Package: geopt (via r-universe)

October 21, 2026

Title Tools to Access and Map Spatial Datasets of Portugal

Version 0.1.0

Description Tools to download, process, and visualize official spatial datasets for Portugal. Get administrative boundaries at all levels and other geographic data ready for mapping. Data from Statistics Portugal (INE), for example retrieved with the 'ineptr2' package, can be joined to the boundaries by place name or code, including names from earlier boundary reforms.

License MIT + file LICENSE

URL https://c-matos.github.io/geopt/, https://github.com/c-matos/geopt

BugReports https://github.com/c-matos/geopt/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports R6, sf, httr2 (>= 1.0.0), geojsonsf, digest

Suggests rmapshaper, stringdist, ggplot2, tmap, echarts4r, htmlwidgets, dplyr, plotly, knitr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Collate 'utils.R' 'compose.R' 'layout.R' 'atoms.R' 'lookup.R' 'match.R' 'match-pool.R' 'join.R' 'derive.R' 'vintage.R' 'censos.R' 'get_map.R' 'plot.R' 'GeoClient.R' 'data.R' 'geopt-package.R'

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Carlos Matos [aut, cre, cph]

Maintainer Carlos Matos <carlosmdmatos@gmail.com>

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

Date/Publication 2026-10-21 16:50:18 UTC

RemoteUrl https://github.com/cran/geopt

RemoteRef HEAD

RemoteSha b981e5a2098a21c3a98fe506bf28653925832bea

Contents

boundaries . . . 2
box_labels_echarts . . . 5
frame_data_echarts . . . 6
GeoClient . . . 7
get_censos . . . 10
get_map . . . 13
island_layout . . . 15
plot_pt . . . 17
pt_censos_vars . . . 19
pt_crosswalk . . . 20
pt_health_codes . . . 21
pt_join . . . 23
pt_lookup . . . 24
pt_match . . . 26
pt_match_report . . . 30
pt_normalize . . . 30
pt_nuts_codes . . . 31
pt_pad_code . . . 32
pt_recode . . . 33
pt_registry . . . 35

Index **37**

`boundaries` *Simplified Portuguese administrative boundaries*

Description

Pre-simplified sf objects derived from CAOP (Carta Administrativa Oficial de Portugal). All levels are topologically consistent: higher-level boundaries are dissolved from simplified freguesias, guaranteeing aligned edges. Each level carries its own identity columns (official codes and names) plus aggregated `area_ha`, so every dataset joins directly against user data or against pt_registry.

Usage

```
pt_freguesias

pt_municipios

pt_distritos
```

`pt_nuts1`

`pt_nuts2`

`pt_nuts3`

`pt_uls`

`pt_reg_saude`

`pt_aces`

`pt_uls_municipio`

`pt_aces_municipio`

Format

An sf data frame with `MULTIPOLYGON` geometry.

An object of class `sf` (inherits from `data.frame`) with 3259 rows and 16 columns.

An object of class `sf` (inherits from `data.frame`) with 308 rows and 13 columns.

An object of class `sf` (inherits from `data.frame`) with 29 rows and 8 columns.

An object of class `sf` (inherits from `data.frame`) with 3 rows and 4 columns.

An object of class `sf` (inherits from `data.frame`) with 9 rows and 6 columns.

An object of class `sf` (inherits from `data.frame`) with 26 rows and 9 columns.

An object of class `sf` (inherits from `data.frame`) with 55 rows and 7 columns.

An object of class `sf` (inherits from `data.frame`) with 7 rows and 6 columns.

An object of class `sf` (inherits from `data.frame`) with 71 rows and 8 columns.

An object of class `sf` (inherits from `data.frame`) with 50 rows and 7 columns.

An object of class `sf` (inherits from `data.frame`) with 66 rows and 8 columns.

Details

Common columns (present where applicable to the level):

dtmnfr 6-digit freguesia code (distrito + municipio + freguesia), zero-padded character. Freguesias only.

dtmn 4-digit municipality code (first 4 digits of `dtmnfr`).

dt 2-digit distrito/ilha code.

nuts1_cod, nuts2_cod, nuts3_cod NUTS 2024 codes in full Eurostat form (`PT1`, `PT19`, `PT191`, ...).

freguesia, municipio, distrito_ilha, nuts1, nuts2, nuts3 Official names.

designacao_simplificada The freguesia's name without the `"União das freguesias de"` prefix. Freguesias only.

perimetro_km Perimeter in kilometres, as CAOP gives it. Freguesias only.

area_ha Area in hectares, summed from the constituent freguesias.

n_freguesias, n_municipios Number of constituent units.

Geometries use WGS84 (EPSG:4326) and cover all of Portugal (Continente, Açores, and Madeira).

A multi-layer GeoPackage containing the same data is also bundled at `inst/extdata/caop_simplified.gpkg`, with one layer per level:

```
gpkg <- system.file("extdata", "caop_simplified.gpkg", package = "geopt")
sf::st_layers(gpkg)
sf::st_read(gpkg, layer = "municipio")
```

The health maps are dissolved from the same simplified freguesias using the pt_health_codes mappings, so their edges align with every administrative level. Each reflects a single health vintage and exists on the current freguesia map only. `pt_uls`, `pt_reg_saude`, and `pt_aces` are freguesia-accurate: units that split municipalities are drawn correctly.

`pt_uls` has 55 units: 39 mainland ULS plus the island health units, which are not ULS — the Açores have nine Unidades de Saúde de Ilha (`"USI São Miguel"`, ...) and Madeira seven centros de saúde (`"CS Funchal Zona I"`, ...). `pt_reg_saude` carries the seven regiões de saúde. Both are the 2024 reform.

`pt_aces` has 71 Agrupamentos de Centros de Saúde — the organization the 2024 reform superseded. Its `reg_saude` is the region of that time (the former ARS), not the 2024 região de saúde.

`pt_uls_municipio` (50 units) and `pt_aces_municipio` (66) are for data by municipality, which can't be divided between the units that share a municipality. Those units are merged into one, named after its members (`"ULS Santo António + São João"`), so every unit is a set of whole municipalities. Lomba, the one freguesia of Gondomar in ULS Vila Nova de Gaia/Espinho and ACES Espinho/Gaia, is drawn with the rest of Gondomar, where data by municipality counts it. They hold the units of the `uls_municipio` and `aces_municipio` columns that `pt_join()` adds.

All datasets carry `nuts1_cod` and `nuts1`, so a health map can be filtered to the mainland, the Açores (`"PT2"`) or Madeira (`"PT3"`), and `pt_aces` and `pt_aces_municipio` carry their `reg_saude`. They carry no other region column, because no other is the same throughout every unit: ULS Médio Tejo, for example, is split between two regiões de saúde.

Source

Direção-Geral do Território (DGT), CAOP. https://www.dgterritorio.gov.pt/cartografia/cartografia-tematica/caop

See Also

`pt_lookup()` for the geometry-free lookup table, `get_map()` for composed maps with island insets.

box_labels_echarts *Add labels to an echarts map*

Description

Pipe-able function that injects series-level label configuration into an existing echarts widget, labelling the rectangles drawn for `islands = "box"`. Uses a JavaScript formatter that shows the `geopt_label` text for those rectangles and an empty string for every other feature.

Usage

```
box_labels_echarts(
  e,
  sf_obj,
  font_size = 12,
  color = "#333",
  position = "right",
  offset = c(92, 0),
  ...
)
```

Arguments

`e`	An echarts widget created by `echarts4r::e_charts()`.
`sf_obj`	An sf object returned by `get_map()`. Features are matched by its `name` column (the one the widget was built with), or by `geopt_label` if there is none.
`font_size`	Font size for labels. Default `12`.
`color`	Color for labels. Default `"#333"`.
`position`	Label position relative to the feature bounding box. Default `"right"`. See echarts label position options.
`offset`	Numeric vector of length 2 (`c(x, y)`) to shift the label in pixels. Default `c(92, 0)`.
`...`	Additional echarts label properties (e.g. `align`, `verticalAlign`, `fontWeight`, `rotate`).

Value

The modified echarts widget.

