Package: tidybrregpanel (via r-universe)

November 2, 2026

Title Snapshots, Synchronisation and Panels for Norway's Business Registry

Version 0.6.0

Description The managed-state companion to 'tidybrreg'. Saves dated bulk downloads of the Norwegian Central Coordinating Register for Legal Entities (Enhetsregisteret) as Hive-partitioned Parquet snapshots, keeps a local mirror current from the register's change data capture feeds with a persistent changelog, replays updates onto a base extract, and derives firm-period panels, entry and exit flows, aggregate time series, role-change diffs, annotation and name history, director networks and survival data. All retrieval goes through 'tidybrreg'; this package adds persistence and analysis.

License MIT + file LICENSE

URL https://sondreskarsten.github.io/tidybrreg/, https://github.com/sondreskarsten/tidybrreg

BugReports https://github.com/sondreskarsten/tidybrreg/issues

Depends R (>= 4.1.0)

Imports cli, dplyr, httr2 (>= 1.0.0), jsonlite, rlang, tibble, tidybrreg (>= 0.6.0), vctrs

Suggests arrow (>= 12.0.0), nanoparquet (>= 0.3.0), curl, duckdb, klassR, knitr, rmarkdown, tidygraph, tidyr, tsibble, yyjsonr, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.1

Repository https://sondreskarsten.r-universe.dev

Date/Publication 2026-11-02 18:57:27 UTC

RemoteUrl https://github.com/sondreskarsten/tidybrreg

RemoteRef HEAD

RemoteSha 55e87e08a95137ad97098a794e327be25ac6aa01

RemoteSubdir tidybrregpanel

Contents

annotation_infotypes . . . 3
as_brreg_tsibble . . . 3
brreg_annotation_summary . . . 4
brreg_annotations . . . 5
brreg_board_network . . . 6
brreg_board_summary . . . 7
brreg_change_summary . . . 8
brreg_changes . . . 8
brreg_cleanup . . . 10
brreg_data_dir . . . 11
brreg_events . . . 11
brreg_flows . . . 12
brreg_harmonize_kommune . . . 14
brreg_harmonize_nace . . . 15
brreg_historical_names . . . 16
brreg_import . . . 17
brreg_manifest . . . 18
brreg_network . . . 18
brreg_open . . . 20
brreg_panel . . . 21
brreg_replay . . . 22
brreg_series . . . 23
brreg_snapshot . . . 24
brreg_snapshots . . . 26
brreg_status . . . 26
brreg_survival_data . . . 27
brreg_sync . . . 28
brreg_sync_status . . . 29
diff_roller_state . . . 30

Index **32**

annotation_infotypes *Annotation infotype codes with English descriptions*

Description

Maps brreg påtegning `infotype` codes to English descriptions. Used by `brreg_annotations()` when `translate = TRUE`. Covers the infotype codes documented in the brreg API reference (`"NAVN"`, `"FADR"`) and those observed in live data (role codes used for missing-role annotations); unknown codes pass through unchanged. Each annotation's own Norwegian text is available in the `tekst` column of `brreg_annotations()` output.

Usage

```
annotation_infotypes
```

Format

A tibble with 7 rows and 2 columns:

code Annotation infotype code (e.g. `"FADR"`, `"NAVN"`).

name_en English description.

See Also

`brreg_annotations()` for the function that uses this table.

Examples

```
annotation_infotypes
```

as_brreg_tsibble *Convert tidybrreg output to tsibble*

Description

Convert the output of `brreg_panel()` or `brreg_series()` to a tsibble for use with the tidyverts ecosystem (fable, feasts, slider). Uses `regular = FALSE` since brreg snapshots are irregularly spaced.

Usage

```
as_brreg_tsibble(x, key = NULL, index = NULL)
```

Arguments

x	A tibble from brreg_panel() or brreg_series().
key	Character vector of key column(s). For panels, typically "org_nr". For series, the grouping variable (e.g. "legal_form"). If NULL, inferred from the brreg_panel_meta attribute.
index	Character. Name of the time index column. Default "period" for series output, "snapshot_date" for panel output.

Value

A tsibble.

See Also

brreg_panel(), brreg_series().

Other tidybrreg panel functions: brreg_change_summary(), brreg_changes(), brreg_events(), brreg_flows(), brreg_panel(), brreg_replay(), brreg_series()

Examples

```
panel <- brreg_panel(cols = c("employees", "legal_form"))
as_brreg_tsibble(panel)
```

---

brreg_annotation_summary

Count entities with active annotations

Description

Quick summary of how many entities currently carry påtegninger, grouped by annotation type.

Usage

```
brreg_annotation_summary()
```

Value

A tibble with infotype and n.

See Also

Other tidybrreg data management functions: brreg_annotations(), brreg_historical_names(), brreg_status(), brreg_sync(), brreg_sync_status(), diff_roller_state(), read_changelog()

Examples

```
brreg_annotation_summary()
```

`brreg_annotations` *Retrieve registry annotations (påtegninger) for entities*

Description

Påtegninger are public annotations placed on entities by the register keeper to warn third parties of irregularities — missing board members, undelivered accounts, deceased officers. They are the earliest formal signal that an entity is in regulatory trouble, preceding forced dissolution warnings by weeks to months.

Usage

```
brreg_annotations(
  org_nr = NULL,
  infotype = NULL,
  active_only = TRUE,
  translate = FALSE
)
```

Arguments

`org_nr`	Optional character vector of organisation numbers. `NULL` returns all annotations.
`infotype`	Optional character vector of annotation type codes to filter by (e.g. `"FADR"`, `"NAVN"`).
`active_only`	Logical. If `TRUE` (default), return only annotations currently in force. If `FALSE`, include cleared annotations from the changelog.
`translate`	Logical. If `TRUE`, add an `infotype_desc` column with English descriptions from annotation_infotypes; unknown codes pass through unchanged. Default `FALSE`.

