terminal cohort reached fan out on the right (who it fed).
depth	How many confluence levels upstream of the roots to draw; NULL (default) means the whole forest when target = NULL, or just the direct feeders (1) when one or more targets are given. Set e.g. depth = 2 to expand a target view to feeders-of-feeders.
min_n	Minimum papers a tributary must transfer to its absorber to be counted; below this it is pruned from the drawing and the river width, together with its own upstream subtree (default 5).
min_prop	Minimum share of a tributary's own cohort that must transfer (default 0.05); guards against incidental trickles.

min_total_size	Minimum total size of a tributary (its own distinct paper count across all years) for it to be drawn (default 0, no filter); prunes small lineages and their upstream subtree, complementing min_n (which gates the transfer, not the trajectory's own size).
min_duration_years	Minimum lifespan in years of a tributary (its last year minus its first year, plus one) for it to be drawn (default 0, no filter); use it to drop short-lived lineages.
width_range	Half-width range <code>c(floor, max)</code> in y-data units (lanes are spaced 1 apart, so keep max below ~0.5 to avoid overlap). The widest stream reaches max; every band is at least floor, so thin streams stay visible. Default <code>c(0, 0.42)</code> is strictly proportional (no floor); a positive floor (e.g. <code>c(0.05, 0.45)</code>) lifts the small streams, raising max thickens everything. Note these are lane-fraction units, not pixel sizes like <code>marker_range</code> .
label_terminal	Label content for the final (terminal) trajectories: "id_count" (default) <code>tr::cNgN</code> (size), "id" <code>tr::cNgN</code> , or NULL to hide them. The count is the central's final-year size (what remains in the final community), matching <code>plot_trajectory_lines_3d()</code> , not its lifetime total.
label_intermediary	Label content for the intermediate (tributary) trajectories: "id_count" (default) <code>trN(n)</code> , "id" <code>trN</code> , or NULL to hide them. <code>n</code> is the tributary's documents that end in its target final (the destination count from <code>plot_trajectory_lines_3d()</code>) – the single target in a single-target view, else the tributary's own dominant central – not the trunk transfer. Objects without destination data fall back to <code>trN(n/cohort)</code> .
label_terminal_size	Text size of the final (terminal) labels (default 4.5).
label_intermediary_size	Text size of the intermediate (tributary) labels (default 4).
labels_text	Optional data.frame with columns <code>id</code> and <code>text</code> mapping central group ids (<code>cNgN</code>) to descriptions; when supplied a bottom legend <code>tr::cNgN: description</code> is added (as in <code>plot_trajectory_dag()</code>). NULL (default) adds no legend.
legend_ncol	Number of columns in the <code>labels_text</code> legend; NULL (default) lets ggplot choose.
palette	Optional named colour vector overriding the default hues (keyed by destination central in the all/multi view, else by tributary <code>traj_id</code>).
axis_text_size	Font size of the x-axis (year) tick labels; NULL (default) keeps the theme default.
year_range	Optional <code>c(from, to)</code> years fixing the time axis (as in <code>plot_trajectory_lines_2d()</code>), so a short confluence can share the window of a longer one. NULL (default) fits the drawn rivers; a wider range only extends the axis (it never drops data). Ignored in the single-intermediate focus view.
show_secondary	Draw faint links from a tributary to the other finals its terminal cohort fed, beyond the dominant one it merges into (default TRUE). A tributary's papers can land in several final groups; the forest keeps only the dominant, and these links reveal the rest. Only drawn where both spines and ribbons are shown and the destination central is in view (so the all-view or several selected centrals, not a single-central view). Needs the <code>destinations</code> field of <code>sniff_trajectory_confluence()</code> .

secondary_min_prop	Minimum share of a tributary's terminal cohort that must reach a secondary final for its link to be drawn (default 0.2); raise it to keep only the strongest secondary destinations.
smooth	Gaussian-smooth the stream outlines (default TRUE): the central spine's cumulative inflow and each tributary's width follow a ~1-year low-pass so the edges flow as brush strokes instead of kinked polygons. FALSE keeps the raw year-to-year steps.
title	Plot title; NULL (default) derives one from target.

Details

- **x = publication year** (never reassigned); the **y-axis has no intrinsic meaning**, it only packs streams to minimise merge crossings.
- **Tributary width** follows its community size each year; **grey** before the first transferred paper, then its **colour**. **Central spine width** is the smoothed cumulative inflow of its counted tributaries (papers transferred), on one proportional scale shared with the tributaries (0 papers give 0 width).
- A tributary hands *n* of its cohort_size papers over; only *n* enters the river, so the merge band keeps a **constant width equal to the transferred papers** (the *n/cohort* label makes the split explicit) and bends over a span that scales with the distance travelled, so its thickness reads the same for near and far feeders.
- min_n / min_prop decide what is **counted**, not merely hidden: dropped tributaries leave the river narrower and are summarised in the caption.
- An intermediate that delivers **0 documents** to the final it flows into (its terminal cohort lands entirely outside that final's community) is always excluded, together with its upstream subtree, regardless of the thresholds: a stream contributing nothing to the destination cannot appear.

Value

A ggplot object.

See Also

[sniff_trajectory_confluence\(\)](#), [plot_trajectory_formation\(\)](#), [sniff_trajectory_braid\(\)](#)

Other visualization: [plot_groups_influence_matrix\(\)](#), [plot_groups_influence_network\(\)](#), [plot_groups_lineage_2d\(\)](#), [plot_groups_lineage_3d\(\)](#), [plot_groups_map\(\)](#), [plot_groups_map_animation\(\)](#), [plot_groups_map_interactive\(\)](#), [plot_groups_per_year\(\)](#), [plot_trajectory_confluence_interactive\(\)](#), [plot_trajectory_confluence_matrix\(\)](#), [plot_trajectory_dag\(\)](#), [plot_trajectory_dag_interactive\(\)](#), [plot_trajectory_dispersion\(\)](#), [plot_trajectory_dynamics\(\)](#), [plot_trajectory_dynamics_interactive\(\)](#), [plot_trajectory_formation\(\)](#), [plot_trajectory_lines_2d\(\)](#), [plot_trajectory_lines_3d\(\)](#)

Examples

```
## Not run:
flow <- sniff_trajectory_braid(docs_per_group)
conf <- sniff_trajectory_confluence(flow)
```

```
# the whole system: how every central forms
plot_trajectory_confluence(conf, min_n = 20)

# one central's direct feeders, coloured per tributary
plot_trajectory_confluence(conf, target = "tr::c1g1")

# focus on one intermediate trajectory: its feeders + the finals it fed
plot_trajectory_confluence(conf, target = "tr7")

# two selected centrals, a description legend, bigger year labels, bare ids,
# a visible floor on thin streams, and only sizeable tributaries
descr <- data.frame(id = c("c1g1", "c1g5"), text = c("Topic A", "Topic B"))
plot_trajectory_confluence(conf, target = c("tr::c1g1", "tr::c1g5"),
                           labels_text = descr, axis_text_size = 14,
                           label_intermediary = "id", min_total_size = 30,
                           min_duration_years = 3, width_range = c(0.05, 0.45))

## End(Not run)
```

```
plot_trajectory_confluence_interactive
```

Interactive trajectory confluence (plotly): hover a stream for its trajectory

Description

The plotly twin of `plot_trajectory_confluence()`. Every trajectory is a filled stream over the publication year, laid out by the same centered lane packing and coloured by the same rule (per destination central in the whole-forest / multi view, per tributary in a single-central view). There is no legend: hovering a **central** river shows `tr::cNgN` (size) (plus its `labels_text` description), and hovering a **tributary** shows `trN` (n/cohort -> `tr::cNgN`). With `label_terminal = TRUE` the central (or selected) rivers are also named in a de-overlapped column on the right, joined to their stream by a leader line. Returns a plotly htmlwidget.

Usage

```
plot_trajectory_confluence_interactive(
  conf,
  target = "all",
  depth = NULL,
  min_n = 5,
  min_prop = 0.05,
  min_total_size = 0,
  min_duration_years = 0,
  width_range = c(0, 0.42),
  palette = NULL,
  labels_text = NULL,
```

```

    label_terminal = TRUE,
    label_size = 12,
    smooth = TRUE,
    title = NULL
  )

```

Arguments

conf	A sniff_trajectory_confluence() object.
target	What to draw, as in plot_trajectory_confluence() minus the focus view: "all" (default) the whole forest, "finals" the central backbones, "intermediary" the tributary streams, a single central tr::cNgN (its feeders, coloured per tributary), or a vector of trajectory ids (centrals stacked, or intermediates shown as their feeders).
depth, min_n, min_prop, min_total_size, min_duration_years, width_range, smooth	Passed through to the same pruning / geometry helpers as plot_trajectory_confluence() ; see there for details.
palette	Optional named colour vector overriding the default hues (keyed by destination central in the all/multi view, else by tributary traj_id).
labels_text	Optional data.frame with columns id and text mapping ids to descriptions, appended to the hover (id: description ...). Central rivers are matched by their group id (cNgN), absorbed tributaries by their own trajectory id (trN); NULL (default) uses the bare id.
label_terminal	Name the central (or selected) rivers in a de-overlapped column to the right, each joined to its stream by a leader line (default TRUE).
label_size	Font size (px) of those right-hand labels (default 12).
title	Plot title; NULL (default) derives one from target.

Details

Unlike the static function this covers the **forest views only** ("all", "finals", "intermediary", one or more centrals, and an intermediate shown as its upstream feeders); the single-intermediate **focus view** and the faint secondary-destination links are not drawn.

Built with plotly, following the same principles as [plot_trajectory_dag_interactive\(\)](#):

- The **x-axis is the publication year**; the **y-axis has no intrinsic meaning** (it only packs streams to minimise merge crossings).
- Each stream is a filled polygon (fill = "toself", hoverable over its area); a tributary is **grey** before the first paper it transfers, then in its colour.
- **There is no legend**: detail is revealed on hover, and only the terminal (central or selected) rivers carry a printed label.
- An intermediate that delivers **0 documents** to the final it flows into is always excluded (as in [plot_trajectory_confluence\(\)](#)).

The widget renders in the RStudio Viewer, embeds in Quarto/Shiny, or saves to a standalone file with `htmlwidgets::saveWidget()`.

Value

A plotly htmlwidget.

See Also

`plot_trajectory_confluence()`, `plot_trajectory_dag_interactive()`, `sniff_trajectory_confluence()`
 Other visualization: `plot_groups_influence_matrix()`, `plot_groups_influence_network()`,
`plot_groups_lineage_2d()`, `plot_groups_lineage_3d()`, `plot_groups_map()`, `plot_groups_map_animation()`,
`plot_groups_map_interactive()`, `plot_groups_per_year()`, `plot_trajectory_confluence()`,
`plot_trajectory_confluence_matrix()`, `plot_trajectory_dag()`, `plot_trajectory_dag_interactive()`,
`plot_trajectory_dispersion()`, `plot_trajectory_dynamics()`, `plot_trajectory_dynamics_interactive()`,
`plot_trajectory_formation()`, `plot_trajectory_lines_2d()`, `plot_trajectory_lines_3d()`

Examples

```
## Not run:
flow <- sniff_trajectory_braid(docs_per_group)
conf <- sniff_trajectory_confluence(flow)

plot_trajectory_confluence_interactive(conf, min_n = 20)

# one central's formation, hover the tributaries for their n/cohort
plot_trajectory_confluence_interactive(conf, target = "tr::c1g1")

## End(Not run)
```

`plot_trajectory_confluence_matrix`

Plot the trajectory confluence matrix (finals vs intermediates)

Description

A heatmap of how each intermediate (absorbed) trajectory's papers spread across the final trajectories. By default **rows are the final trajectories** `tr::cNgN` and **columns are the intermediate trajectories** `trN` (swap with orientation); the fill is the coupling strength (shared documents) and each intermediate's dominant-final cell is outlined. Which intermediates appear is decided by the same pruning the other flow plots use (`plot_trajectory_lines_2d()`, `plot_trajectory_confluence()`), so the column set is consistent across views.

Usage

```
plot_trajectory_confluence_matrix(
  flow,
  conf = NULL,
  fill = c("n_shared", "prop_of_group", "prop_of_traj"),
  orientation = c("finals-cols", "finals-rows"),
```

```

order_by = c("terminal", "reach", "size", "alpha", "cluster"),
min_n = 5,
min_prop = 0.05,
min_total_size = 0,
min_duration_years = 0,
min_target_n = 0,
mark_terminal = TRUE,
show_values = FALSE,
label_size = 2.5,
axis_text_size = NULL,
axis_title_size = NULL,
legend_position = "right",
legend_text_size = NULL,
bg = "white",
title = "Trajectory matrix"
)

```

Arguments

flow	A sniff_trajectory_braid() object.
conf	An optional precomputed sniff_trajectory_confluence() object for flow; NULL (default) computes it.
fill	"n_shared" (default, log-scaled), "prop_of_group", or "prop_of_traj".
orientation	"finals-cols" (default): final trajectories tr: : cNgN on the rows (y-axis), intermediate trajectories trN on the columns (x-axis) – the "finals (rows) x intermediates (cols)" matrix. "finals-rows" is the transpose (finals on the columns, intermediates on the rows).
order_by	How the intermediate trajectories are ordered along their axis: "terminal" (default, grouped by dominant final then reach), "reach" (total shared documents), "size" (trajectory size at handoff), "alpha" (alphabetical), or "cluster" (seriation that reorders both axes by hierarchical clustering – intermediates by their destination profile and finals by their source profile – so alike rows and columns form blocks).
min_n, min_prop, min_total_size, min_duration_years	Thresholds selecting which intermediate trajectories are shown, passed to the same prune as plot_trajectory_lines_2d() : an intermediate is kept only if it transfers $\geq$ min_n papers and $\geq$ min_prop of its cohort, has total size $\geq$ min_total_size and lifespan $\geq$ min_duration_years.
min_target_n	Minimum shared documents a cell must have to be drawn (default 0, all cells); raise it to hide the long tail of small overlaps. Cells with zero shared documents are never drawn. Final trajectories are always kept on their axis, even when the thresholds leave them with no cell.
mark_terminal	Outline each intermediate's dominant-final cell (default TRUE).
show_values	Print the value inside each cell (default FALSE). The text colour auto-contrasts (dark on light cells, light on dark) for legibility.
label_size	Font size of the in-cell values (default 2.5).

axis_text_size	Font size of the axis (trajectory id) tick labels; NULL (default) keeps the theme default.
axis_title_size	Font size of the axis titles ("Final Trajectory" / "Intermediate Trajectory"); NULL (default) keeps the theme default.
legend_position	Where to put the colourbar: "right" (default), "left", "top", "bottom", "none", or a c(x, y) coordinate inside the panel (each in 0..1). "top"/"bottom" draw the colourbar horizontally.
legend_text_size	Font size of the legend text; NULL (default) keeps the theme default.
bg	Panel/plot background colour (default "white"). A darker background (e.g. "grey20") raises contrast against the bright cells; the grid lines and the terminal outline auto-adapt to its luminance.
title	Plot title (default "Trajectory matrix"); NULL removes it.

Value

A ggplot object.

See Also

[sniff_trajectory_braid\(\)](#), [plot_trajectory_lines_2d\(\)](#), [plot_trajectory_confluence\(\)](#)

Other visualization: [plot_groups_influence_matrix\(\)](#), [plot_groups_influence_network\(\)](#), [plot_groups_lineage_2d\(\)](#), [plot_groups_lineage_3d\(\)](#), [plot_groups_map\(\)](#), [plot_groups_map_animation\(\)](#), [plot_groups_map_interactive\(\)](#), [plot_groups_per_year\(\)](#), [plot_trajectory_confluence\(\)](#), [plot_trajectory_confluence_interactive\(\)](#), [plot_trajectory_dag\(\)](#), [plot_trajectory_dag_interactive\(\)](#), [plot_trajectory_dispersion\(\)](#), [plot_trajectory_dynamics\(\)](#), [plot_trajectory_dynamics_interactive\(\)](#), [plot_trajectory_formation\(\)](#), [plot_trajectory_lines_2d\(\)](#), [plot_trajectory_lines_3d\(\)](#)

Examples

```
## Not run:
flow <- sniff_trajectory_braid(docs_per_group)
plot_trajectory_confluence_matrix(flow, min_n = 20, show_values = TRUE)
plot_trajectory_confluence_matrix(flow, order_by = "cluster", bg = "grey20")

## End(Not run)
```

plot_trajectory_dag *Plot the soft trajectory DAG (all intermediate nodes)*

Description

The braided-river panorama: every cluster-year node laid out by year (x), with succession edges, node colour = terminal_group (or group), node size = paper count, and births ringed. Uses the year-aware Sugiyama layout from [sniff_trajectory_dag\(\)](#)'s graph. Nodes from dormant lineages that never reach the final year (a colour value outside the final groups) are drawn grey and kept out of the colour legend.

Usage

```
plot_trajectory_dag(
  dag,
  color_by = c("terminal_group", "group"),
  edge_alpha = 0.25,
  point_size = 2.5,
  show_legend = FALSE,
  label_terminal = TRUE,
  labels_text = NULL,
  label_size = 3,
  title = "Cumulative network groups",
  legend_text_size = NULL,
  legend_ncol = NULL,
  axis_text_size = NULL
)
```

Arguments

dag	A sniff_trajectory_dag() object.
color_by	How to colour the nodes (default "terminal_group"). "terminal_group" uses the final group each node's lineage flows into (its heaviest-successor destination), so a whole trajectory keeps one colour from source to mouth and the colour legend has one entry per final group. "group" uses the node's own per-year id (c1gN), which is a within-year size rank rather than a stable identity, so the same trajectory changes colour across years. Pick "terminal_group" to read how the final groups formed, "group" to inspect the raw annual clustering labels.
edge_alpha	Base edge transparency, scaled by Jaccard weight (default 0.25).
point_size	Base node size (default 2.5).
show_legend	Show the colour legend (default FALSE; many final groups make a busy legend).
label_terminal	Label the final-year nodes with their bare group id (c1gN), to the right of each vertex, with a leader line back to the vertex (default TRUE).
labels_text	Optional data.frame with columns id and text. When supplied, the node-colour legend moves to the bottom and each group is relabelled "c1gN: text" (groups without a match keep the bare id); NULL (default) leaves the legend governed by show_legend. The final-node labels (label_terminal) are always the bare group id, independent of this.
label_size	Font size of the final-node labels (default 3).

title	Plot title (default "Cumulative network groups"); NULL for no title.
legend_text_size	Font size of the colour-legend text; NULL (default) keeps the theme default.
legend_ncol	Number of columns in the colour legend; NULL (default) lets ggplot choose.
axis_text_size	Font size of the x-axis (year) tick labels; NULL (default) keeps the theme default.

