Package 'closecity'

August 2, 2026

Title Client for the 'Close' API

Version 1.5.0

Description Access travel times from every US census block to nearby points of interest, by walking, biking, and public transit, over the 'Close' API. Wraps the metered, read-only public endpoints with keyset pagination, ETag/304 conditional requests, token accounting, typed RFC 9457 error handling, and tabular output by default ('sf' where geometry applies, a data frame otherwise).

License MIT + file LICENSE

Encoding UTF-8

URL https://henryspatialanalysis.github.io/closecity-r/, https://github.com/henryspatialanalysis/closecity-r

BugReports https://github.com/henryspatialanalysis/closecity-r/issues

Depends R (>= 4.1)

Imports httr2 (>= 1.0.0), jsonlite, R6, rlang, sf

Suggests testthat (>= 3.0.0), withr, tigris, plotly, geojsonsf, knitr, rmarkdown

Config/testthat/edition 3

Config/Needs/website pkgdown

RoxygenNote 8.0.0

VignetteBuilder knitr

NeedsCompilation no

Author Nathaniel Henry [aut, cre], Henry Spatial Analysis [aut, cph, fnd]

Maintainer Nathaniel Henry <nat@henryspatialanalysis.com>

Repository CRAN

Date/Publication 2026-08-02 16:30:07 UTC

Contents

`CloseClient` *Close API client*

Description

An R6 object that holds your connection settings and gives you one method per public route. Create it with `close_client()` rather than calling `$new()`. Results come back per the `output` field: an sf object where geometry applies, a data frame otherwise, or a close_reply when `output` is `'raw'`.

Public fields

`output` (`character(1)`)
: How results come back: `'spatial'`, `'tabular'`, or `'raw'`. Change it any time, or override it per call with the method's `output` argument.

Methods

Public methods:

- `CloseClient$new()`
- `CloseClient$health()`
- `CloseClient$last_updated()`
- `CloseClient$modes()`
- `CloseClient$destination_types()`
- `CloseClient$vintage()`
- `CloseClient$places()`
- `CloseClient$block_summary()`
- `CloseClient$block_pois()`
- `CloseClient$point_summary()`
- `CloseClient$point_pois()`
- `CloseClient$pois_search()`
- `CloseClient$poi()`
- `CloseClient$poi_catchment()`
- `CloseClient$blocks_query()`
- `CloseClient$place_blocks()`
- `CloseClient$place_pois()`

- CloseClient$place_boundary()
- CloseClient$isochrone()
- CloseClient$isochrone_meta()
- CloseClient$records()
- CloseClient$clone()

`CloseClient$new()`: Create a client. Prefer `close_client()`.

Usage:

```
CloseClient$new(
  api_key = NULL,
  base_url = DEFAULT_BASE_URL,
  timeout = 30,
  output = "spatial"
)
```

Arguments:

`api_key` Your API key, or NULL for the free routes. When NULL, the `CLOSECITY_KEY` environment variable is used if set.

`base_url` API base URL.

`timeout` Request timeout, in seconds.

`output` One of `'spatial'`, `'tabular'`, or `'raw'`.

`CloseClient$health()`: Liveness check (free). Always a raw close_reply.

Usage:

```
CloseClient$health()
```

Returns: A close_reply.

`CloseClient$last_updated()`: Publication time of the newest data (free). Always a raw reply.

Usage:

```
CloseClient$last_updated()
```

Returns: A close_reply.

`CloseClient$modes()`: Travel modes and their numeric ids (free).

Usage:

```
CloseClient$modes(output = NULL)
```

Arguments:

`output` Override the client's output mode for this call.

Returns: A data frame, or a close_reply when `output` is `'raw'`.

`CloseClient$destination_types()`: Destination-type taxonomy (free). Use it to look up the numeric `type` ids the data routes filter on; a parent type expands to its `leaf_ids`.

Usage:

```
CloseClient$destination_types(output = NULL)
```

Arguments:

`output` Override the client's output mode for this call.

Returns: A data frame, or a close_reply when `output` is `'raw'`.