Details

Requires `brreg_sync()` to have been run at least once to populate the local påtegninger state.

Value

A tibble with columns: `org_nr`, `position` (array index), `infotype`, `tekst` (annotation text), `innfoert_dato` (date introduced). With `translate = TRUE`, an `infotype_desc` column is inserted after `infotype`.

See Also

`brreg_sync()` to populate annotation data, `brreg_changes()` to track annotation events over time, annotation_infotypes for the English `infotype` lookup table.

Other tidybrreg data management functions: `brreg_annotation_summary()`, `brreg_historical_names()`, `brreg_status()`, `brreg_sync()`, `brreg_sync_status()`, `diff_roller_state()`, `read_changelog()`

Examples

```
brreg_sync()
brreg_annotations()
brreg_annotations(infotype = "FADR")
brreg_annotations(translate = TRUE)
```

`brreg_board_network` *Build a director interlock network*

Description

Construct a bipartite graph of entities and persons linked by board/officer roles. Returns a `tbl_graph` (tidygraph) object suitable for centrality analysis and ggraph visualization.

Usage

```
brreg_board_network(org_nrs = NULL, roles_data = NULL)
```

Arguments

`org_nrs`	Character vector of organization numbers to include. Roles are fetched via `tidybrreg::brreg_roles()` for each entity.
`roles_data`	Optional pre-fetched roles tibble (from `tidybrreg::brreg_roles()`). If provided, `org_nrs` is ignored.

Details

For a full ego network including sub-units, child entities, and legal roles in addition to board roles, use `brreg_network()` instead.

Value

A `tbl_graph` with two node types: `"entity"` (identified by `org_nr`) and `"person"` (identified by `person_id`). Edge attributes include `role_code`, `role_group_code`, and `org_nr`.

See Also

`brreg_network()` for full entity network graphs, `tidybrreg::brreg_roles()` to fetch role data, `brreg_board_summary()` for board-level covariates.

Other tidybrreg governance functions: `brreg_board_summary()`, `brreg_network()`, `brreg_survival_data()`

Examples

```
net <- brreg_board_network(c("923609016", "984851006"))
net
```

`brreg_board_summary` *Derive board-level summary covariates from role data*

Description

Compute firm-level variables commonly used in corporate governance research: board size, composition counts, and officer indicators.

Usage

```
brreg_board_summary(roles)
```

Arguments

`roles`	A tibble returned by `tidybrreg::brreg_roles()`.

Value

A 1-row tibble with columns: `org_nr`, `board_size`, `n_chair`, `n_deputy_chair`, `n_members`, `n_alternates`, `n_observers`, `n_employee_elected`, `has_ceo`, `has_auditor`, `auditor_org_nr`. Counts exclude resigned and deregistered roles.

See Also

`tidybrreg::brreg_roles()` for the underlying role data.

Other tidybrreg governance functions: `brreg_board_network()`, `brreg_network()`, `brreg_survival_data()`

Examples

```
brreg_roles("923609016") |> brreg_board_summary()
```

brreg_change_summary *Summarize changes by field and type*

Description

Produces a count table of how many changes occurred per field and change type, useful for understanding the volume and distribution of registry mutations.

Usage

```
brreg_change_summary(from = NULL, to = NULL, registry = NULL)
```

Arguments

from, to	Date range (inclusive).
registry	Character vector of streams to include. Default includes all four.

Value

A tibble with registry, change_type, field, n.

See Also

brreg_changes() for raw changelog rows, brreg_sync() to populate the changelog.

Other tidybrreg panel functions: as_brreg_tsibble(), brreg_changes(), brreg_events(), brreg_flows(), brreg_panel(), brreg_replay(), brreg_series()

Examples

```
brreg_change_summary(from = Sys.Date() - 7)
```

brreg_changes *Query the change log for field-level mutations*

Description

Returns a filtered view of all recorded changes across the four sync streams: enheter, underenheter, roller, and påtegninger. Every call to brreg_sync() appends events to the changelog; this function reads and filters them.

Usage

```
brreg_changes(
  track = NULL,
  registry = NULL,
  change_type = NULL,
  from = NULL,
  to = NULL,
  org_nr = NULL
)
```

Arguments

track	Character vector of fields to include (e.g. c("nace_1", "municipality_code", "employees")). NULL returns all fields.
registry	Character vector of streams to include. Default includes all four.
change_type	Character vector of event types to include. Options: "entry", "exit", "change", "annotation_added", "annotation_cleared".
from, to	Date range (inclusive).
org_nr	Optional character vector of organisation numbers.

Value

A tibble with columns: timestamp, org_nr, registry, change_type, field, value_from, value_to, update_id.

See Also

brreg_sync() to populate the changelog, brreg_flows() for aggregated entry/exit counts.

Other tidybrreg panel functions: as_brreg_tsibble(), brreg_change_summary(), brreg_events(), brreg_flows(), brreg_panel(), brreg_replay(), brreg_series()

Examples

```
brreg_sync()

# All changes this month
brreg_changes(from = Sys.Date() - 30)

# NACE reclassifications only
brreg_changes(track = "nace_1", change_type = "change")

# Entries and exits for a specific entity
brreg_changes(org_nr = "923609016", change_type = c("entry", "exit"))

# All annotation events
brreg_changes(registry = "paategninger")
```

brreg_cleanup *Remove old snapshots from the local store*

Description

Delete snapshot partitions by count or age. At least one of `keep_n` or `max_age_days` must be provided.

