Package 'OPCC'

September 22, 2026

Title Open Postal Code Correspondence

Version 1.0.1

Description An open, fully reproducible alternative to closed postal code conversion files, giving source-qualified many-to-many correspondence between Ontario postal codes and Statistics Canada 2021 census geographies. Every published artifact can be rebuilt step by step from public sources using the included build functions, so any user can reproduce and audit the conversion end to end. Lookup results retain allocation weights, evidence source, lineage, method, and vintage; unmatched postal codes remain explicit, and versioned release artifacts are checksum-verified before use. A built-in 'shiny' application provides a point-and-click interface for joining, mapping, and exporting results. Correspondences are derived from the Statistics Canada National Address Register <https://www150.statcan.gc.ca/n1/pub/46-26-0002/462600022022001-eng.htm>, the 2021 Census Geographic Attribute File <https://www12.statcan.gc.ca/census-recensement/2021/geo/aip-pia/attribute-attribs/index-eng.cfm>, and the GeoNames postal code export <https://download.geonames.org/export/zip/>. The package does not redistribute Canada Post, PCCF, or PCCF+ data and does not claim authoritative postal assignments.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

URL https://github.com/lennon-li/OPCC

BugReports https://github.com/lennon-li/OPCC/issues

Depends R (>= 4.1)

Imports digest, jsonlite

Suggests testthat (>= 3.0.0), curl, dplyr, readr, readxl, rlang, sf, knitr, rmarkdown, withr, shiny (>= 1.8.0), bslib (>= 0.6.0), DT, leaflet, htmlwidgets, chromote, processx, httpuv

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Lennon Li [aut, cre, cph]

Maintainer Lennon Li <yeli@biostats.ai>

Repository CRAN

Date/Publication 2026-09-22 05:50:02 UTC

Contents

aggregate_da_correspondence . . . 3
build_centroids . . . 3
build_db_assignment . . . 4
build_m2 . . . 5
build_source_layer . . . 6
clear_opcc_cache . . . 7
contribution_bundle . . . 8
contribution_issue_url . . . 9
download_census_boundaries . . . 10
download_da_boundaries . . . 11
download_gaf . . . 12
download_geonames . . . 12
download_nar . . . 13
export_postal_points . . . 14
geonames_supplementary_adapter . . . 15
get_correspondence . . . 15
get_da_correspondence . . . 16
list_vintages . . . 17
new_source_adapter . . . 18
normalize_postal_code . . . 19
OPCC . . . 20
open_contribution_issue . . . 20
pc_to_geo . . . 21
pc_to_point . . . 22
profile_source_layer . . . 24
release_manifest . . . 24
run_app . . . 25
validate_release . . . 26
validate_source_data . . . 27

Index **29**

aggregate_da_correspondence *Aggregate postal-code-to-DB evidence to DA links*

Description

Aggregate postal-code-to-DB evidence to DA links

Usage

```
aggregate_da_correspondence(correspondence)
```

Arguments

correspondence	A postal-code-to-DB correspondence data frame.

Value

A data frame with one row per postal_code and DAUID.

Examples

```
# A small in-line postal-code-to-DB evidence table.  Allocation weights
# must sum to one within each postal code.
db_links <- data.frame(
  postal_code = c("M5V 3A8", "M5V 3A8", "M5V 3A8"),
  DBUID = c("35200001000", "35200001001", "35200002000"),
  DAUID = c("35200001", "35200001", "35200002"),
  allocation_weight = c(0.5, 0.25, 0.25),
  source_vintage = "2026-06-26",
  census_vintage = "2021",
  evidence_class = "NAR",
  stringsAsFactors = FALSE
)
aggregate_da_correspondence(db_links)
```

build_centroids *Build Ontario postal code centroids from NAR and GeoNames*

Description

Reads the extracted NAR Address and Location part files, computes the mean coordinate per postal code, and merges with GeoNames Ontario centroids. NAR centroids take priority; GeoNames fills gaps.

Usage

```
build_centroids(nar_dir, geonames_txt, output_dir = NULL)
```

Arguments

nar_dir	Path returned by download_nar().
geonames_txt	Path returned by download_geonames().
output_dir	Required directory for output files.

Value

Invisibly, the path to ontario_postal_centroids.csv.

