# Package: latlon2map (via r-universe)

September 18, 2024

**Title** Facilitates matching lat/lon data with administrative units and other geographic shapes

**Version** 0.0.0.9008

**Description** Facilitates matching lat/lon data with administrative units and other geographic shapes

**License** GPL-3

**Imports** config, golem, shiny, attempt, DT, glue, htmltools, readr, fs, magrittr, sf, tibble, ggplot2, dplyr, htmlwidgets, xml2, purrr, countrycode, cli

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.3

**Suggests** leaflet, knitr, rmarkdown

**VignetteBuilder** knitr

**Roxygen** list(markdown = TRUE)

**Depends** R (>= 2.10)

**URL** `https://giocomai.github.io/latlon2map/`

**Repository** https://giocomai.r-universe.dev

**RemoteUrl** https://github.com/giocomai/latlon2map

**RemoteRef** HEAD

**RemoteSha** 4cedceeba3131080f273eac9a3951291c1198851

## Contents

## `ll_app` *Run the Shiny Application*

### Description

Run the Shiny Application

### Usage

```
ll_app(max_file_size = 100, ll_folder_path = NULL, ...)
```

### Arguments

| | |
|---|---|
| `max_file_size` | Maximum file size to accept for upload expressed in MB, defaults to 100. |
| `ll_folder_path` | If given, sets the folder to use for caching, corresponds to `ll_set_folder()`. Useful e.g. for Docker deployments. Defaults to NULL. |
| `...` | A series of options to be used inside the app. |

## ll_bbox *Provide a bounding box with a consistent, user given ratio*

### Description

This is useful in particular to make geom_sf()-based ggplots with consistent aspect ratio.

### Usage

```
ll_bbox(sf, ratio = "4:3")
```

### Arguments

| | |
|---|---|
| `sf` | An sf object. |
| `ratio` | Defaults to "4:3". A chacters string, in the form of e.g. "4:3" or "16:9" or "1:1" (other values possible) |

### Value

A bounding box vector, same as with `sf::st_bbox()`, but with the given ratio set and compatible with crs 4326.

### Examples

```
## Not run:
# The following two graphs will have same 4:3 aspect ratio
ll_set_folder("~/R/")
library("ggspatial")

sf_location <- ll_get_nuts_it(name = "Palmanova", level = "lau", resolution = "low")

ggplot() +
  annotation_map_tile(type = "osm", zoomin = -1, cachedir = fs::path(ll_set_folder(), "ll_data")) +
   geom_sf(data = sf::st_as_sfc(ll_bbox(sf_location)), fill = NA, color = NA) +
   geom_sf(
     data = sf_location,
     colour = "darkred",
     size = 2,
     fill = NA,
     alpha = 0.8
   )


sf_location <- ll_get_nuts_it(name = "Pinzolo", level = "lau", resolution = "low")

ggplot() +
  annotation_map_tile(type = "osm", zoomin = -1, cachedir = fs::path(ll_set_folder(), "ll_data")) +
   geom_sf(data = sf::st_as_sfc(ll_bbox(sf_location)), fill = NA, color = NA) +
   geom_sf(
     data = sf_location,
     colour = "darkred",
     size = 2,
     fill = NA,
     alpha = 0.8
   )

## End(Not run)
```

---

`ll_create_folders` *Create folders to store geographic data*

---

## Description

Create folders to store geographic data

## Usage

```
ll_create_folders(
  geo,
  level,
  resolution,
  year,
  file_type = c("shp", "zip", "rds")
)
```

## Arguments

| | |
|---|---|
| `geo` | The geographic unit of reference as a two-letter code |
| `level` | E.g. NUTS0, NUTS1, or county, state, ecc. |
| `resolution` | Either resolution level as given by the data distributor, or generic such as "high", "low", or "default |
| `file_type` | By defaults, it creates folder for zip, shp, and rds files. |

---

`ll_export_sf_to_kml` *Export sf objects into kml file that can be used with Google Earth, Google Maps, etc.*

---

## Description

Attention: this function requires libkml.

**Usage**

```
ll_export_sf_to_kml(
  sf,
  path,
  name = NULL,
  keep_other_columns = TRUE,
  description = NULL,
  label_text = NULL,
  label_font = "Roboto Sans, Noto Sans, Helvetica",
  label_size = "24pt",
  label_placement = "m",
  label_scale = NULL,
  line_colour = "#ffffffff",
  line_width = "3px",
  icon_url = "",
  icon_colour = "#000000ff",
  icon_scale = NULL,
  fill_colour = NULL
)
```