Examples

```
m <- get_map("nuts3", islands = "box")
m$name <- dplyr::coalesce(m$geopt_label, m$nuts3)
geojson <- geojsonsf::sf_geojson(m)

m |>
```

```
  echarts4r::e_charts(name) |>
  echarts4r::e_map_register("pt", geojson) |>
  echarts4r::e_map(area_ha, map = "pt", roam = TRUE) |>
  box_labels_echarts(m) |>
  echarts4r::e_visual_map(area_ha)
```

`frame_data_echarts` *Build echarts frame styling data*

Description

Creates a list of data items suitable for passing to `echarts4r::e_map(data = ...)` to style the frames around the island insets, with line widths based on `geopt_frame_level`. Because the `data` parameter in echarts replaces the entire series data, this function returns entries for **all** features — preserving the mapped values for real regions and styling frames as transparent overlays.

Usage

```
frame_data_echarts(
  sf_obj,
  value_col,
  level_widths = c(0.8, 0.4, 0.2),
  border_color = "#333"
)
```

Arguments

`sf_obj`	An sf object returned by `get_map()`.
`value_col`	Name of the value column being mapped (as a string).
`level_widths`	Numeric vector mapping frame level to border width. Position 1 corresponds to `geopt_frame_level == 1`, etc.
`border_color`	Border color string. Default `"#333"`.

Value

A list of named lists, each with `name`, `value`, and (for frames) `itemStyle`. Pass this to `echarts4r::e_map(data = ...)`.

Examples

```
m <- get_map("municipio")
m$name <- dplyr::coalesce(m$municipio, m$geopt_label)
geojson <- geojsonsf::sf_geojson(m)

m |>
  echarts4r::e_charts(name) |>
  echarts4r::e_map_register("pt", geojson) |>
```

```
echarts4r::e_map(area_ha, map = "pt", roam = TRUE,
                 data = frame_data_echarts(m, "area_ha")) |>
echarts4r::e_visual_map(area_ha)
```

GeoClient *GeoClient*

Description

R6 class for downloading, processing, and caching Portuguese geographic boundaries from CAOP (Carta Administrativa Oficial de Portugal), and for filtered queries to DGT's OGC API. `get_censos()` and `get_map()` use one for their downloads too.

Failed downloads

When a download fails, because a server can't be reached or a file has moved or changed, the method stops with an error saying which and why. The error has class `geopt_download_error`, so a script can catch download failures alone with `tryCatch(..., geopt_download_error = ...)`.

Usage

```
client <- GeoClient$new()
sf <- client$get_caop("continente", "municipio")
```

Active bindings

`cache_dir` Character. Path to the cache directory. Defaults to `tools::R_user_dir("geopt", "cache")`.

`use_cache` Logical. Whether to read and write cached results (default `TRUE`). With `FALSE`, cached results are neither read nor written: each call builds its result again, and the latest CAOP year is looked up online instead of taken from the cache. Downloaded files (CAOP and census GeoPackages) are kept and reused either way; `$clear_cache("raw")` removes them.

`crs` Integer EPSG code for the default coordinate reference system. Defaults to 4326 (WGS84). Output can be transformed to any CRS via the `crs` argument of `$get_caop()`.

Methods

Public methods:

- `GeoClient$new()`
- `GeoClient$get_caop()`
- `GeoClient$get_ogc()`
- `GeoClient$list_ogc_collections()`
- `GeoClient$download_caop()`
- `GeoClient$simplify_boundaries()`
- `GeoClient$export_boundaries()`

- GeoClient$list_cached()
- GeoClient$clear_cache()
- GeoClient$print()
- GeoClient$clone()

`GeoClient$new()`: Create a new GeoClient.

Usage:

```
GeoClient$new(cache_dir = NULL, use_cache = TRUE, crs = DEFAULT_CRS)
```

Arguments:

`cache_dir` Path to cache directory. Defaults to user cache dir.

`use_cache` Whether to read and write cached results. Downloaded files are kept either way; see the `use_cache` field.

`crs` Default EPSG code. Defaults to 4326 (WGS84).

`GeoClient$get_caop()`: Get geographic boundaries as an sf object.

Usage:

```
GeoClient$get_caop(region, level, year = NULL, crs = NULL, detail = "full")
```

Arguments:

`region` One of "continente", "raa", "ram".

`level` One of "freguesia", "municipio", "distrito", "nuts1", "nuts2", "nuts3".

`year` CAOP year. If NULL, resolves to the latest available.

`crs` Output EPSG code. If NULL, uses the client default.

`detail` Simplification level: "full", "high", "medium", "low" or "very_low".

Returns: An sf object.

`GeoClient$get_ogc()`: Get features from a DGT OGC API collection.

Queries `https://ogcapi.dgterritorio.gov.pt/` and returns the response as an sf object. Use this for filtered or bounded subsets (single municipio, bbox around a city, COS land-cover by area). For bulk country-wide downloads of CAOP, `$get_caop()` is faster.

Usage:

```
GeoClient$get_ogc(
  collection,
  bbox = NULL,
  bbox_crs = 4326L,
  properties = NULL,
  cql_filter = NULL,
  crs = NULL,
  limit = OGC_DEFAULT_LIMIT,
  max_features = Inf
)
```

Arguments:

`collection` Collection id, e.g. "freguesias", "municipios", "nuts3", "cos2023v1". See `$list_ogc_collections()`; an id geopt doesn't know is an error that lists the ones it does. The large collections (`"cos2018v3"`, `"cos2023v1"`, `"cadastro"`) need a `bbox`, `properties` or `cql_filter`.

`bbox` Numeric length-4 vector `c(xmin, ymin, xmax, ymax)` or an sf `bbox` object. If NULL, no spatial filter is applied.

`bbox_crs` Integer EPSG code that `bbox` is expressed in. Defaults to 4326. Only sent to the server when not 4326.

`properties` Named list of equality filters on attribute columns, e.g. `list(distrito = "Lisboa")`. Use `cql_filter` for more complex predicates.

`cql_filter` Optional CQL-Text filter string, passed through verbatim. Example: `"distrito_ilha = 'Lisboa'"`. DGT's server accepts CQL1 syntax (`cql-text`), not CQL2.

`crs` Output EPSG code. If NULL, uses the client default. Requested server-side to avoid client-side reprojection.

`limit` Page size sent to the server. Default 1000.

`max_features` Stop pagination after this many features. Defaults to `Inf` (fetch everything).

Returns: An sf object, or NULL if the query returned zero features.

`GeoClient$list_ogc_collections()`: List collections reported live by the DGT OGC API. Hits `/collections?f=json`. Used for discovery; the package itself validates against the hard-coded `OGC_COLLECTIONS` registry.

Usage:

```
GeoClient$list_ogc_collections()
```

Returns: A data.frame with columns `id`, `title`, `item_type`, `description`.

`GeoClient$download_caop()`: Download boundaries to cache without loading into memory.

Usage:

```
GeoClient$download_caop(region, year = NULL)
```

Arguments:

`region` One of "continente", "raa", "ram".

`year` CAOP year. If NULL, resolves to the latest available.

Returns: The paths to the downloaded GeoPackage files (invisibly).

`GeoClient$simplify_boundaries()`: Simplify boundary geometries.

Usage:

```
GeoClient$simplify_boundaries(sf_obj, detail = "medium")
```

Arguments:

`sf_obj` An sf object.

`detail` One of "full", "high", "medium", "low" or "very_low".

Returns: A simplified sf object.

`GeoClient$export_boundaries()`: Export boundaries to a file format suitable for dashboards.

Usage:

```
GeoClient$export_boundaries(
  sf_obj,
  path,
  format = "geojson",
  detail = NULL,
  crs = 4326L
)
```

Arguments:

`sf_obj` An sf object.

`path` Output file path.

`format` Output format: "geojson" or "gpkg".

`detail` Optional simplification level before export.

`crs` Output EPSG code. Defaults to 4326 (WGS84) for web compatibility.

Returns: The output path (invisibly).

`GeoClient$list_cached()`: List cached data files.

Usage:

```
GeoClient$list_cached()
```

Returns: A list with `processed` and `raw` character vectors.

`GeoClient$clear_cache()`: Clear cached data.

Usage:

```
GeoClient$clear_cache(type = c("all", "processed", "raw"))
```

Arguments:

`type` One of "all", "processed", or "raw".

`GeoClient$print()`: Print method.