Details

The visual encoding:

- The **x-axis is the publication year** (a Sugiyama layered layout); the **y-axis has no intrinsic meaning**, it only spreads nodes to reduce edge crossings.
- **Node colour** is the trajectory basin (see `color_by`), **node size** is the paper count, and **succession edges** fade with their Jaccard weight.
- **Births** (nodes with no strong predecessor) are ringed in black.
- **Dormant lineages** that never reach the final year are drawn grey and kept out of the colour legend, so the colours stand for the final groups only.
- With `label_terminal = TRUE` the final-year nodes are labelled c1gN to the right (with a leader line); with `labels_text` the colour legend moves to the bottom as "c1gN: description" entries.

Value

A ggplot object.

See Also

`sniff_trajectory_dag()`, `plot_groups_per_year()`, `plot_trajectory_dag_interactive()`

Other visualization: `plot_groups_influence_matrix()`, `plot_groups_influence_network()`, `plot_groups_lineage_2d()`, `plot_groups_lineage_3d()`, `plot_groups_map()`, `plot_groups_map_animation()`, `plot_groups_map_interactive()`, `plot_groups_per_year()`, `plot_trajectory_confluence()`, `plot_trajectory_confluence_interactive()`, `plot_trajectory_confluence_matrix()`, `plot_trajectory_dag_in`, `plot_trajectory_dispersion()`, `plot_trajectory_dynamics()`, `plot_trajectory_dynamics_interactive()`, `plot_trajectory_formation()`, `plot_trajectory_lines_2d()`, `plot_trajectory_lines_3d()`

Examples

```
## Not run:
# docs_per_group comes from sniff_groups_lineage()
dag <- sniff_trajectory_dag(docs_per_group)

# the whole braid, coloured by destination group
plot_trajectory_dag(dag)

# human-readable bottom legend, two columns, larger year labels
descr <- data.frame(id = c("c1g1", "c1g2"), text = c("Topic A", "Topic B"))
plot_trajectory_dag(dag, labels_text = descr, legend_ncol = 2, axis_text_size = 14)
```

```
## End(Not run)
```

```
plot_trajectory_dag_interactive
```

Interactive trajectory DAG (plotly): hover a node for its trajectory

Description

The plotly twin of `plot_trajectory_dag()`. Nodes are laid out by the same year-aware Sugiyama layout, sized by paper count, and coloured by the final group their lineage reaches. Dormant lineages that never reach the final year are drawn grey. There is no legend: hovering a final-year node shows its group id and `labels_text` description (c1gN: description), and hovering any other (intermediate or dormant) node shows its raw id (e.g. y2021c1g2). When `label_terminal` is TRUE, the final group id (c1gN) is also printed to the right of each final-year node. Returns a plotly `htmlwidget`.

Usage

```
plot_trajectory_dag_interactive(
  dag,
  labels_text = NULL,
  marker_range = c(4, 14),
  edge_alpha = 0.3,
  label_terminal = TRUE,
  label_size = 12,
  title = "Cumulative network groups"
)
```

Arguments

<code>dag</code>	A <code>sniff_trajectory_dag()</code> object.
<code>labels_text</code>	Optional <code>data.frame</code> with columns <code>id</code> and <code>text</code> mapping final group ids to descriptions, shown after the group id when hovering a final-year node (c1gN: description); NULL (default) falls back to the bare group id.
<code>marker_range</code>	Min/max marker size in pixels; paper count is scaled into this range (default <code>c(4, 14)</code>).
<code>edge_alpha</code>	Edge opacity, 0-1 (default 0.3).
<code>label_terminal</code>	Show the final group id (c1gN) to the right of each final-year node (default TRUE).
<code>label_size</code>	Font size (px) of the final-node labels (default 12).
<code>title</code>	Plot title (default "Cumulative network groups"); NULL for none.

Details

The interactive twin of `plot_trajectory_dag()`, built with plotly:

- The **x-axis is the publication year** (Sugiyama layout); the **y-axis has no intrinsic meaning**.
- **Node colour** is the trajectory basin (the final group reached); **dormant lineages** that never reach the final year are grey. **Node size** scales with paper count over `marker_range`.
- **There is no legend**. Hovering a **final-year node** shows `c1gN`: description (its group id plus the `labels_text` description); hovering any **other node** shows its raw id (`yYYYYcXgN`).
- **Edges** are a single faint layer (`edge_alpha`) and are excluded from hover. With `label_terminal = TRUE` the final group ids (`c1gN`) are printed in a de-overlapped column to the right, each joined to its node by a leader line.

The widget renders in the RStudio Viewer, embeds in Quarto/Shiny, or saves to a standalone file with `htmlwidgets::saveWidget()`.

Value

A plotly htmlwidget.

See Also

`plot_trajectory_dag()`, `plot_groups_per_year()`

Other visualization: `plot_groups_influence_matrix()`, `plot_groups_influence_network()`, `plot_groups_lineage_2d()`, `plot_groups_lineage_3d()`, `plot_groups_map()`, `plot_groups_map_animation()`, `plot_groups_map_interactive()`, `plot_groups_per_year()`, `plot_trajectory_confluence()`, `plot_trajectory_confluence_interactive()`, `plot_trajectory_confluence_matrix()`, `plot_trajectory_dag()`, `plot_trajectory_dispersion()`, `plot_trajectory_dynamics()`, `plot_trajectory_dynamics_interactive()`, `plot_trajectory_formation()`, `plot_trajectory_lines_2d()`, `plot_trajectory_lines_3d()`

Examples

```
## Not run:
# docs_per_group comes from sniff_groups_lineage()
dag <- sniff_trajectory_dag(docs_per_group)

plot_trajectory_dag_interactive(dag)

# hover the final nodes for descriptions; fainter edges
descr <- data.frame(id = c("c1g1", "c1g2"), text = c("Topic A", "Topic B"))
p <- plot_trajectory_dag_interactive(dag, labels_text = descr, edge_alpha = 0.2)
# htmlwidgets::saveWidget(p, "trajectory_dag.html", selfcontained = TRUE)

## End(Not run)
```

plot_trajectory_dispersion

Plot where a stagnant trajectory's cohort dispersed (its dispersion fan)

Description

Draws the selected intermediate (stagnant) trajectory as a short spine on the left and fans out, from its handoff year, one curved link to **every final trajectory its terminal cohort reached** (the full destination split), each link's width scaled by the papers that landed there and labelled `tr: :cNgN (n)`. It answers, visually, "where did this trajectory go?" – not just the dominant absorber but the complete dispersion. Styling follows the other flow plots ([plot_trajectory_lines_2d\(\)](#), [plot_trajectory_confluence\(\)](#)): a Set2/hue palette keyed by final group, a year-aware x-axis, and the shared `year_range` / `axis_text_size` / `palette` controls.

Usage

```
plot_trajectory_dispersion(
  destination,
  title = NULL,
  label_size = 4.5,
  year_range = NULL,
  axis_text_size = NULL,
  palette = NULL,
  min_n = 0
)
```

Arguments

<code>destination</code>	Output of sniff_trajectory_destination() . Uses its destination table (<code>g_final, n</code>) and <code>source_info</code> .
<code>title</code>	Plot title. Defaults to a source -> N finals summary.
<code>label_size</code>	Text size for the in-plot labels (default: 4.5).
<code>year_range</code>	Optional <code>c(from, to)</code> years fixing the time axis, as in plot_trajectory_lines_2d() . NULL (default) fits the drawn data; a wider range only extends the axis.
<code>axis_text_size</code>	Font size of the x-axis (year) tick labels; NULL (default) keeps the built-in size.
<code>palette</code>	Optional named colour vector overriding the default hues (keyed by final group <code>cNgN</code>). NULL (default) uses the package's Set2 (≤ 8) / hue scheme.
<code>min_n</code>	Minimum papers a final must receive to be drawn (default 0, all impacted finals). Raise it to hide the long tail of tiny destinations.

Value

A ggplot object.

See Also

[sniff_trajectory_destination\(\)](#), [plot_trajectory_confluence\(\)](#), [plot_trajectory_lines_2d\(\)](#)

Other visualization: [plot_groups_influence_matrix\(\)](#), [plot_groups_influence_network\(\)](#), [plot_groups_lineage_2d\(\)](#), [plot_groups_lineage_3d\(\)](#), [plot_groups_map\(\)](#), [plot_groups_map_animation\(\)](#), [plot_groups_map_interactive\(\)](#), [plot_groups_per_year\(\)](#), [plot_trajectory_confluence\(\)](#), [plot_trajectory_confluence_interactive\(\)](#), [plot_trajectory_confluence_matrix\(\)](#), [plot_trajectory_dag\(\)](#), [plot_trajectory_dag_interactive\(\)](#), [plot_trajectory_dynamics\(\)](#), [plot_trajectory_dynamics_interactive\(\)](#), [plot_trajectory_formation\(\)](#), [plot_trajectory_lines_2d\(\)](#), [plot_trajectory_lines_3d\(\)](#)

Examples

```
## Not run:
flow <- sniff_trajectory_braid(docs_per_group)
d <- sniff_trajectory_destination(flow, "tr1")
plot_trajectory_dispersion(d)

## End(Not run)
```

plot_trajectory_dynamics

Strategic map of trajectory dynamic states

Description

A quadrant scatter of [sniff_trajectory_dynamics\(\)](#) output, one figure per population, designed to make the dynamic states legible at a glance. Each point is a trajectory, its area the community size, its colour the state; the regions are shaded and named after the state they classify, with the thresholds drawn as reference lines.

Usage

```
plot_trajectory_dynamics(
  dyn,
  target = c("finals", "intermediary"),
  thresholds = NULL,
  labels_text = NULL,
  label_id = FALSE,
  size_range = c(2, 14),
  label_size = 3.5,
  axis_text_size = NULL,
  palette = NULL,
  show_thresholds = TRUE,
  show_quadrants = TRUE,
  xlim = NULL,
  ylim = NULL,
  legend_position = "right",
```

```

    title = NULL
  )

```

Arguments

<code>dyn</code>	A <code>sniff_trajectory_dynamics()</code> tibble.
<code>target</code>	What to draw, following <code>plot_trajectory_confluence()</code> . "finals" (default) maps the living (central) trajectories on a life-cycle plane (novelty x growth) that separates emergence, maturity and dormancy. "intermediary" maps the declining (absorbed) trajectories on a destination plane (entropy x dormant share) that separates convergence, divergence and dormancy. The two populations live on different axes, so there is no combined view.
<code>thresholds</code>	The <code>fixed_state_thresholds()</code> -shaped list whose cut values bound the shaded regions and the reference lines. NULL (default) reuses the cuts the classification was built with – <code>attr(dyn, "state_thresholds")</code> , attached by <code>sniff_trajectory_dynamics()</code> – so regions and colours always agree; if that attribute is absent it derives them from <code>dyn</code> via <code>data_state_thresholds()</code> . Pass an explicit list to override.
<code>labels_text</code>	Optional <code>data.frame</code> with columns <code>id</code> and <code>text</code> mapping group ids (cNgN) to descriptions; used to label the living cores in the "finals" view (as in <code>plot_trajectory_dag()</code>). NULL (default) labels by <code>id</code> .
<code>label_id</code>	Prefix each living-core label with its group id, as <code>cNgN: description</code> (default FALSE, description only). No effect when <code>labels_text</code> is NULL (the label is already the <code>id</code>) or in the "intermediary" view (labelled by <code>traj_id</code>).
<code>size_range</code>	Point-area range <code>c(min, max)</code> passed to <code>ggplot2::scale_size()</code> (default <code>c(2, 14)</code>).
<code>label_size</code>	Text size of the point labels (default 3.5); the region labels are drawn one point larger.
<code>axis_text_size</code>	Font size of the axis tick labels; NULL (default) keeps the theme default.
<code>palette</code>	Optional named colour vector overriding the default semantic hues, keyed by state (emergence/maturity/dormancy for "finals", convergence/divergence/dormancy for "intermediary").
<code>show_thresholds</code>	Draw the classification cut lines (default TRUE).
<code>show_quadrants</code>	Shade and name the state regions (default TRUE).
<code>xlim, ylim</code>	Optional <code>c(lo, hi)</code> to zoom the axes via <code>ggplot2::coord_cartesian()</code> without dropping data from the fit; NULL (default) expands the view to keep every state region visible (the data, the threshold cuts, and the <code>[0, 1]</code> share bounds, with padding), so the bands are shown even when the points cluster. Pass a range to override, e.g. to zoom past a single young core's outlying <code>growth_rate</code> .
<code>legend_position</code>	Legend placement (default "right").
<code>title</code>	Plot title; NULL (default) derives one from <code>target</code> .

Details

- "finals" – x = novelty, y = growth_rate; emergence is the novel-and-growing region, dormancy the loss-of-momentum band, maturity the saturated middle.
- "intermediary" – x = dest_entropy, y = dormant_share; convergence is the low-entropy (single destination) side, divergence the high-entropy side, and dormancy the high-dropout band (rare, so this band is usually sparse). Extinct (group NA) lineages have no coordinates and are reported in the caption rather than drawn.

Value

A ggplot object.

See Also

`sniff_trajectory_dynamics()`, `fixed_state_thresholds()`, `plot_trajectory_confluence()`

Other visualization: `plot_groups_influence_matrix()`, `plot_groups_influence_network()`, `plot_groups_lineage_2d()`, `plot_groups_lineage_3d()`, `plot_groups_map()`, `plot_groups_map_animation()`, `plot_groups_map_interactive()`, `plot_groups_per_year()`, `plot_trajectory_confluence()`, `plot_trajectory_confluence_interactive()`, `plot_trajectory_confluence_matrix()`, `plot_trajectory_dag()`, `plot_trajectory_dag_interactive()`, `plot_trajectory_dispersion()`, `plot_trajectory_dynamics_interactive`, `plot_trajectory_formation()`, `plot_trajectory_lines_2d()`, `plot_trajectory_lines_3d()`

Examples

```
## Not run:
flow <- sniff_trajectory_braid(docs_per_group)
dyn  <- sniff_trajectory_dynamics(flow)
descr <- data.frame(id = "c1g1", text = "Anaerobic digestion")

plot_trajectory_dynamics(dyn, target = "finals", labels_text = descr)
plot_trajectory_dynamics(dyn, target = "intermediary")

## End(Not run)
```

`plot_trajectory_dynamics_interactive`

Interactive strategic map of trajectory dynamic states (plotly)

Description

The plotly twin of `plot_trajectory_dynamics()`. The same quadrant map – shaded and named state regions, classification thresholds as reference lines, points sized by community and coloured by state – rendered as an HTML widget. Hovering a point reveals the full indicator panel (growth, novelty, recruitment, inflow, destination entropy, dormant share, and any CCT / hub-role columns), and the state legend is clickable to isolate one state at a time.

Usage

```
plot_trajectory_dynamics_interactive(
  dyn,
  target = c("finals", "intermediary"),
  thresholds = NULL,
  labels_text = NULL,
  label_id = FALSE,
  marker_range = c(6, 42),
  show_labels = FALSE,
  label_size = 11,
  palette = NULL,
  show_thresholds = TRUE,
  show_quadrants = TRUE,
  xlim = NULL,
  ylim = NULL,
  title = NULL
)
```

Arguments

<code>dyn</code>	A <code>sniff_trajectory_dynamics()</code> tibble.
<code>target</code>	"finals" (default) maps the living (central) trajectories (novelty x growth), "intermediary" the declining (absorbed) ones (destination entropy x dormant share), as in <code>plot_trajectory_dynamics()</code> .
<code>thresholds</code>	The <code>fixed_state_thresholds()</code> -shaped list bounding the regions and the reference lines. NULL (default) reuses the cuts attached to <code>dyn</code> by <code>sniff_trajectory_dynamics()</code> (<code>attr(dyn, "state_thresholds")</code>), else derives them via <code>data_state_thresholds()</code> . Pass an explicit list to override.
<code>labels_text</code>	Optional data.frame (id, text) mapping ids to descriptions, used in the hover (and labels). The "finals" view matches by group id (cNgN); the "intermediaries" view matches by trajectory id (trN).
<code>label_id</code>	Put the id on the label's own first line, above the description (cNgN then the text for finals, trN then the text for intermediaries), so the hover and pinned labels stay identifiable; the line break is plotly's <code>
</code> . Default FALSE.
<code>marker_range</code>	Marker diameter range in px <code>c(min, max)</code> (default <code>c(6, 42)</code>); area scales with the square root of size.
<code>show_labels</code>	Pin a text label to each point (default FALSE; the detail lives in the hover). Text may overlap when points are dense.
<code>label_size</code>	Font size (px) of the pinned labels (default 11).
<code>palette</code>	Optional named colour vector overriding the default semantic hues, keyed by state.
<code>show_thresholds</code>	Draw the classification cut lines (default TRUE).
<code>show_quadrants</code>	Shade and name the state regions (default TRUE).

xlim,ylim	Optional c(lo, hi) axis ranges; NULL (default) expands to keep every state region visible (the data, the threshold cuts, and the $[0,1]$ share bounds, with padding).
title	Plot title; NULL (default) derives one from target.

Value

A plotly htmlwidget.