Usage

```
brreg_cleanup(
  keep_n = NULL,
  max_age_days = NULL,
  type = c("enheter", "underenheter", "roller")
)
```

Arguments

`keep_n`	Integer. Keep the `keep_n` most recent snapshots and delete the rest. `NULL` to skip this criterion.
`max_age_days`	Integer. Delete snapshots older than this many days. `NULL` to skip this criterion.
`type`	One of `"enheter"` or `"underenheter"`.

Value

A tibble of deleted snapshots (invisibly).

See Also

`brreg_snapshots()` to list available snapshots, `brreg_data_dir()` for storage location.

Other tidybrreg snapshot functions: `brreg_data_dir()`, `brreg_import()`, `brreg_manifest()`, `brreg_open()`, `brreg_snapshot()`, `brreg_snapshots()`

Examples

```
brreg_cleanup(keep_n = 12)
brreg_cleanup(max_age_days = 365)
```

brreg_data_dir *Path to the tidybrreg snapshot store*

Description

Returns (and creates if needed) the directory where tidybrreg stores Parquet snapshots. Location follows R's standard user data directory convention via `tools::R_user_dir("tidybrreg", "data")`. Override with `options(brreg.data_dir = "/custom/path")`.

Usage

```
brreg_data_dir()
```

Value

Character path.

See Also

`brreg_snapshot()` to save snapshots, `brreg_open()` to read them.

Other tidybrreg snapshot functions: `brreg_cleanup()`, `brreg_import()`, `brreg_manifest()`, `brreg_open()`, `brreg_snapshot()`, `brreg_snapshots()`

Examples

```
brreg_data_dir()
```

brreg_events *Detect changes between two snapshots*

Description

Compare two dated snapshots and return a tibble of entity-level events: entries (new entities), exits (deleted entities), and field-level changes. Unlike the CDC stream, snapshot diffs provide both old and new values for every changed field.

Usage

```
brreg_events(
  date_from,
  date_to,
  cols = NULL,
  type = c("enheter", "underenheter")
)
```

Arguments

`date_from`, `date_to`
: Dates identifying the two snapshots to compare. Both must exist in the snapshot store. `date_from` is the "before" state, `date_to` is the "after" state.

`cols`
: Character vector of columns to track for changes. `NULL` (default) tracks all columns in tidybrreg::field_dict.

`type`
: One of `"enheter"` or `"underenheter"`.

Value

A tibble with columns: `org_nr`, `event_type` (`"entry"`, `"exit"`, `"change"`), `event_date` (the `date_to` snapshot date), `field` (column name, `NA` for entry/exit), `value_from` (character, previous value), `value_to` (character, new value).

See Also

`tidybrreg::brreg_updates()` for the CDC stream (API-level changes), `brreg_panel()` for full panel construction.

Other tidybrreg panel functions: `as_brreg_tsibble()`, `brreg_change_summary()`, `brreg_changes()`, `brreg_flows()`, `brreg_panel()`, `brreg_replay()`, `brreg_series()`

Examples

```
snaps <- brreg_snapshots()
if (nrow(snaps) >= 2) {
  events <- brreg_events(snaps$snapshot_date[1], snaps$snapshot_date[2])
  events
}
```

---

brreg_flows *Compute daily entry and exit flows*

Description

Calculate daily counts of entity registrations (entries) and deletions (exits) classified by industry (NACE code) and geography (municipality code). Three data paths, selected automatically:

Usage

```
brreg_flows(
  data = NULL,
  updates = NULL,
  by = c("nace_1", "municipality_code"),
  from = NULL,
```

```
  to = NULL,
  legal_form = NULL
)
```

Arguments

data	Optional. A tibble from tidybrreg::brreg_download() or a snapshot. Required when no changelog exists. Must contain org_nr, registration_date, nace_1, and municipality_code.
updates	Optional. A tibble from tidybrreg::brreg_updates() with CDC events.
by	Character vector of grouping columns. Default c("nace_1", "municipality_code"). Use NULL for national totals, or any column present in data.
from, to	Date range for the output. NULL defaults to the range of observed events.
legal_form	Optional character vector of legal form codes to include (e.g. c("AS", "ENK")). NULL includes all.

Details

1. **Changelog path** (preferred) — when brreg_sync() has been run, reads directly from the persistent changelog. Provides timestamped entries, exits, and field-level transitions. No arguments needed.
2. **Bulk + CDC path** — pass data (from tidybrreg::brreg_download()) and optionally updates (from tidybrreg::brreg_updates()). Registration dates provide historical entries; CDC provides recent entries + exits.
3. **Bulk-only path** — pass data alone. Only entries are computed (no exit data available).

Value

A tibble with columns: date (Date), grouping columns from by, entries (integer), exits (integer), net (integer: entries - exits). An attribute flow_source records which data sources contributed.

Entry vs. founding date

This function uses registration_date (registreringsdatoEnhetsregisteret), NOT founding_date (stiftelsesdato). Registration date is when the entity entered the registry. Founding date can precede registration by months (AS companies) or years (associations).

See Also

tidybrreg::brreg_download() to get bulk data, tidybrreg::brreg_updates() to get CDC events, brreg_series() for snapshot-based time series, as_brreg_tsibble() for tsibble conversion.

Other tidybrreg panel functions: as_brreg_tsibble(), brreg_change_summary(), brreg_changes(), brreg_events(), brreg_panel(), brreg_replay(), brreg_series()

Examples

```
entities <- brreg_download()
flows <- brreg_flows(entities)

# With CDC exits
cdc <- brreg_updates(since = "2026-01-01", size = 10000)
flows <- brreg_flows(entities, updates = cdc)

# Monthly by NACE section
flows |>
  dplyr::mutate(month = format(date, "%Y-%m"),
                nace_section = substr(nace_1, 1, 2)) |>
  dplyr::summarise(entries = sum(entries), exits = sum(exits),
                   .by = c(month, nace_section))
```

---

brreg_harmonize_kommune

Harmonize municipality codes across boundary reforms

Description

Remap `municipality_code` (kommunenummer) to the classification valid at a target date, using correspondence tables from SSB's KLASS system (classification 131). Handles the 2020 municipal reform (428 → 356 municipalities) and 2024 county reversals.