Examples

```
# Consumes the multi-gigabyte NAR and GeoNames downloads, so it is not run
# automatically.

if (interactive()) {
  cache <- tempfile("opcc-build")
  nar_dir <- download_nar(cache_dir = cache)
  geonames_txt <- download_geonames(cache_dir = cache)
  build_centroids(nar_dir, geonames_txt,
                  output_dir = file.path(cache, "centroids"))
}
```

---

build_db_assignment *Assign postal centroids to Dissemination Blocks and join GAF*

Description

Validates that centroids fall within the Ontario boundary, assigns each to exactly one 2021 Dissemination Block via point-in-polygon, and joins the Geographic Attribute File for higher geographies.

Usage

```
build_db_assignment(
  centroids_csv,
  province_shp,
  db_shp,
  gaf_csv,
  output_dir = NULL
)
```

Arguments

centroids_csv	Path returned by build_centroids().
province_shp	Path to the province boundary .shp from download_census_boundaries().
db_shp	Path to the DB boundary .shp from download_census_boundaries().
gaf_csv	Path returned by download_gaf().
output_dir	Required directory for output files.

Value

Invisibly, the path to ontario_postal_gaf_rollup.csv.

Examples

```
# Consumes the census boundary and GAF downloads, so it is not run
# automatically.

if (interactive()) {
  cache <- tempfile("opcc-build")
  boundaries <- download_census_boundaries(cache_dir = cache)
  gaf_csv <- download_gaf(cache_dir = cache)
  centroids_csv <- build_centroids(
    download_nar(cache_dir = cache),
    download_geonames(cache_dir = cache),
    output_dir = file.path(cache, "centroids")
  )
  build_db_assignment(centroids_csv, boundaries$province, boundaries$db,
                      gaf_csv, output_dir = file.path(cache, "rollup"))
}
```

---

build_m2 *Build the M2 postal-code-to-DB correspondence*

Description

Reads raw NAR address points, assigns them to 2021 Dissemination Blocks via point-in-polygon, joins the GAF, aggregates evidence per postal-code/DB pair, and appends GeoNames supplementary links from the rollup produced by build_db_assignment().

Usage

```
build_m2(nar_dir, db_shp, gaf_csv, rollup_csv, output_dir = NULL)
```

Arguments

nar_dir	Path returned by download_nar().
db_shp	Path to the DB boundary .shp from download_census_boundaries().
gaf_csv	Path returned by download_gaf().
rollup_csv	Path returned by build_db_assignment().
output_dir	Required directory for output files.

Value

Invisibly, the path to m2_correspondence.csv.

Examples

```
# Consumes the NAR, boundary, and GAF downloads plus the DB-assignment
# rollup, so it is not run automatically.

if (interactive()) {
  cache <- tempfile("opcc-build")
  nar_dir <- download_nar(cache_dir = cache)
  boundaries <- download_census_boundaries(cache_dir = cache)
  gaf_csv <- download_gaf(cache_dir = cache)
  rollup_csv <- file.path(cache, "rollup",
                          "ontario_postal_gaf_rollup.csv")
  build_m2(nar_dir, boundaries$db, gaf_csv, rollup_csv,
           output_dir = file.path(cache, "m2"))
}
```

---

build_source_layer *Build a source-separated local evidence layer*

Description

Build a source-separated local evidence layer

Usage

```
build_source_layer(
  data,
  adapter,
  on_invalid = c("error", "drop", "quarantine")
)
```

Arguments

data	A user-supplied postal-code evidence data frame.
adapter	Source metadata created by new_source_adapter().
on_invalid	How to handle invalid rows: error, drop them, or retain them in the opcc_quarantine attribute.

Value

An opcc_source_layer data frame, never merged into a release.

Examples

```
adapter <- new_source_adapter(
  source_id = "example_open_addresses",
  licence = "Open Government Licence - Ontario",
  lineage = "Municipal open address points, retrieved from a city portal",
  retrieval_date = as.Date("2026-01-15"),
  schema_map = list(postal_code = "pc", latitude = "lat", longitude = "lon")
)
raw <- data.frame(
  pc = c("m5v3a8", "K1A 0A6", "not a postal code"),
  lat = c(43.6426, 45.4215, NA),
  lon = c(-79.3871, -75.6972, NA),
  stringsAsFactors = FALSE
)
layer <- build_source_layer(raw, adapter, on_invalid = "drop")
layer[c("postal_code", "source_id", "source_retrieval_date")]
class(layer)
```

---

clear_opcc_cache *Clear the downloaded OPCC release cache*

Description

Removes only OPCC files named opcc-* from cache_dir, including the Shiny app's reusable map artifact in its shiny-app subdirectory. This is an explicit maintenance action; routine downloads retain a small, verified cache capped at 64 MiB.