See Also

[plot_trajectory_dynamics\(\)](#), [sniff_trajectory_dynamics\(\)](#), [fixed_state_thresholds\(\)](#)

Other visualization: [plot_groups_influence_matrix\(\)](#), [plot_groups_influence_network\(\)](#), [plot_groups_lineage_2d\(\)](#), [plot_groups_lineage_3d\(\)](#), [plot_groups_map\(\)](#), [plot_groups_map_animation\(\)](#), [plot_groups_map_interactive\(\)](#), [plot_groups_per_year\(\)](#), [plot_trajectory_confluence\(\)](#), [plot_trajectory_confluence_interactive\(\)](#), [plot_trajectory_confluence_matrix\(\)](#), [plot_trajectory_dag\(\)](#), [plot_trajectory_dag_interactive\(\)](#), [plot_trajectory_dispersion\(\)](#), [plot_trajectory_dynamics\(\)](#), [plot_trajectory_formation\(\)](#), [plot_trajectory_lines_2d\(\)](#), [plot_trajectory_lines_3d\(\)](#)

Examples

```
## Not run:
flow <- sniff_trajectory_braid(docs_per_group)
dyn  <- sniff_trajectory_dynamics(flow)
plot_trajectory_dynamics_interactive(dyn, target = "finals", labels_text = descr)

## End(Not run)
```

plot_trajectory_formation

Plot the trajectories that fed into a target (confluence timeline)

Description

Draws the target trajectory as a central spine and each feeder as a parallel timeline that grows along its length (see `feeder_curve`) and merges into the spine at its handoff year. The spine is a cumulative river: it widens smoothly as each drawn feeder joins, so by the final year its width reflects the total papers absorbed. Spine and feeder widths share one strictly proportional scale (width 0 at zero papers), so their thickness is directly comparable. It generalises [plot_trajectory_dispersion\(\)](#) (one source into one absorber) to many feeders converging on one trajectory, answering visually "which trajectories formed this one?".

Usage

```
plot_trajectory_formation(
  formation,
  feeder_curve = c("inflow", "size"),
  max_feeders = 8,
  title = NULL,
  label_size = 4.5,
  year_range = NULL,
  axis_text_size = NULL,
  palette = NULL
)
```

Arguments

formation	Output of sniff_trajectory_formation() .
feeder_curve	What the feeder's growing width traces: "inflow" (default) is the cumulative arrival of just the papers that entered the target, a monotone curve ending at n on the river's scale; "size" is the feeder's measured cluster size per year, ending at cohort_size (its whole community, which can dwarf and shrink the river since the 2D scale is linear). Each feeder rises from zero at its start year to this end value, on the same proportional scale as the spine, so it grows like the target does.
max_feeders	Maximum number of feeders to draw, taken by descending papers (default: 8). Feeders beyond this are summarised in the caption.
title	Plot title. Defaults to the target trajectory key.
label_size	Text size for the feeder and target labels (default: 4.5).
year_range	Optional c(from, to) years fixing the time axis, so a short formation can be shown on the same window as a long one (as in plot_trajectory_lines_2d()). NULL (default) fits the drawn data; a wider range only extends the axis (it never drops data).
axis_text_size	Font size of the x-axis (year) tick labels; NULL (default) keeps the built-in size.
palette	Optional named colour vector overriding the default hues (keyed by feeder source_key), to match colours across plots. NULL (default) uses the package's Set2 (<= 8 feeders) / hue scheme.

Value

A ggplot object.

See Also

[sniff_trajectory_formation\(\)](#), [plot_trajectory_lines_2d\(\)](#), [plot_trajectory_dispersion\(\)](#)

Other visualization: [plot_groups_influence_matrix\(\)](#), [plot_groups_influence_network\(\)](#), [plot_groups_lineage_2d\(\)](#), [plot_groups_lineage_3d\(\)](#), [plot_groups_map\(\)](#), [plot_groups_map_animation\(\)](#), [plot_groups_map_interactive\(\)](#), [plot_groups_per_year\(\)](#), [plot_trajectory_confluence\(\)](#), [plot_trajectory_confluence_interactive\(\)](#), [plot_trajectory_confluence_matrix\(\)](#), [plot_trajectory_dag\(\)](#),

```
plot_trajectory_dag_interactive(), plot_trajectory_dispersion(), plot_trajectory_dynamics(),
plot_trajectory_dynamics_interactive(), plot_trajectory_lines_2d(), plot_trajectory_lines_3d()
```

Examples

```
## Not run:
flow <- sniff_trajectory_braid(groups_lineage)
f <- sniff_trajectory_formation(flow, target = "tr::c1g3")
plot_trajectory_formation(f)

## End(Not run)
```

```
plot_trajectory_lines_2d
```

Plot the global trajectories as 2D variable-width lines

Description

The line counterpart of `plot_trajectory_confluence()` over the **global** flow DAG. Every global trajectory is drawn as a variable-width line through the year-aware Sugiyama layout: the central finals `tr::cNgN` and the intermediate `trN` that feed them, each line widening with the documents accumulated along its path. A bottom legend reports, for every intermediate, how many papers of its terminal cohort reached each final group – a single `trN` can feed more than one final (its terminal cohort splits across groups), which the legend makes explicit.

Usage

```
plot_trajectory_lines_2d(
  flow,
  target = "all",
  conf = NULL,
  min_n = 5,
  min_prop = 0.05,
  min_total_size = 0,
  min_duration_years = 0,
  min_target_n = 0,
  width_range = c(0.8, 6),
  use_raw_papers = FALSE,
  label_size = 4,
  dest_min_prop = 0.05,
  lowlight_alpha = 0.18,
  lowlight_color = "#9AA5B1",
  legend_ncol = NULL,
  legend_text_size = NULL,
  axis_text_size = NULL,
  year_range = NULL,
  palette = NULL,
```

```

    title = NULL
  )

```

Arguments

flow	A sniff_trajectory_braid() object, or a docs_per_group tibble / sniff_trajectory_dag() object the flow is built from.
target	What to draw. "all" (default) draws every central and the trajectories feeding it; a central traj_id (e.g. "tr::c1g1") or a vector of them draws only those finals and their feeders.
conf	An optional precomputed sniff_trajectory_confluence() object for flow; pass it to skip recomputing the confluence on every call (e.g. in a Shiny app that re-renders as a selector changes). NULL (default) computes it.
min_n, min_prop, min_total_size, min_duration_years	Feeder thresholds, passed to the same pruning used by plot_trajectory_confluence() : an intermediate is drawn only if it transfers $\geq$ min_n papers and $\geq$ min_prop of its cohort, has total size $\geq$ min_total_size and lifespan $\geq$ min_duration_years; a pruned intermediate takes its own feeders with it.
min_target_n	Minimum papers an intermediate must leave in the selected final(s) by the last year to be drawn (default 0, no filter). Unlike min_n (the tree-edge transfer to its absorber), this counts the intermediate's own papers that end up in the target group, so a feeder that merges in but barely persists there (e.g. tr39 with 12 of its papers in c1g1) is dropped by min_target_n = 20. Only applies to a targeted final view.
width_range	Line-width range c(min, max) (default c(0.8, 6)), scaled by the cumulative tracked documents along each trajectory.
use_raw_papers	Scale widths by raw quantity_papers (TRUE) or by the weighted quantity_papers * prop_tracked_intra_group (FALSE, default).
label_size	Text size of the trN / tr::cNgN end labels (default 4).
dest_min_prop	In the target = "all" view only, the minimum share of an intermediate's terminal cohort that must reach a final for that final to appear in its legend entry (default 0.05). A targeted view reports only the selected final(s), so this does not apply.
lowlight_alpha, lowlight_color	Opacity and colour of the background (non-trajectory) edges (defaults 0.18, neutral grey).
legend_ncol	Number of columns in the feeding legend; NULL (default) lets ggplot choose.
legend_text_size	Font size (pt) of the legend text; NULL (default) keeps the theme default.
axis_text_size	Font size (pt) of the x-axis (year) tick labels; NULL (default) keeps the theme default.
year_range	Optional c(from, to) years fixing the time axis, so a short trajectory can be shown on the same window as a long one (e.g. compare a young tr::c1g14 against tr::c1g1). NULL (default) fits the drawn trajectories; a wider range only extends the axis (it never drops data).

palette	Optional named colour vector overriding the default hues (keyed by trajectory traj_id).
title	Plot title; NULL (default) derives one from target.

Details

- **x = publication year** (Sugiyama layers); the **y-axis has no intrinsic meaning** (a branching position).
- **Line width** grows with cumulative tracked documents along the trajectory.
- A single-year **feeder** has no edge to draw as a line and is dropped (it would otherwise show a label/legend with no visible line); a single-year **central** (the selected final, e.g. a final-year-only community like `tr::c1g17`) is kept and drawn as a point.
- **Colour** is per trajectory. With a single targeted final the legend is summarised to `trN: p% (n)` – `n` of `trN`'s papers landed in that final, which is `p%` of `trN`'s size (the feeders' counts overlap, so they do not sum to the final's total). With `target = "all"` it shows the full split `trN -> c1g1:n, c1g2:n`. A central is `tr::cNgN (final, n)`, where `n` is the documents that **remained** in its final-year community (the last point of its size curve), not the cumulative documents the trajectory ever held.

Value

A ggplot object.

See Also

`plot_trajectory_lines_3d()`, `plot_trajectory_confluence()`, `sniff_trajectory_braid()`, `sniff_trajectory_confluence()`

Other visualization: `plot_groups_influence_matrix()`, `plot_groups_influence_network()`, `plot_groups_lineage_2d()`, `plot_groups_lineage_3d()`, `plot_groups_map()`, `plot_groups_map_animation()`, `plot_groups_map_interactive()`, `plot_groups_per_year()`, `plot_trajectory_confluence()`, `plot_trajectory_confluence_interactive()`, `plot_trajectory_confluence_matrix()`, `plot_trajectory_dag()`, `plot_trajectory_dag_interactive()`, `plot_trajectory_dispersion()`, `plot_trajectory_dynamics()`, `plot_trajectory_dynamics_interactive()`, `plot_trajectory_formation()`, `plot_trajectory_lines_3d()`

Examples

```
## Not run:
flow <- sniff_trajectory_braid(docs_per_group)
plot_trajectory_lines_2d(flow, min_n = 20)
plot_trajectory_lines_2d(flow, target = "tr::c1g1")

## End(Not run)
```

plot_trajectory_lines_3d

Plot the global trajectories as 3D variable-width lines

Description

The 3D counterpart of `plot_trajectory_lines_2d()`. Every global trajectory is drawn over the year-aware Sugiyama layout as a growing-thickness line: **x** = **publication year**, **y** = **route** (the Sugiyama branching coordinate that separates trajectories), and **z** = **cumulative tracked documents** along the path. The central finals `tr:cNgN` and the intermediate `trN` that feed them share one colour-per-trajectory scheme; the legend reports, for every intermediate, how many papers of its terminal cohort reached each final (a single `trN` can feed more than one final, which the legend makes explicit). Selection and thresholds are identical to `plot_trajectory_lines_2d()`; only the rendering adds the document z-axis and interactive hover.

Usage

```
plot_trajectory_lines_3d(
  flow,
  target = "all",
  conf = NULL,
  min_n = 5,
  min_prop = 0.05,
  min_total_size = 0,
  min_duration_years = 0,
  min_target_n = 0,
  width_range = c(4, 12),
  use_raw_papers = FALSE,
  log_scale = FALSE,
  label_size = 16,
  hover_font_size = 12,
  dest_min_prop = 0.05,
  lowlight_alpha = 0.75,
  lowlight_color = "#9AA5B1",
  lowlight_width = 3,
  show_background = TRUE,
  year_range = NULL,
  palette = NULL,
  title = NULL,
  legend_position = c("bottom", "right", "top", "left", "none"),
  legend_ncol = NULL,
  labels_text = NULL
)
```

Arguments

flow	A <code>sniff_trajectory_braid()</code> object, or a docs_per_group tibble / <code>sniff_trajectory_dag()</code> object the flow is built from.
target	What to draw. "all" (default) draws every central and the trajectories feeding it; a central traj_id (e.g. "tr::c1g1") or a vector of them draws only those finals and their feeders.
conf	An optional precomputed <code>sniff_trajectory_confluence()</code> object for flow; pass it to skip recomputing the confluence on every call. NULL (default) computes it.
min_n, min_prop, min_total_size, min_duration_years	Feeder thresholds, passed to the same pruning used by <code>plot_trajectory_lines_2d()</code> : an intermediate is drawn only if it transfers $\geq$ min_n papers and $\geq$ min_prop of its cohort, has total size $\geq$ min_total_size and lifespan $\geq$ min_duration_years; a pruned intermediate takes its own feeders with it.
min_target_n	Minimum papers an intermediate must leave in the selected final(s) by the last year to be drawn (default 0, no filter). Only applies to a targeted final view.
width_range	Line-width range c(min, max) (default c(4, 12)), scaled per trajectory by its cumulative tracked documents.
use_raw_papers	Scale z and width by raw quantity_papers (TRUE) or by the weighted quantity_papers * prop_tracked_intra_group (FALSE, default).
log_scale	Apply log1p() to the z-axis (cumulative documents), which keeps large trajectories from dwarfing small ones (default FALSE).
label_size	Font size of the trN / tr::cNgN end labels (default 16).
hover_font_size	Font size for hover tooltips (default 12).
dest_min_prop	In the target = "all" and single-target views, the minimum share of an intermediate's terminal cohort that must reach a final for that final to appear in its legend split (default 0.05). In the single-target view the targeted final is always kept regardless of this threshold.
lowlight_alpha, lowlight_color, lowlight_width	Opacity, colour and width of the light-grey links between trajectories.
show_background	Draw the light-grey links between trajectories – the edges of the DAG that are not on any single trajectory's path (the hand-offs and forks), at the nodes' real heights, as in <code>plot_trajectory_lines_2d()</code> (default TRUE).
year_range	Optional c(from, to) years fixing the x-axis range, so a short trajectory can be shown on the same window as a long one. NULL (default) fits the drawn trajectories.
palette	Optional named colour vector overriding the default hues (keyed by trajectory traj_id).
title	Plot title; NULL (default) derives one from target.
legend_position	Where to place the legend: "bottom" (default, matching <code>plot_trajectory_lines_2d()</code>), "right", "top", "left", or "none" to hide it.

legend_ncol	Number of columns in the legend (default NULL, plotly chooses). Applies to the horizontal legends ("bottom" / "top"): each entry is given 1 / legend_ncol of the width, so legend_ncol entries fit per row. Ignored for the vertical legends ("right" / "left").
labels_text	Optional data.frame with columns id and text mapping group ids (e.g. "c1g1") to descriptions, as in plot_trajectory_confluence() . When supplied, hovering a trajectory's nodes shows the description of its group beneath the feeding legend. NULL (default) shows the bare group ids.

Details

- **z = cumulative tracked documents** along each trajectory (raw or weighted), accumulated node by node; log_scale = TRUE compresses it.
- **Line width** grows per trajectory with the same cumulative measure, so each line is thinnest at its birth and thickest at its terminal node (the width is rescaled within each trajectory, matching [plot_trajectory_lines_2d\(\)](#)).
- A single-year **central** (e.g. a final-year-only community like tr::c1g17) has no segment to draw and is rendered as a terminal ball; a single-year **feeder** is dropped (it would show a label with no visible line).
- **Colour** is per trajectory and the legend carries the feeding split: a central reads tr::cNgN (final, n) (n = documents that remained in its final-year community), an intermediate reads trN -> c1g1:n, c1g2:n (its terminal cohort split across finals). In a single-target view it reads trN: p% (n) -> c1g15:n, c1g1:n, ... - p% and the parenthesised (n) are the share and the document count that reached the target (as in [plot_trajectory_lines_2d\(\)](#)); the split then shows the feeder's full destination, so a feeder of c1g1 reveals that more of it may flow elsewhere.
- **Hovering a node** shows that same feeding-legend text, the trajectory's labels_text description (when supplied), the node's year, and the absolute documents in that cluster-year (quantity_papers, not the running cumulative that sets the height).

Value

A plotly 3D plot object.

See Also

[plot_trajectory_lines_2d\(\)](#), [sniff_trajectory_braid\(\)](#), [sniff_trajectory_confluence\(\)](#)

Other visualization: [plot_groups_influence_matrix\(\)](#), [plot_groups_influence_network\(\)](#), [plot_groups_lineage_2d\(\)](#), [plot_groups_lineage_3d\(\)](#), [plot_groups_map\(\)](#), [plot_groups_map_animation\(\)](#), [plot_groups_map_interactive\(\)](#), [plot_groups_per_year\(\)](#), [plot_trajectory_confluence\(\)](#), [plot_trajectory_confluence_interactive\(\)](#), [plot_trajectory_confluence_matrix\(\)](#), [plot_trajectory_dag\(\)](#), [plot_trajectory_dag_interactive\(\)](#), [plot_trajectory_dispersion\(\)](#), [plot_trajectory_dynamics\(\)](#), [plot_trajectory_dynamics_interactive\(\)](#), [plot_trajectory_formation\(\)](#), [plot_trajectory_lines_2d\(\)](#)

Examples

```
## Not run:
flow <- sniff_trajectory_braid(docs_per_group)
```

```
plot_trajectory_lines_3d(flow, min_n = 20)
plot_trajectory_lines_3d(flow, target = "tr::c1g1")

## End(Not run)
```

read_openalex	<i>Read and Process OpenAlex data</i>
---------------	---------------------------------------

Description

Parse datasets exported from **OpenAlex** in two ways: (1) a CSV file exported in the browser, or (2) a data frame obtained via the {openalexR} API helpers. The function standardizes fields to common bibliographic tags (e.g., AU, SO, CR, PY, DI) and returns a tidy tibble.

Usage

```
read_openalex(file, format = "csv")
```

Arguments

file	For format = "csv", a character string with a local path or an HTTP(S) URL to a CSV export. For format = "api", a data frame produced by {openalexR} for the works entity.
format	Either "csv" (CSV export) or "api" (data frame from {openalexR}).