Usage:

```
GeoClient$print(...)
```

Arguments:

`...` Ignored.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`get_censos` *Get INE census geographies (BGRI, secções, GRID)*

Description

Downloads and reads the census geographies that INE distributes at https://mapas.ine.pt/download/index2021.phtml: BGRI subsecções (city-block/place-part polygons) for the 1991, 2001, 2011, and 2021 censuses, statistical secções, and the 1 km grid — each with the census variables INE embeds in the files. Files are fetched at the smallest scope that covers the request (per-municipality files are typically 1–3 MB) and cached.

Usage

```
get_censos(
  municipio = NULL,
  year = 2021,
  level = "subseccao",
  region = NULL,
  freguesia = NULL,
  variables = NULL,
  geometry = TRUE,
  crs = NULL,
  client = NULL
)
```

Arguments

`municipio`	Municipality name(s) or DTMN code(s). See *Scope*.
`year`	Census year: 1991, 2001, 2011, or 2021 (default).
`level`	`"subseccao"` (default), `"seccao"`, or `"grid"`.
`region`	Region keyword, NUTS2/NUTS3 name, or NUTS 2013 code. See *Scope*.
`freguesia`	Freguesia name(s) or DTMNFR code(s). See *Scope*.
`variables`	Optional character vector of census variables to keep (case-insensitive; identifier columns are always kept). See pt_censos_vars.
`geometry`	Logical. `FALSE` drops the geometry and returns a plain data.frame. Default `TRUE`.
`crs`	Output EPSG code. If `NULL`, uses the client default (4326).
`client`	Optional GeoClient to use (for a custom cache directory or CRS). Defaults to `GeoClient$new()`.

Value

An sf object (or a data.frame when `geometry = FALSE`) with one row per subsecção, secção, or grid cell. A failed download stops with an error of class `geopt_download_error` saying why (see GeoClient).

Scope

Exactly one scope must be given (there is no default: a whole-country request is ~250 MB and must be explicit):

- `municipio`: one or more municipality names or 4-digit DTMN codes, resolved via `pt_match()` (qualifiers like `"Lagoa (Açores)"` work). Downloads the per-municipality file(s).
- `freguesia`: one or more freguesia names or 6-digit DTMNFR codes. Downloads the enclosing municipality file(s) and filters. May be combined with `municipio` to disambiguate names.
- `region`: `"portugal"`, `"continente"`, `"raa"`/`"açores"`, `"ram"`/`"madeira"`, a NUTS2 or NUTS3 name, or a NUTS 2013 code (`"PT16D"`/`"16D"`). Census files follow the NUTS 2013 geography; NUTS 2024 names are accepted when they map unambiguously to it.

Levels and years

- "subseccao" (default): BGRI subsecções, available for all four censuses. 2021 files embed 32 census variables, 2011 files 122, 1991/2001 files seven basic counts.
- "seccao": statistical secções. For 2021 these are INE's native files with ~180 variables (more detail than the subsecção set). For earlier censuses no secção product exists: subsecções are downloaded and dissolved, summing that year's variables (adds n_subseccoes).
- "grid": the 2021-only 1 km grid (EPSG:3035 cells crossing administrative boundaries), region scope only.

See pt_censos_vars for every variable with its INE description.

Identifiers

INE's identifier columns are year-suffixed and inconsistent (DTCC91, DTMN21, FR11, ...). Year-stable tidy columns are added alongside the originals: censos_year, bgri (full subsecção code), dtmn, dtmnfr, seccao, subseccao, and — where the file carries them (1991–2011) — lugar/lugar_nome. GIS artifacts (OBJECTID, SHAPE_*) and INE join metadata are dropped.

Freguesia codes are vintage-specific: each census uses the freguesia map in force at the time. The dtmnfr codes of the 2021 census are the "2013" vintage, those of 2011 the "2011" vintage and those of 2001 the "2001" vintage, so pt_recode() converts them to today's map. The 1991 census used an earlier map (4208 freguesias in 305 municípios, before Odivelas, Trofa and Vizela were created in 1998), which geopt has no vintage for.

Name-based freguesia scoping resolves against the current registry and, for 2021, also matches through pt_crosswalk(); for older censuses pass that era's codes if a name does not resolve. In the 2001 files, two freguesias still carry the code of the municipality they belonged to before a transfer: Vale de Amoreira is 090756 (Guarda, now Manteigas) and Moita 100117 (Alcobaça, now Marinha Grande). pt_recode() and pt_match() accept those codes for the "2001" vintage, and a freguesia filter finds either form.

See Also

pt_censos_vars, GeoClient (for client), get_map() for administrative boundaries.

Examples

```
# City blocks of Braga with the 2021 census variables (~2 MB download)
bgri <- get_censos("Braga")

# Population density by subsecção
bgri$dens <- bgri$N_INDIVIDUOS / sf::st_area(bgri)

# 2011 vintage, only the resident population variables
get_censos("Braga", year = 2011,
           variables = c("N_INDIVIDUOS_RESIDENT", "N_INDIVIDUOS_RESIDENT_H"))

# Secção-level detail (native 2021 product, ~180 variables)
get_censos("Braga", level = "seccao")

# A whole NUTS3, and the 1 km grid for the mainland
```

```
get_censos(region = "Cávado")
get_censos(region = "continente", level = "grid")

# Just the data, no geometry
get_censos("Braga", geometry = FALSE)
```

---

get_map *Get a composed map of Portugal*

Description

Returns an sf object with mainland Portugal and the islands moved next to it as insets, ready for plotting. Uses the pre-simplified bundled datasets.

Usage

```
get_map(
  level = "municipio",
  islands = "compact",
  data = NULL,
  by = NULL,
  fuzzy = FALSE,
  aliases = NULL,
  admin_vintage = NULL,
  nuts_vintage = NULL,
  client = NULL
)
```

Arguments

level	Administrative level: "freguesia", "municipio", "distrito", "nuts1", "nuts2", "nuts3" — or one of the health geographies dissolved from freguesias via pt_health_codes: "uls", "reg_saude" (regiões de saúde) for the 2024 reform, or "aces". If you have data by municipality, you might need "uls_municipio" and "aces_municipio": the units that share a municipality are merged into one (see vignette("health-geographies")). Health levels are defined on the 2025 map only.
islands	How to show the Açores and Madeira: "compact" (the default), "groups", "box", "in_place" or "none" (mainland only), or a layout made with island_layout() for anything else, such as a different setting per archipelago or no frames.
data	Optional data.frame to join onto the boundaries, producing a map ready for a choropleth. Requires by.
by	How to join data: which column identifies the unit and what it contains. Same forms as pt_join(). The unit keyword must correspond to level. Rows of data for an archipelago left out with "none" are dropped without a warning.

fuzzy	Logical. Enable fuzzy name matching for the join (see pt_match()). Default FALSE.
aliases	Optional named character vector of extra name aliases for the join (see pt_match()).
admin_vintage	Which freguesia map to draw. Available values are "2025" (the default, the bundled data) or "2013", the map in force from the 2013 reform until freguesias were restored in 2025. The 2013 map is downloaded from DGT (CAOP 2024, the last release that used the 2013 freguesias division), simplified and cached the first time, which needs the rmapshaper package and an internet connection. For "2011" and "2001" maps, use INE's BGRI by merging census areas of that year instead (see get_censos()), or convert the data with pt_recode(). Names in data are matched against this vintage. See vignette("vintages").
nuts_vintage	Which NUTS revision to use: "2024" (the default), "2013" or "2002". Independent of admin_vintage, because every revision groups whole municipalities.
client	Optional GeoClient to download and cache with (for a custom cache directory).

Details

Every result has the same columns, in the same order, whatever the options: the level's own columns (and any joined data), then geopt_type, geopt_label and geopt_frame_level, then the geometry. geopt_type is "boundary" for the regions and "frame" for the frames drawn around the insets (see island_layout()). geopt_label holds a frame's name, or the label of an archipelago drawn as a rectangle (islands = "box").

For a map of one region, filter the result: rows keep their codes and names. For an archipelago, use islands = "in_place" first, so it is in its true position rather than moved and enlarged.

Value

An sf object in WGS84 (EPSG:4326).