Usage

```
clear_opcc_cache(cache_dir = NULL)
```

Arguments

cache_dir	Directory containing downloaded OPCC release artifacts. Defaults to NULL, meaning the resolved location described above.

Value

The paths removed, invisibly.

Where the cache lives

OPCC never writes to your home filespace by default. When `cache_dir` is `NULL`, the location is resolved in this order:

1. the `OPCC.cache_dir` option, if set;
2. the `OPCC_CACHE_DIR` environment variable, if set;
3. a recorded consent answer, if you have been asked before;
4. otherwise a session temporary directory, discarded when R exits.

In an interactive session with no answer on record, OPCC asks once whether it may keep artifacts in `tools::R_user_dir()` and remembers the reply. A non-interactive session is never asked, and writes nothing outside `tempdir()` unless you previously granted permission or set the option or environment variable above.

Examples

```
cache <- tempfile("opcc-cache")
dir.create(cache)
clear_opcc_cache(cache)
```

`contribution_bundle` *Create a reviewable local-source contribution bundle*

Description

Create a reviewable local-source contribution bundle

Usage

```
contribution_bundle(layer, output_dir = NULL, fixture_rows = 100L)
```

Arguments

`layer`	A layer created by `build_source_layer()`.
`output_dir`	Explicit directory in which to create a new bundle directory.
`fixture_rows`	Maximum normalized sample rows to include.

Value

A named list of generated bundle paths.

Examples

```
adapter <- new_source_adapter(
  source_id = "example_open_addresses",
  licence = "Open Government Licence - Ontario",
  lineage = "Municipal open address points, retrieved from a city portal",
  retrieval_date = as.Date("2026-01-15")
)
layer <- build_source_layer(
  data.frame(postal_code = c("M5V 3A8", "K1A 0A6")),
  adapter
)

# `output_dir` is required and must be explicit; a session temporary
# directory keeps the example self-contained.
output_dir <- tempfile("opcc-bundle")
bundle <- contribution_bundle(layer, output_dir = output_dir)
basename(unlist(bundle))

unlink(output_dir, recursive = TRUE)
```

`contribution_issue_url` *Create a GitHub source-proposal issue URL for a contribution bundle*

Description

The returned URL opens GitHub's issue composer with bundle provenance prefilled. GitHub does not support file attachments through this URL, so the contributor must attach the generated bundle manually before submitting.

Usage

```
contribution_issue_url(bundle, repository = "lennon-li/OPCC")
```

Arguments

`bundle`	A bundle returned by `contribution_bundle()`.
`repository`	GitHub repository in `owner/repository` form.

Value

A GitHub issue-composer URL.

Examples

```
adapter <- new_source_adapter(
  source_id = "example_open_addresses",
  licence = "Open Government Licence - Ontario",
  lineage = "Municipal open address points, retrieved from a city portal",
  retrieval_date = as.Date("2026-01-15")
)
layer <- build_source_layer(
  data.frame(postal_code = c("M5V 3A8", "K1A 0A6")),
  adapter
)
output_dir <- tempfile("opcc-bundle")
bundle <- contribution_bundle(layer, output_dir = output_dir)

issue_url <- contribution_issue_url(bundle)
substr(issue_url, 1, 55)

unlink(output_dir, recursive = TRUE)
```

---

download_census_boundaries

Download Statistics Canada 2021 census boundary shapefiles

Description

Downloads the province/territory and Dissemination Block boundary shapefiles. Skips downloads if already cached.

Usage

```
download_census_boundaries(cache_dir = NULL)
```

Arguments

cache_dir	Required directory for downloaded and extracted files.

Value

Invisibly, a named list with `province` and `db` paths to the extracted `.shp` files.

Examples

```
# Downloads two Statistics Canada boundary zips (hundreds of megabytes),
# so it is not run automatically.

if (interactive()) {
  boundaries <- download_census_boundaries(cache_dir = tempfile("opcc"))
  boundaries$province
```

```
}
```

`download_da_boundaries` *Download Statistics Canada 2021 dissemination area boundary files*

Description

Downloads the Canada-wide dissemination area cartographic boundary shapefiles (~200 MB zip) and extracts the `.shp`. Skips the download if the zip is already cached. On first download a SHA-256 sidecar file is written next to the zip and every later reuse is verified against it; Statistics Canada does not publish an official checksum for this file.