Details

CSV mode (format = "csv"):

- If file is a URL, it is downloaded to a temporary file before parsing (a progress message is printed).
- Selected fields are mapped to standardized tags: id_short (short OpenAlex ID), SR (= id_short), PY (= publication_year), TI (= title), DI (= doi), DT (= type), DE (= keywords.display_name), AB (= abstract), AU (= authorships.author.display_name), SO (= locations.source.display_name), C1 (= authorships.countries), TC (= cited_by_count), SC (= primary_topic.field.display_name), CR (= referenced_works, with the https://openalex.org/ prefix stripped), and DB = "openalex_csv".
- PY is coerced to numeric; a helper column DI2 (uppercase, punctuation-stripped variant of DI) is added; columns with all-caps tags are placed first and DI2 is relocated after DI.

API mode (format = "api"):

- file must be a data frame containing at least column id; typically this is returned by openalexR::oa_request() + openalexR::oa2df() or similar.
- Records are filtered to type %in% c("article", "review") and deduplicated by id.
- The function derives:
 - id_short (= id without the https://openalex.org/ prefix) and SR (= id_short);

- CR: concatenated short IDs from referenced_works (semicolon-separated);
 - DE: concatenated keyword names (lower case) from keywords;
 - AU: concatenated author names (upper case) from authorships;
 - plus core fields PY (= publication_year), TC (= cited_by_count), TI (= title), AB (= abstract), DI (= doi), and DB = "openalex_api".
- The result keeps one row per id and may include original columns from the input (via a right join), after constructing the standardized fields above.

Value

A tibble with standardized bibliographic columns. Typical output includes: id_short, AU, DI, CR, SO, DT, DE, AB, C1, TC, SC, SR, PY, and DB (source flag: "openalex_csv" or "openalex_api"). See **Details**.

Supported inputs

- format = "csv" — a local path or an HTTP(S) URL to an OpenAlex CSV export.
- format = "api" — a **data frame** produced by {openalexR} for the **works** entity (with the usual OpenAlex columns, including list-columns such as keywords, authorships, and referenced_works).

See Also

OpenAlex R client: [oa_request](#), [oa2df](#). Importers for Web of Science: [read_wos](#).

Examples

```
## Not run:
## CSV export (local path)
x <- read_openalex("openalex-works.csv", format = "csv")

## Using the API with openalexR
library(openalexR)
url_api <- "https://api.openalex.org/works?page=1&filter=primary_location.source.id:s121026525"
df_api <- openalexR::oa_request(query_url = url_api) |>
  openalexR::oa2df(entity = "works")
y <- read_openalex(df_api, format = "api")

## End(Not run)
```

read_wos

Read Web of Science exported files

Description

Parse Web of Science (WoS) export files in multiple formats and return a tidy table. The function automatically dispatches to a specialized parser based on the format argument and can also **download from a URL** if file points to an http:// or https:// resource.

Usage

```
read_wos(file, format = "bib", normalized_names = TRUE)
```

Arguments

file	Character scalar or vector. Path(s) to a WoS export file, or a single URL (http:// or https://) pointing to a WoS export.
format	Character scalar. Export format; one of "bib", "ris", "txt-plain-text", or "txt-tab-delimited".
normalized_names	Logical. If TRUE (default), use standardized column names when possible; if FALSE, keep original WoS field tags.

Details

- file may be a single path/URL or a **vector of paths**; multiple files will be combined row-wise when applicable.
- When file is a URL, the file is downloaded to a temporary path before parsing (a progress message is printed).
- If normalized_names = TRUE, selected WoS tags are mapped to standardized names (e.g., AU -> author, TI -> title, PY -> year, DI -> doi, DE -> keywords, SR -> unique_id, etc.; the exact mapping depends on the format). Otherwise, original field tags are preserved.
- The output includes:
 - DI2: an uppercase, punctuation-stripped variant of DI (if present),
 - PY: coerced to numeric (when present),
 - DB: a provenance flag indicating the source/format and whether names were normalized.
- Columns with ALL-CAPS tags (e.g., AU, TI, PY) are placed first, followed by other columns, and DI2 is relocated just after DI.

Value

A tibble with the parsed WoS records. See **Details** for notes on added/coerced columns (DI2, PY, DB) and column ordering.

Supported formats

- "bib" — BibTeX export
- "ris" — RIS export
- "txt-plain-text" — Plain-text export
- "txt-tab-delimited" — Tab-delimited export

See Also

Internal parsers used by this function: [read_wos_bib](#), [read_wos_ris](#), [read_wos_plain](#), [read_wos_tab](#).

Examples

```
bib_file <- system.file("extdata", "sample_wos.bib", package = "birddog")
M <- read_wos(bib_file, format = "bib", normalized_names = TRUE)
head(M)

## Not run:
# load data from a URL
M <- read_wos("https://example.com/savedrecs.bib", format = "bib")

## End(Not run)
```

sniff_cct

Calculate Citation Cycle Time (CCT) indicator

Description

Calculates the Citation Cycle Time (CCT) to measure the pace of scientific or technological progress in a publication network. Based on Kayal (1999), the indicator measures the median age of cited publications, where lower values indicate faster knowledge replacement cycles.

Usage

```
sniff_cct(
  network,
  scope = "groups",
  start_year = NULL,
  end_year = NULL,
  tracked_cr_py = NULL,
  batch_size = 50,
  min_papers_per_year = 3,
  rolling_window = NULL,
  normalize = "none",
  dispersion = FALSE,
  citation_scope = "global"
)
```

Arguments

network	Required. Network object containing publication data. For scope = "groups": object returned by sniff_groups(). For scope = "network": network object (tbl_graph or igraph).
scope	Analysis scope. Either "groups" (default) for separate group analysis or "network" for complete network analysis.
start_year, end_year	Start and end years for temporal analysis. If not specified, uses minimum and maximum years found in the data.

tracked_cr_py	Pre-processed citation year data (optional). A tibble with columns CR (OpenAlex work ID) and CR_PY (publication year). If provided, skips fetching data from OpenAlex API. Useful for avoiding repeated API calls.
batch_size	For OpenAlex data: number of IDs to process per API call (default: 50). Smaller batches help avoid API rate limits, larger batches process data faster but may trigger rate limiting.
min_papers_per_year	Minimum number of papers required in a given year to compute CCT. Years with fewer papers are reported as NA (default: 3).
rolling_window	Optional integer for rolling window smoothing. If provided, CCT values are smoothed using a centered moving average of the specified width (e.g., 3 for a 3-year window). Default is NULL (no smoothing).
normalize	Normalization mode. Either "none" (default) for raw CCT values in years, or "domain" to divide each group's annual CCT by the per-year network median across all groups. With normalize = "domain", values around 1 indicate average cycle time for the domain; values below 1 = short-cycle relative to the domain; values above 1 = long-cycle. Only applies when scope = "groups".
dispersion	Logical. If TRUE, the returned data tibble includes p25 and p75 columns (25th and 75th percentiles of per-paper citation-age medians), useful for detecting bimodal citation-age distributions and wide within-group heterogeneity. Default is FALSE.
citation_scope	Either "global" (default) to count all references, or "local" to restrict references to those that are also nodes in the birddog network (i.e., intra-corpus citations). Local CCT captures the renewal pace of the domain's own knowledge stock and is closer to Lee's (2024) sector-level TCT. Currently supported for OpenAlex data only; Web of Science data falls back to "global" with a warning.

Details

The Citation Cycle Time (CCT) is calculated following Kayal (1999):

1. Extract citation IDs from the network's CR column
2. Fetch publication years for cited works from OpenAlex API using `get_openalex_fields()`
3. For each publication, calculate the age of each cited reference (PY - CR_PY)
4. Calculate the median citation age per publication
5. For each year, calculate the median of per-publication medians across all publications in that year (annual mode)

Lower CCT values indicate that publications are citing more recent work, suggesting a faster pace of knowledge replacement. A sudden drop in CCT within a group signals potential scientific emergence.

The function automatically handles:

- Splitting semicolon-separated citation IDs
- Batch processing of OpenAlex API requests

- Filtering invalid citations (where cited work was published after citing work)
- Skipping years with too few papers (min_papers_per_year)
- Optional rolling window smoothing for noisy time series
- Creating temporal plots for each group

Value

A list with the following components:

data	Tibble with CCT data containing columns: group, year, index. When dispersion = TRUE, also includes p25 and p75 columns with the 25th and 75th percentiles of per-paper citation-age medians.
plots	Named list of plotly objects showing temporal evolution of CCT for each group. Each plot shows both absolute CCT values and year-over-year differences.
years_range	Named vector with start_year and end_year used in the analysis
tracked_cr_py	Citation year data with columns CR and CR_PY. Can be saved and reused in subsequent analyses to avoid repeated API calls.

References

Kayal AA, Waters RC. An empirical evaluation of the technology cycle time indicator as a measure of the pace of technological progress in superconductor technology. IEEE Transactions on Engineering Management. 1999;46(2):127-31. doi:10.1109/17.759138

See Also

[sniff_groups\(\)](#), [get_openalex_fields\(\)](#), [indexes_plots\(\)](#)

Examples

```
## Not run:
# Group analysis
results <- sniff_cct(network_groups, scope = "groups")

# Network analysis
results_network <- sniff_cct(complete_network, scope = "network")

# With rolling window smoothing
results_smooth <- sniff_cct(
  network_groups,
  scope = "groups",
  rolling_window = 3
)

# Domain-normalized CCT (values around 1; <1 = short-cycle relative to domain)
results_norm <- sniff_cct(
  network_groups,
  scope = "groups",
  normalize = "domain"
```

```

)

# With dispersion (adds p25 and p75 columns to detect bimodal distributions)
results_disp <- sniff_cct(
  network_groups,
  scope = "groups",
  dispersion = TRUE
)

# Local citations only (intra-corpus references)
results_local <- sniff_cct(
  network_groups,
  scope = "groups",
  citation_scope = "local"
)

# Accessing results
cct_data <- results$data
plots <- results$plots
plots$c1g1 # View plot for specific group

# Reuse citation data to avoid repeated API calls
saved_citations <- results$tracked_cr_py
results2 <- sniff_cct(
  network_groups,
  tracked_cr_py = saved_citations
)

## End(Not run)

```

sniff_components	<i>Identify and Analyze Network Components</i>
------------------	--

Description

Detects connected components in a citation network and computes summary statistics for each component. Returns both the component information and an updated network with component labels.

Usage

```
sniff_components(net)
```

Arguments

net	A network object (tbl_graph or igraph) generated by sniff_network()
-----	---

Value

A list with two elements:

components A tibble with component statistics containing:

- **component**: Component identifier (e.g., "c1")
- **quantity_publications**: Number of publications in component
- **average_age**: Mean publication year of component

network The input network with added component labels

See Also

Other groups (stock): [sniff_groups\(\)](#), [sniff_groups_attributes\(\)](#), [sniff_groups_cumulative\(\)](#), [sniff_groups_cumulative_citations\(\)](#), [sniff_groups_hubs\(\)](#), [sniff_groups_influence\(\)](#), [sniff_groups_lineage\(\)](#), [sniff_network\(\)](#)

Examples

```
## Not run:
# Create a network first
data <- read_wos("savedreecs.txt")
net <- sniff_network(data)

# Analyze components
result <- sniff_components(net)

# Access component information
result$components

# Get network with component labels
component_net <- result$network

## End(Not run)
```

sniff_entropy

Calculate Entropy Based on Keywords Over Time

Description

Computes the normalized Shannon entropy of keyword distributions from scientific publications over a specified time range. Entropy measures the diversity and evenness of keyword usage within research groups or the entire network.

Usage

```
sniff_entropy(
  network,
  scope = "groups",
  start_year = NULL,
  end_year = NULL,
  mode = "rolling",
  window_size = 5
)
```

Arguments

network	A network object to analyze. For scope = "groups", this should be the output of sniff_groups(). For scope = "network", this should be a tbl_graph or igraph object from sniff_network().
scope	Character specifying the analysis scope: "groups" for multiple groups or "network" for the entire network (default: "groups").
start_year	Starting year for entropy calculation. If NULL, uses the minimum publication year found in the network data.
end_year	Ending year for entropy calculation. If NULL, uses the maximum publication year found in the network data.
mode	Character specifying the temporal mode for entropy calculation: "annual" Uses only publications from each specific year (default). "cumulative" Uses all publications from the start up to each year. "rolling" Uses a sliding window of window_size years ending at each year.
window_size	Integer specifying the rolling window size in years (default: 5). Only used when mode = "rolling".

Details

The function calculates the normalized Shannon entropy (Pielou's evenness index) based on Shannon's information theory (Shannon, 1948). The temporal scope of keyword data depends on the mode parameter:

- **annual:** entropy from keywords published in each specific year. Values tend to be high (near 1) since within-year distributions are typically even.
- **cumulative:** entropy from all keywords published up to each year. Shows long-term trends in thematic concentration as the keyword pool grows.
- **rolling:** entropy from a sliding window of recent years. Balances sensitivity to recent shifts with enough data for stable estimates.

The normalized entropy (Pielou's J') is calculated as:

$$J' = \frac{H}{H_{max}} = \frac{-\sum_{i=1}^n p_i \log_2 p_i}{\log_2 n}$$

where p_i is the relative frequency of keyword i , n is the number of unique keywords, and $H_{max} = \log_2 n$ is the maximum possible entropy for n categories.

Entropy values range from 0 to 1, where:

- 0 indicates minimal diversity (one dominant keyword)
- 1 indicates maximal diversity (all keywords equally frequent)

A sudden increase in entropy may signal the emergence of new research topics, while a decrease suggests thematic convergence.

Value

A list with three components:

data	A tibble containing entropy values for each group and year
plots	A list of plotly objects visualizing entropy trends for each group
years_range	A vector with the start_year and end_year used in calculations

References

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 379-423. doi:10.1002/j.15387305.1948.tb01338.x

Pielou, E. C. (1966). The measurement of diversity in different types of biological collections. *Journal of Theoretical Biology*, 13, 131-144.

See Also

[sniff_groups](#), [sniff_network](#), [indexes_plots](#)

Examples

```
## Not run:
# Rolling window (default: 5 years)
entropy_results <- sniff_entropy(groups_data, scope = "groups")

# Cumulative mode
entropy_results <- sniff_entropy(groups_data, mode = "cumulative")

# Annual mode
entropy_results <- sniff_entropy(groups_data, mode = "annual")

# Rolling window with custom size
entropy_results <- sniff_entropy(groups_data, mode = "rolling", window_size = 3)

# Access results
entropy_data <- entropy_results$data
entropy_plots <- entropy_results$plots

## End(Not run)
```

sniff_groups

Detect and analyze groups in a scientific network

Description

This function identifies and analyzes groups (communities) within scientific networks created from articles and patents data. It can apply different clustering algorithms to detect technological trajectories and emerging scientific fields.

Usage

```
sniff_groups(
  comps,
  min_group_size = 10,
  keep_component = c("c1"),
  cluster_component = c("c1"),
  algorithm = "fast_greedy",
  seed = 888L
)
```

Arguments

<code>comps</code>	A list containing network components, typically generated by <code>sniff_components()</code> . Must include a network object with 'component' and 'PY' (publication year) vertex attributes.
<code>min_group_size</code>	Minimum size for a group to be included in results (default = 10). Groups with fewer members will be filtered out.
<code>keep_component</code>	Character vector specifying which network components to process (default = "c1"). Can include multiple components.
<code>cluster_component</code>	Character vector specifying which components should be clustered (default = "c1"). Components not listed here will be treated as single groups.
<code>algorithm</code>	Community detection algorithm to use (default = "fast_greedy"). Options include: "louvain", "walktrap", "edge_betweenness", "fast_greedy", or "leiden".
<code>seed</code>	Random seed for reproducible results (default = 888L). Only applies to algorithms that use random initialization like Louvain.

Details

The function first validates the input network, then applies the specified clustering algorithm to detect communities within the network. It calculates statistics for each detected group and returns the results along with the augmented network. The function can handle multiple network components simultaneously, applying clustering only to specified components.

Value

A list with three elements:

- `aggregate`: A data frame with group statistics including group name, number of papers, and average publication year
- `network`: The input network with added group attributes
- `pubs_by_year`: Publication counts by group and year

See Also

`sniff_components()` for creating the input network components

Other groups (stock): `sniff_components()`, `sniff_groups_attributes()`, `sniff_groups_cumulative()`, `sniff_groups_cumulative_citations()`, `sniff_groups_hubs()`, `sniff_groups_influence()`, `sniff_groups_lineage()`, `sniff_network()`

Examples

```
## Not run:
# Assuming 'comps' is output from sniff_components()
groups <- sniff_groups(comps,
  min_group_size = 15,
  algorithm = "leiden",
  seed = 888L
)

# Access group statistics
groups$aggregate
groups$network
groups$pubs_by_year

## End(Not run)
```

`sniff_groups_attributes`

Calculate and Visualize Group Attributes from Scientific Networks

Description

This function analyzes publication growth rates and other attributes for research groups identified in scientific networks. It calculates growth rates using exponential models, creates horizon plots for visualization, and generates summary tables.

Usage

```
sniff_groups_attributes(
  groups,
  growth_rate_period = 2010:2022,
  horizon_plot = TRUE,
  show_results = TRUE,
  assign_result = NULL
)
```

Arguments

<code>groups</code>	A list containing network data with publications by year and group information. Must include elements: <code>network</code> , <code>pubs_by_year</code> , and <code>aggregate</code> .
<code>growth_rate_period</code>	Numeric vector of years to use for growth rate calculation (default: 2010:2024).
<code>horizon_plot</code>	Logical indicating whether to include horizon plots in the output table (default: TRUE).
<code>show_results</code>	Logical indicating whether to print results to console (default: TRUE).
<code>assign_result</code>	Character string specifying a variable name to assign the results to in the global environment (default: NULL).