Usage

```
brreg_harmonize_kommune(
  data,
  target_date = Sys.Date(),
  col = "municipality_code"
)
```

Arguments

`data`	A tibble with a municipality code column.
`target_date`	Date. Remap all codes to the classification valid at this date. Default: today.
`col`	Column name containing municipality codes. Default `"municipality_code"`.

Value

The input tibble with two added columns: `{col}_harmonized` (the remapped code) and `{col}_target_name` (municipality name at the target date). Unmatched codes pass through unchanged.

See Also

brreg_harmonize_nace() for NACE code harmonization.

Other tidybrreg harmonization functions: brreg_harmonize_nace()

Examples

```
# Remap old codes to current boundaries
df <- tibble::tibble(municipality_code = c("0301", "1201", "0602"))
brreg_harmonize_kommune(df)
```

---

brreg_harmonize_nace *Harmonize NACE industry codes across classification revisions*

Description

Remap NACE codes between SN2007 (NACE Rev. 2) and SN2025 (NACE Rev. 2.1) using SSB KLASS correspondence tables.

Usage

```
brreg_harmonize_nace(data, from = "SN2007", to = "SN2025", col = "nace_1")
```

Arguments

data	A tibble with a NACE code column.
from	Source classification: "SN2007" (default) or "SN2025".
to	Target classification: "SN2025" (default) or "SN2007".
col	Column name containing NACE codes. Default "nace_1".

Value

The input tibble with {col}_harmonized (remapped code) and {col}_ambiguous (logical, TRUE when the mapping is one-to-many and the first match was used). Unmatched codes pass through unchanged.

See Also

brreg_harmonize_kommune() for municipality harmonization.

Other tidybrreg harmonization functions: brreg_harmonize_kommune()

Examples

```
df <- tibble::tibble(nace_1 = c("06.100", "64.190", "62.010"))
brreg_harmonize_nace(df, from = "SN2007", to = "SN2025")
```

`brreg_historical_names` *Retrieve historical names (historiskeNavn) for entities*

Description

Norwegian entities keep their organization number across renames; the register exposes prior names with the interval each was in force. BRREG added the `historiskeNavn` field to enheter and underenheter on 2026-06-16. Name history accrues through `brreg_sync()`: every rename in the CDC feed appends the superseded name here. History predating your first sync comes from the JSON bulk download and is backfilled when `brreg_sync()` bootstraps.

Usage

```
brreg_historical_names(org_nr = NULL)
```

Arguments

`org_nr`	Optional character vector of organisation numbers. `NULL` returns all name history.

Details

Requires `brreg_sync()` to have been run at least once to populate the local name-history state.

Value

A tibble with columns: `org_nr`, `position` (array index), `name` (the superseded name), `from_date`, `to_date` (register timestamps as observed, format `"YYYY-MM-DD HH:MM:SS"`, kept as character).

See Also

`brreg_sync()` to populate name-history data, `brreg_changes()` with `registry = "historiske_navn"` to track rename events over time.

Other tidybrreg data management functions: `brreg_annotation_summary()`, `brreg_annotations()`, `brreg_status()`, `brreg_sync()`, `brreg_sync_status()`, `diff_roller_state()`, `read_changelog()`

Examples

```
brreg_sync()
brreg_historical_names()
brreg_historical_names("923609016")
```

`brreg_import` *Import a historical bulk JSON file as a snapshot partition*

Description

Read a brreg bulk JSON file (as downloaded by `tidybrreg::brreg_download()` with `type_output = "path"`, or from the brreg website), normalize column names via tidybrreg::field_dict, and save as a dated Parquet partition in the snapshot store.

Usage

```
brreg_import(
  path,
  snapshot_date,
  type = c("enheter", "underenheter", "roller"),
  force = FALSE
)
```

Arguments

`path`	Path to a brreg bulk JSON file (gzipped or plain): the entity array of `/enheter/lastned` or `/underenheter/lastned`, or the `/roller/totalbestand` array.
`snapshot_date`	The date this file represents. Required — the file itself contains no date metadata.
`type`	One of `"enheter"`, `"underenheter"` or `"roller"`.
`force`	Logical. Overwrite existing partition.

Value

The file path to the written Parquet partition (invisibly).

See Also

`brreg_snapshot()` to download and save today's register.

Other tidybrreg snapshot functions: `brreg_cleanup()`, `brreg_data_dir()`, `brreg_manifest()`, `brreg_open()`, `brreg_snapshot()`, `brreg_snapshots()`

Examples

```
# Import a historical download
brreg_import("enheter_bulk_2024-12-31.json.gz", snapshot_date = "2024-12-31")
```

`brreg_manifest` *Read the snapshot manifest*

Description

Returns the provenance catalog recording every download: endpoint, timestamps, HTTP headers, file hashes, and CDC bridge metadata. The manifest lives at `brreg_data_dir()/manifest.json`.

Usage

```
brreg_manifest()
```

Value

A tibble with one row per snapshot. Returns an empty tibble if no manifest exists.

See Also

`brreg_snapshot()` to create snapshots, `brreg_snapshots()` to list them.

Other tidybrreg snapshot functions: `brreg_cleanup()`, `brreg_data_dir()`, `brreg_import()`, `brreg_open()`, `brreg_snapshot()`, `brreg_snapshots()`

Examples

```
brreg_manifest()
```

`brreg_network` *Build an entity network graph*

Description

Construct a `tbl_graph` (tidygraph) representing the relationships around one or more seed entities. At depth 1, the graph includes sub-units, child entities, role holders, and legal role targets — all reachable via direct API calls. At depth 2, the graph expands through person nodes to discover board interlocks, requiring local bulk data (see Details).