Usage

```
download_da_boundaries(cache_dir = NULL)
```

Arguments

`cache_dir`	Required directory for downloaded and extracted files.

Value

Invisibly, a named list with `da` pointing to the extracted `lda_000b21a_e.shp`.

Examples

```
# Downloads a ~200 MB Statistics Canada boundary zip, so it is not run
# automatically.

if (interactive()) {
  boundaries <- download_da_boundaries(cache_dir = tempfile("opcc-build"))
  boundaries$da
}
```

download_gaf *Download the Statistics Canada 2021 Geographic Attribute File*

Description

Downloads the GAF zip and extracts the CSV. Skips if already cached.

Usage

```
download_gaf(cache_dir = NULL)
```

Arguments

cache_dir	Required directory for downloaded and extracted files.

Value

Invisibly, the path to the extracted GAF CSV.

Examples

```
# Downloads the Statistics Canada Geographic Attribute File zip, so it is
# not run automatically.

if (interactive()) {
  gaf_csv <- download_gaf(cache_dir = tempfile("opcc-build"))
}
```

download_geonames *Download GeoNames Canadian postal codes*

Description

Downloads the GeoNames CA_full.csv.zip and extracts the tab-delimited text file. Skips the download if already cached.

Usage

```
download_geonames(cache_dir = NULL)
```

Arguments

cache_dir	Required directory for downloaded and extracted files.

Value

Invisibly, the path to the extracted `CA_full.txt`.

Examples

```
# Downloads the GeoNames CA_full.csv.zip, so it needs network access and
# is not run automatically.

if (interactive()) {
  geonames_txt <- download_geonames(cache_dir = tempfile("opcc-build"))
}
```

`download_nar` *Download the Statistics Canada National Address Register*

Description

Downloads the NAR release zip (~1.6 GB) and extracts the Ontario Address and Location part files. Skips the download if the zip is already cached.

Usage

```
download_nar(cache_dir = NULL)
```

Arguments

`cache_dir`	Required directory for downloaded and extracted files.

Value

Invisibly, the path to the NAR scratch directory containing the extracted Ontario CSV files.

Examples

```
# Downloads the ~1.6 GB Statistics Canada NAR release, so it is not run
# automatically.  Pass an explicit cache directory to keep files local.

if (interactive()) {
  nar_dir <- download_nar(cache_dir = tempfile("opcc-build"))
}
```

`export_postal_points` *Write a postal-code point shapefile for an OPCC join*

Description

Builds a point layer for the supplied postal codes, one feature per unique code, carrying the dissemination-area roll-up from `joined` as attributes, and writes it to a zipped ESRI Shapefile. The points are the same source-qualified postal centroids the OPCC Shiny app draws on its map, so a script generated by the app reproduces exactly what the app displayed.

Usage

```
export_postal_points(
  postal_code,
  joined,
  zipfile,
  vintage = "2026-06-26",
  cache_dir = NULL,
  offline = FALSE
)
```

Arguments

`postal_code`	Character vector of postal codes.
`joined`	A data frame produced by joining records to an OPCC correspondence, carrying the `opcc_postal_code_col` and `opcc_dauid_col` attributes set by the app. Plain `opcc_postal_code` and `DAUID` columns are used when those attributes are absent.
`zipfile`	Path of the `.zip` archive to write.
`vintage`	Centroid artifact vintage.
`cache_dir`	Directory for the small, verified, actively managed runtime cache. Defaults to `NULL`, which uses a session temporary directory unless a persistent location has been configured or consented to; see `clear_opcc_cache()` for how the location is resolved.
`offline`	If `TRUE`, use only already-cached artifacts.

Details

Codes with no point evidence cannot appear in the output; that is a property of the evidence, not a failure. When no supplied code has any point evidence, nothing is written and the function returns `NULL`.

Value

The `zipfile` path, invisibly, or `NULL` when no points were available to write.