Details

The function performs the following steps:

1. Calculates growth rates using exponential models for each group
2. Processes publication age and doubling time metrics
3. Optionally creates horizon plots for each group's publication trend
4. Generates a comprehensive summary table

Value

A list with two components:

- `attributes_table`: A gt table showing group attributes including growth rates
- `regression`: A list of model summaries for each group's growth rate calculation

See Also

Other groups (stock): [sniff_components\(\)](#), [sniff_groups\(\)](#), [sniff_groups_cumulative\(\)](#), [sniff_groups_cumulative_citations\(\)](#), [sniff_groups_hubs\(\)](#), [sniff_groups_influence\(\)](#), [sniff_groups_lineage\(\)](#), [sniff_network\(\)](#)

Examples

```
## Not run:
# Assuming groups is output from sniff_groups()
groups_attributes <- sniff_groups_attributes(groups,
  growth_rate_period = 2010:2022,
  horizon_plot = TRUE
)

# View the results table
print(groups_attributes$attributes_table)

# Access model summaries
groups_attributes$regression

## End(Not run)
```

sniff_groups_cumulative

Analyze Cumulative Network Groups Over Time

Description

Performs cumulative community detection on a network over specified time spans, returning group statistics and keyword analysis for each time period.

Usage

```
sniff_groups_cumulative(
  comps,
  time_span = NULL,
  min_group_size = 10,
  keep_component = c("c1"),
  cluster_component = c("c1"),
  top_n_keywords = 10,
  algorithm = "fast_greedy",
  seed = 888L
)
```

Arguments

comps	A list containing network components, typically generated by sniff_components() . Must include a network object with 'component' and 'PY' (publication year) vertex attributes.
time_span	Numeric vector of years to analyze (default: 2000:2024).
min_group_size	Minimum size for a cluster to be retained (default = 10).

keep_component Character vector specifying which network components to process (default = "c1"). Can include multiple components.

cluster_component Character vector specifying which components should be clustered (default = "c1"). Components not listed here will be treated as single groups.

top_n_keywords Number of top keywords to extract per group (default = 10).

algorithm Community detection algorithm to use. One of: "louvain", "walktrap", "edge_betweenness", "fast_greedy" (default), or "leiden".

seed Random seed for reproducible results (default = 888L). Only applies to algorithms that use random initialization like Louvain.

Value

A named list (by year) where each element contains:

groups A tibble with group statistics and top keywords

documents A tibble mapping documents to groups

network The cumulative network up to that year

See Also

Other groups (stock): [sniff_components\(\)](#), [sniff_groups\(\)](#), [sniff_groups_attributes\(\)](#), [sniff_groups_cumulative_citations\(\)](#), [sniff_groups_hubs\(\)](#), [sniff_groups_influence\(\)](#), [sniff_groups_lineage\(\)](#), [sniff_network\(\)](#)

Examples

```
## Not run:
# Typical pipeline:
data <- read_wos("savedrecs.txt")
net <- sniff_network(data)
comps <- sniff_components(net)

# Cumulative analysis
groups_cumulative <- sniff_groups_cumulative(
  comps,
  time_span = 2010:2020,
  keep_component = c("c1", "c2"),
  cluster_component = c("c1"),
  algorithm = "leiden",
  seed = 888L
)

# Access results for 2015
groups_cumulative[["network_until_2015"]]$groups

## End(Not run)
```

```
sniff_groups_cumulative_citations
```

Calculate Cumulative Citations by Group and Year

Description

This function calculates cumulative citations for papers within research groups, tracking how citations accumulate over time for highly cited papers.

Usage

```
sniff_groups_cumulative_citations(groups, min_citations = 5)
```

Arguments

<code>groups</code>	A list containing network data with the following components: • <code>network</code>: A tidygraph network object • <code>pubs_by_year</code>: Publication counts by year • <code>aggregate</code>: Aggregate network statistics
<code>min_citations</code>	Minimum number of citations for a paper to be included in analysis (default: 10).

Details

For each research group, the function:

- Identifies papers with citations above the threshold
- Tracks citations to these papers year by year
- Calculates cumulative citation patterns
- Computes various growth metrics for citation analysis

Works with both Web of Science (WOS) and OpenAlex data formats.

Value

A named list (by research group) where each element contains a tibble with:

- `group`: Research group identifier
- `SR`: Paper identifier
- `TC`: Total citations
- `PY`: Publication year
- `Ki`: Total network citations
- `citations_by_year`: A tibble with annual citation counts (PY: year, citations: count)
- `growth_power`: Growth power score (0-100)

- growth_consistency: Percentage of years with citations
- peak_momentum: Highest 3-year rolling average citation count
- early_impact: Citations in first 5 years
- recent_momentum: Citations in last 3 years
- acceleration_factor: Ratio of late to early citations

See Also

Other groups (stock): `sniff_components()`, `sniff_groups()`, `sniff_groups_attributes()`, `sniff_groups_cumulative()`, `sniff_groups_hubs()`, `sniff_groups_influence()`, `sniff_groups_lineage()`, `sniff_network()`

Examples

```
## Not run:
# Assuming groups is output from sniff_groups()
# Calculate cumulative citations
groups_cumulative_citations <- sniff_groups_cumulative_citations(groups, min_citations = 5)
# View results for first group
head(groups_cumulative_citations[[1]])

## End(Not run)
```

sniff_groups_hubs	<i>Identify Hub Papers in Research Groups</i>
-------------------	---

Description

This function analyzes citation networks to identify hub papers within research groups based on their citation patterns. It calculates several metrics (Z_i , P_i) to classify papers into different hub categories.

Usage

```
sniff_groups_hubs(groups, min_citations = 1)
```

Arguments

groups	A list containing network data with the following components: • network: A tidygraph network object • pubs_by_year: Publication counts by year • aggregate: Aggregate network statistics
min_citations	Minimum number of citations for a paper to be considered (default: 1)

Details

The function classifies papers into hub categories based on:

- R5: Knowledge hubs ($Z_i \geq 2.5$ and $P_i \leq 0.3$)
- R6: Bridging hubs ($Z_i \geq 2.5$ and $0.3 < P_i \leq 0.75$)
- R7: Boundary-spanning hubs ($Z_i \geq 2.5$ and $P_i > 0.75$)

Value

A tibble containing:

- group: Research group identifier
- SR: Paper identifier
- TC: Total citations
- Ki: Total citations from all groups
- ki: Citations from within the same group
- Zi: Standardized within-group citation score
- Pi: Citation diversity index
- zone: Hub classification ("noHub", "R5", "R6", "R7")

See Also

Other groups (stock): [sniff_components\(\)](#), [sniff_groups\(\)](#), [sniff_groups_attributes\(\)](#), [sniff_groups_cumulative\(\)](#), [sniff_groups_cumulative_citations\(\)](#), [sniff_groups_influence\(\)](#), [sniff_groups_lineage\(\)](#), [sniff_network\(\)](#)

Examples

```
## Not run:

# Assuming 'groups' is output from sniff_groups()

# Identify hub papers
hubs <- sniff_groups_hubs(groups, min_citations = 5)

# View results
head(hubs)

## End(Not run)
```

sniff_groups_influence

Measure directed citation influence between research groups

Description

Experimental. Lifts the internal citations of a `sniff_groups()` network to the group level and measures how much each group's output flows into every other group's. Influence is **directional**: if group B 's papers cite group A 's papers, knowledge flows $A \rightarrow B$. The function returns the cross-citation matrix, four size-corrected indices per ordered pair, the net flow between pairs, and a source / broker / sink role per group.

Usage

```
sniff_groups_influence(groups, self = TRUE, null_reps = 0, seed = NULL)
```

```
is_influence(x)
```

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

Arguments

groups	A <code>sniff_groups()</code> object: a list with a network component, a directed tidygraph/igraph whose nodes carry a group attribute and whose edges are the internal citations (citing -> cited).
self	Keep the diagonal (intra-group citations) in the matrix and in the out/in strengths. Default TRUE. FALSE studies only between-group flow.
null_reps	Number of group-label permutations for a per-channel p-value: shuffle the group labels over documents, recompute the cross-citation matrix, and record how often the random flow meets or beats the observed count. Default 0 (no p-values). When > 0, flow gains a p_value column.
seed	Optional integer seed for the permutation null, for reproducibility. Default NULL.
x	A <code>birddog_influence</code> object.
...	Ignored.

Details

Writing C_{ij} for the number of citations from group i to group j (rows cite columns), out-strength $o_i = \sum_k C_{ik}$, in-strength $\iota_j = \sum_k C_{kj}$, and total $m = \sum_{ij} C_{ij}$, each ordered pair carries four normalizations:

- **debt** C_{ij}/o_i – the share of i 's citations owed to j (how much i leans on j);
- **audience** C_{ij}/ι_j – the share of j 's citations coming from i ;
- **salton** $C_{ij}/\sqrt{o_i \iota_j}$ – a size-free channel strength, symmetric in the pair;

- **surprise** $C_{ij}/(o_i \iota_j/m)$ – the flow against the configuration-model expectation; > 1 over-represented, < 1 under-represented.

The **net influence** $\nu_{ij} = C_{ij} - C_{ji}$ removes reciprocal flow, and the group **balance** $\beta_i = \iota_i - o_i$ (received minus made) classifies each group as a *source* ($\beta > 0$, foundational), a *sink* ($\beta < 0$, frontier consumer), or a *broker* ($\beta \approx 0$). The balances of a closed system sum to zero. The diagonal C_{ii} is intra-group cohesion; `self = FALSE` drops it before any normalization (the balances and net are unchanged, since the diagonal cancels).

Value

An object of class `birddog_influence`, a list with:

- `matrix`: the $G \times G$ cross-citation matrix C (rows cite columns), groups in `mixed_sort()` order.
- `flow`: a tibble, one row per observed ordered pair, sorted by descending surprise – influencer (cited group), recipient (citing group), citations, debt, audience, salton, surprise (and `p_value` when `null_reps > 0`).
- `groups`: per-group received, made, balance, role (source / broker / sink), sorted by descending balance.
- `net`: the net-influence edge list – from (source), to (recipient), net ($\nu_{ij} > 0$).
- `params`: the call settings.

See Also

`sniff_groups()`, `sniff_groups_hubs()`, `plot_groups_influence_matrix()`, `plot_groups_influence_network()`

Other groups (stock): `sniff_components()`, `sniff_groups()`, `sniff_groups_attributes()`, `sniff_groups_cumulative()`, `sniff_groups_cumulative_citations()`, `sniff_groups_hubs()`, `sniff_groups_lineage()`, `sniff_network()`

Examples

```
## Not run:
groups <- sniff_groups(net)
infl <- sniff_groups_influence(groups)
infl
infl$groups      # source / broker / sink per group
infl$net         # who, on balance, leads whom
plot_groups_influence_matrix(infl)

## End(Not run)
```

sniff_groups_keywords *Extract representative keywords from grouped nodes*

Description

This function processes nodes grouped in a network (typically by community detection), and extracts the most frequent and the most distinctive keywords (using TF-IDF) from a descriptor field such as keywords or subject terms.

Usage

```
sniff_groups_keywords(net_groups, n_terms = 15, min_freq = 1, sep = ";")
```

Arguments

net_groups	A list containing a network component of class <code>tbl_graph</code> , where each node has at least two attributes: <code>group</code> and <code>DE</code> .
n_terms	Integer. The number of top terms to return per group, both by frequency and by TF-IDF. Default is 15.
min_freq	Integer. Minimum frequency a term must have in a group to be considered. Default is 2.
sep	Character. Separator used in the <code>DE</code> field to split multiple terms. Default is <code>" ; "</code> .

Value

A tibble with one row per group, containing two columns:

- `term_freq`: the most frequent terms (with raw frequency).
- `term_tfidf`: the most distinctive terms (with TF-IDF scores).

Examples

```
## Not run:  
# Assuming 'groups' is output from sniff_groups()  
groups_keywords <- sniff_groups_keywords(groups)  
  
## End(Not run)
```

sniff_groups_lineage *Per-group lineage data*

Description

This function analyzes the evolution of document groups over time to detect technological trajectories and scientific emergence patterns. It computes similarity measures between groups across time periods and tracks their attributes.

Usage

```
sniff_groups_lineage(
  groups_cumulative,
  min_group_size = 10,
  top_n_keywords = 3
)
```

Arguments

<code>groups_cumulative</code>	A list of cumulative group data over time, typically produced by other functions in the <code>birddog</code> package. Each element should contain network, documents, and groups data.
<code>min_group_size</code>	Minimum number of documents required for a group to be considered (default: 10). Smaller groups will be filtered out.
<code>top_n_keywords</code>	Number of top keywords to consider when analyzing group characteristics (default: 3).

Details

Prepares lineage data (`docs_per_group`, `groups_similarity`, `groups_attributes`); detects no trajectories – see [sniff_trajectory_braid\(\)](#).

Value

A list with three components:

- `groups_attributes`: A list of data frames containing attributes for each tracked group
- `groups_similarity`: A list of data frames containing Jaccard similarity measures between groups across time periods
- `docs_per_group`: A data frame containing document IDs for all groups across time periods

See Also

Other groups (stock): [sniff_components\(\)](#), [sniff_groups\(\)](#), [sniff_groups_attributes\(\)](#), [sniff_groups_cumulative\(\)](#), [sniff_groups_cumulative_citations\(\)](#), [sniff_groups_hubs\(\)](#), [sniff_groups_influence\(\)](#), [sniff_network\(\)](#)

Examples

```
## Not run:
# Assuming you have cumulative group data:
trajectories <- sniff_groups_lineage(groups_cumulative, min_group_size = 15)

## End(Not run)
```

sniff_groups_stm_prepare

Prepare Text Data and Analyze Topic Models

Description

Processes text data for structural topic modeling and performs topic number selection analysis, returning both the processed data and diagnostic plots.

Usage

```
sniff_groups_stm_prepare(
  groups,
  group_to_stm = "g01",
  search_topics = c(5:40, 45, 50, 55, 60),
  seed = 1234,
  cores = 1
)
```

Arguments

groups	A list containing network data with a 'network' component
group_to_stm	Character string specifying which research group to process (default: 'g01')
search_topics	Numeric vector of topic numbers to evaluate (default: c(5:40, 45, 50, 55, 60))
seed	Random seed for reproducibility (default: 1234)
cores	Number of CPU cores to use (default: 1)

Value

A list containing:

- result: The searchK results object
- plots: A list containing two ggplot objects (p1: metrics by K, p2: exclusivity vs coherence)
- df_prep: Output from stm::textProcessor
- df_doc: Output from stm::prepDocuments
- df: Original filtered data

Examples

```
## Not run:
output <- sniff_groups_stm_prepare(network_data)
output$plots$p1 # View first plot
output$result # Access search results

## End(Not run)
```

sniff_groups_stm_run *Run Structural Topic Modeling Analysis*

Description

Performs structural topic modeling on prepared text data and returns topic proportions and top documents for each topic.

Usage

```
sniff_groups_stm_run(groups_stm_prepare, k_topics = 12, n_top_documents = 50)
```

Arguments

groups_stm_prepare	A prepared STM object from sniff_groups_stm_prepare()
k_topics	Number of topics to model (default: 12)
n_top_documents	Number of top documents to each topic (default: 50)

Details

This function:

- Fits an STM model with specified number of topics
- Identifies top terms for each topic
- Calculates topic proportions
- Identifies top documents for each topic

Value

A list containing:

- topic_proportion2: Data frame with topic proportions and top terms
- tab_top_documents: Data frame of top documents for each topic

Examples

```
## Not run:
# Prepare data first
stm_data <- sniff_groups_stm_prepare(network_data)

# Run topic modeling
stm_results <- sniff_groups_stm_run(stm_data, k_topics = 15)

# Access results
stm_results$topic_proportion2 # Topic proportions and terms
stm_results$tab_top_documents # Top documents per topic

## End(Not run)
```

sniff_groups_terms	<i>Extract and Analyze Key Terms from Research Groups</i>
--------------------	---

Description

Identifies and extracts key terms from titles and abstracts of publications within different research groups using natural language processing techniques, and computes term statistics including TF-IDF scores.

Usage

```
sniff_groups_terms(
  net_groups,
  algorithm = "rake",
  phrase_pattern = "(A|N)*N(P+D*(A|N)*N)*",
  model_dir = tempdir(),
  n_cores = 1,
  show_progress = TRUE,
  n_terms = 15,
  min_freq = 2,
  digits = 4
)
```

Arguments

net_groups	A list containing network data with publication information. Must include elements: network (with vertex attributes 'group', 'TI', 'AB'), pubs_by_year, and aggregate.
algorithm	Term extraction algorithm to use. Options are: "rake" - Rapid Automatic Keyword Extraction (default) "pointwise" - Pointwise Mutual Information "phrase" - Phrase pattern matching

phrase_pattern	Regular expression pattern for phrase extraction when algorithm = "phrase" (default: "(A N)N(P+D(A N)N)")
model_dir	Directory where UDPipe models are stored (default: tempdir())
n_cores	Number of CPU cores to use for parallel processing (default: 1)
show_progress	Logical indicating whether to show progress bar (default: TRUE)
n_terms	Number of top terms to return in summary table (default: 15)
min_freq	Minimum frequency threshold for terms (default: 2)
digits	Number of decimal places to round numerical values (default: 4)

Details

This function performs the following steps:

1. Validates input structure and parameters
2. Loads the UDPipe language model from the specified directory
3. Processes text data (titles and abstracts) for each group
4. Applies the selected term extraction algorithm (RAKE, PMI, or phrase patterns)
5. Computes term frequencies and TF-IDF scores
6. Returns ranked terms for each research group with comprehensive statistics

The function uses UDPipe for tokenization, lemmatization and POS tagging before term extraction. For phrase extraction, the default pattern finds noun phrases.