`CloseClient$vintage()`: Active version of each dataset component (free).

Usage:

```
CloseClient$vintage(output = NULL)
```

Arguments:

`output` Override the client's output mode for this call.

Returns: A data frame, or a close_reply when `output` is `'raw'`.

`CloseClient$places()`: Look up a city or town by name (free). Each match carries its census place GEOID and centre point.

Usage:

```
CloseClient$places(q, limit = NULL, output = NULL)
```

Arguments:

`q` Name to search for, such as "Providence".

`limit` Most matches to return (1 to 20).

`output` Override the client's output mode for this call.

Returns: An sf of points (a data frame in tabular mode, a close_reply when `output` is `'raw'`).

`CloseClient$block_summary()`: Fastest travel time from a census block to each destination category, by mode. Pass a vector of GEOIDs to query many blocks in one call (one request, one rate-limit tick).

Usage:

```
CloseClient$block_summary(
  geoid,
  mode = NULL,
  type = NULL,
  if_none_match = NULL,
  output = NULL
)
```

Arguments:

`geoid` A 15-digit census block GEOID, or a vector of them for a single multi-origin call. Every result row carries its origin `geoid`; per-origin `errors` / `truncated` / `truncated_reason` ride on the frame's attributes.

`mode` Travel mode(s) to keep: "walk", "bike", "transit".

`type` Destination type id(s) to keep.

`if_none_match` An ETag from an earlier reply, to revalidate for free (single-block form only).

`output` Override the client's output mode for this call.

Returns: A data frame with a broadcast `geoid` column, or a close_reply when `output` is `'raw'`.

`CloseClient$block_pois()`: Nearby points of interest and their travel time from a block, one row per (POI, mode). Reads every page by default. Pass a vector of GEOIDs to query many blocks in one call (one request, one rate-limit tick; the multi-origin form is not paginated).

Usage:

```
CloseClient$block_pois(
  geoid,
  mode = NULL,
  type = NULL,
  dest_id = NULL,
  max_minutes = NULL,
  limit = NULL,
  cursor = NULL,
  paginate = TRUE,
  output = NULL
)
```

Arguments:

`geoid` A 15-digit census block GEOID, or a vector of them for a single multi-origin call. Every row carries its origin `geoid`; per-origin `errors` / `truncated` ride on the frame's attributes.

`mode` Travel mode(s) to keep.

`type` Destination type id(s) to keep.

`dest_id` Specific destination id(s) to keep.

`max_minutes` Upper bound on travel time (up to 30).

`limit` Rows per page (up to 1000; single-block form only).

`cursor` Page cursor from a previous reply's `next_cursor`; supplying one fetches only that page.

`paginate` Follow `next_cursor` and return every page (the default); set `FALSE` for the first page only.

`output` Override the client's output mode for this call.

Returns: An sf of points (a data frame in tabular mode, a close_reply when `output` is `'raw'`).

`CloseClient$point_summary()`: Like `$block_summary()`, but from the block containing a lat/lon point. The resolved block is echoed as `resolved_block` and broadcast to a `geoid` column. Pass equal-length `lat` / `lon` vectors to query many points in one call (one request, one rate-limit tick).

Usage:

```
CloseClient$point_summary(
  lat,
  lon,
  mode = NULL,
  type = NULL,
  if_none_match = NULL,
  output = NULL
)
```

Arguments:

`lat` Latitude, or a vector of latitudes for a multi-origin call.

`lon` Longitude, or a matching vector of longitudes. In the multi-origin form each row carries its `origin_lat` / `origin_lon`, and resolved blocks / `errors` / `truncated` ride on the frame's attributes.

`mode` Travel mode(s) to keep.

`type` Destination type id(s) to keep.

`if_none_match` An ETag to revalidate for free (single-point form only).

`output` Override the client's output mode for this call.

Returns: A data frame, or a close_reply when `output` is `'raw'`.

`CloseClient$point_pois()`: Like `$block_pois()`, but from the block containing a lat/lon point. Reads every page by default. Pass equal-length `lat` / `lon` vectors to query many points in one call (one request, one rate-limit tick; the multi-origin form is not paginated).

