

Package: terroir (via r-universe)

October 10, 2026

Title Geo-Point Environmental and Agricultural-Economics Data
Collector

Version 0.4.0

Description A uniform, point-keyed interface for retrieving climate, weather, vegetation-index, soil, elevation and agricultural-economics data for a geographic coordinate from many free data sources. Each source is wrapped as an optional adapter behind one request and response contract with mandatory provenance; the package builds and checks with no source package installed. The data-collector member of the ORCHESTRA.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports cli, data.table

Suggests apsimx, chirps, geodata, ggplot2, GSODR, httr2, jsonlite, knitr, MODISTools, nasapower, nert, ocular, read.abares, rmarkdown, rsi, rstac, sf, terra, testthat (>= 3.0.0), weatherOz, withr

Config/testthat/edition 3

VignetteBuilder knitr

Remotes AAGI-AUS/nert,
AAGI-AUS/ocular@3317d806806185d1cf509515212d4f2c9cb16e80

URL <https://github.com/max578/terroir>

BugReports <https://github.com/max578/terroir/issues>

Config/roxygen2/version 8.1.0

Repository <https://max578.r-universe.dev>

Date/Publication 2026-10-09 22:27:43 UTC

RemoteUrl <https://github.com/max578/terroir>

RemoteRef main

RemoteSha 85bdfa2ad9e7d768bc4e9d27b4cbdb4721fd2d26

Contents

collect_point	2
collect_points	3
geo_point_request	5
geo_point_schema	6
is_refusal	7
is_replayed	7
list_sources	8
new_geo_point_series	8
register_source	9
terroir_external_facts	10
terroir_fact_status	11
verify_geo_series	11
Index	12

collect_point	<i>Collect geo-point data for one request</i>
---------------	---

Description

The single entry point of the C6 contract. Dispatches a `geo_point_request()` to its registered source adapter and returns a verified `geo_point_series`, or a structured `refusal` when the source is unknown or its package is not installed.

Usage

```
collect_point(  
  request,  
  retrieved_at = NULL,  
  offline = getOption("terroir.offline", FALSE)  
)
```

Arguments

request	A <code>geo_point_request</code> from <code>geo_point_request()</code> .
retrieved_at	Optional character scalar timestamp (provenance). Defaults to the current time. Ignored in offline mode (a replayed series keeps the fixture's recorded timestamp).
offline	Logical scalar. When TRUE, answer from a recorded fixture with zero network egress (see Details). Defaults to the <code>terroir.offline</code> option, otherwise FALSE.

Details

`collect_point()` performs network input/output, so it stamps `retrieved_at` with the current time by default; pass `retrieved_at` for reproducible runs.

In **offline mode** (`offline = TRUE`, or globally via `options(terroir.offline = TRUE)`) the request is answered from a recorded fixture with no network egress at all: the matching cached series is replayed with its provenance marked `replayed = TRUE` and its original `retrieved_at` preserved (see [is_replayed\(\)](#)). This lets a sovereign-coordinate trial run with zero upstream contact, and makes a pipeline reproducible bit-for-bit. When no recorded fixture answers the request, offline mode returns a structured refusal rather than silently falling back to the network. The fixture directory is the `terroir.fixture_dir` option if set, otherwise the installed package's `fixtures/` directory.

Value

A `geo_point_series`, or a `terroir_refusal` – including when the returned values fail the value-plausibility conformer (an implausible NDVI/EVI or temperature reading), which is refused rather than aborted so one bad cell does not stop a multi-site collection.