Usage

```
brreg_network(
  org_nr,
  depth = 1L,
  include = c("underenheter", "children", "roles", "legal_roles"),
  download = FALSE
)
```

Arguments

org_nr	Character vector of seed organization numbers.
depth	Integer. 0 = seed only, 1 = ego network (default), 2 = expand through persons (requires bulk data).
include	Character vector of relationship types to include. Default includes all available types. Current types: "underenheter", "children", "roles", "legal_roles".
download	Logical. If TRUE and depth > 1, offer to download missing bulk data interactively. Default FALSE.

Value

A tbl_graph with node attributes node_id, node_type, name, org_nr, person_id, and edge attributes from, to, edge_type, role_code, role.

Depth and data requirements

- **Depth 0**: Seed entity only. 1 API call per seed.
- **Depth 1**: Full ego network. 5-7 API calls per seed.
- **Depth 2**: Board interlocks via person-to-entity reverse lookup. Requires local bulk data for enheter, underenheter, and roller. Run brreg_snapshot() for each type first, or call with download = TRUE to trigger downloads interactively.

Extensibility

The include parameter controls which relationship types are traversed. Each type maps to an internal collector function. Future versions may add types such as "addresses", "prior_owners", or "accounting".

See Also

tidybrreg::brreg_entity() for single lookups, brreg_board_network() for the roles-only subgraph, brreg_status() to check bulk data availability.

Other tidybrreg governance functions: brreg_board_network(), brreg_board_summary(), brreg_survival_data()

Examples

```
net <- brreg_network("923609016")
net

tidygraph::as_tibble(net, "nodes")
tidygraph::as_tibble(net, "edges")
```

`brreg_open` *Open the snapshot store as a lazy Arrow Dataset*

Description

Returns an Arrow Dataset with Hive-style partitioning on `snapshot_date`. No data is loaded until `dplyr::collect()`. Requires the arrow package.

Usage

```
brreg_open(type = c("enheter", "underenheter", "roller"))
```

Arguments

`type`	One of `"enheter"` or `"underenheter"`.

Value

An `arrow::Dataset` object.

See Also

`brreg_panel()` for the higher-level panel constructor.

Other tidybrreg snapshot functions: `brreg_cleanup()`, `brreg_data_dir()`, `brreg_import()`, `brreg_manifest()`, `brreg_snapshot()`, `brreg_snapshots()`

Examples

```
ds <- brreg_open()
ds
```

brreg_panel *Construct a firm-period panel from accumulated snapshots*

Description

Build an unbalanced firm × period panel from the Hive-partitioned snapshot store. For each target period, selects the most recent prior snapshot (LOCF). Requires at least two snapshots saved via `brreg_snapshot()` or `brreg_import()`.

Usage

```
brreg_panel(
  frequency = c("year", "quarter", "month", "custom"),
  cols = NULL,
  from = NULL,
  to = NULL,
  dates = NULL,
  max_gap = NULL,
  type = c("enheter", "underenheter", "roller"),
  label = FALSE
)
```

Arguments

`frequency`	One of `"year"` (default), `"quarter"`, `"month"`, or `"custom"`. For `"custom"`, supply target dates via `dates`.
`cols`	Character vector of column names to include. `NULL` (default) returns all columns mapped by tidybrreg::field_dict.
`from, to`	Start and end dates for the panel. `NULL` defaults to the range of available snapshots.
`dates`	Date vector for `frequency = "custom"`.
`max_gap`	Integer. Maximum number of periods a snapshot may carry forward via LOCF. `NULL` (default) carries forward indefinitely. Set `max_gap = 2` to prevent a quarterly snapshot from representing a firm as active 3+ quarters after its last observation.
`type`	One of `"enheter"` or `"underenheter"`.
`label`	Logical. If `TRUE`, apply `tidybrreg::brreg_label()` to the result.

Value

A tibble with columns: `org_nr`, `period` (character label for the period), `snapshot_date` (the actual snapshot used), plus requested `cols`. Attribute `date_mapping` records which snapshot was used for each period.

See Also

brreg_snapshot() to accumulate snapshots, brreg_events() for change detection between snapshots.

Other tidybrreg panel functions: as_brreg_tsibble(), brreg_change_summary(), brreg_changes(), brreg_events(), brreg_flows(), brreg_replay(), brreg_series()

Examples

```
# Annual panel of all entities
panel <- brreg_panel()

# Monthly panel for specific columns
panel <- brreg_panel("month", cols = c("employees", "nace_1", "legal_form"))
```

---

brreg_replay *Reconstruct register state by replaying CDC updates*

Description

Given a base snapshot and a stream of CDC updates from tidybrreg::brreg_updates(), reconstruct the state of the register at any arbitrary date. Uses dplyr::rows_upsert() for Ny/Endring events and dplyr::rows_delete() for Sletting events, applied chronologically.

Usage

```
brreg_replay(base, updates, target_date = Sys.Date(), cols = NULL)
```

Arguments

base	A tibble from tidybrreg::brreg_download(), brreg_import(), or a snapshot read via brreg_open(). Must contain org_nr.
updates	A tibble from tidybrreg::brreg_updates() with include_changes = TRUE. Must contain org_nr, change_type, timestamp, and changes (list-column of patch tibbles).
target_date	Date. Reconstruct state as of this date. Only updates with timestamp <= target_date are applied.
cols	Character vector of columns to track. If NULL, all columns present in base are used.

Value

A tibble with the same columns as base, reflecting all applied changes up to target_date. Attribute replay_info records the number of inserts, updates, and deletes applied.