Examples

```
# Writing the shapefile downloads a versioned centroid artifact, so this
# runs only in an interactive session.

if (interactive() && requireNamespace("sf", quietly = TRUE)) {
  joined <- data.frame(opcc_postal_code = "M5S1A1", DAUID = "35200001")
  export_postal_points(
    "M5S1A1", joined, file.path(tempdir(), "opcc_postal_points.zip")
  )
}
```

---

geonames_supplementary_adapter *Load the versioned GeoNames supplementary-point adapter*

Description

Load the versioned GeoNames supplementary-point adapter

Usage

```
geonames_supplementary_adapter()
```

Value

An `opcc_source_adapter` for the packaged GeoNames point artifact.

Examples

```
adapter <- geonames_supplementary_adapter()
adapter$source_id
adapter$coordinate_method
adapter$licence
```

---

get_correspondence *Download, cache, and verify a correspondence release*

Description

Download, cache, and verify a correspondence release

Usage

```
get_correspondence(vintage = "2026-06-26", cache_dir = NULL, offline = FALSE)
```

Arguments

vintage	A value returned by list_vintages().
cache_dir	Directory for the small, verified, actively managed runtime cache. Defaults to NULL, which uses a session temporary directory unless a persistent location has been configured or consented to; see clear_opcc_cache() for how the location is resolved.
offline	Require an already cached verified file.

Value

A data frame of postal-code-to-DB links.

Examples

```
list_vintages("DB")

# Pass an explicit cache directory; never write to the default user cache
# from an example or a test.
cache <- tempfile("opcc-cache")


# Downloads and checksum-verifies a release artifact, so it needs network
# access and is not run automatically.
if (interactive()) {
  m2 <- get_correspondence("2026-06-26", cache_dir = cache)
  utils::head(m2[c("postal_code", "DBUID", "DAUID", "allocation_weight")])
}
```

---

get_da_correspondence *Download, cache, and verify a direct DA correspondence release*

Description

Download, cache, and verify a direct DA correspondence release

Usage

```
get_da_correspondence(
  vintage = "2026-06-26",
  cache_dir = NULL,
  offline = FALSE
)
```

Arguments

vintage	A value returned by list_vintages() for level = "DA".
cache_dir	Directory for the small, verified, actively managed runtime cache. Defaults to NULL, which uses a session temporary directory unless a persistent location has been configured or consented to; see clear_opcc_cache() for how the location is resolved.
offline	Require an already cached verified file.

Value

A data frame of postal-code-to-DA links with contributing DB lineage.

Examples

```
list_vintages("DA")

# Pass an explicit cache directory; never write to the default user cache
# from an example or a test.
cache <- tempfile("opcc-cache")


# Downloads and checksum-verifies a release artifact, so it needs network
# access and is not run automatically.
if (interactive()) {
  m5 <- get_da_correspondence("2026-06-26", cache_dir = cache)
  utils::head(m5[c("postal_code", "DAUID", "allocation_weight")])
}
```

---

list_vintages *List supported correspondence release vintages*

Description

List supported correspondence release vintages

Usage

```
list_vintages(level = c("DB", "DA"))
```

Arguments

level	Geography level, "DB" or "DA".

Value

A character vector of release vintages.

Examples

```
list_vintages("DB")
list_vintages("DA")
```

`new_source_adapter` *Define a source adapter for a local evidence layer*

Description

Define a source adapter for a local evidence layer

Usage

```
new_source_adapter(
  source_id,
  licence,
  lineage,
  retrieval_date = Sys.Date(),
  schema_map = list(postal_code = "postal_code"),
  endpoint = NULL,
  checksum = NULL,
  location_type = "unknown",
  coordinate_method = "unknown",
  authority_level = "unknown",
  coverage_type = "unknown",
  update_frequency = "unknown"
)
```

Arguments

`source_id`	Stable, lower-case source identifier.
`licence`	Licence or permission statement for the source.
`lineage`	Source lineage and collection method.
`retrieval_date`	Source retrieval or creation date.
`schema_map`	Named mapping from OPCC fields to source fields.
`endpoint`	Optional public retrieval endpoint.
`checksum`	Optional SHA-256 checksum of the source artifact.
`location_type`	Source location type: `physical`, `mailing`, or `unknown`.
`coordinate_method`	Coordinate derivation method: `address_point`, `entrance`, `building`, `parcel`, `centroid`, or `unknown`.
`authority_level`	Source authority classification.
`coverage_type`	Source coverage classification.
`update_frequency`	Expected source update frequency.

Value

An opcc_source_adapter object.