See Also

[geo_point_request\(\)](#), [list_sources\(\)](#), [verify_geo_series\(\)](#), [is_replayed\(\)](#)

Examples

```
req <- geo_point_request(
  lon = 138.6, lat = -34.92,
  start = "2020-01-01", end = "2020-01-10",
  variables = c("T2M", "PRECTOTCORR"),
  source = "nasapower"
)
# offline = TRUE answers from the shipped fixture -- no network egress
series <- collect_point(req, offline = TRUE)
series
is_replayed(series)

# an unknown source returns a structured refusal, never an error
out <- collect_point(geo_point_request(0, 0, source = "not_a_source"))
is_refusal(out)
```

collect_points

Collect geo-point data for a batch of requests

Description

Batch/panel form of [collect_point\(\)](#). Stacks the results of every request in `requests` into one long `data.table`, one row per (request x date x variable) for a successful collection or exactly one row for a refused request. Every per-request refusal path of [collect_point\(\)](#) (unknown source, missing package, out-of-extent point, implausible value) applies identically.

Usage

```
collect_points(
  requests,
  retrieved_at = NULL,
  offline = getOption("terroir.offline", FALSE)
)
```

Arguments

`requests` A non-empty list of `geo_point_request` objects (see [geo_point_request\(\)](#)).

`retrieved_at, offline` As for [collect_point\(\)](#); the same value is passed through to every request in the batch.

Details

Requests are grouped by source. A source registered with a batch function (see [register_source\(\)](#)) receives all of its requests in one call, so a raster is read once for many points. Other sources are called one request at a time. Offline mode always answers one request at a time from fixtures.

A refusal is never dropped from the stack: it contributes one row with `status == "refused"` and reason set to the refusal's message, so a 40-site collection with two dead sources still returns 40 requests' worth of accounting, not 38.

Value

A data.table with a leading `request_index` (position in requests) and `status` ("ok" or "refused") column, a `reason` column (NA_character_ for an "ok" row, the refusal message otherwise), followed by the fixed C6 `geo_point_series` columns. `lon`, `lat` and `source` are populated on a refused row from the request itself (the values known even when nothing was collected); the remaining C6 columns are NA on a refused row. A final `call` column carries the upstream call recorded in each series' provenance.

See Also

[collect_point\(\)](#), [geo_point_request\(\)](#), [is_refusal\(\)](#)

Examples

```
reqs <- list(
  geo_point_request(138.6, -34.9, "2020-01-01", "2020-01-31",
    source = "nasapower"),
  geo_point_request(0, 0, source = "no_such_source")
)
out <- suppressWarnings(collect_points(reqs, offline = TRUE))
out$status
```

geo_point_request	<i>Construct a geo-point data request</i>
-------------------	---

Description

Build a request for environmental or agricultural-economics data at a single geographic point. The request is the input half of the constellation's C6 `geo_point_series` contract; pass it to `collect_point()`.

Usage

```
geo_point_request(
  lon,
  lat,
  start = NULL,
  end = NULL,
  variables = NULL,
  source,
  sovereign = FALSE,
  coarsen = NULL
)
```

Arguments

lon, lat	Numeric scalars. Longitude and latitude in WGS84 decimal degrees (EPSG:4326). lon in $[-180, 180]$, lat in $[-90, 90]$.
start, end	Character scalars (ISO-8601 "YYYY-MM-DD") or NULL. The inclusive date range. Ignored by static layers (for example soil or elevation).
variables	Character vector of source-native variable names, or NULL to let the adapter choose sensible defaults.
source	Character scalar. The id of one registered source adapter (see <code>list_sources()</code>).
sovereign	Logical scalar. When TRUE, the point is treated as sensitive (a sovereign-data coordinate – a grower's paddock, a trial site) and coarsening is applied by default so the exact GPS never leaves the host (see <code>coarsen</code>). Defaults to FALSE.
coarsen	Coordinate-coarsening policy applied to lon/lat before any upstream call, so the exact coordinate is never transmitted. One of: NULL (no coarsening; the default for a non-sovereign point); a single positive number giving the rounding precision in degrees (for example 0.01 rounds to roughly a 1 km cell); or a <code>list(grid = <size>)</code> snapping each coordinate to the nearest multiple of <size> degrees. For a sovereign = TRUE point with coarsen = NULL, a default precision of 0.01 degrees is applied automatically. The request records both the raw-requested coordinate and the coarsened-sent coordinate.