See Also

island_layout(), plot_pt()

Examples

```
library(ggplot2)

m <- get_map("municipio")
ggplot(m) + geom_sf()

# Mainland only
m <- get_map("municipio", islands = "none")

# Regions and frames drawn separately
m <- get_map("nuts2")
ggplot() +
  geom_sf(data = m[m$geopt_type == "boundary", ], aes(fill = nuts2)) +
  geom_sf(data = m[m$geopt_type == "frame", ], fill = NA, linewidth = 0.5)
```

```
# The archipelagos as labelled rectangles
get_map("nuts3", islands = "box")

# The Açores in their three official groups, no frames
get_map("municipio", islands = island_layout(raa = "groups", frames = FALSE))

# Join your own data by municipality name
df <- data.frame(concelho = c("Porto", "Lisboa"), value = c(10, 20))
m <- get_map("municipio", data = df, by = c(municipio = "concelho"))
ggplot(m) + geom_sf(aes(fill = value))

# The NUTS 3 regions of the 2013 revision (25, against 26 today)
get_map("nuts3", nuts_vintage = "2013")
```

`island_layout` *Lay out the islands*

Description

The Açores are about 1,500 km west of the mainland and Madeira about 1,000 km to the south-west, so a map of Portugal drawn to scale is mostly ocean. `get_map()` and `plot_pt()` move the islands next to the mainland, as insets. `island_layout()` says how: which archipelagos to show, how to split them into insets, and whether to draw frames around them.

Usage

```
island_layout(
  both = "compact",
  raa = both,
  ram = both,
  frames = TRUE,
  box_position = "bottom_left"
)
```

Arguments

`both`	How to show both archipelagos: `"compact"`, `"groups"`, `"box"`, `"in_place"` or `"none"`. See Choices.
`raa`, `ram`	How to show the Açores (`raa`) or Madeira (`ram`), when it should differ from `both`: one of the choices, or a layout list (see Moving and resizing the islands section below).
`frames`	Draw frames around the insets? Only applies to archipelagos drawn as insets.
`box_position`	Where to put the rectangles, for `"box"`: `"bottom_left"` or `"top_left"` of the mainland, or `c(x, y)`, the longitude and latitude of the bottom-left corner of the stack.

Details

Pass the result as `islands` to `get_map()` or `plot_pt()`. For the common cases a string is enough: `islands = "none"` is the same as `islands = island_layout("none")`.

Value

An island layout, to pass as `islands` to `get_map()` or `plot_pt()`.

Choices

Each archipelago takes one of:

`"compact"` (The default.) Insets arranged to take the least room: the Açores in four insets (the western group, the central group, São Miguel, and Santa Maria on its own), and Porto Santo in its own small inset, moved next to Madeira.

`"groups"` The Açores in their three official groups (Ocidental, Central and Oriental). Madeira has no groups, so for Madeira this is the same as `"compact"`.

`"whole"` Madeira and Porto Santo in one inset, at their true distance apart. Not available for the Açores: at a size that fits next to the mainland, the smaller islands would be too small to read.

`"box"` A labelled rectangle instead of the islands, stacked to the left of the mainland. Only works at levels where each archipelago is a single region: `"nuts1"`, `"nuts2"`, `"nuts3"` and `"reg_saude"`. Can't be combined with insets.

`"in_place"` The islands where they really are, without frames. The starting point for a map of one archipelago (see Examples).

`"none"` Leave the archipelago out.

Frames

With `frames = TRUE`, `get_map()` adds the frames drawn around the insets, which show that the islands have been moved. They are extra rows with `geopt_type = "frame"`, their name in `geopt_label`, and `geopt_frame_level` saying what they surround: 1 for a whole archipelago, 2 for a group of islands and 3 for a single island.

Moving and resizing the islands

`island_layout()` returns a list, which you can change before passing it on. Each archipelago drawn as insets has:

`anchor` Longitude and latitude its insets are placed around.

`label` The name on its frame (and on its rectangle, for `"box"`).

`frame_buffer` Space between the islands and the frame around the whole archipelago, in degrees.

`groups` One entry per inset, each with:

- `crop` The part of the archipelago this inset shows, as `c(xmin =, ymin =, xmax =, ymax =)` in longitude and latitude. `NULL` shows the whole archipelago.
- `offset` Where the inset goes, as `c(x, y)` in degrees from `anchor`.
- `scale` How much to enlarge it. Default 1.

`label` Optional. Draws a frame around this inset alone, with this name.

`frame_level`, `frame_buffer` Optional. That frame's `geopt_frame_level` (default 3) and its space around the islands.

`sub_frames` Optional. Frames around several insets at once, each with `groups` (the inset names), `label`, and optionally `frame_buffer` and `frame_level` (default 2).

Print `island_layout()$raa` to see a complete example. To lay out an archipelago from scratch, pass a list with the same structure as `raa` or `ram`.

See Also

`get_map()`, `plot_pt()`

Examples

```
island_layout()
island_layout(raa = "groups", ram = "whole")


get_map("municipio", islands = "none")    # mainland only
get_map("nuts3", islands = "box")
get_map("nuts3", islands = island_layout("box", box_position = "top_left"))
get_map("municipio", islands = island_layout(frames = FALSE))

# The Açores alone, in their true position
m <- get_map("municipio", islands = "in_place")
acores <- m[m$nuts2_cod %in% "PT20", ]

# Move the Açores half a degree west, with an English label
lay <- island_layout()
lay$raa$anchor <- lay$raa$anchor + c(-0.5, 0)
lay$raa$label <- "Azores"
get_map("nuts3", islands = lay)
```

`plot_pt` *Quick map of Portugal*

Description

One-liner choropleths on the composed Portugal map (mainland + island insets), rendered with ggplot2 or echarts4r. Returns the renderer's native object – a `ggplot` you can extend with `+`, or an `echarts4r` widget you can pipe into `e_*()` functions – with deliberately light styling.

Usage

```
plot_pt(
  data = NULL,
  value = NULL,
  level = "municipio",
  by = NULL,
  renderer = "ggplot",
  palette = NULL,
  na_color = "grey85",
  legend = TRUE,
  ...
)
```

Arguments

data	NULL (blank map), a data.frame to join (requires by), or an sf object returned by get_map().
value	Name of the column to fill by (a column of data). Numeric columns get a continuous scale; character/factor columns a discrete one. NULL draws boundaries only.
level	Administrative level, as in get_map(). Ignored (detected) when data is an sf map.
by	How to join a data.frame data; see get_map().
renderer	"ggplot" or "echarts".
palette	Fill palette: a vector of colors, or a single string naming a grDevices::hcl.colors() palette (e.g. "Blues 3", "Viridis"). Defaults to "Viridis".
na_color	Fill for boundaries without a data value.
legend	Logical. Show the fill legend?
...	Passed to get_map() (islands, admin_vintage, nuts_vintage, fuzzy, aliases, client). Ignored when data is already an sf map.

Value

A ggplot object or an echarts4r widget, depending on renderer.

Input forms

- plot_pt() – blank administrative map.
- plot_pt(df, value = "cases", by = "dtmn") – joins df onto the boundaries via get_map() and fills by value.
- plot_pt(m, value = "cases") where m is an sf returned by get_map() – renders an already-composed map directly. Build the map once, render it with several renderers.

Renderers

- "ggplot" (default): returns a ggplot object. Suited to static reports; style further with + theme(...), + labs(...), etc.
- "echarts": returns an echarts4r htmlwidget with tooltip and roam enabled, suited to Shiny. Features are keyed internally by official code (not by name), so duplicate unit names (two "Lagoa" municipalities, 196 reused freguesia names) cannot mis-map values; tooltips show the display name. Style further with |> echarts4r::e_title(...), etc.

See Also

get_map(), island_layout(), pt_join()

Examples

```
# blank municipality map
plot_pt()

df <- data.frame(concelho = c("Porto", "Lisboa"), value = c(10, 20))
plot_pt(df, value = "value", by = c(municipio = "concelho"))
plot_pt(df, value = "value", by = c(municipio = "concelho"), islands = "none")

# Build once, render twice
m <- get_map("nuts3", islands = "box")
m$area_ha <- as.numeric(m$area_ha)
plot_pt(m, value = "area_ha")
plot_pt(m, value = "area_ha", renderer = "echarts")
```

---

pt_censos_vars — *Census variables available from get_censos()*

Description

Dictionary of the census variables embedded in the INE geographies returned by get_censos(), with INE's Portuguese descriptions. One row per variable per census year and level, derived from the actual GeoPackage columns (INE's description CSVs use spelling variants of the column names, which are matched to them).