Examples

```
adapter <- new_source_adapter(
  source_id = "example_open_addresses",
  licence = "Open Government Licence - Ontario",
  lineage = "Municipal open address points, retrieved from a city portal",
  retrieval_date = as.Date("2026-01-15"),
  schema_map = list(postal_code = "pc", latitude = "lat", longitude = "lon"),
  location_type = "physical",
  coordinate_method = "address_point"
)
adapter$source_id
adapter$schema_map
```

---

normalize_postal_code *Normalize Canadian postal codes*

Description

Normalize Canadian postal codes

Usage

```
normalize_postal_code(x, strict = FALSE)
```

Arguments

x	A character vector of postal codes.
strict	If TRUE, reject any non-missing invalid value instead of returning NA for it.

Value

A character vector in A1A 1A1 form.

Examples

```
normalize_postal_code(c("m5v3a8", "M5V 3A8", "M5V-3A8"))

# Invalid values become NA unless strict = TRUE, which raises an error.
normalize_postal_code(c("M5V 3A8", "not a postal code"))
try(normalize_postal_code("not a postal code", strict = TRUE))
```

OPCC *Open Postal Code Correspondence*

Description

Checksum-verified, source-qualified correspondence releases and postal-code lookup helpers for linking Ontario postal codes to census dissemination blocks and dissemination areas.

Details

OPCC does not substitute GeoNames point evidence for NAR address evidence. Local source layers are always source-separated and never modify a canonical OPCC release. Canada Post, PCCF, and PCCF+ data are rejected. Each local-data function invites users to contribute redistributable evidence through a reviewable bundle.

`open_contribution_issue` *Open a GitHub source-proposal issue for a contribution bundle*

Description

Open a GitHub source-proposal issue for a contribution bundle

Usage

```
open_contribution_issue(bundle, repository = "lennon-li/OPCC", open = FALSE)
```

Arguments

`bundle`	A bundle returned by `contribution_bundle()`.
`repository`	GitHub repository in `owner/repository` form.
`open`	Whether to open the returned URL in a browser. Defaults to `FALSE` so submission remains an explicit user action.

Value

Invisibly, the GitHub issue-composer URL.

Examples

```
adapter <- new_source_adapter(
  source_id = "example_open_addresses",
  licence = "Open Government Licence - Ontario",
  lineage = "Municipal open address points, retrieved from a city portal",
  retrieval_date = as.Date("2026-01-15")
)
layer <- build_source_layer(
  data.frame(postal_code = c("M5V 3A8", "K1A 0A6")),
  adapter
)
output_dir <- tempfile("opcc-bundle")
bundle <- contribution_bundle(layer, output_dir = output_dir)

# open = FALSE keeps submission an explicit user action, so nothing is
# sent and no browser is opened here.
issue_url <- suppressMessages(open_contribution_issue(bundle))
substr(issue_url, 1, 55)

unlink(output_dir, recursive = TRUE)
```

---

pc_to_geo *Look up postal-code-to-geography links*

Description

All DB links are returned by default. Set `all_links = FALSE` only when a single best link is specifically needed.

Usage

```
pc_to_geo(
  postal_code,
  level = c("DB", "DA"),
  all_links = TRUE,
  correspondence = NULL,
  ...
)
```

Arguments

`postal_code`	Character vector of Canadian postal codes.
`level`	Geography level, `"DB"` or `"DA"`.
`all_links`	Whether to retain every allocated DB link.
`correspondence`	Optional already-loaded correspondence data.
`...`	Passed to `get_correspondence()` when `correspondence` is NULL.

Value

A data frame; unmatched normalized postal codes are stored in its `unmatched` attribute.

Examples

```
# Supplying `correspondence` keeps the lookup fully offline.
da_links <- data.frame(
  postal_code = c("M5V 3A8", "M5V 3A8"),
  DAUID = c("35200001", "35200002"),
  allocation_weight = c(0.75, 0.25),
  n_contributing_dbs = c(2L, 1L),
  contributing_dbuids = c("35200001000|35200001001", "35200002000"),
  source_vintages = "2026-06-26",
  census_vintages = "2021",
  evidence_classes = "NAR",
  best_link = c(TRUE, FALSE),
  stringsAsFactors = FALSE
)
pc_to_geo("M5V 3A8", level = "DA", correspondence = da_links)

# A single best link, when one is specifically needed.
pc_to_geo("m5v3a8", level = "DA", correspondence = da_links,
          all_links = FALSE)

# Unmatched postal codes stay explicit rather than being dropped silently.
found <- pc_to_geo(c("M5V 3A8", "K1A 0A6"), level = "DA",
                   correspondence = da_links)
attr(found, "unmatched")
```