Limitations

The brreg CDC stream provides only new values (not old values) in RFC 6902 JSON Patch format, and field-level changes are only available from September 2025. Before that date, only the change type (Ny/Endring/Sletting) is recorded — field-level replay is not possible for those periods. Use `brreg_events()` (snapshot diff) for pre-September 2025 field-level changes.

See Also

`brreg_panel()` for multi-snapshot panels, `tidybrreg::brreg_updates()` to fetch the CDC stream.

Other tidybrreg panel functions: `as_brreg_tsibble()`, `brreg_change_summary()`, `brreg_changes()`, `brreg_events()`, `brreg_flows()`, `brreg_panel()`, `brreg_series()`

Examples

```
base <- brreg_download(type_output = "tibble")
updates <- brreg_updates(since = Sys.Date() - 30,
                         size = 10000, include_changes = TRUE)
state <- brreg_replay(base, updates, target_date = Sys.Date())
```

---

brreg_series *Compute aggregate time series from snapshots*

Description

Produce period-level summary statistics from the snapshot store for any combination of variables and summary functions. Returns a tibble suitable for ggplot2 or `as_brreg_tsibble()` conversion.

Usage

```
brreg_series(
  .vars = NULL,
  .fns = list(total = function(x) sum(x, na.rm = TRUE)),
  by = NULL,
  frequency = c("year", "quarter", "month"),
  from = NULL,
  to = NULL,
  type = c("enheter", "underenheter", "roller"),
  label = FALSE
)
```

Arguments

`.vars`	Character vector of column names to aggregate. `NULL` (default) counts entities per period.

.fns	Named list of summary functions applied to each column in .vars. Default: list(total = \(x) sum(x, na.rm = TRUE)). Use list(avg = mean, sd = sd) for multiple summaries. Output columns are named {variable}_{function}.
by	Character vector of grouping column names (e.g. "nace_1", c("legal_form", "municipality_code")). NULL for national totals.
frequency	One of "year", "quarter", "month".
from, to	Start and end dates for the panel. NULL defaults to the range of available snapshots.
type	One of "enheter", "underenheter", "roller".
label	Logical. Translate group codes to English labels.

Value

A tibble with period (character), optional grouping columns, and one column per variable-function combination. Attribute brreg_panel_meta records metadata for as_brreg_tsibble() conversion.

See Also

brreg_panel() for entity-level panels, as_brreg_tsibble() for tsibble conversion.

Other tidybrreg panel functions: as_brreg_tsibble(), brreg_change_summary(), brreg_changes(), brreg_events(), brreg_flows(), brreg_panel(), brreg_replay()

Examples

```
brreg_series(.vars = "employees", by = "legal_form")

brreg_series(.vars = "employees",
             .fns = list(avg = mean, total = sum),
             by = "nace_1")
```

---

brreg_snapshot *Save a dated snapshot of the full register*

Description

Download today's complete register and save as a Parquet partition in the local snapshot store. Each call adds one partition to a Hive-partitioned dataset at tools::R_user_dir("tidybrreg", "data")/{type}/snapshot_da
Subsequent calls to brreg_panel() and brreg_events() query this partitioned dataset lazily via arrow::open_dataset().

Usage

```
brreg_snapshot(
  type = c("enheter", "underenheter", "roller"),
  date = Sys.Date(),
  force = FALSE,
  ask = interactive()
)
```

Arguments

type	One of "enheter" (main entities, default), "underenheter" (sub-entities / establishments), or "roller" (all roles for all entities, via /roller/totalbestand). Roller snapshots parse the nested JSON into a flat tibble matching tidybrreg::brreg_roles() output.
date	Date for this snapshot (default: today). Used as the partition key, not as an API parameter — the brreg bulk endpoint always returns the current-day state.
force	Logical. If TRUE, overwrite an existing partition for this date. Default FALSE skips if partition exists.
ask	Logical. If TRUE (the default in interactive sessions), prompt before downloading ~145 MB.

Value

The file path to the written Parquet partition (invisibly).

See Also

brreg_import() to add historical snapshots from bulk JSON files, brreg_snapshots() to list available snapshots, brreg_panel() to construct panels from accumulated snapshots.

Other tidybrreg snapshot functions: brreg_cleanup(), brreg_data_dir(), brreg_import(), brreg_manifest(), brreg_open(), brreg_snapshots()

Examples

```
brreg_snapshot()
brreg_snapshots()
```

brreg_snapshots *List available snapshots*

Description

Scan the local snapshot store and return metadata for each partition.

Usage

```
brreg_snapshots(type = c("enheter", "underenheter", "roller"))
```

Arguments

type	One of "enheter" or "underenheter".

Value

A tibble with columns: snapshot_date (Date), file_size (numeric, bytes), path (character).

See Also

Other tidybrreg snapshot functions: brreg_cleanup(), brreg_data_dir(), brreg_import(), brreg_manifest(), brreg_open(), brreg_snapshot()

Examples

```
brreg_snapshots()
```

brreg_status *Check availability of local bulk datasets*

Description

Inspects the snapshot store and download cache for each requested dataset type. Used internally by brreg_network() to gate depth > 1 operations that require local data for person-to-entity reverse lookups.

Usage

```
brreg_status(datasets = c("enheter", "underenheter", "roller"), quiet = FALSE)
```

Arguments

datasets	Character vector of dataset types to check.
quiet	Logical. If TRUE, suppress informational messages.

Value

A list with components: `available` (character vector of datasets found locally), `missing` (character vector of datasets not found), `all_ready` (logical).

See Also

`brreg_snapshot()` to download and cache bulk data, `tidybrreg::brreg_download()` for one-off downloads.