Usage

```
pt_censos_vars
```

Format

A data.frame with columns:

year Census year (1991, 2001, 2011, 2021).

level "subseccao", "seccao", or "grid".

variable Column name as it appears in the returned object.

description INE's description (Portuguese).

Details

The variable sets differ widely by year: 2021 subsecções carry 32 variables, 2011 subsecções 122, 1991/2001 subsecções seven basic counts, 2021 secções ~180 variables, and the 2021 grid ten.

Source

INE, Censos — importação de dados alfanuméricos e geográficos (BGRI, GRID e Lugares): https://mapas.ine.pt/download/index2021.phtml

See Also

get_censos()

Examples

```
# What is available for the 2011 census?
vars <- pt_censos_vars[pt_censos_vars$year == 2011, ]
head(vars[, c("variable", "description")])
```

---

pt_crosswalk *Correspondence between two administrative vintages*

Description

Which units of one vintage correspond to which of another, and how each changed. Derived from shared territory rather than stored, so any pair of vintages works and nothing goes stale when a new one is added.

Usage

```
pt_crosswalk(
  from = "2013",
  to = NULL,
  level = c("freguesia", "municipio", "distrito")
)
```

Arguments

from, to	Administrative vintages: "2001", "2011", "2013" or "2025". from defaults to "2013"; leave to out for the newest ("2025").
level	Which units to relate. Only "freguesia" changes across these vintages; "municipio" and "distrito" are included for completeness.

Value

A data.frame with one row per corresponding pair: `from`, `to`, `name_from`, `name_to`, `relation`.

Codes are vintage-local

A DTMNFR identifies a unit *within its own vintage only*. None of the 302 freguesias restored in 2025 reused its pre-2013 code, so this table — not string comparison — is the only safe way to relate codes across vintages.

Relations

`relation` is computed from the correspondence itself:

`unchanged` same territory, same code, same name.

`renamed` same territory and code, different name.

`renumbered` same territory, different code. Colos kept its territory across the 2025 reform and changed only its code (`021118` → `021124`).

`merged` several `from` units became one — the 2013 uniões.

`split` one `from` unit became several — the 2025 reposição.

See Also

`pt_recode()` to carry values across vintages, pt_registry for the units themselves.

Examples

```
x <- pt_crosswalk("2013", "2025")
table(x$relation)

# what became of the 2013 "030882" code
x[x$from == "030882", ]
```

---

pt_health_codes *Health geography of Portuguese freguesias, both vintages*

Description

One row per freguesia per health vintage. Keyed on the freguesia rather than the município because `unit` varies *within* a município in five of them — the reason this cannot live alongside pt_nuts_codes, which is município grained.

Usage

```
pt_health_codes
```

Format

A data.frame with 6518 rows and columns:

health_vintage `"2022"` or `"2024"`.

dtmnfr 6-digit freguesia code, current vintage.

reg_saude Região de saúde (2024) or the former ARS (2022).

scheme `"uls"` for 2024, `"aces"` for 2022 — different concepts, not two names for one thing.

unit The local unit: a ULS or an ACES.

unit_municipio Município-grained approximation of `unit`.

Details

Mappings are recorded against the current (2025) freguesia set; `pt_lookup()` reaches them from an earlier `admin_vintage`.

Vintages

`"2024"` The reform: 5 regiões de saúde on the mainland plus the island services, and ULS (Unidades Locais de Saúde) as the local unit.

`"2022"` What it superseded: the former ARS, and ACES (Agrupamentos de Centros de Saúde) as the local unit.

`unit_municipio` is the unit to use for data by município. Units that share a município are merged into one pseudo-unit, and merging follows chains, so it can differ from `unit` even in a município that lies whole in one unit. For example, ULS São José also covers part of Loures, which merges ULS Loures-Odivelas with Lisboa's three ULS, and Odivelas gets the merged unit too. Everywhere else it equals `unit`.

Source

Decreto-Lei 102/2023 (ULS); prior ARS/ACES organization.

See Also

`pt_lookup()`, pt_registry

Examples

```
table(pt_health_codes$health_vintage, pt_health_codes$scheme)
```

pt_join *Add official codes and administrative context to a data frame*

Description

Matches a column of `data` against the pt_registry (via `pt_match()`) and appends the administrative columns that are well-defined at that level: codes, names of parent units, the NUTS regions of one revision, and health mappings of both vintages. A column is only appended when its value is constant within each matched unit. If you have data by municipio, for example, pt_join will add `uls_municipio` and not `uls`.

Usage

```
pt_join(
  data,
  by,
  fuzzy = FALSE,
  aliases = NULL,
  admin_vintage = NULL,
  nuts_vintage = NULL,
  quiet = FALSE
)
```

Arguments

`data` A data.frame.

`by` Which column of `data` identifies the unit, and what it contains. Forms:

- `"municipio"` – the column `municipio` in `data` holds municipality names or DTMN codes. Any keyword from the by registry works: `freguesia`/`dtmnfr`, `municipio`/`concelho`/`dtmn`, `distrito`/`dt`, `nuts1`-`nuts3` (`*_cod`), `uls`, `reg_saude`, `aces`, `uls_municipio`, `aces_municipio`.
- `c(municipio = "muni_name")` – similar to tidyverse join logic. This example assumes the municipio column in your data is called `muni_name`.
- `c(freguesia = "freg", municipio = "mun")` – first entry is the unit to match; further entries give context columns used to disambiguate. In this example, freguesia names scoped by municipio

`fuzzy`, `aliases`, `quiet`
Passed to `pt_match()`.

`admin_vintage` Which administrative vintage `data` is on: `"2025"`, `"2013"`, `"2011"`, `"2001"`, or `"auto"` to infer it from the `by` column. Leave it out for the newest (`"2025"`). Ignored at the health levels, which exist on the current map only. See *Vintages*.

`nuts_vintage` Which NUTS revision the NUTS columns are: `"2024"`, `"2013"` or `"2002"`. Leave it out for the newest (`"2024"`). See *Vintages*.

Value

data with the matched code column and administrative columns appended. Columns already present in data are not overwritten. The match report is attached and retrievable with pt_match_report().

Vintages

admin_vintage says which freguesia map data is on, and the codes and names appended are that map's. INE publishes freguesias on the 2013 map, so its data joins with admin_vintage = "2013" (or "auto") and keeps its 2013 codes; on the default, the current map, the uniões split in 2025 would not match. Health mappings are recorded on the current map, so on an earlier one they are reached through shared territory, as in pt_lookup(): a unit whose territory now lies in more than one ULS gets NA.

nuts_vintage says which NUTS revision the NUTS columns are, and at a NUTS level which revision by is matched against.

See Also

pt_match(), get_map() (which joins directly onto geometry)

Examples

```
df <- data.frame(concelho = c("Porto", "Lagoa (Acores)"), value = 1:2)
pt_join(df, by = c(municipio = "concelho"))

# Data on the 2013 map, as INE publishes it: the uniao of Conde and
# Gandarela (Guimaraes) was split in 2025, so its code only exists there
ine <- data.frame(dtmnfr = "030882", value = 3)
pt_join(ine, by = "dtmnfr", admin_vintage = "2013")

# The NUTS regions of the 2013 revision, under which Lisboa was PT17
pt_join(data.frame(concelho = "Lisboa"), by = c(municipio = "concelho"),
        nuts_vintage = "2013")
```

---

pt_lookup *Administrative, NUTS and health codes for one choice of each vintage*

Description

Assembles a wide lookup table from pt_registry, pt_nuts_codes and pt_health_codes. Each axis is chosen independently, because they are independent: the same freguesia set is published by INE under more than one NUTS revision, and a NUTS revision says nothing about which freguesias exist.