Usage:

```
CloseClient$point_pois(
  lat,
  lon,
  mode = NULL,
  type = NULL,
  dest_id = NULL,
  max_minutes = NULL,
  limit = NULL,
  cursor = NULL,
  paginate = TRUE,
  output = NULL
)
```

Arguments:

`lat` Latitude, or a vector of latitudes for a multi-origin call.

`lon` Longitude, or a matching vector of longitudes. In the multi-origin form each row carries its `origin_lat` / `origin_lon`.

`mode` Travel mode(s) to keep.

`type` Destination type id(s) to keep.

`dest_id` Specific destination id(s) to keep.

`max_minutes` Upper bound on travel time (up to 30).

`limit` Rows per page (up to 1000; single-point form only).

`cursor` Page cursor; supplying one fetches only that page.

`paginate` Follow `next_cursor` and return every page (the default); set `FALSE` for the first page only.

`output` Override the client's output mode for this call.

Returns: An sf of points (a data frame in tabular mode, a close_reply when `output` is `'raw'`).

`CloseClient$pois_search()`: Search points of interest by bounding box, or by a circle (`lat` + `lon` + `radius_m`). Reads every page by default.

Usage:

```
CloseClient$pois_search(
  lat = NULL,
  lon = NULL,
  radius_m = NULL,
  bbox = NULL,
```

```
  type = NULL,
  q = NULL,
  limit = NULL,
  cursor = NULL,
  paginate = TRUE,
  output = NULL
)
```

Arguments:

`lat, lon` Circle centre.

`radius_m` Circle radius, in metres (up to 50000).

`bbox` Bounding box, "min_lon,min_lat,max_lon,max_lat".

`type` Destination type id(s) to keep.

`q` Name text to match.

`limit` Rows per page (up to 1000).

`cursor` Page cursor; supplying one fetches only that page.

`paginate` Follow `next_cursor` and return every page (the default); set `FALSE` for the first page only.

`output` Override the client's output mode for this call.

Returns: An sf of points (a data frame in tabular mode, a close_reply when `output` is `'raw'`).

`CloseClient$poi()`**:** Details for one point of interest.

Usage:

```
CloseClient$poi(dest_id, if_none_match = NULL, output = NULL)
```

Arguments:

`dest_id` Destination id.

`if_none_match` An ETag to revalidate for free.

`output` Override the client's output mode for this call.

Returns: An sf of one point (a data frame in tabular mode, a close_reply when `output` is `'raw'`).

`CloseClient$poi_catchment()`**:** Every census block that can reach a point of interest, one row per (block, mode). Reads every page by default. Pass a vector of dest_ids to query many POIs in one call (one request, one rate-limit tick; the multi-origin form is not paginated).

Usage:

```
CloseClient$poi_catchment(
  dest_id,
  mode = NULL,
  block = NULL,
  max_minutes = NULL,
  limit = NULL,
  cursor = NULL,
  paginate = TRUE,
  output = NULL
)
```

Arguments:

`dest_id` A destination id, or a vector of them for a single multi-origin call. Every row carries its origin `dest_id`; per-POI `errors` / `truncated` ride on the frame's attributes.

`mode` Travel mode(s) to keep.

`block` Specific block id(s) to keep.

`max_minutes` Upper bound on travel time (up to 30).

`limit` Rows per page (up to 1000; single-POI form only).

`cursor` Page cursor; supplying one fetches only that page.

`paginate` Follow `next_cursor` and return every page (the default); set `FALSE` for the first page only.

`output` Override the client's output mode for this call.

Returns: An sf of block polygons (a data frame in tabular mode, a close_reply when `output` is `'raw'`).

`CloseClient$blocks_query()`: Blocks inside a GeoJSON polygon, or a circle (`center` + `radius_m`), one row per (block, category, mode). Rows carry the numeric `mode_id` (join `$modes()` to label it). Reads every page by default.