Other tidybrreg data management functions: `brreg_annotation_summary()`, `brreg_annotations()`, `brreg_historical_names()`, `brreg_sync()`, `brreg_sync_status()`, `diff_roller_state()`, `read_changelog()`

Examples

```
brreg_status()
```

---

brreg_survival_data *Prepare firm survival data*

Description

Compute time-to-event and censoring indicators from entity registration data, ready for use with `survival::Surv()` and `flexsurv`.

Usage

```
brreg_survival_data(
  data,
  entry_var = "founding_date",
  censoring_date = Sys.Date()
)
```

Arguments

`data`	A tibble from `tidybrreg::brreg_download()`, `tidybrreg::brreg_search()`, or `brreg_panel()`. Must contain at least `org_nr` and a date column for entry.
`entry_var`	Column name for the entry date. Default `"founding_date"` (stiftelsesdato).
`censoring_date`	Date at which surviving firms are right-censored. Default: today.

Value

A tibble with added columns: `entry_date`, `exit_date` (Date or NA), `duration_years` (numeric), `event` (integer: 1 = exit observed, 0 = right-censored). Compatible with `survival::Surv(duration_years, event)`.

See Also

tidybrreg::brreg_download() for full register data.

Other tidybrreg governance functions: brreg_board_network(), brreg_board_summary(), brreg_network()

Examples

```
firms <- brreg_search(legal_form = "AS", municipality_code = "0301",
                      max_results = 100)
surv <- brreg_survival_data(firms)
surv[, c("org_nr", "entry_date", "exit_date", "duration_years", "event")]
```

---

brreg_sync *Synchronize local state with the brreg CDC stream*

Description

Maintains a local mirror of the Enhetsregisteret by applying incremental CDC (change data capture) events to a persistent state table. On first run, bootstraps from a bulk download. Subsequent runs poll the CDC endpoints from the last cursor position and apply mutations.

Usage

```
brreg_sync(
  types = c("enheter", "underenheter", "roller"),
  size = 10000L,
  roller_method = c("bulk", "cdc"),
  verbose = TRUE
)
```

Arguments

types	Character vector of streams to sync. Default syncs all four.
size	Integer. CDC page size per API call (max 10000).
roller_method	One of "bulk" (default) or "cdc". "bulk" downloads the full totalbestand (~131 MB) and computes a field-level diff against previous state — fast and produces granular changelogs. "cdc" fetches current roles per-org via the API for each CDC event — slower but provides sub-daily attribution when syncing multiple times per day.
verbose	Logical. Print progress messages.

Details

Four state files are maintained:

- `enheter.parquet` — main entities (~1M rows)
- `underenheter.parquet` — sub-entities (~500K rows)
- `roller.parquet` — all roles (~3.4M rows)
- `paategninger.parquet` — registry annotations

Every mutation is logged to a Hive-partitioned changelog under `state/changelog/sync_date={date}/`. The changelog drives `brreg_changes()` and `brreg_flows()`.

Value

A list with sync summary: events processed per type, changelog rows written, elapsed time.

Write ordering

Changelog is written first (WAL), then state files, then cursor. If a crash occurs between state and cursor, the next sync replays from the old cursor. Mutations are idempotent (upsert by org_nr), so replay is safe.

See Also

`brreg_sync_status()` to check current state, `brreg_changes()` to query the changelog, `brreg_flows()` for entry/exit counts.

Other tidybrreg data management functions: `brreg_annotation_summary()`, `brreg_annotations()`, `brreg_historical_names()`, `brreg_status()`, `brreg_sync_status()`, `diff_roller_state()`, `read_changelog()`

Examples

```
brreg_sync()
brreg_sync_status()
```

---

brreg_sync_status *Display sync status*

Description

Shows the current state of the sync engine: which state files exist, when the last sync occurred, cursor positions, and changelog size.

Usage

```
brreg_sync_status()
```

Value

A list with status components (invisibly).

See Also

`brreg_sync()` to run a sync, `brreg_changes()` to query the changelog.

Other tidybrreg data management functions: `brreg_annotation_summary()`, `brreg_annotations()`, `brreg_historical_names()`, `brreg_status()`, `brreg_sync()`, `diff_roller_state()`, `read_changelog()`

Examples

```
brreg_sync_status()
```

---

diff_roller_state *Compute field-level diffs between two roller state tibbles*

Description

Pure function: takes two flattened roller tibbles (as produced by `tidybrreg:::flatten_roles()` or `tidybrreg:::parse_roles_bulk()`) and returns a long-format changelog recording every field-level mutation. Detects three categories of change: role additions, role removals, and field-level modifications on continuing roles.

Usage

```
diff_roller_state(
  old_state,
  new_state,
  timestamp = format(Sys.time(), "%Y-%m-%dT%H:%M:%S"),
  update_id = NA_integer_
)
```

Arguments

`old_state`	Tibble. Previous roller state from `read_state()` or an earlier `tidybrreg:::flatten_roles()` call. `NULL` or zero-row tibble treats all current roles as additions.
`new_state`	Tibble. Current roller state from `tidybrreg::brreg_download()` or `tidybrreg:::flatten_roles()`.
`timestamp`	Character or POSIXct. Timestamp for changelog entries (typically the CDC event time or sync time).
`update_id`	Integer or character. Identifier for the sync batch, used as `update_id` in the changelog.

Value

A tibble matching the changelog schema: `timestamp`, `org_nr`, `registry` (always `"roller"`), `change_type` (`"entry"`, `"exit"`, `"change"`), `field`, `value_from`, `value_to`, `update_id`.

Composite key

Each role assignment is identified by `(org_nr, role_group_code, role_code, holder_id)` where `holder_id` is derived from `person_id` for person-held roles and `entity:{entity_org_nr}` for entity-held roles (auditors, accountants). Roles with neither are keyed as `unknown:{row_index}` within their respective state — these are rare and produce conservative add/remove pairs rather than false modifications.