`pc_to_point` *Look up source-qualified point observations*

Description

Look up source-qualified point observations

Usage

```
pc_to_point(
  postal_code,
  vintage = "2026-07-19",
  point_file = NULL,
  cache_dir = NULL,
  offline = FALSE,
  source = NULL
)
```

Arguments

postal_code	Character vector of Canadian postal codes.
vintage	Point-release vintage.
point_file	Optional local gzip CSV file. Supplying this enables fully offline and air-gapped use.
cache_dir	Directory for the small, verified, actively managed runtime cache. Defaults to NULL, which uses a session temporary directory unless a persistent location has been configured or consented to; see clear_opcc_cache() for how the location is resolved.
offline	Require an already cached verified file.
source	Optional character vector of point_source values to retain. By default, observations from every source are returned.

Value

All matching source-qualified point observations, including DB/DA fields when a point intersects a 2021 Ontario dissemination block.

Examples

```
# `point_file` accepts any local gzip CSV with the required columns, which
# makes point lookups fully offline and air-gapped.
points <- data.frame(
  postal_code = c("M5V 3A8", "M5V 3A8"),
  latitude = c(43.6426, 43.6430),
  longitude = c(-79.3871, -79.3875),
  point_source = c("nar", "geonames"),
  point_method = c("address_point", "centroid"),
  stringsAsFactors = FALSE
)
point_file <- tempfile("opcc-points", fileext = ".csv.gz")
connection <- gzfile(point_file, "w")
utils::write.csv(points, connection, row.names = FALSE)
close(connection)

pc_to_point("M5V 3A8", point_file = point_file)

# Restrict the evidence to one source.
pc_to_point("M5V 3A8", point_file = point_file, source = "nar")

unlink(point_file)
```

profile_source_layer *Profile a local source layer*

Description

Profile a local source layer

Usage

```
profile_source_layer(layer)
```

Arguments

layer	A layer created by build_source_layer().

Value

A list of coverage and coordinate-quality metrics.

Examples

```
adapter <- new_source_adapter(
  source_id = "example_open_addresses",
  licence = "Open Government Licence - Ontario",
  lineage = "Municipal open address points, retrieved from a city portal",
  retrieval_date = as.Date("2026-01-15"),
  schema_map = list(postal_code = "pc", latitude = "lat", longitude = "lon")
)
raw <- data.frame(
  pc = c("M5V 3A8", "K1A 0A6"),
  lat = c(43.6426, 45.4215),
  lon = c(-79.3871, -75.6972),
  stringsAsFactors = FALSE
)
layer <- build_source_layer(raw, adapter)
profile_source_layer(layer)
```

release_manifest *Read and verify a release manifest*

Description

Read and verify a release manifest

Usage

```
release_manifest(
  vintage = "2026-06-26",
  cache_dir = NULL,
  offline = FALSE,
  level = c("DB", "DA")
)
```

Arguments

vintage	A value returned by `list_vintages()`.
cache_dir	Directory for the small, verified, actively managed runtime cache. Defaults to `NULL`, which uses a session temporary directory unless a persistent location has been configured or consented to; see `clear_opcc_cache()` for how the location is resolved.
offline	Require an already cached verified file.
level	Geography level, `"DB"` or `"DA"`.

Value

A parsed JSON list.

Examples

```
list_vintages("DB")
cache <- tempfile("opcc-cache")


# Downloads and checksum-verifies the release manifest, so it needs
# network access and is not run automatically.
if (interactive()) {
  manifest <- release_manifest("2026-06-26", cache_dir = cache)
  names(manifest)
}
```

---

run_app *Launch the OPCC Shiny app*

Description

Launches a Shiny app that joins a user-uploaded CSV to the OPCC postal-code-to-DA correspondence: pick the postal-code column (an auto-detected candidate is preselected), click Join to see the result table, and draw the matched dissemination areas on a map. The joined CSV, the map as HTML, and an R script reproducing both artifacts are downloadable.

Usage

```
run_app(...)
```

Arguments

`...`	Passed to `shiny::runApp()`.