Usage:

```
CloseClient$blocks_query(
  polygon = NULL,
  center = NULL,
  radius_m = NULL,
  type = NULL,
  mode = NULL,
  include_population = NULL,
  limit = NULL,
  cursor = NULL,
  paginate = TRUE,
  output = NULL
)
```

Arguments:

`polygon` A GeoJSON polygon or multipolygon (a list).

`center` A circle centre, `list(lon =, lat =)`.

`radius_m` Circle radius, in metres (up to 28000).

`type` Destination type id(s) to keep.

`mode` Travel mode(s) to keep.

`include_population` Add each block's population to its rows.

`limit` Rows per page (up to 1000).

`cursor` Page cursor; supplying one fetches only that page.

`paginate` Follow `next_cursor` and return every page (the default); set `FALSE` for the first page only.

`output` Override the client's output mode for this call.

Returns: An sf of block polygons (a data frame in tabular mode, a close_reply when `output` is `'raw'`).

`CloseClient$place_blocks()`: Per-block travel times for every block in a place (a city or town), by place GEOID. Rows carry the numeric `mode_id` (join `$modes()` to label it). Reads every page by default.

Usage:

```
CloseClient$place_blocks(
  geoid,
  mode = NULL,
  type = NULL,
  include_population = NULL,
  limit = NULL,
  cursor = NULL,
  paginate = TRUE,
  output = NULL
)
```

Arguments:

`geoid` Census place GEOID.

`mode` Travel mode(s) to keep.

`type` Destination type id(s) to keep.

`include_population` Add each block's population to its rows.

`limit` Rows per page (up to 1000).

`cursor` Page cursor; supplying one fetches only that page.

`paginate` Follow `next_cursor` and return every page (the default); set `FALSE` for the first page only.

`output` Override the client's output mode for this call.

Returns: An sf of block polygons (a data frame in tabular mode, a close_reply when `output` is `'raw'`).

`CloseClient$place_pois()`: Every point of interest within a census place (a city or town), by place GEOID. The place analog of `$pois_search()`; pass `type` to get, e.g., all supermarkets in a city. Spatial only, no travel times. Reads every page by default.

Usage:

```
CloseClient$place_pois(
  geoid,
  type = NULL,
  q = NULL,
  limit = NULL,
  cursor = NULL,
  paginate = TRUE,
  output = NULL
)
```

Arguments:

`geoid` Census place GEOID.

`type` Destination type id(s) to keep.

`q` Name substring to match.

`limit` Rows per page (up to 1000).

`cursor` Page cursor; supplying one fetches only that page.

`paginate` Follow `next_cursor` and return every page (the default); set `FALSE` for the first page only.

`output` Override the client's output mode for this call.

Returns: An sf of points (a data frame in tabular mode, a close_reply when `output` is `'raw'`).

`CloseClient$place_boundary()`: The boundary polygon of a census place, as a one-row sf, handy as a `boundary` layer for `close_map()` when mapping a city's blocks or POIs. Free (no API key).

Usage:

```
CloseClient$place_boundary(geoid, output = NULL)
```

Arguments:

`geoid` Census place GEOID.

`output` Override the client's output mode for this call.

Returns: A one-row sf polygon (a close_reply when `output` is `'raw'`).

`CloseClient$isochrone()`: Travel-time contours from a block or a lat/lon point. Give `minutes` for one threshold, or `contours` for up to four.

Usage:

```
CloseClient$isochrone(
  block = NULL,
  lon = NULL,
  lat = NULL,
  mode = NULL,
  direction = NULL,
  minutes = NULL,
  contours = NULL,
  format = NULL,
  v = NULL,
  if_none_match = NULL,
  output = NULL
)
```

Arguments:

`block` Origin block GEOID (or give `lon` + `lat`).

`lon, lat` Origin point, instead of `block`.

`mode` "walk", "bike", or "transit".

`direction` "to" (blocks that can reach the origin) or "from".

`minutes` A single threshold (1 to 60).

`contours` Up to four ascending levels, instead of `minutes`.

`format` "geojson" (polygons) or "blocks" (a block list).

`v` Optional cache-buster, echoed back.

`if_none_match` An ETag to revalidate for free.

`output` Override the client's output mode for this call.