Value

An object of class `geo_point_request`. When coarsening is applied, `lon/lat` hold the coarsened coordinate (what is sent upstream) and `lon_raw/lat_raw` hold the exact requested coordinate (kept on the host for provenance); `coarsen_policy` records the policy as a string.

See Also

`collect_point()`, `list_sources()`

Examples

```
geo_point_request(138.6, -34.92, "2020-01-01", "2020-01-31",
                  variables = c("T2M", "PRECTOTCORR"), source = "nasapower")

# A sovereign point: the exact GPS is coarsened to ~1 km before any call.
geo_point_request(138.63124, -34.92871, "2020-01-01", "2020-01-31",
                  source = "nasapower", sovereign = TRUE)
```

`geo_point_schema`

The fixed C6 response schema

Description

Returns the ten column names every `geo_point_series` carries, in contract order. Column order is part of contract C6, so a sibling package typing against `terroir`'s output should read the schema from here rather than hard-coding it.

Usage

```
geo_point_schema()
```

Value

A character vector of column names.

Examples

```
geo_point_schema()
```

is_refusal	<i>Is an object a terroir refusal?</i>
------------	--

Description

Is an object a terroir refusal?

Usage

is_refusal(x)

Arguments

x An object.

Value

Logical scalar.

is_replayed	<i>Is a series a replayed (offline) fixture?</i>
-------------	--

Description

Reports whether a [geo_point_series](#) was produced by the offline replay path (`collect_point(request, offline = TRUE)`) rather than a live upstream call. A replayed series carries `replayed = TRUE` in its provenance and keeps the recording's `retrieved_at` rather than a fresh timestamp, so a consumer can always tell a cached series from a live one.

Usage

is_replayed(x)

Arguments

x A `geo_point_series` (or any object).

Value

Logical scalar; TRUE only for a replayed series.

See Also

[collect_point\(\)](#)

list_sources	<i>List registered data sources</i>
--------------	-------------------------------------

Description

List registered data sources

Usage

```
list_sources()
```

Value

A data.table with one row per registered source.

Examples

```
list_sources()
```

new_geo_point_series	<i>Build a geo_point_series from a long-format table</i>
----------------------	--

Description

Adapter-facing constructor. Coerces a long table to the fixed C6 schema, attaches mandatory provenance, and stamps the geo_point_series class. Most users call [collect_point\(\)](#) rather than this directly.

Usage

```
new_geo_point_series(  
  dt,  
  lon,  
  lat,  
  source,  
  source_version,  
  license,  
  retrieved_at,  
  call  
)
```

Arguments

dt	A <code>data.frame</code> / <code>data.table</code> with at least date, variable, value columns (plus optional unit).
lon, lat	Numeric scalars echoed onto every row.
source, source_version, license	Character scalars (provenance).
retrieved_at	Character scalar retrieval timestamp.
call	Character scalar describing the upstream call (provenance).

Value

A `geo_point_series` (a `data.table`) with provenance attached.

register_source	<i>Register a data-source adapter</i>
-----------------	---------------------------------------

Description

Adds a source adapter to the registry. Enforces the data-collector role's **free-only** rule: an adapter that does not reach a free source is refused.