Usage

```
pt_lookup(
  level = "freguesia",
  admin_vintage = NULL,
  nuts_vintage = NULL,
  health_vintage = NULL
)
```

Arguments

`level`	Grain of the result: `"freguesia"` (default), `"municipio"`, `"distrito"`, `"nuts1"`, `"nuts2"`, `"nuts3"`, or the health levels `"reg_saude"`, `"uls"`, `"aces"`, `"uls_municipio"`, `"aces_municipio"`. Coarser levels return one row per unit with only the columns that are constant within it. A health level reads its own health geography unless `health_vintage` says otherwise (see *Vintages*).
`admin_vintage`	Which administrative unit set: `"2025"`, `"2013"`, `"2011"` or `"2001"`. Leave it out for the newest (`"2025"`).
`nuts_vintage`	Which NUTS revision: `"2024"`, `"2013"` or `"2002"`. Leave it out for the newest (`"2024"`); `NA` drops the NUTS columns.
`health_vintage`	Which health geography: `"2024"` or `"2022"`. Unlike the other two, leaving it out omits the health columns, except at a health `level`. ULS exist only in `"2024"` and ACES only in `"2022"`, so `level = "uls"` with `"2022"` returns that geography's ACES, and `level = "aces"` with `"2024"` its ULS (likewise for the `*_municipio` levels), with a warning.

Value

A data.frame, one row per unit of `level`.

Vintages

`admin_vintage` selects which administrative units exist:

- `"2025"` — 3259 freguesias, the current set (Lei 25-A/2025, in force from the 2025 autárquicas, which restored units aggregated in 2013).
- `"2013"` — 3092 freguesias (Lei 11-A/2013). **This is what INE still publishes**, including on indicators labelled "NUTS - 2024".
- `"2011"` — 4260 freguesias, the map the 2011 census used, in force from 2004 until the 2013 reform.
- `"2001"` — 4241 freguesias, the map the 2001 census used. 22 freguesias were created and 3 dissolved between the two censuses.

The pre-2013 maps are named after the census that recorded them, because neither was made by a single law. The 1991 census used an earlier map again (4208 freguesias in 305 municípios, before Odivelas, Trofa and Vizela were created in 1998), which geopt does not have.

`nuts_vintage` selects the NUTS revision: `"2024"` (9 NUTS II, 26 NUTS III), `"2013"` (7 / 25) or `"2002"` (7 / 30).

health_vintage selects the health geography: "2024" (regiões de saúde + ULS) or "2022" (the former ARS + ACES). Health columns are omitted unless asked for, or unless level is a health level: those read the geography their units belong to, "2024" for "reg_saude", "uls" and "uls_municipio", and "2022" for "aces" and "aces_municipio".

Codes are vintage-local

A DTMNFR identifies a unit *within its own vintage only*. None of the 302 freguesias restored in 2025 reused its pre-2013 code, so comparing a code across vintages — or joining data of one vintage onto codes of another — silently mismatches. Use pt_crosswalk() to convert, and pt_recode() to carry values across.

See Also

pt_registry, pt_nuts_codes, pt_health_codes, pt_crosswalk(), pt_match()

Examples

```
head(pt_lookup())

# the combination INE actually publishes: 2013 freguesias, 2024 NUTS
head(pt_lookup(admin_vintage = "2013", nuts_vintage = "2024"))

# one row per municipio, with its 2002 NUTS grouping
head(pt_lookup("municipio", nuts_vintage = "2002"))

# one row per ULS (the 2024 health geography, without asking)
head(pt_lookup("uls"))
```

---

pt_match — *Match place names to official codes*

Description

Resolves a character vector of Portuguese administrative unit names to their official codes using the pt_registry table. Matching is exact on normalized names (see pt_normalize()), optionally preceded by a user-supplied alias lookup and followed by an opt-in fuzzy pass.

Usage

```
pt_match(
  x,
  level = "municipio",
  to = NULL,
  context = NULL,
  admin_vintage = NULL,
  nuts_vintage = NULL,
```

```
  health_vintage = NULL,
  fuzzy = FALSE,
  max_dist = 2,
  aliases = NULL,
  historic = TRUE,
  quiet = FALSE
)
```

Arguments

x	Character vector of names to match. Codes are automatically padded to the correct length, if needed.
level	Which units x refers to: "freguesia", "municipio" (default), "distrito", "nuts1", "nuts2", "nuts3", "uls", "reg_saude", "aces", "uls_municipio", "aces_municipio". "uls"/"reg_saude" are the 2024 health reform; "aces" is the organization it superseded. The *_municipio levels merge the units that share a municipality, and are matched by the merged name ("ULS Santo António + São João").
to	Column of pt_lookup() to return, at the vintages requested below. Defaults to the level's own code column (dtmnfr, dtmn, dt, nuts*_cod, ...). A column defined only at a finer level — "dtmnfr" for a municipio — yields NA.
context	Optional character vector (length 1 or length(x)) naming a parent unit to scope the match. Any vintage of any parent scheme is accepted, so INE's own parent column works whichever nomenclature that indicator was published under.
admin_vintage	Which administrative vintage to match against and return codes for: "2025", "2013", "2011", "2001", or "auto" to infer it from x. Leave it out for the newest ("2025"). See *Vintages*.
nuts_vintage	Which NUTS revision to is expressed in: "2024", "2013" or "2002". Also the vintage matched at level = "nuts1"/"nuts2"/"nuts3". Leave it out for the newest ("2024").
health_vintage	Which health geography to is expressed in: "2024" or "2022". Leave it out for the newest ("2024"). Ignored at level = "uls" and "aces" (and their *_municipio forms), which each exist in one health geography only; a value that isn't a health vintage at all is still an error.
fuzzy	Logical. Enable the fuzzy fallback pass? Default FALSE.
max_dist	Maximum string distance for a fuzzy match. Default 2.
aliases	Optional named character vector mapping variant spellings to either canonical names or codes, e.g. c("VN Gaia" = "Vila Nova de Gaia"). Checked before exact matching; names are compared in normalized form.
historic	Logical. Also match names from the level's other vintages? Default TRUE. Set FALSE to accept only names in use at admin_vintage — useful when validating that a file really is that vintage.
quiet	Logical. Suppress warnings/messages about ambiguous, unmatched, and fuzzy-matched entries? Default FALSE.

Value

Character vector of codes (same length as x), with a "report" attribute retrievable via pt_match_report().

Ambiguity and context

Many names are not unique (~200 freguesia names are reused; the municipality "Lagoa" exists in Faro and in Sao Miguel). Ambiguous names return NA with a warning listing the candidates. To resolve them, either:

- pass context: a parent unit name per element (municipio for freguesias; distrito/ilha, NUTS region, or "Acores"/"Madeira" for municipios), or
- embed a qualifier in the name itself: "Lagoa (Acores)", "Lagoa (Faro)", INE-style "Calheta (R.A.M.)", "Lagoa [R.A. Acores]" and dash-separated forms like "Calheta - RAA" or "Calheta-R.A.A" are all parsed automatically. A dash is only read as a separator when what follows it is a recognized region — ordinary names containing dashes ("Albergaria-a-Velha") are never split.

A qualifier is only read as one when it is not already part of an official name. Many freguesias end in a parenthetical of their own — "Bouro (Santa Maria)", "Uniao das freguesias de Braga (Maximinos, Se e Cividade)" — and those are matched whole, before any split is attempted.

Spelling tolerance

Matching ignores case, accents, and punctuation, and expands S./Sta./ Sto.. If that still does not match, a second pass ignores Portuguese particles, so official designations and their working short forms both resolve: "ULS da Guarda" → "ULS Guarda", "ULS Trás-os-Montes Alto Douro" → "ULS Trás-os-Montes e Alto Douro". Exact spellings always win; the report's method column shows "loose" when this pass was needed.

A few units accept a second official spelling, so that data labelled either way resolves: "Vila da Praia da Vitoria" for Praia da Vitória, and "Calheta" for the Azorean Calheta (which pt_registry stores as "Calheta de Sao Jorge"). Because Madeira also has a Calheta, a bare "Calheta" is therefore *ambiguous* and returns NA with a warning — qualify it as "Calheta (R.A.A.)" or "Calheta (Madeira)".

Historic names

Administrative reforms rename and dissolve units, so names in older census and administrative files often have no current counterpart. With historic = TRUE (the default) names from *every* vintage of the level are matched, returning the code of the unit covering that territory in the requested vintage; the report's method column shows "historic". A name found in the 2001 map is only used when no later vintage has it: most 2001 names were still in use in 2011, and for the few freguesias that lost territory in between, the 2011 sense is the one that matches a single unit today. This is one mechanism, not a freguesia special case — it is also what makes "Minho-Lima" resolve at level = "nuts3", where the current name is "Alto Minho".