Returns: An sf (contour polygons for geojson, block polygons for blocks), a data frame in tabular mode, or a close_reply when `output` is `'raw'`.

`CloseClient$isochrone_meta()`: Isochrone version, directions, modes, and assumptions (free). Always a raw close_reply.

Usage:

```
CloseClient$isochrone_meta(if_none_match = NULL)
```

Arguments:

`if_none_match` An ETag to revalidate for free.

Returns: A close_reply.

`CloseClient$records()`: Read every record from a paginated method, following the cursor to the last page. The paginated methods now read every page by default, so this is rarely needed; it remains for explicit control and back-compat.

Usage:

```
CloseClient$records(endpoint, ..., output = NULL)
```

Arguments:

`endpoint` Name of a paginated method, such as "pois_search".

`...` Arguments passed on to that method.

`output` Override the client's output mode for this call.

Returns: A data frame (an sf in spatial mode), or a list of records when `output` is `'raw'`.

Examples:

```
close$records("pois_search", lat = 41.82, lon = -71.41, radius_m = 1500)
```

`CloseClient$clone()`: The objects of this class are cloneable with this method.

Usage:

```
CloseClient$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `CloseClient$records()`
## ------------------------------------------------

## Not run:
close$records("pois_search", lat = 41.82, lon = -71.41, radius_m = 1500)

## End(Not run)
```

close_as_df *Convert a Close reply to a data.frame*

Description

Turns a close_reply into a plain data.frame, one row per record. This is what the client returns in the "tabular" output mode; close_as_sf() adds geometry on top of the same rows. Metering and envelope metadata (token counts, block_geoid, assumptions, ...) are attached as attributes.

Usage

```
close_as_df(x, key = NULL)
```

Arguments

x	(close_reply) A reply, or the same list shape.
key	(character(1), default NULL) Name of the record array to read. Detected from the body when omitted.

Value

A data.frame. Summary replies gain a broadcast geoid column.

Examples

```
## Not run:
close <- close_client(output = 'raw')
close_as_df(close$modes())

## End(Not run)
```

close_as_sf *Convert a Close reply to an sf object*

Description

Detects the geometry from the payload. POI replies (from $pois_search(), $block_pois(), $point_pois(), $poi(), $places()) become points from their lat/lon. An isochrone reply with format = "geojson" becomes polygons. Block replies ($blocks_query(), $place_blocks(), $poi_catchment(), isochrone format = "blocks") carry only GEOIDs, so the block boundaries are joined from block_geometry, or downloaded with tigris when fetch = TRUE.

Usage

```
close_as_sf(
  x,
  block_geometry = NULL,
  geoid_col = "GEOID20",
  crs = 4326,
  fetch = FALSE
)
```

Arguments

x	(close_reply) A reply, or the same list shape.
block_geometry	(sf, default NULL) Block boundaries with a geoid_col column, joined to block replies on the 15-digit GEOID.
geoid_col	(character(1), default 'GEOID20') Name of the GEOID column in block_geometry (TIGER 2020 blocks).
crs	(default 4326) Coordinate reference system for point and polygon geometry.
fetch	(logical(1), default FALSE) If TRUE and block_geometry is NULL, download the needed TIGER blocks with tigris (inferring state and county from the GEOIDs).

Value

An sf data frame. Metering and envelope metadata are attached as attributes.

Examples

```
## Not run:
close <- close_client(output = 'raw')
close_as_sf(close$pois_search(lat = 41.82, lon = -71.41, radius_m = 1500))

## End(Not run)
```

---

close_client *Create a Close API client*

Description

Builds a CloseClient. The catalog and health routes are free, so a key is optional. Every data route needs one (a ck_live_ key), created at https://account.close.city (5,000 free tokens on signup, no card).

Usage

```
close_client(
  api_key = NULL,
  base_url = DEFAULT_BASE_URL,
  timeout = 30,
  output = "spatial"
)
```

Arguments

api_key	(character(1), default NULL) Your API key, or NULL for the free routes. When NULL, the CLOSECITY_KEY environment variable is used if set.
base_url	(character(1)) API base URL.
timeout	(numeric(1), default 30) Request timeout, in seconds.
output	(character(1), default 'spatial') How results come back: 'spatial' returns an sf object where geometry applies and a data frame otherwise; 'tabular' returns a data frame for every route and never downloads block boundaries; 'raw' returns the close_reply.