Value

A list with two components:

- terms_by_group: A named list (by group) of data frames containing extracted terms with statistics
- terms_table: A summary tibble with top terms by frequency and TF-IDF for each group

Examples

```
## Not run:
# Assuming groups is output from sniff_groups()
terms <- sniff_groups_terms(groups, algorithm = "rake")

# View terms for first group
head(terms$terms_by_group[[1]])

# View summary table
print(terms$terms_table)

# Customized extraction with custom model directory
net_groups_terms <- sniff_groups_terms(net_groups,
  algorithm = "phrase",
  model_dir = tempdir(),
  n_terms = 10,
```

```

    min_freq = 3,
    n_cores = 4
)

## End(Not run)

```

sniff_key_route

Identify Key Routes in Citation Networks

Description

This function identifies and visualizes key citation routes within scientific networks by analyzing the most significant citation paths between publications. The algorithm implements the key-route search from the integrated main path analysis approach described in Liu & Lu (2012).

Usage

```

sniff_key_route(
  network,
  scope = "network",
  n_routes = 1,
  compact_gaps = FALSE,
  direction = "vertical"
)

```

Arguments

network	A network object of class <code>tbl_graph</code> or <code>igraph</code> containing citation data, or a list object generated by <code>sniff_groups()</code> when <code>scope = "groups"</code>
scope	Character string specifying the analysis scope. Must be either "network" (for full network analysis) or "groups" (for group-wise analysis of a grouped network)
n_routes	Positive integer specifying the number of key-route starting edges to use (default: 1). Each iteration selects the next-highest SPC edge as a new starting point and extends it into a full path. Higher values reveal auxiliary knowledge diffusion paths and divergence-convergence structures (see Liu & Lu, 2012, Figure 8).
compact_gaps	Logical. If TRUE, compresses the vertical axis by removing empty year gaps between publications. This produces shorter plots suitable for academic publications. Default is FALSE (chronological spacing).
direction	Character. Plot direction: "vertical" (default, top-to-bottom) or "horizontal" (left-to-right, oldest on the left).

Details

The function implements the key-route search from Liu & Lu (2012):

1. Computes Search Path Count (SPC) for each citation link using an efficient $O(V+E)$ algorithm based on topological sort. SPC measures how many source-to-sink paths traverse each link.
2. Selects the key-route: the link with the highest SPC value.
3. Searches forward from the end node of the key-route, greedily following the outgoing link with the highest SPC, until a sink is reached.
4. Searches backward from the start node of the key-route, greedily following the incoming link with the highest SPC, until a source is reached.

When $n_routes > 1$, the procedure is repeated: each iteration selects the edge with the next-highest SPC as a new starting point and extends it forward and backward. Routes can share nodes and edges, producing a divergence-convergence-divergence structure that reveals how knowledge flows through multiple sub-streams.

The SPC is computed as $forward[u] * backward[v]$ for each edge (u, v) , where $forward[u]$ counts paths from any source to u and $backward[v]$ counts paths from v to any sink (Batagelj, 2003). This guarantees the most significant link is always included in the key-route path.

Value

A list containing for each group:

- `plot` - A ggplot2 object visualizing the key citation route(s)
- `data` - A tibble with publication details (name, TI, AU, PY) of nodes in the key route(s)

References

Liu JS, Lu LYY. An integrated approach for main path analysis: Development of the Hirsch index as an example. *Journal of the American Society for Information Science and Technology*. 2012;63(3):528-542. doi:10.1002/asi.21692

Batagelj V. Efficient algorithms for citation network analysis. University of Ljubljana, Institute of Mathematics, Physics and Mechanics, Department of Theoretical Computer Science, Preprint Series. 2003;41:897.

Examples

```
## Not run:
# Single key-route (default)
result <- sniff_key_route(my_network, scope = "network")

# Multiple key-routes for richer structure
result <- sniff_key_route(my_network, scope = "network", n_routes = 5)

# Group-wise analysis
grouped_network <- sniff_groups(data)
result <- sniff_key_route(grouped_network, scope = "groups", n_routes = 3)

# Access results for a specific group
```

```

result$group_name$plot
result$group_name$data

## End(Not run)

```

sniff_network

Create Citation Networks from Bibliographic Data

Description

Constructs different types of citation networks from bibliographic data imported from Web of Science or OpenAlex using birddog's reading functions.

Usage

```
sniff_network(dataframe, type = "direct citation", external_references = FALSE)
```

Arguments

dataframe	A data frame imported via <code>read_openalex()</code> or <code>read_wos()</code>
type	Type of network to create. One of: "direct citation": Direct citation links between documents "bibliographic coupling": Documents linked by shared references
external_references	Logical indicating whether to include external references (references not in the original dataset) as nodes in the network

Value

A `tbl_graph` object from the `tidygraph` package representing the citation network. Node attributes include bibliographic information from the input data.

See Also

Other groups (stock): `sniff_components()`, `sniff_groups()`, `sniff_groups_attributes()`, `sniff_groups_cumulative()`, `sniff_groups_cumulative_citations()`, `sniff_groups_hubs()`, `sniff_groups_influence()`, `sniff_groups_lineage()`

Examples

```

## Not run:
# Using OpenAlex data
oa_data <- read_openalex("works.csv", format = "csv")
net <- sniff_network(oa_data, type = "direct citation")

# Using WoS data
wos_data <- read_wos("savedrecs.txt")

```

```
net <- sniff_network(wos_data, type = "bibliographic coupling", external_references = TRUE)

## End(Not run)
```

```
sniff_trajectory_braid
```

Braid: the canonical detector of trajectories (central + absorbed)

Description

Decomposes the cumulative-clustering DAG into trajectories via **trunk edges** (mutual dominance: $v \rightarrow u$ where u is v 's heaviest successor and v is u 's heaviest predecessor). Trunk edges form disjoint chains. A chain ending at a final-year group node is a **central** trajectory ($tr::<group>$, one per final group, reaches the final year); every other chain is **absorbed** ($tr1 \dots trN$), merging into an absorber (central or another absorbed) where its head flows out via a non-trunk edge. Absorption is transitive, forming a confluence forest.

Usage

```
sniff_trajectory_braid(x, min_group_size = 10, jaccard_min = 0.05, k_out = 2)
```

Arguments

x A [sniff_trajectory_dag\(\)](#) object, a [sniff_groups_lineage\(\)](#) object, or a docs_per_group tibble (the DAG is built internally).

min_group_size, jaccard_min, k_out Passed to [sniff_trajectory_dag\(\)](#) when **x** is not already a DAG object.

Value

A `birddog_flow` object: a classed list with trajectories (tibble: `traj_id`, `type` ("central"/"absorbed"), `group` (destination group; NA if extinct), `start`, `end`, `size` (distinct papers), `depth` (0 = central), `absorbed_into` (absorber `traj_id`, NA for central), `absorption_year` (NA for central), `nodes` list-col), `tree` (edges `child`, `parent`, `absorption_year`), `graph`, `docs_per_group`, `last_year`.

See Also

[sniff_trajectory_channel\(\)](#), [sniff_trajectory_dag\(\)](#), [sniff_trajectory_dynamics\(\)](#)

Other trajectory detection: [is_flow\(\)](#), [sniff_trajectory_channel\(\)](#), [sniff_trajectory_dag\(\)](#), [subset.birddog_flow\(\)](#), [validate_flow\(\)](#)

sniff_trajectory_cct *Per-year citation cycle time (renewal pace) along each flow trajectory*

Description

Recomputes the Citation Cycle Time (Kayal, 1999) from the documents in each [sniff_trajectory_braid\(\)](#) trajectory's own group-year nodes, one value per year of the trajectory's life: per node-year, the median over its documents of each document's median cited-reference age. A low or falling series marks a fast-renewing knowledge stock, a diagnostic signal of emergence.

Usage

```
sniff_trajectory_cct(flow, references)
```

Arguments

flow	A sniff_trajectory_braid() object.
references	A long data frame with columns <code>document_id</code> and <code>ref_age</code> , one row per document-cited-reference, <code>ref_age</code> the reference's age in years (<code>citing_year - cited_year</code>). Build it from the corpus CR field joined to the cited works' publication years (e.g. the <code>tracked_cr_py</code> of sniff_cct()).

Details

This is the faithful, per-document counterpart of [sniff_cct\(\)](#) (which scores a static [sniff_groups\(\)](#) partition by group-year): it follows the trajectory's actual membership, so an absorbed tributary is scored on its own documents rather than its destination group's.

Value

A tibble, one row per trajectory: `traj_id`, `type`, `group`, and `cct`, a list-column whose each cell is a tibble with columns `year` and `cct` (the median citation age over that node-year's documents; NA when none of them carry references).

References

Kayal AA, Waters RC (1999). An empirical evaluation of the technology cycle time indicator. IEEE Trans. Eng. Manag. 46(2):127-31.

See Also

[sniff_cct\(\)](#), [sniff_trajectory_entropy\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_braid\(\)](#)
 Other trajectory analysis: [sniff_trajectory_community\(\)](#), [sniff_trajectory_confluence\(\)](#),
[sniff_trajectory_contribution\(\)](#), [sniff_trajectory_destination\(\)](#), [sniff_trajectory_dynamics\(\)](#),
[sniff_trajectory_emergence_owners\(\)](#), [sniff_trajectory_entropy\(\)](#), [sniff_trajectory_formation\(\)](#),
[sniff_trajectory_hubs\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#)

sniff_trajectory_channel

Detect channel trajectories by global optimal-path routing

Description

An alternative to [sniff_trajectory_braid\(\)](#). Instead of flow's local mutual-dominance trunk edges, each node is routed to the successor on its globally cheapest path to a final-year node, where edge cost is $-\log(\text{weight})$ (the Jaccard). The minimum-cost path is the maximum-product Jaccard path (highest geometric-mean coherence). Final-year nodes root one in-tree each (the watershed of a final community); the cheapest birth $\rightarrow$ final chain is the **central** trajectory ($\text{tr}::\langle\text{group}\rangle$), the rest are **absorbed** tributaries ($\text{tr1} \dots \text{trN}$), mirroring the flow object exactly. "Flow" names the object kind, not the algorithm, so the result is a `birddog_flow` and passes [validate_flow\(\)](#).

Usage

```
sniff_trajectory_channel(x, min_group_size = 10, jaccard_min = 0.05, k_out = 2)
```

Arguments

`x` A [sniff_trajectory_dag\(\)](#) object, a [sniff_groups_lineage\(\)](#) object, or a `docs_per_group` tibble (the DAG is built internally).

`min_group_size, jaccard_min, k_out` Passed to [sniff_trajectory_dag\(\)](#) when `x` is not already a DAG object.

Details

Adds one column to trajectories: `latest_departure`, the year of the most recent birth that can reach the trajectory's tail by any time-respecting path (Wu et al. 2014 latest-departure). For a central this is the freshest input anywhere in the final community's ancestry; combined with `start` it separates old-but-freshly-fed communities from genuinely young ones.

Value

A `birddog_flow` object (see [sniff_trajectory_braid\(\)](#)) with the extra `latest_departure` column.

References

Wu H, Cheng J, Huang S, Ke Y, Lu Y, Xu Y (2014). Path Problems in Temporal Graphs. Proc. VLDB Endowment 7(9).

See Also

[sniff_trajectory_braid\(\)](#), [validate_flow\(\)](#), [sniff_trajectory_dag\(\)](#)

Other trajectory detection: [is_flow\(\)](#), [sniff_trajectory_braid\(\)](#), [sniff_trajectory_dag\(\)](#), [subset.birddog_flow\(\)](#), [validate_flow\(\)](#)

sniff_trajectory_coherence

Appraise the content coherence of a trajectory flow

Description

Scores how content-coherent a [sniff_trajectory_braid\(\)](#) / [sniff_trajectory_channel\(\)](#) partition is, using a silhouette over an independent content signal (shared references or keywords) the detectors never optimized. Each group-year node is profiled from its documents' features; node-to-node distance is $1 - \text{Salton cosine}$; the silhouette is computed at two cluster resolutions: trajectory (each backbone/tributary chain) and final_group (each central + its tributaries).

Usage

```
sniff_trajectory_coherence(
  flow,
  content,
  profile = "incremental",
  signals = names(content)
)
```

Arguments

flow	A <code>birddog_flow</code> (from sniff_trajectory_braid() or sniff_trajectory_channel()).
content	Named list; each element a data frame with columns <code>document_id</code> and <code>feature</code> (long, one row per document-feature pair). The element name labels the signal in the output (e.g. <code>coupling</code> , <code>keywords</code>).
profile	"incremental" (default: profile a node from the papers new to its year, stripping cumulative carry-over) and/or "full" (all of a node's papers).
signals	Which <code>names(content)</code> to score (default all).

Value

An `sniff_trajectory_coherence` object: a list with nodes (one row per node x signal x profile x resolution) and summary (one row per signal x profile x resolution, with `mean_sil`, `n_nodes`, `n_singletons`, `n_excluded`, `coverage`).

See Also

[sniff_trajectory_comparison\(\)](#), [sniff_trajectory_braid\(\)](#), [sniff_trajectory_channel\(\)](#), [validate_flow\(\)](#)

Other flow utilities: [sniff_trajectory_comparison\(\)](#)

sniff_trajectory_community

Community breadth (distinct authors) of each flow trajectory

Description

Measures the FUSE *community* pillar (Carley et al., 2018) for every [sniff_trajectory_braid\(\)](#) trajectory: how many distinct authors stand behind its documents, and how broad that authorship is per document. A raw author count tracks trajectory size; `authors_per_doc` is the size-independent breadth — a high value marks a lineage carried by many distinct contributors (a wide community), a low value a lineage concentrated in a tight group.

Usage

```
sniff_trajectory_community(flow, authors)
```

Arguments

<code>flow</code>	A sniff_trajectory_braid() object.
<code>authors</code>	A long data frame with columns <code>document_id</code> and <code>author</code> , one row per document-author pair (e.g. the corpus AU field split on ; and joined to the membership key). Documents absent from <code>authors</code> contribute no authors but still count in <code>n_docs</code> , so keep author coverage high.

Details

This is the document-cluster analogue of the term-level community measure in the emergence-indicator literature, where a term counts as emergent only when used by more than one independent author. It complements [sniff_trajectory_dynamics\(\)](#), which names the community pillar but scores it only through document counts (recruitment, size).

Value

A tibble, one row per trajectory: `traj_id`, `type`, `group`, `n_docs` (distinct documents of the trajectory), `n_authors` (distinct authors across them), `authors_per_doc` (`n_authors / n_docs`, the breadth; NA for an empty trajectory).

See Also

[sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_hubs\(\)](#), [sniff_trajectory_braid\(\)](#)

Other trajectory analysis: [sniff_trajectory_cct\(\)](#), [sniff_trajectory_confluence\(\)](#), [sniff_trajectory_contribution\(\)](#), [sniff_trajectory_destination\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_emergence_owners\(\)](#), [sniff_trajectory_entropy\(\)](#), [sniff_trajectory_formation\(\)](#), [sniff_trajectory_hubs\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#)

`sniff_trajectory_comparison`*Compare trajectory detectors by content coherence*

Description

Runs [sniff_trajectory_coherence\(\)](#) on each named flow, across every content signal and the chosen profile(s), and reports which detector produces the more content-coherent partition, whether the signals agree, and the contested nodes (those a pair of detectors place in different final groups).

Usage

```
sniff_trajectory_comparison(  
    flows,  
    content,  
    profile = "incremental",  
    signals = names(content)  
)
```

Arguments

<code>flows</code>	Named list of <code>birddog_flow</code> objects, e.g. <code>list(flow = ..., channel = ...)</code> .
<code>content</code>	Named list of (<code>document_id</code> , <code>feature</code>) data frames (see sniff_trajectory_coherence()).
<code>profile</code>	Passed to sniff_trajectory_coherence() (default "incremental").
<code>signals</code>	Which names(<code>content</code>) to score (default all).

Value

A `sniff_trajectory_comparison` object: a list with `summary` (per method x signal x profile x resolution), `verdict` (best_method, delta per signal x profile x resolution), `agreement` (do all signals pick the same best_method, per profile x resolution), and `contested` (nodes the detectors place in different final groups, with each method's `final_group` and `silhouette`, plus the `closer_method`; NULL unless exactly two flows are given).

See Also

[sniff_trajectory_coherence\(\)](#)

Other flow utilities: [sniff_trajectory_coherence\(\)](#)

sniff_trajectory_confluence

Build the render-ready confluence forest of the soft DAG

Description

Turns a `sniff_trajectory_braid()` object into the data a trajectory confluence needs: one rivers row per trajectory (with a per-year size curve) and one confluences row per child to parent merge edge (papers transferred, the first year a transferred paper was published, the handoff year, and the cumulative-inflow curve). Every edge of the flow tree is kept; thresholds are applied later by `plot_trajectory_confluence()`, so a slider can re-filter without recomputing.

Usage

```
sniff_trajectory_confluence(
  x,
  min_group_size = 10,
  jaccard_min = 0.05,
  k_out = 2
)
```

Arguments

x A `sniff_trajectory_braid()` object, or a docs_per_group tibble / `sniff_groups_lineage()` object / `sniff_trajectory_dag()` object (the flow is built internally).

min_group_size, jaccard_min, k_out Passed to `sniff_trajectory_braid()` when x is not already a flow object.

Value

A list with:

rivers Tibble, one row per trajectory: traj_id, type ("central"/"absorbed"), central (destination group cNgN), size, start, handoff_year (= last_year for centrals, else absorption year), depth, parent (absorber traj_id, NA for central), and size_curve (list-col of tibble(year, size), the trajectory's community size per year).

confluences Tibble, one row per child to parent merge: child, parent, n (papers transferred), cohort_size (the child's size), first_feed_year (publication year of the first transferred paper), handoff_year, and inflow_curve (list-col of tibble(year, size)).

destinations Tibble, one row per (absorbed trajectory, final group) it fed: traj_id, g_final, n (papers of the trajectory's terminal cohort that ended up in g_final). A trajectory usually appears under several finals, so this is the multi-destination split the forest collapses to one.

centrals The central group ids (cNgN) in mixed_sort() order.

last_year Final year of the analysis.