Usage

```
register_source(
  id,
  fun,
  package,
  free,
  description = "",
  extent = NULL,
  batch = NULL
)
```

Arguments

id	Character scalar. Unique source id (used as source in geo_point_request()).
fun	A function <code>function(request, retrieved_at)</code> returning a geo_point_series .
package	Character scalar. The upstream package the adapter needs (declared in <code>Suggests</code> ; checked at collect time).
free	Logical scalar. Must be TRUE – the free-only role rule.
description	Character scalar. One-line description.
extent	Optional numeric <code>c(xmin, ymin, xmax, ymax)</code> in WGS84 giving the source's valid spatial extent, or NULL for a global source. A request outside a declared extent is refused by collect_point() rather than silently returning an out-of-coverage value (Independent Oracle Principle: an Australia-only source must not be queried over Argentina).

batch Optional function function(requests, retrieved_at) that answers many requests for this source in one upstream call. It returns a list of the same length as requests, each element a [geo_point_series](#) or a refusal. [collect_points\(\)](#) uses it for raster sources, where one file read serves every point. NULL (the default) means requests are answered one at a time by fun.

Value

Invisibly, the registered id.

Examples

```
toy_adapter <- function(request, retrieved_at) {
  dt <- data.frame(date = request$start, variable = "toy", value = 1)
  new_geo_point_series(
    dt, lon = request$lon, lat = request$lat,
    source = "toy", source_version = "1.0", license = "CC0",
    retrieved_at = retrieved_at, call = "toy_adapter()"
  )
}
register_source("toy", toy_adapter, package = "base", free = TRUE,
  description = "Illustrative in-memory adapter")
"toy" %in% list_sources()$id
```

terroir_external_facts

terroir's external-authority fact registry

Description

Enumerates every fact terroir asserts about an external data authority (MODIS product / band / scale, SILO variable codes, ABARES farm-data layer names, Sentinel-2 band mapping) together with its grounding provenance. A fact is *grounded* only when it has been diffed against the authority that owns it on a recorded date; until then it is ungrounded. The catalogue-diff oracle test grounds them at check time.

Usage

```
terroir_external_facts()
```

Value

A data.table keyed by source, fact_family, with columns asserted_value, oracle_kind (the authority that grounds the fact), verified_on (Date or NA), evidence_path, and a derived grounding token ("grounded" / "ungrounded").

Examples

```
facts <- terroir_external_facts()
facts[fact_family == "scale_factor"]
```

terroir_fact_status	<i>Grounding status summary for terroir's external facts</i>
---------------------	--

Description

One row per (source, fact_family) with the count of grounded vs unverified asserted values – a glance-able audit of how much of terroir's external-fact surface has been checked against its authorities.

Usage

```
terroir_fact_status()
```

Value

A data.table with source, fact_family, n_facts, n_grounded, n_unverified.

verify_geo_series	<i>Verify a geo_point_series against contract C6</i>
-------------------	--

Description

Checks the fixed schema, units column, and mandatory provenance. Fails loudly (the constellation never silently accepts a malformed series).

Usage

```
verify_geo_series(x)
```

Arguments

x An object to verify.

Value

Invisibly TRUE on success; otherwise an error.

Examples

```
req <- geo_point_request(
  lon = 138.6, lat = -34.92,
  start = "2020-01-01", end = "2020-01-10",
  variables = c("T2M", "PRECTOTCORR"),
  source = "nasapower"
)
series <- collect_point(req, offline = TRUE)
verify_geo_series(series)
```

Index

`collect_point`, [2](#)
`collect_point()`, [3–9](#)
`collect_points`, [3](#)
`collect_points()`, [10](#)

`geo_point_request`, [5](#)
`geo_point_request()`, [2–4, 9](#)
`geo_point_schema`, [6](#)
`geo_point_series`, [2, 6, 7, 9, 10](#)

`is_refusal`, [7](#)
`is_refusal()`, [4](#)
`is_replayed`, [7](#)
`is_replayed()`, [3](#)

`list_sources`, [8](#)
`list_sources()`, [3, 5, 6](#)

`new_geo_point_series`, [8](#)

`refusal`, [2](#)
`register_source`, [9](#)
`register_source()`, [4](#)

`terroir_external_facts`, [10](#)
`terroir_fact_status`, [11](#)

`verify_geo_series`, [11](#)
`verify_geo_series()`, [3](#)