Live names win outright. A name still in use resolves to the unit bearing it now, whatever it also meant in the past. Where a predecessor was later split between several current units the name is genuinely ambiguous and returns NA — pass a context or use codes. Historic names are matched exactly and particle-loosely, never fuzzily.

A freguesia renamed while its map was in force keeps its earlier names as names of that vintage, not as historic ones. The 2013 freguesias are named as they were last, in CAOP 2024 and in INE's tables, and 25 of them were renamed by law before that. `"Buarcos"`, the name of `060520` until 2015, therefore matches it with `admin_vintage = "2013"` even when `historic = FALSE`.

Vintages

The three axes are independent and each has its own argument. `admin_vintage` chooses which units exist *and which vintage's codes come back*; `nuts_vintage` and `health_vintage` choose the nomenclature that NUTS and health columns are expressed in. See `pt_lookup()`.

`admin_vintage` defaults to the current set, which is what you want for joining old data onto today's map. To keep the data on its own vintage instead — to map it against boundaries of the era — name that vintage:

```
pt_match(c("Sé", "Maximinos"), level = "freguesia", context = "Braga")
#> [1] "030364" "030364"   # the union covering both today

pt_match(c("Sé", "Maximinos"), level = "freguesia", context = "Braga",
         admin_vintage = "2011")
#> [1] "030352" "030324"   # the units they were
```

Codes are vintage-local. None of the 302 freguesias restored in 2025 reused its pre-2013 code, even if it has the same name and same boundaries, so a code is only accepted against the vintage it belongs to. `pt_match("030882", level = "freguesia")` is `NA`, because that is a 2013 code; with `admin_vintage = "2013"` it resolves.

`admin_vintage = "auto"` infers the vintage from the input and says which it chose. This is the setting for INE data, which is published on the 2013 freguesia set even under the 2024 NUTS revision — so 134 units that were disaggregated in 2025 are ambiguous against the current registry and exact against their own.

INE prefixes a city's pre-2013 parishes with the city (`"Braga (Sé)"`) but names the same kind of unit plainly elsewhere (`"Sé"` in Lisboa). All the spellings resolve — `"Braga (Sé)"`, `"Sé (Braga)"`, and `"Sé"` with `context = "Braga"` — because a bare name plus a context is recombined into the parenthesised forms and retried, once the name as given has failed. A trailing parenthetical never *displaces* a `context` you passed: where it is part of the official name rather than a qualifier, the context still applies.

Fuzzy matching

With `fuzzy = TRUE` (requires the `stringdist` package), elements that fail exact matching are compared against candidate names by optimal string alignment distance. A fuzzy match is accepted only when the best candidate is within `max_dist` and strictly better than the second-best candidate; every accepted fuzzy match is reported via message. Fuzzy matching is most reliable with a `context`, which shrinks the candidate pool.

See Also

`pt_match_report()`, `pt_normalize()`, pt_registry

Examples

```
pt_match(c("Porto", "lisboa", "S. Joao da Madeira"))
pt_match(c("Lagoa (Faro)", "Lagoa (Acores)"))
pt_match("Pinheiro", level = "freguesia", context = "Guimarães")
```

---

pt_match_report — *Retrieve the match report from a pt_match result*

Description

Retrieve the match report from a pt_match result

Usage

```
pt_match_report(x)
```

Arguments

x	The return value of `pt_match()`.

Value

A data.frame with one row per input: `input`, `matched_name`, `code`, `method` (`"exact"`, `"loose"` — matched once Portuguese particles were ignored, `"alias"`, `"historic"` — matched a name from a vintage other than the one requested, such as a pre-2013 name, `"fuzzy"`, `"code"`, `"ambiguous"`, `"context_unrecognized"` — the `context` is not a unit geopt knows, `"unmatched"`, `"empty"`), and `dist` (fuzzy matches only).

Examples

```
m <- pt_match(c("Porto", "Lagoa"))
pt_match_report(m)
```

---

pt_normalize — *Normalize Portuguese place names for matching*

Description

Lowercases, expands common abbreviations (`S.` -> `Sao`, `Sta.` -> `Santa`, `Sto.` -> `Santo`), strips accents, and collapses punctuation and whitespace, so that variant spellings of the same unit produce identical keys: `"S. Jorge"`, `"são jorge"`, and `"Sao-Jorge"` all normalize to `"sao jorge"`.

Usage

```
pt_normalize(x)
```

Arguments

x	Character vector of names.

Details

This is the normalization used internally by `pt_match()`; it is exported so the same rules can be applied to arbitrary user columns.

Value

Character vector of normalized names.

Examples

```
pt_normalize(c("S. Jorge", "SÃO JORGE", "Sao-Jorge"))
```

---

pt_nuts_codes *NUTS classification of Portuguese municípios, every revision*

Description

One row per município per NUTS revision. NUTS revisions are set at European level and are **independent of the administrative vintage**: INE publishes the same 3092-freguesia set under both the 2013 and the 2024 revision.

Usage

```
pt_nuts_codes
```

Format

A data.frame with 924 rows and columns:

nuts_vintage `"2002"`, `"2013"` or `"2024"`.

dtmn 4-digit município code.

nuts1_cod, nuts1 NUTS I code and name.

nuts2_cod, nuts2 NUTS II code and name.

nuts3_cod, nuts3 NUTS III code and name.

Details

Keyed on `dtmn` because every NUTS revision Portugal has had is a regrouping of the same 308 municípios — stable since the 1998 creation of Trofa, Vizela and Odivelas. NUTS is therefore a function of `(dtmn, nuts_vintage)` and not a freguesia-level attribute at all, which is why 308 rows per revision is the whole fact.

Revisions

"2024" 9 NUTS II, 26 NUTS III. Splits the former Lisboa region into Grande Lisboa, Península de Setúbal and Oeste e Vale do Tejo.

"2013" 7 NUTS II, 25 NUTS III, the CIM/AM-based sub-regions.

"2002" 7 NUTS II, 30 NUTS III. PT17 is "Lisboa" here and "Área Metropolitana de Lisboa" from 2013.

Source

INE / Eurostat NUTS nomenclature. NUTS II 2002 names come from INE's own indicator dimension labels.

See Also

pt_lookup(), pt_registry

Examples

```
# how a municipio's NUTS III changed across revisions
pt_nuts_codes[pt_nuts_codes$dtmn == "1106", c("nuts_vintage", "nuts3")]
```

---

pt_pad_code *Restore leading zeros in administrative codes*

Description

Spreadsheet round-trips commonly convert codes like "0115" to the number 115. This pads digit codes back to their official width: 6 for DTMNFR (freguesia), 4 for DTMN (municipio), 2 for DT (distrito/ilha).

Usage

```
pt_pad_code(x, level = "municipio")
```

Arguments

x	Vector of codes (character or numeric).
level	One of "freguesia"/"dtmnfr", "municipio"/"dtmn", "distrito"/"dt".

Value

Character vector of zero-padded codes. Non-digit entries are returned unchanged.

Examples

```
pt_pad_code(c(115, 1106), "municipio")
```

pt_recode *Convert freguesia-level data between code vintages*

Description

Portugal's freguesias were aggregated into "uniões de freguesias" in 2013 and partially de-aggregated in 2025; the maps before 2013 are finer still. This function converts freguesia-level data between four vintages: `"2001"` (the map the 2001 census used, 4241 freguesias), `"2011"` (the map the 2011 census used, 4260), `"2013"` (the uniões, 3092) and `"2025"` (the current map, 3259). The correspondence is derived from shared territory rather than stored, so any pair of vintages works; it handles the value semantics explicitly:

Usage

```
pt_recode(
  data,
  by = "dtmnfr",
  from,
  to = NULL,
  values = NULL,
  aggregate = NULL,
  split = c("error", "replicate", "weight"),
  weights = NULL,
  quiet = FALSE
)
```

Arguments

`data`	A data.frame with freguesia-level rows.
`by`	Name of the column in `data` holding the freguesia codes (DTMNFR, zero-padding is repaired). Default `"dtmnfr"`.
`from`	Which vintage the codes in `data` belong to (required): `"2025"` (the current map), `"2013"` (the uniões, in force 2013–2025), `"2011"` (the map the 2011 census used, in force 2004–2013) or `"2001"` (the map the 2001 census used). The 1991 census used an earlier map again, which geopt does not have.
`to`	Target vintage, one of the same four. Leave it out for the newest (`"2025"`); when `from` is already the newest, `to` must be given.
`values`	Character vector naming the numeric value columns to convert. Other columns are carried along: replicated on splits, kept on merges when constant within the group (otherwise set to `NA`). Columns called `dtmnfr` or `freguesia` are not carried: the result's own are the target vintage's.
`aggregate`	Function (or named list of functions, one per value column) used to combine values on merges, e.g. `sum`. Required when any merge occurs and `values` is non-empty.
`split`	How to distribute values on splits: `"error"` (default), `"replicate"`, or `"weight"`.

weights	For split = "weight": "area" (proportional to the area of the newest vintage's freguesias) or a named numeric vector keyed by the target dtmnfr (e.g. population).
quiet	Logical. Suppress informational warnings? Default FALSE.