See Also

[sniff_trajectory_braid\(\)](#), [plot_trajectory_confluence\(\)](#)

Other trajectory analysis: [sniff_trajectory_cct\(\)](#), [sniff_trajectory_community\(\)](#), [sniff_trajectory_contribution\(\)](#), [sniff_trajectory_destination\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_emergence_owners\(\)](#), [sniff_trajectory_entropy\(\)](#), [sniff_trajectory_formation\(\)](#), [sniff_trajectory_hubs\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#)

sniff_trajectory_contribution

Documents an intermediate trajectory contributes to a target in a given year

Description

Selects a source trajectory's node in a specific year and returns its documents, flagging which of them belong to the target trajectory's final-year community. This is the fine-grained, per-document view of the inter-trajectory contribution that [sniff_trajectory_destination\(\)](#) aggregates: where destination splits a terminal cohort across every final group, contribution answers "which papers of source, as they stood in year, consolidate into this one target".

Usage

```
sniff_trajectory_contribution(flow, source, year, target)
```

Arguments

flow	A sniff_trajectory_braid() object.
source	Trajectory id whose year-node is read, typically an intermediate trN (e.g. "tr14"). Any trajectory id present in flow is accepted; each trajectory has at most one node per year.
year	Integer publication year. source must have a node in this year.
target	Trajectory id whose final-year community is the contribution basis, typically a central tr::cNgN. Its destination group must be known.

Details

The result is document ids only (plus the contribution flag), so it composes with any downstream join: keywords (via the corpus DE field or [sniff_groups_keywords\(\)](#)), authors, citations, or STM topics. Filter on in_target for the contributed set.

Value

A tibble with one row per document in source's year node: source, target, year, document_id, and in_target (logical: the document is in target's final-year community). The source/target/year columns are constant, so results for several (source, year, target) triples row-bind cleanly. Summarise with one line, e.g. `dplyr::summarise(x, n = dplyr::n(), contributed = sum(in_target))`.

See Also

[sniff_trajectory_destination\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#), [sniff_groups_keywords\(\)](#)

Other trajectory analysis: [sniff_trajectory_cct\(\)](#), [sniff_trajectory_community\(\)](#), [sniff_trajectory_confluence\(\)](#), [sniff_trajectory_destination\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_emergence_owners\(\)](#), [sniff_trajectory_entropy\(\)](#), [sniff_trajectory_formation\(\)](#), [sniff_trajectory_hubs\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#)

sniff_trajectory_dag *Build the soft cumulative-clustering trajectory DAG*

Description

Reconstructs, from docs_per_group alone, the full temporal DAG of cluster-year nodes: vertices are every cluster whose distinct-document count is at least min_group_size (isolated nodes included), edges are consecutive-year Jaccard links (top k_out per source, weight >= jaccard_min). Each node is labelled with the final group its heaviest-successor walk reaches (terminal_group) and whether it is a birth (no strong predecessor). This DAG is the first-class object the soft trajectories and group-formation views are read from.

Usage

```
sniff_trajectory_dag(
  x,
  min_group_size = 10,
  jaccard_min = 0.05,
  k_out = 2,
  similarity = c("overlap", "coupling"),
  references = NULL
)
```

Arguments

x	The membership tibble docs_per_group (group_id, document_id, network_until, group) or a sniff_groups_lineage() object (its \$docs_per_group is used).
min_group_size	Minimum distinct-document count for a node (default 10).
jaccard_min	Minimum Jaccard weight to keep an edge (default 0.05).
k_out	Max outgoing edges kept per source node (default 2).
similarity	How to weight consecutive-year edges: "overlap" (default) uses the Jaccard of the nodes' shared documents (the cumulative carry-over); "coupling" uses the Jaccard of their shared cited references (bibliographic coupling) and requires references. Both detectors (sniff_trajectory_braid() , sniff_trajectory_channel()) route either DAG unchanged.
references	When similarity = "coupling", a data frame with columns document_id and feature (one cited reference per row), the same shape as a content element of sniff_trajectory_coherence() . Ignored for "overlap".

Value

A birddog_dag object: a classed list with graph (scored igraph DAG with vertex attrs year, size, group, is_birth, terminal_group, plus doc_ids), nodes (tidy tibble name, year, group, size, is_birth, terminal_group), edges (from, to, weight, documents), births (character vector of birth node names), last_year, and the de-duplicated docs_per_group.

See Also

[plot_trajectory_dag\(\)](#)

Other trajectory detection: [is_flow\(\)](#), [sniff_trajectory_braid\(\)](#), [sniff_trajectory_channel\(\)](#), [subset.birddog_flow\(\)](#), [validate_flow\(\)](#)

sniff_trajectory_destination

Where a trajectory's terminal cohort goes

Description

Returns the destination of source's terminal cohort: its per-final-group split and dormant share, plus its immediate absorber and ultimate destination.

Usage

```
sniff_trajectory_destination(x, ...)
```

Arguments

x A [sniff_trajectory_braid\(\)](#) object.
 ... source, the trajectory whose terminal cohort to follow.

Value

A list (target, target_info, destination, cohort_size, dormant_share, continuation_traj, last_year).

See Also

[sniff_trajectory_braid\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#)

Other trajectory analysis: [sniff_trajectory_cct\(\)](#), [sniff_trajectory_community\(\)](#), [sniff_trajectory_confluence\(\)](#), [sniff_trajectory_contribution\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_emergence_owners\(\)](#), [sniff_trajectory_entropy\(\)](#), [sniff_trajectory_formation\(\)](#), [sniff_trajectory_hubs\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#)

sniff_trajectory_dynamics

Dynamic-state indicators and classification for flow trajectories

Description

Characterizes each `sniff_trajectory_braid()` trajectory along the evolutionary states of the method (emergence, convergence, divergence, dormancy), grounding the four emergence pillars (novelty, growth, community, persistence) in two complementary lenses tied to the flow model.

Usage

```
sniff_trajectory_dynamics(
  flow,
  thresholds = NULL,
  k = 1,
  growth_window = 3,
  novelty_window = 5,
  winsorize = 3,
  cct = NULL,
  entropy = NULL,
  hubs = NULL
)
```

Arguments

<code>flow</code>	A <code>sniff_trajectory_braid()</code> object.
<code>thresholds</code>	A <code>fixed_state_thresholds()</code> -shaped list. NULL (default) derives the cut points from the data via <code>data_state_thresholds()</code> with the <code>k</code> below, so the classification adapts to the dataset; pass <code>fixed_state_thresholds()</code> to force the fixed constants instead. The thresholds actually used are attached to the result as <code>attr(result, "state_thresholds")</code> , which the plots read back.
<code>k</code>	Robust deviations for the data-driven thresholds when <code>thresholds</code> is NULL (default 1); see <code>data_state_thresholds()</code> . Ignored when an explicit thresholds list is supplied.
<code>growth_window</code>	Curve points over which <code>growth_rate</code> is fitted (default 3).
<code>novelty_window</code>	Years counted as recent for novelty (default 5).
<code>winsorize</code>	Cap each pillar's robust z-score at <code>c(-winsorize, +winsorize)</code> before summing into <code>emergence_index</code> , so no single extreme pillar (e.g. a young core 18 MADs out on growth) dominates the ranking. Default 3 (near the Iglewicz–Hoaglin modified-z outlier boundary of 3.5); NULL leaves the robust z-scores uncapped.
<code>cct</code>	Optional <code>sniff_trajectory_cct()</code> output (one row per <code>traj_id</code>), left-joined to add the <code>cct</code> list-column (per-year renewal-pace series); NULL (default) omits it.

entropy	Optional sniff_trajectory_entropy() output (one row per traj_id), left-joined to add the keyword_entropy list-column (per-year keyword-diversity series); NULL (default) omits it.
hubs	Optional sniff_trajectory_hubs() output (one row per traj_id), left-joined to add the hub-role columns; NULL (default) omits them.

Details

Forward lens (all trajectories). growth_rate and doubling_time (an exponential fit of the recent size curve), novelty (share of documents that arrived recently), recruitment (net documents gained over the lifespan, the power of attraction), emergence_density (staying power: the share of the lineage’s year-steps still growing at the emergence rate, after Carley et al. 2017), and age/size (persistence and community).

Backward lens (absorbed trajectories only). Where the terminal cohort goes, via [sniff_trajectory_destination\(\)](#): dest_entropy (normalized Shannon entropy of the destination split) and dormant_share (share dropping out).

Inflow lens (central trajectories only). attraction_inflow, the documents of the final community delivered by absorbed tributaries (the consolidating side of convergence), from [sniff_trajectory_self_sufficiency\(\)](#).

Reach lens (all trajectories). reach_ratio, the share of the future field a lineage’s origin seeds – its birth node’s forward temporal reach in the DAG over all nodes born later (after the temporal-reachability of Marino & Silva 2023). Structural downstream influence, distinct from size and, via the ratio, from age.

Classification uses two axes: a life-cycle phase for living (central) trajectories – emergence / maturity / dormancy(stall) by growth and novelty – and a terminal fate for declining (absorbed) ones – convergence / divergence / dormancy(death) by destination. state is phase for centrals and fate for absorbed; phase and fate are kept separate so the two senses of dormancy (a stalled living core vs a dead cohort) stay distinguishable. The emergence_index (a within-central combination of growth, novelty and recruitment) is computed for living trajectories only, so an absorbed lineage that grew before dying is never ranked as emerging. Because recruitment is an extensive (size-carrying) pillar, the full index partly tracks community size; emergence_index_intensive drops it (growth and novelty only) for a rate-based ranking that does not reward a large core for its volume alone.

Value

A tibble, one row per trajectory, sorted by descending emergence_index (absorbed last): traj_id, type, group, start, end, age, size, growth_rate, doubling_time, novelty, recruitment, emergence_density, attraction_inflow, dest_entropy, dormant_share, phase, fate, state, emergence_index, emergence_index_intensive (the same index without the extensive recruitment pillar), reach_ratio, plus any cct / keyword_entropy (per-year list-columns) and hubs columns.

See Also

[sniff_trajectory_braid\(\)](#), [sniff_trajectory_destination\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#), [sniff_trajectory_cct\(\)](#), [sniff_trajectory_entropy\(\)](#), [sniff_trajectory_hubs\(\)](#), [fixed_state_thresholds\(\)](#)

Other trajectory analysis: [sniff_trajectory_cct\(\)](#), [sniff_trajectory_community\(\)](#), [sniff_trajectory_confluence\(\)](#), [sniff_trajectory_contribution\(\)](#), [sniff_trajectory_destination\(\)](#), [sniff_trajectory_emergence_owners\(\)](#),

[sniff_trajectory_entropy\(\)](#), [sniff_trajectory_formation\(\)](#), [sniff_trajectory_hubs\(\)](#),
[sniff_trajectory_self_sufficiency\(\)](#)

sniff_trajectory_emergence_owners

Roll trajectory emergence up to the authors who own it

Description

The actor-level companion to [sniff_trajectory_dynamics\(\)](#), after the player scoring of Garner et al. (2017): it credits each author with the emergence of the living trajectories their documents belong to. An author who publishes many documents inside high-emergence_index centrals is an owner of the field's emergence; one whose work sits in mature or absorbed lineages is not.

Usage

```
sniff_trajectory_emergence_owners(
  flow,
  dynamics,
  authors,
  min_docs = 1,
  by_trajectory = FALSE,
  top_n = NULL
)
```

Arguments

flow	A sniff_trajectory_braid() object.
dynamics	A sniff_trajectory_dynamics() tibble (supplies traj_id and emergence_index).
authors	A long data frame with columns document_id and author, one row per document-author pair.
min_docs	Drop authors with fewer than this many credited documents: summed across trajectories when aggregated, or within the trajectory when by_trajectory = TRUE. Default 1 keeps all. Use a higher cut to read norm/share without one-paper noise.
by_trajectory	If TRUE, return the per-trajectory breakdown (one row per trajectory-author) instead of the field-level aggregate (one row per author). Default FALSE.
top_n	Keep only the highest-scoring authors – overall (by total) when aggregated, or per trajectory (by contribution) when by_trajectory = TRUE. Default NULL keeps all.

Details

For every central trajectory with a defined `emergence_index` (the living population), each author is credited `emergence_index(t)` per distinct document they hold in `t`. Absorbed lineages carry no `emergence_index` and are skipped, so a contributor to a dead tributary is never scored as owning emergence.

$$Total(a) = \sum_t emergence_index(t) n_{a,t}, \quad Norm(a) = Total(a) / \sqrt{\sum_t n_{a,t}}$$

where $n_{a,t}$ is author a 's distinct documents in trajectory t . Total favours prolific owners; Norm (per square-root of output, Garner's normalization) surfaces high-intensity authors with fewer papers.

With `by_trajectory = TRUE` the per-trajectory term $n_{a,t}$ is returned instead of being summed away: one row per (trajectory, author), so you can read which authors *dominate each living trajectory* rather than the field as a whole. `top_n` then trims to the leading authors per trajectory.

Value

A tibble.

When `by_trajectory = FALSE` (default): one row per author, sorted by descending total – author, total (emergence ownership), `ndocs` (credited document-trajectory incidences), `norm` (total / `sqrt(ndocs)`).

When `by_trajectory = TRUE`: one row per trajectory-author, sorted by trajectory then descending contribution – `traj_id`, group, author, `ndocs` (the author's distinct documents in the trajectory, $n_{a,t}$), `emergence_index` (the trajectory's weight), `contribution` (`emergence_index * ndocs`, the author's share of that trajectory's emergence), and `share` (`ndocs` over the trajectory's distinct documents – the author's coverage of its papers).

See Also

[sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_community\(\)](#), [sniff_trajectory_contribution\(\)](#)

Other trajectory analysis: [sniff_trajectory_cct\(\)](#), [sniff_trajectory_community\(\)](#), [sniff_trajectory_confluence\(\)](#), [sniff_trajectory_contribution\(\)](#), [sniff_trajectory_destination\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_entropy\(\)](#), [sniff_trajectory_formation\(\)](#), [sniff_trajectory_hubs\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#)

sniff_trajectory_entropy

Per-year keyword diversity (Pielou's J') along each flow trajectory

Description

Recomputes, from the documents in each [sniff_trajectory_braid\(\)](#) trajectory's own group-year nodes, the normalized Shannon entropy (Pielou's evenness J') of their keyword distribution, one value per year of the trajectory's life. A falling series marks thematic convergence, a rising one diversification. Unlike [sniff_entropy\(\)](#), which scores a static `sniff_groups()` partition by group-year, this follows the trajectory's actual membership, so an absorbed tributary is scored on its own documents rather than its destination group's.

Usage

```
sniff_trajectory_entropy(flow, keywords)
```

Arguments

flow	A sniff_trajectory_braid() object.
keywords	A long data frame with columns <code>document_id</code> and <code>keyword</code> , one row per document-keyword pair (e.g. the corpus DE field split on ; and keyed by the membership id).

Details

This is **keyword** entropy (thematic diversity), distinct from the dynamics `dest_entropy` (the dispersion of an absorbed cohort across destination groups); the dynamics column is named `keyword_entropy` to keep the two apart.

Value

A tibble, one row per trajectory: `traj_id`, `type`, `group`, and `keyword_entropy`, a list-column whose each cell is a tibble with columns `year` and `keyword_entropy` (Pielou's J' over that node-year's documents, in $[0, 1]$; 0 for a single keyword, NA when the year's documents carry none).

See Also

[sniff_entropy\(\)](#), [sniff_trajectory_cct\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_braid\(\)](#)

Other trajectory analysis: [sniff_trajectory_cct\(\)](#), [sniff_trajectory_community\(\)](#), [sniff_trajectory_confluence\(\)](#), [sniff_trajectory_contribution\(\)](#), [sniff_trajectory_destination\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_emergence_owners\(\)](#), [sniff_trajectory_formation\(\)](#), [sniff_trajectory_hubs\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#)

sniff_trajectory_formation

Trajectories that fed into a target (confluence)

Description

Returns the target trajectory's direct tributaries: its children in the confluence tree, the trajectories whose terminal cohort was absorbed into target. The result is the formation shape that [plot_trajectory_formation\(\)](#) renders.

Usage

```
sniff_trajectory_formation(x, ...)
```

Arguments

x A `sniff_trajectory_braid()` object.

... target (a `traj_id`), then `min_papers` (default 5) and `min_prop` (default 0.05), the thresholds behind the feeders' kept flag.

Value

A formation list (`target`, `target_info`, `feeders`, `total_inflow`, `last_year`). The feeders carry `n_dest` (the feeder's documents that end in the target's final community) and `target_info` carries `final_size` (the target's final-year size); `plot_trajectory_formation()` uses these for its labels.

See Also

`sniff_trajectory_braid()`, `plot_trajectory_formation()`

Other trajectory analysis: `sniff_trajectory_cct()`, `sniff_trajectory_community()`, `sniff_trajectory_confluence()`, `sniff_trajectory_contribution()`, `sniff_trajectory_destination()`, `sniff_trajectory_dynamics()`, `sniff_trajectory_emergence_owners()`, `sniff_trajectory_entropy()`, `sniff_trajectory_hubs()`, `sniff_trajectory_self_sufficiency()`

`sniff_trajectory_hubs` *Hub roles aggregated to each flow trajectory (provincial vs bridging)*

Description

Summarizes the document-level hub roles of `sniff_groups_hubs()` (the Guimera–Amaral within-module z-score Z_i and participation coefficient P_i) over the documents of every `sniff_trajectory_braid()` trajectory. Provincial hubs (low P_i) keep a lineage self-contained, consistent with convergence / paradigmatic homogeneity; connector and boundary-spanning hubs (high P_i) bridge across groups, consistent with divergence / branching.

Usage

```
sniff_trajectory_hubs(flow, hubs)
```

Arguments

flow A `sniff_trajectory_braid()` object.

hubs A `sniff_groups_hubs()` tibble: a document-id column (`name`, as returned, or `SR`), plus Z_i , P_i , `zone` (zone one of "noHub", "R5", "R6", "R7"). Documents are matched to a trajectory by that id against the trajectory's node documents.