Value

A CloseClient. Make calls through its methods.

Examples

```
## Not run:
close <- close_client('ck_live_your_key')   # use your own key here
close$block_summary('440070008001068', mode = 'walk')

## End(Not run)
```

---

close_map *Interactive map of Close spatial results*

Description

Draw the sf object a client method returns as an interactive map on a CARTO Positron basemap. Points (POIs, places) render as bright hoverable markers; polygons (census blocks) are filled, optionally greying the features that do not meet a criterion so the ones that matter stand out. The view auto-zooms to fit every layer with a margin, and hover shows all attributes.

Usage

```
close_map(
  x,
  color = "#e8590c",
  highlight = NULL,
  fill = NULL,
  palette = "YlGnBu",
  reverse = FALSE,
  label = NULL,
  size = 15,
  opacity = 0.65,
  boundary = NULL,
  background = NULL,
  background_color = "#3b6fb0",
  background_opacity = 0.3,
  background_fill = TRUE,
  points = NULL,
  points_color = "#e8590c",
  mark = NULL,
  buffer = 0.15,
  zoom = NULL
)
```

Arguments

x	An sf from a client method: points or polygons.
color	Marker/fill colour for flat features (or for highlighted ones).
highlight	Optional. A logical vector (length nrow(x)) or the name of a logical/0-1 column in x. When supplied, features that do not meet it render grey (#888) and the rest use color, so you can show every block in a study area and pick out the matches, rather than dropping the others.
fill	Optional. The name of a numeric column to shade features by, on a continuous ColorBrewer scale with a legend (e.g. travel time, or an access score). Use this OR highlight, not both.
palette	A plotly ColorBrewer colorscale name for fill (default "YlGnBu").
reverse	Which end of the YlGnBu scale is blue. FALSE (default) puts blue at the low values, TRUE at the high values. Choose so blue marks the most-accessible end: FALSE for travel time (low is best), TRUE for a score.
label	Optional. A column shown first (bold) in the hover; the rest of the attributes follow.
size	Marker size, for point maps.
opacity	Fill opacity, for polygon maps.
boundary	Optional. A polygon sf drawn as a grey outline underneath the data, e.g. a city boundary from place_boundary().
background	Optional. A polygon sf, or a list of them, drawn as semi-transparent fills underneath the data, e.g. commute isochrones, or a walkshed under its POIs.

background_color
: Fill colour(s) for background, recycled across the layers.

background_opacity
: Fill opacity for background layers.

background_fill
: Fill the background layers (default TRUE); FALSE draws them as outlines only.

points
: Optional. A point sf drawn as markers on top of the main layer, e.g. POI locations over a block map.

points_color
: Marker colour for points.

mark
: Optional. A point to mark on top with an X, either a c(lon, lat) pair or a point sf/sfc (e.g. a starting point).

buffer
: Fraction of the data extent to pad the view by (default 0.15).

zoom
: Deprecated/ignored; the view auto-zooms to the data.

Value

A plotly map object.

Examples

```
## Not run:
close <- close_client()
close_map(close$place_pois(geoid = "4459000", type = 30), color = "#e8590c")

## End(Not run)
```

---

close_reply *The reply object*

Description

When output is 'raw', every method returns a close_reply: a list with the parsed body plus the metering and caching information as named fields.

Fields

data The parsed body (a list), or NULL for a 304.

results The results array for list routes (else an empty list).

next_cursor The cursor for the next page, or NULL.

tokens_charged, tokens_remaining Token counts; NULL on free routes.

etag The response ETag, to pass back as if_none_match.

status HTTP status code.

not_modified TRUE for a 304 (data is NULL).

request_id Server request id, handy for support.

See Also

`close_as_df()` and `close_as_sf()` to convert a reply by hand.

Index