NA handling

For additions, fields that are `NA` in the new state are excluded from the changelog (no value to report). For removals, fields that are `NA` in the old state are excluded. For modifications, a change from `NA` to a non-NA value (or vice versa) is recorded.

See Also

`brreg_sync()` for automated sync with changelog persistence, `tidybrreg::brreg_roles()` for fetching current roles, `brreg_changes()` for querying stored changelogs.

Other tidybrreg data management functions: `brreg_annotation_summary()`, `brreg_annotations()`, `brreg_historical_names()`, `brreg_status()`, `brreg_sync()`, `brreg_sync_status()`, `read_changelog()`

Examples

```
# Detect board changes for a single company
old <- brreg_roles("923609016")
Sys.sleep(1)
new <- brreg_roles("923609016")
diff_roller_state(old, new)

# Bootstrap: NULL old state treats all roles as entries
roles <- brreg_roles("923609016")
diff_roller_state(NULL, roles)
```

Index

∗ **datasets**
annotation_infotypes, 3
∗ **tidybrreg data management functions**
brreg_annotation_summary, 4
brreg_annotations, 5
brreg_historical_names, 16
brreg_status, 26
brreg_sync, 28
brreg_sync_status, 29
diff_roller_state, 30
∗ **tidybrreg governance functions**
brreg_board_network, 6
brreg_board_summary, 7
brreg_network, 18
brreg_survival_data, 27
∗ **tidybrreg harmonization functions**
brreg_harmonize_kommune, 14
brreg_harmonize_nace, 15
∗ **tidybrreg panel functions**
as_brreg_tsibble, 3
brreg_change_summary, 8
brreg_changes, 8
brreg_events, 11
brreg_flows, 12
brreg_panel, 21
brreg_replay, 22
brreg_series, 23
∗ **tidybrreg snapshot functions**
brreg_cleanup, 10
brreg_data_dir, 11
brreg_import, 17
brreg_manifest, 18
brreg_open, 20
brreg_snapshot, 24
brreg_snapshots, 26
∗ **tidybrregpanel reference data**
annotation_infotypes, 3

annotation_infotypes, 3, *5*
as_brreg_tsibble, 3, *8, 9*, *12, 13*, *22–24*
as_brreg_tsibble(), *13*, *23, 24*

brreg_annotation_summary, 4, *5*, *16*, *27*, *29–31*
brreg_annotations, *4*, 5, *16*, *27*, *29–31*
brreg_annotations(), *3*
brreg_board_network, 6, *7*, *19*, *28*
brreg_board_network(), *19*
brreg_board_summary, *6*, 7, *19*, *28*
brreg_board_summary(), *6*
brreg_change_summary, *4*, 8, *9*, *12, 13*, *22–24*
brreg_changes, *4*, 8, *8*, *12, 13*, *22–24*
brreg_changes(), *5*, *8*, *16*, *29–31*
brreg_cleanup, 10, *11*, *17, 18*, *20*, *25, 26*
brreg_data_dir, *10*, 11, *17, 18*, *20*, *25, 26*
brreg_data_dir(), *10*
brreg_events, *4*, *8, 9*, 11, *13*, *22–24*
brreg_events(), *22–24*
brreg_flows, *4*, *8, 9*, 12, *12*, *22–24*
brreg_flows(), *9*, *29*
brreg_harmonize_kommune, 14, *15*
brreg_harmonize_kommune(), *15*
brreg_harmonize_nace, 15, *15*
brreg_harmonize_nace(), *15*
brreg_historical_names, *4, 5*, 16, *27*, *29–31*
brreg_import, *10, 11*, 17, *18*, *20*, *25, 26*
brreg_import(), *21, 22*, *25*
brreg_manifest, *10, 11*, *17*, 18, *20*, *25, 26*
brreg_network, *6, 7*, 18, *28*
brreg_network(), *6*, *26*
brreg_open, *10, 11*, *17, 18*, 20, *25, 26*
brreg_open(), *11*, *22*
brreg_panel, *4*, *8, 9*, *12, 13*, 21, *23, 24*
brreg_panel(), *3, 4*, *12*, *20*, *23–25*, *27*
brreg_replay, *4*, *8, 9*, *12, 13*, *22*, *22*, *24*
brreg_series, *4*, *8, 9*, *12, 13*, *22*, 23, *23*
brreg_series(), *3, 4*, *13*
brreg_snapshot, *10, 11*, *17, 18*, *20*, 24, *26*
brreg_snapshot(), *11*, *17–19*, *21, 22*, *27*

brreg_snapshots, *10, 11*, *17, 18*, *20*, *25*, 26
brreg_snapshots(), *10*, *18*, *25*
brreg_status, *4, 5*, *16*, 26, *29–31*
brreg_status(), *19*
brreg_survival_data, *6, 7*, *19*, 27
brreg_sync, *4, 5*, *16*, *27*, 28, *30, 31*
brreg_sync(), *5*, *8, 9*, *13*, *16*, *30, 31*
brreg_sync_status, *4, 5*, *16*, *27*, 29, *29*, *31*
brreg_sync_status(), *29*

diff_roller_state, *4, 5*, *16*, *27*, *29*, 30, *30*

read_changelog, *4, 5*, *16*, *27*, *29–31*
read_state(), *30*

tidybrreg::brreg_download(), *13*, *17*, *22*, *27, 28*, *30*
tidybrreg::brreg_entity(), *19*
tidybrreg::brreg_label(), *21*
tidybrreg::brreg_roles(), *6, 7*, *25*, *31*
tidybrreg::brreg_search(), *27*
tidybrreg::brreg_updates(), *12, 13*, *22, 23*
tidybrreg::field_dict, *12*, *17*, *21*