Details

- **finer → coarser (merges),** e.g. 2025 → 2013, 2011 → 2025: freguesias collapse into a shared unit. Value columns must be aggregated, and only values for which aggregation is meaningful may be converted (raw counts can be summed; rates and medians cannot) — so aggregate is required whenever a merge occurs. If only some constituents of a target are present in data, the aggregate covers the present ones and a warning reports the partial groups.
- **coarser → finer (splits),** e.g. 2013 → 2025, 2025 → 2011, 2001 → 2011 (22 freguesias created between the two censuses): a unit's value cannot be exactly disaggregated. You must opt in to a strategy: split = "replicate" copies the value to every constituent; split = "weight" distributes it proportionally to weights ("area", only when splitting towards the newest vintage, or a named vector keyed by the target dtmnfr, e.g. population). The default ("error") refuses.

Codes not present in the relevant crosswalk (including current freguesias created after 2011, which have no pre-2013 predecessor) convert to NA with a warning. For "2001", the codes the 2001 census areas use for Vale de Amoreira (090756) and Moita (100117) are accepted alongside INE's (090804, 101003).

Value

A data.frame with the target vintage's freguesia code (dtmnfr) and name (freguesia), followed by the carried and converted columns. The column names are the same whichever vintage you convert to, as in pt_lookup().

See Also

pt_crosswalk(), pt_registry, pt_join()

Examples

```
# 2013-era data onto the current map: sum-able counts, split by area
old <- data.frame(dtmnfr = c("010121", "010109"), cases = c(30, 10))
new <- pt_recode(old, from = "2013", values = "cases",
                 split = "weight", weights = "area")
get_map("freguesia", data = new, by = "dtmnfr")

# Current data backwards to 2013 units (counts: aggregate = sum). Conde and
# Gandarela (Guimarães) were one união from 2013 until 2025.
now <- data.frame(dtmnfr = c("030895", "030897"), n = c(12, 7))
pt_recode(now, from = "2025", to = "2013", values = "n", aggregate = sum)

# 2011-census freguesia counts onto the current map: many old freguesias
# merged into today's uniões, so aggregate = sum
old <- data.frame(dtmnfr = c("010101", "010102"), pop = c(500, 800))
pt_recode(old, from = "2011", to = "2025", values = "pop",
```

```
          aggregate = sum)

# a 2001-census rate onto the 2011 map: Agualva-Cacém became four
# freguesias, and a rate is copied to each
pt_recode(data.frame(dtmnfr = "111101", rate = 12.5), from = "2001",
          to = "2011", values = "rate", split = "replicate")
```

`pt_registry` *Portuguese administrative units, every vintage*

Description

One row per freguesia per administrative vintage. "Administrative vintage" means *which units exist*, which changes by Portuguese law and is unrelated to the NUTS revision (see pt_nuts_codes) or the health geography (see pt_health_codes).

Usage

```
pt_registry
```

Format

A data.frame with 14852 rows and columns:

admin_vintage `"2001"`, `"2011"`, `"2013"` or `"2025"`.

dtmnfr 6-digit freguesia code, unique within the vintage.

freguesia Official designation. For 2013, the last one the freguesia had on that map.

designacao_simplificada Without the `"União das freguesias de"` prefix. Official for 2025; derived by the same rule for 2013, because DGT's list for that map leaves 32 uniões unsimplified. Equal to `freguesia` for 2001 and 2011, which predate uniões.

dtmn 4-digit município code (the first four of `dtmnfr`).

municipio Município name.

dt, distrito_ilha District or island.

Details

`dtmn` is denormalised onto each row, so a vintage's set of municípios is the distinct `dtmn` values within it.

Vintages

`"2025"` 3259 freguesias. Lei 25-A/2025, in force from the 2025 autárquicas, which restored units aggregated in 2013.

`"2013"` 3092 freguesias. Lei 11-A/2013. **This is the set INE still publishes**, including on indicators labelled "NUTS - 2024" — the label refers to the NUTS revision, not to the freguesia set. Each freguesia has the last name it had on that map, as in CAOP 2024 and INE's tables. 25 were renamed by law while the map was in force (`"Buarcos"` became `"Buarcos e São Julião"` in 2015), and `pt_match()` recognizes the earlier names too.

`"2011"` 4260 freguesias, the map the 2011 census used, in force from 2004 until the 2013 reform.

`"2001"` 4241 freguesias, the map the 2001 census used. Between the two censuses 22 freguesias were created and 3 dissolved.

The pre-2013 maps are named after the census that recorded them, because neither was made by a single law; the later ones after their reform.

Codes are vintage-local

A DTMNFR identifies a unit *within its own vintage only*. None of the 302 freguesias restored in 2025 reused its pre-2013 code, so comparing codes across vintages, or joining data of one vintage onto codes of another, silently mismatches. Convert with `pt_crosswalk()`; carry values across with `pt_recode()`.

Source

Carta Administrativa Oficial de Portugal (DGT): CAOP 2025, and CAOP 2024 for the 2013 names, which INE's code list for that map matches name for name; Lei 11-A/2013 and Lei 25-A/2025; INE's freguesia code lists for the 2001 and 2011 censuses.

See Also

`pt_lookup()`, pt_nuts_codes, pt_health_codes, `pt_crosswalk()`

Examples

```
table(pt_registry$admin_vintage)

# the set INE publishes today
head(pt_registry[pt_registry$admin_vintage == "2013", ])
```

Index

∗ **datasets**
boundaries, 2
pt_censos_vars, 19
pt_health_codes, 21
pt_nuts_codes, 31
pt_registry, 35

boundaries, 2
box_labels_echarts, 5

frame_data_echarts, 6

GeoClient, 7, *11, 12*, *14*
get_censos, 10
get_censos(), *7*, *14*, *19, 20*
get_map, 13
get_map(), *4–7*, *12*, *15–19*, *24*
grDevices::hcl.colors(), *18*

island_layout, 15
island_layout(), *13, 14*, *19*

plot_pt, 17
plot_pt(), *14–17*
pt_aces *(*boundaries*)*, 2
pt_aces_municipio *(*boundaries*)*, 2
pt_censos_vars, *11, 12*, 19
pt_crosswalk, 20
pt_crosswalk(), *12*, *26*, *34*, *36*
pt_distritos *(*boundaries*)*, 2
pt_freguesias *(*boundaries*)*, 2
pt_health_codes, *4*, *13*, 21, *24*, *26*, *35, 36*
pt_join, 23
pt_join(), *4*, *13*, *19*, *34*
pt_lookup, 24
pt_lookup(), *4*, *22*, *24*, *27*, *29*, *32*, *34*, *36*
pt_match, 26
pt_match(), *11, 12*, *14*, *23, 24*, *26*, *30, 31*, *36*
pt_match_report, 30
pt_match_report(), *24*, *28, 29*
pt_municipios *(*boundaries*)*, 2
pt_normalize, 30
pt_normalize(), *26*, *29*
pt_nuts1 *(*boundaries*)*, 2
pt_nuts2 *(*boundaries*)*, 2
pt_nuts3 *(*boundaries*)*, 2
pt_nuts_codes, *21*, *24*, *26*, 31, *35, 36*
pt_pad_code, 32
pt_recode, 33
pt_recode(), *12*, *14*, *21*, *26*, *36*
pt_reg_saude *(*boundaries*)*, 2
pt_registry, *2*, *21–24*, *26*, *29*, *32*, *34*, 35
pt_uls *(*boundaries*)*, 2
pt_uls_municipio *(*boundaries*)*, 2