Details

Reusing the simplified map between sessions requires a writable cache. The app never writes one without permission: set `OPCC.shiny_da_cache_dir` (or `OPCC_SHINY_DA_CACHE_DIR`) to choose the location yourself, or answer the one-time prompt shown in an interactive session. The answer is remembered, so a recorded "yes" also applies to later non-interactive launches, and a recorded "no" is honored until you delete the record. Without permission the map is rebuilt in a session-only temporary directory, which is correct but takes several minutes per session.

Value

Called for its side effect (launches the Shiny app). Invisibly returns `NULL`.

Examples

```
if (interactive()) {
  run_app()
}
```

---

validate_release *Validate a verified correspondence release*

Description

Validate a verified correspondence release

Usage

```
validate_release(
  vintage = "2026-06-26",
  cache_dir = NULL,
  offline = FALSE,
  level = c("DB", "DA")
)
```

Arguments

vintage	A value returned by list_vintages().
cache_dir	Directory for the small, verified, actively managed runtime cache. Defaults to NULL, which uses a session temporary directory unless a persistent location has been configured or consented to; see clear_opcc_cache() for how the location is resolved.
offline	Require an already cached verified file.
level	Geography level, "DB" or "DA".

Value

Invisibly TRUE, or an error describing a failed invariant.

Examples

```
list_vintages("DB")
cache <- tempfile("opcc-cache")


# Downloads the release and its manifest, so it needs network access and
# is not run automatically.
if (interactive()) {
  validate_release("2026-06-26", cache_dir = cache)
}
```

---

validate_source_data *Validate local postal-code evidence*

Description

Validate local postal-code evidence

Usage

```
validate_source_data(
  data,
  adapter,
  on_invalid = c("error", "drop", "quarantine")
)
```

Arguments

data	A data frame with a postal-code field named by adapter.
adapter	Source metadata created by new_source_adapter().
on_invalid	How to handle invalid rows: error, drop them, or retain them in the opcc_quarantine attribute.

Value

A normalized data frame with `postal_code` and validation metadata. Coordinate-bearing rows outside the inclusive broad Ontario bounds (latitude 41.6 to 56.9, longitude -95.2 to -74.3) are invalid and counted in `outside_ontario_bounds_rows`.

Examples

```
adapter <- new_source_adapter(
  source_id = "example_open_addresses",
  licence = "Open Government Licence - Ontario",
  lineage = "Municipal open address points, retrieved from a city portal",
  retrieval_date = as.Date("2026-01-15"),
  schema_map = list(postal_code = "pc", latitude = "lat", longitude = "lon")
)
raw <- data.frame(
  pc = c("m5v3a8", "K1A 0A6", "not a postal code"),
  lat = c(43.6426, 45.4215, NA),
  lon = c(-79.3871, -75.6972, NA),
  stringsAsFactors = FALSE
)
clean <- validate_source_data(raw, adapter, on_invalid = "quarantine")
clean$postal_code
attr(clean, "opcc_validation_report")[c("accepted_rows", "rejected_rows")]
attr(clean, "opcc_quarantine")$.opcc_validation_reason
```

Index

aggregate_da_correspondence, 3

build_centroids, 3
build_centroids(), *5*
build_db_assignment, 4
build_db_assignment(), *5, 6*
build_m2, 5
build_source_layer, 6
build_source_layer(), *8*, *24*

clear_opcc_cache, 7
clear_opcc_cache(), *14*, *16, 17*, *23*, *25*, *27*
contribution_bundle, 8
contribution_bundle(), *9*, *20*
contribution_issue_url, 9

download_census_boundaries, 10
download_census_boundaries(), *5, 6*
download_da_boundaries, 11
download_gaf, 12
download_gaf(), *5, 6*
download_geonames, 12
download_geonames(), *4*
download_nar, 13
download_nar(), *4*, *6*

export_postal_points, 14

geonames_supplementary_adapter, 15
get_correspondence, 15
get_correspondence(), *21*
get_da_correspondence, 16

list_vintages, 17
list_vintages(), *16, 17*, *25*, *27*

new_source_adapter, 18
new_source_adapter(), *7*, *27*
normalize_postal_code, 19

OPCC, 20
OPCC-package *(*OPCC*)*, 20
open_contribution_issue, 20

pc_to_geo, 21
pc_to_point, 22
profile_source_layer, 24

release_manifest, 24
run_app, 25

shiny::runApp(), *26*

tempdir(), *8*
tools::R_user_dir(), *8*

validate_release, 26
validate_source_data, 27