Value

A tibble, one row per trajectory: `traj_id`, `type`, `group`, `n_docs` (documents of the trajectory), `mean_Zi`, `mean_Pi`, `hub_share` (share that are hubs), `connector_share` (share that are connector/boundary hubs R6/R7, the bridging signal), `provincial_share` (share that are provincial hubs R5).

See Also

[sniff_groups_hubs\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_braid\(\)](#)

Other trajectory analysis: [sniff_trajectory_cct\(\)](#), [sniff_trajectory_community\(\)](#), [sniff_trajectory_confluence\(\)](#), [sniff_trajectory_contribution\(\)](#), [sniff_trajectory_destination\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_emergence_owners\(\)](#), [sniff_trajectory_entropy\(\)](#), [sniff_trajectory_formation\(\)](#), [sniff_trajectory_self_sufficiency\(\)](#)

sniff_trajectory_self_sufficiency

Self-sufficiency of each central trajectory

Description

Returns each central trajectory's endogenous fraction: the share of its **final-year community** that was NOT delivered by any absorbed tributary (1 = fully endogenous, ~0 = a confluence of tributaries). Inflow follows the same destination semantics as [plot_trajectory_confluence\(\)](#) and [sniff_trajectory_destination\(\)](#): a document counts as imported when it sits in the central's final community **and** in the terminal cohort of some absorbed tributary, counted once even if several tributaries carried it, and **regardless of which central the tributary is dominantly assigned to** – so secondary inflow (tributaries owned by other centrals) is included, not only the central's own feeders.

Usage

```
sniff_trajectory_self_sufficiency(x, ...)
```

Arguments

x A [sniff_trajectory_braid\(\)](#) object.

... min_size (default 30, the minimum final-year community size).

Value

A tibble (central, group, final_size, inflow, self_sufficiency), sorted by descending self_sufficiency.

See Also

[sniff_trajectory_braid\(\)](#), [sniff_trajectory_destination\(\)](#)

Other trajectory analysis: [sniff_trajectory_cct\(\)](#), [sniff_trajectory_community\(\)](#), [sniff_trajectory_confluence\(\)](#), [sniff_trajectory_contribution\(\)](#), [sniff_trajectory_destination\(\)](#), [sniff_trajectory_dynamics\(\)](#), [sniff_trajectory_emergence_owners\(\)](#), [sniff_trajectory_entropy\(\)](#), [sniff_trajectory_formation\(\)](#), [sniff_trajectory_hubs\(\)](#)

subset.birddog_flow	<i>Subset a trajectory flow, preserving the contract</i>
---------------------	--

Description

Filters the trajectories of a flow object while keeping it a valid flow: `$tree` and `absorbed_into` are reattached by transitive bypass (a kept child points to its nearest kept ancestor; NA if none survives) and `$docs_per_group` follows the kept nodes. The result is a view, not a census: the call is recorded in `attr(, "pruned")` and shown by `print()`. Dispatches for any detector whose result inherits from `birddog_flow` (the contract class), so future detection methods are subsettable for free.

Usage

```
## S3 method for class 'birddog_flow'
subset(x, subset, target = NULL, ...)
```

Arguments

<code>x</code>	A <code>birddog_flow</code> object (or a subclass from a future detector).
<code>subset</code>	Logical expression on the columns of <code>x\$trajectories</code> , e.g. <code>size >= 50</code> or <code>type == "central"</code> .
<code>target</code>	Optional character vector of <code>traj_ids</code> : keep each target and its whole tributary subtree (the watershed). Intersects with <code>subset</code> .
<code>...</code>	Ignored (base generic compatibility).

Value

An object of the same class as `x`, passing `validate_flow()`.

See Also

`sniff_trajectory_braid()`, `validate_flow()`

Other trajectory detection: `is_flow()`, `sniff_trajectory_braid()`, `sniff_trajectory_channel()`, `sniff_trajectory_dag()`, `validate_flow()`

 validate_flow

 Validate the trajectory flow contract

Description

The authoritative checker of the flow contract (trajectories, tree, graph, docs_per_group, last_year). "Flow" names the object kind, not the algorithm: any detector – [sniff_trajectory_braid\(\)](#) today, alternative sniff_trajectory_<algo>() detectors in the future – must return an object that passes this validator and inherits the birddog_flow class.

Usage

```
validate_flow(x)
```

Arguments

x The object to validate, typically a [sniff_trajectory_braid\(\)](#) object or the output of an alternative detector shaped like one.

Details

Beyond the column schema, the checked invariants are the ones that otherwise fail silently: every trajectory node label must be a docs_per_group\$group_id key and a graph vertex (the triple key documents and analyses share), the graph must carry the quantity_papers / prop_tracked_intra_group vertex attributes the line plots read, absorbed_into must reference existing trajectories without forming a cycle, each final group must have exactly one central present at last_year, central ids must use the tr:: convention, and start/end must match the node years.

Value

x, invisibly, when valid; otherwise an error listing every violation.

See Also

[sniff_trajectory_braid\(\)](#), [is_flow\(\)](#)

Other trajectory detection: [is_flow\(\)](#), [sniff_trajectory_braid\(\)](#), [sniff_trajectory_channel\(\)](#), [sniff_trajectory_dag\(\)](#), [subset.birddog_flow\(\)](#)

Index

- * **flow utilities**
 - sniff_trajectory_coherence, [81](#)
 - sniff_trajectory_comparison, [83](#)
- * **groups (stock)**
 - sniff_components, [55](#)
 - sniff_groups, [59](#)
 - sniff_groups_attributes, [60](#)
 - sniff_groups_cumulative, [62](#)
 - sniff_groups_cumulative_citations, [64](#)
 - sniff_groups_hubs, [65](#)
 - sniff_groups_influence, [67](#)
 - sniff_groups_lineage, [70](#)
 - sniff_network, [77](#)
- * **package**
 - birddog-package, [3](#)
- * **trajectory analysis**
 - sniff_trajectory_cct, [79](#)
 - sniff_trajectory_community, [82](#)
 - sniff_trajectory_confluence, [84](#)
 - sniff_trajectory_contribution, [85](#)
 - sniff_trajectory_destination, [87](#)
 - sniff_trajectory_dynamics, [88](#)
 - sniff_trajectory_emergence_owners, [90](#)
 - sniff_trajectory_entropy, [91](#)
 - sniff_trajectory_formation, [92](#)
 - sniff_trajectory_hubs, [93](#)
 - sniff_trajectory_self_sufficiency, [94](#)
- * **trajectory detection**
 - is_flow, [8](#)
 - sniff_trajectory_braid, [78](#)
 - sniff_trajectory_channel, [80](#)
 - sniff_trajectory_dag, [86](#)
 - subset.birddog_flow, [95](#)
 - validate_flow, [96](#)
- * **visualization**
 - plot_groups_influence_matrix, [10](#)
 - plot_groups_influence_network, [11](#)
 - plot_groups_lineage_2d, [13](#)
 - plot_groups_lineage_3d, [15](#)
 - plot_groups_map, [17](#)
 - plot_groups_map_animation, [18](#)
 - plot_groups_map_interactive, [20](#)
 - plot_groups_per_year, [21](#)
 - plot_trajectory_confluence, [23](#)
 - plot_trajectory_confluence_interactive, [27](#)
 - plot_trajectory_confluence_matrix, [29](#)
 - plot_trajectory_dag, [31](#)
 - plot_trajectory_dag_interactive, [34](#)
 - plot_trajectory_dispersion, [36](#)
 - plot_trajectory_dynamics, [37](#)
 - plot_trajectory_dynamics_interactive, [39](#)
 - plot_trajectory_formation, [41](#)
 - plot_trajectory_lines_2d, [43](#)
 - plot_trajectory_lines_3d, [46](#)
- birddog (birddog-package), [3](#)
- birddog-package, [3](#)
- data_state_thresholds, [5](#)
- data_state_thresholds(), [6](#), [38](#), [40](#), [88](#)
- fixed_state_thresholds, [6](#)
- fixed_state_thresholds(), [5](#), [38–41](#), [88](#), [89](#)
- get_openalex_fields, [6](#)
- get_openalex_fields(), [54](#)
- gganimate::anim_save(), [19](#)
- gganimate::animate(), [19](#)
- ggplot2::coord_cartesian(), [38](#)
- ggplot2::scale_size(), [38](#)
- ggraph::create_layout(), [18](#), [19](#), [21](#)

- `indexes_plots`, 58
- `indexes_plots()`, 54
- `is_flow`, 8, 78, 80, 87, 95, 96
- `is_flow()`, 96
- `is_influence` (`sniff_groups_influence`), 67
- `mixed_sort`, 9
- `oa2df`, 50
- `oa_request`, 50
- `plot_groups_influence_matrix`, 10, 12, 14, 16, 18, 20, 21, 23, 26, 29, 31, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_groups_influence_matrix()`, 12, 68
- `plot_groups_influence_network`, 11, 11, 14, 16, 18, 20, 21, 23, 26, 29, 31, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_groups_influence_network()`, 11, 68
- `plot_groups_lineage_2d`, 11, 12, 13, 16, 18, 20, 21, 23, 26, 29, 31, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_groups_lineage_3d`, 11, 12, 14, 15, 18, 20, 21, 23, 26, 29, 31, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_groups_map`, 11, 12, 14, 16, 17, 20, 21, 23, 26, 29, 31, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_groups_map()`, 19–21
- `plot_groups_map_animation`, 11, 12, 14, 16, 18, 18, 21, 23, 26, 29, 31, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_groups_map_interactive`, 11, 12, 14, 16, 18, 20, 20, 23, 26, 29, 31, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_groups_map_interactive()`, 20
- `plot_groups_per_year`, 11, 12, 14, 16, 18, 20, 21, 21, 26, 29, 31, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_groups_per_year()`, 33, 35
- `plot_trajectory_confluence`, 11, 12, 14, 16, 18, 20, 21, 23, 23, 29, 31, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_trajectory_confluence()`, 27–29, 31, 36–39, 43–45, 48, 84, 85, 94
- `plot_trajectory_confluence_interactive`, 11, 12, 14, 16, 18, 20, 21, 23, 26, 27, 31, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_trajectory_confluence_matrix`, 11, 12, 14, 16, 18, 20, 21, 23, 26, 29, 29, 33, 35, 37, 39, 41, 42, 45, 48
- `plot_trajectory_confluence_matrix()`, 10, 11
- `plot_trajectory_dag`, 11, 13, 14, 16, 18, 20, 21, 23, 26, 29, 31, 31, 35, 37, 39, 41, 42, 45, 48
- `plot_trajectory_dag()`, 18, 21, 23, 25, 34, 35, 38, 87
- `plot_trajectory_dag_interactive`, 11, 13, 14, 16, 18, 20, 21, 23, 26, 29, 31, 33, 34, 37, 39, 41, 43, 45, 48
- `plot_trajectory_dag_interactive()`, 23, 28, 29, 33
- `plot_trajectory_dispersion`, 11, 13, 14, 16, 18, 20, 21, 23, 26, 29, 31, 33, 35, 36, 39, 41, 43, 45, 48
- `plot_trajectory_dispersion()`, 41, 42
- `plot_trajectory_dynamics`, 11, 13, 14, 16, 18, 20, 21, 23, 26, 29, 31, 33, 35, 37, 37, 41, 43, 45, 48
- `plot_trajectory_dynamics()`, 5, 39–41
- `plot_trajectory_dynamics_interactive`, 11, 13, 14, 16, 18, 20, 21, 23, 26, 29, 31, 33, 35, 37, 39, 39, 43, 45, 48
- `plot_trajectory_formation`, 11, 13, 14, 16, 18, 20, 21, 23, 26, 29, 31, 33, 35, 37, 39, 41, 41, 45, 48
- `plot_trajectory_formation()`, 26, 92, 93
- `plot_trajectory_lines_2d`, 11, 13, 14, 16, 18, 20, 21, 23, 26, 29, 31, 33, 35, 37, 39, 41, 43, 43, 48
- `plot_trajectory_lines_2d()`, 25, 29–31, 36, 37, 42, 46–48
- `plot_trajectory_lines_3d`, 11, 13, 14, 16, 18, 20, 21, 23, 26, 29, 31, 33, 35, 37, 39, 41, 43, 45, 46
- `plot_trajectory_lines_3d()`, 25, 45
- `print.birdog_influence` (`sniff_groups_influence`), 67
- `read_openalex`, 49
- `read_wos`, 50, 50
- `read_wos_bib`, 51
- `read_wos_plain`, 51
- `read_wos_ris`, 51
- `read_wos_tab`, 51

sniff_cct, [52](#)
 sniff_cct(), [79](#)
 sniff_components, [55](#), [59–63](#), [65](#), [66](#), [68](#), [70](#), [77](#)
 sniff_entropy, [56](#)
 sniff_entropy(), [91](#), [92](#)
 sniff_groups, [56](#), [58](#), [59](#), [61](#), [63](#), [65](#), [66](#), [68](#), [70](#), [77](#)
 sniff_groups(), [54](#), [67](#), [68](#)
 sniff_groups_attributes, [56](#), [60](#), [60](#), [63](#), [65](#), [66](#), [68](#), [70](#), [77](#)
 sniff_groups_cumulative, [56](#), [60](#), [61](#), [62](#), [65](#), [66](#), [68](#), [70](#), [77](#)
 sniff_groups_cumulative(), [17–21](#)
 sniff_groups_cumulative_citations, [56](#), [60](#), [61](#), [63](#), [64](#), [66](#), [68](#), [70](#), [77](#)
 sniff_groups_hubs, [56](#), [60](#), [61](#), [63](#), [65](#), [65](#), [68](#), [70](#), [77](#)
 sniff_groups_hubs(), [68](#), [93](#), [94](#)
 sniff_groups_influence, [56](#), [60](#), [61](#), [63](#), [65](#), [66](#), [67](#), [70](#), [77](#)
 sniff_groups_influence(), [10–12](#)
 sniff_groups_keywords, [69](#)
 sniff_groups_keywords(), [85](#), [86](#)
 sniff_groups_lineage, [56](#), [60](#), [61](#), [63](#), [65](#), [66](#), [68](#), [70](#), [77](#)
 sniff_groups_lineage(), [78](#), [80](#), [84](#), [86](#)
 sniff_groups_stm_prepare, [71](#)
 sniff_groups_stm_run, [72](#)
 sniff_groups_terms, [73](#)
 sniff_key_route, [75](#)
 sniff_network, [56](#), [58](#), [60](#), [61](#), [63](#), [65](#), [66](#), [68](#), [70](#), [77](#)
 sniff_trajectory_braid, [9](#), [78](#), [80](#), [87](#), [95](#), [96](#)
 sniff_trajectory_braid(), [8](#), [9](#), [14](#), [16](#), [26](#), [30](#), [31](#), [44](#), [45](#), [47](#), [48](#), [70](#), [79–82](#), [84–96](#)
 sniff_trajectory_cct, [79](#), [82](#), [85–87](#), [89](#), [91–94](#)
 sniff_trajectory_cct(), [88](#), [89](#), [92](#)
 sniff_trajectory_channel, [9](#), [78](#), [80](#), [87](#), [95](#), [96](#)
 sniff_trajectory_channel(), [78](#), [81](#), [86](#)
 sniff_trajectory_coherence, [81](#), [83](#)
 sniff_trajectory_coherence(), [83](#), [86](#)
 sniff_trajectory_community, [79](#), [82](#), [85–87](#), [89](#), [91–94](#)
 sniff_trajectory_community(), [91](#)
 sniff_trajectory_comparison, [81](#), [83](#)
 sniff_trajectory_comparison(), [81](#)
 sniff_trajectory_confluence, [79](#), [82](#), [84](#), [86](#), [87](#), [89](#), [91–94](#)
 sniff_trajectory_confluence(), [23–26](#), [28–30](#), [44](#), [45](#), [47](#), [48](#)
 sniff_trajectory_contribution, [79](#), [82](#), [85](#), [85](#), [87](#), [89](#), [91–94](#)
 sniff_trajectory_contribution(), [91](#)
 sniff_trajectory_dag, [9](#), [78](#), [80](#), [86](#), [95](#), [96](#)
 sniff_trajectory_dag(), [22](#), [23](#), [32–34](#), [44](#), [47](#), [78](#), [80](#), [84](#)
 sniff_trajectory_destination, [79](#), [82](#), [85](#), [86](#), [87](#), [89](#), [91–94](#)
 sniff_trajectory_destination(), [36](#), [37](#), [85](#), [86](#), [89](#), [94](#)
 sniff_trajectory_dynamics, [79](#), [82](#), [85–87](#), [88](#), [91–94](#)
 sniff_trajectory_dynamics(), [5](#), [6](#), [37–41](#), [78](#), [79](#), [82](#), [90–92](#), [94](#)
 sniff_trajectory_emergence_owners, [79](#), [82](#), [85–87](#), [89](#), [90](#), [92–94](#)
 sniff_trajectory_entropy, [79](#), [82](#), [85–87](#), [90](#), [91](#), [91](#), [93](#), [94](#)
 sniff_trajectory_entropy(), [79](#), [89](#)
 sniff_trajectory_formation, [79](#), [82](#), [85–87](#), [90](#), [91](#), [92](#), [92](#), [94](#)
 sniff_trajectory_formation(), [42](#)
 sniff_trajectory_hubs, [79](#), [82](#), [85–87](#), [90–92](#), [93](#), [93](#), [94](#)
 sniff_trajectory_hubs(), [82](#), [89](#)
 sniff_trajectory_self_sufficiency, [79](#), [82](#), [85–87](#), [90–93](#), [94](#), [94](#)
 sniff_trajectory_self_sufficiency(), [86](#), [87](#), [89](#)
 sort(), [9](#)
 stringr::str_extract_all(), [9](#)
 subset.birdog_flow, [9](#), [78](#), [80](#), [87](#), [95](#), [96](#)
 validate_flow, [9](#), [78](#), [80](#), [87](#), [95](#), [96](#)
 validate_flow(), [8](#), [9](#), [80](#), [81](#), [95](#)