**Arguments**

| | |
|---|---|
| `sf` | An object of class `sf` |
| `path` | Path where to save the .kml output. |
| `name` | Column to be used for names. |
| `keep_other_columns` | Logical, defaults to TRUE. If you don't want to keep in the output data columns present in the original `sf` object, set this to FALSE. |
| `description` | Column to be used for description. |
| `label_text` | Column to be used as label text. Defaults to NULL. Corresponds to "LABEL" element in OGR. |
| `label_font` | Font family to be used for the font. Defaults to "Roboto Sans, Noto Sans, Helvetica" |
| `label_size` | Size of the label. Defaults to "24pt" |
| `label_placement` | Defaults to "m" (centre and middle-aligned). For more options, check: https://gdal.org/user/ogr_feature_s |
| `label_scale` | Scale of label. Defaults to NULL. If given, changes label size (e.g. 1 = default, 2 = twice as big, 0.5, half as big, etc.) |
| `line_colour` | Defaults to "#ffffffff" (i.e. white, with 100% opacity). Line corresponds to "PEN" in OGR. Accepts 8-digit hex codes to include transparency. |
| `line_width` | Defaults to "3pt". Line corresponds to "PEN" in OGR. Besides pt (points), other acceptable units are `g`: Map Ground Units (whatever the map coordinate units are), `px` Pixels, `pt` Points (1/72 inch), `mm` Millimeters, `cm` Centimeters, `in` Inches. |
| `icon_url` | Defaults to "" for no URL. Corresponds to "SYMBOL" in OGR. In case of wrong inputs, Google Earth may show you an ugly yellow pushpin instead |

| | |
|---|---|
| | (i.e. default to http://maps.google.com/mapfiles/kml/pushpin/ylw-pushpin.png). Available icons offered by Google available at this link: http://kml4earth.appspot.com/icons.html |
| `icon_colour` | Defaults to "#000000ff" (i.e. black, with 100% opacity). |
| `icon_scale` | Defaults to NULL. If given, changes icon size (e.g. 1 = default, 2 = twice as big, 0.5, half as big, etc.) |
| `fill_colour` | Defaults to NULL. Fill corresponds to "BRUSH" in OGR. If given, colour to be used for filling polygons. |

### Details

Attention: label styling is not currently functional, likely due to issues in passing arguments to libkml. In order to change label size, use label_scale, which directly edits the xml file.

For further details on the exact meaning of each of the parameters, please consult the documentation of OGR (used by GDAL to pass parameters to .kml): https://gdal.org/user/ogr_feature_style.html

## `ll_find_file` *Find file names. Mostly used internally*

### Description

Find file names. Mostly used internally

### Usage

```
ll_find_file(geo, level, resolution, year, name = "abl", file_type = "rds")
```

### Arguments

| | |
|---|---|
| `name` | Name of specific dataset being downloaded. Defaults to abl, i.e. administrative boundary line |
| `file_type` | |

## `ll_find_pop_centre` *Find the population-weighted centre of a municipality*

### Description

Find the population-weighted centre of a municipality

## Usage

```
ll_find_pop_centre(
  sf_location,
  sf_population_grid,
  power = 2,
  join = sf::st_intersects,
  adjusted = FALSE
)
```

## Arguments

| | |
|---|---|
| `power` | Defaults to 2. To give more weight to cells with higher population density, raise the number of residents by the power of. |
| `join` | Defaults to sf::st_intersects. |
| `adjusted` | If adjusted is set to TRUE, join is ignored. The population of cells along the boundary line are weighted by the share of the cell included within the border. |

## Examples

```
ll_set_folder("~/R/")
name <- "Pinzolo"
sf_location <- ll_get_nuts_it(name = name, level = "lau", resolution = "high")

lau_grid_name_temp <- stringr::str_c(name, "_lau_high-st_intersects")

sf_location_grid <- ll_get_population_grid(
  match_sf = sf_location,
  match_name = lau_grid_name_temp,
  match_country = "IT",
  join = sf::st_intersects
)


pop_centre <- ll_find_pop_centre(
  sf_location = sf_location,
  sf_population_grid = sf_location_grid,
  power = 2
)
```

---

`ll_get_adm_ocha` *Get administrative boundary lines from OCHA database*

---

## Description

Source: https://data.humdata.org/

### Usage

```
ll_get_adm_ocha(
  geo,
  level = 0,
  match_name = NULL,
  source_url = NULL,
  silent = FALSE
)
```

### Arguments

| | |
|---|---|
| `geo` | A twe letter country code, such as "IT" for Italy and "DE" for Germany |
| `match_name` | A name to be used for local caching if a subset of the grid is used. It is the responsibility of the user to keept it consistent. If not given, data are not cached locally. |
| `source_url` | A direct link to the zipped version of the csv file in the original database, if automatic download with the country code does not work. For example, for Italy this would be "https://data.humdata.org/dataset/0eb77b21-06be-42c8-9245-2edaff79952f/resource/1e96f272-7d86-4108-b4ca-5a951a8b11a0/download/population_ita_2019-07-01.csv.zip" |
| `silent` | |

### Examples

```
if (interactive) {
  ll_get_adm_ocha(geo = "UA", level = 3)
}
```

---

## `ll_get_electoral_districts_it` *Get Italian electoral districts (CC-BY Istat)*

---

### Description

2022 / WGS 84 / UTM zone 32N

### Usage

```
ll_get_electoral_districts_it(
  name = NULL,
  level = "Circoscrizioni_Camera",
  year = 2022,
  silent = FALSE,
  no_check_certificate = FALSE
)
```

## Arguments

| | |
|---|---|
| `level` | Defaults to "Circoscrizioni_Camera". Valid values: • "Circoscrizioni_Camera": Basi geografiche delle circoscrizioni elettorali - Camera dei deputati • "Regioni_Senato": Basi geografiche delle circoscrizioni elettorali - Senato della Repubblica • "CAMERA_CollegiPLURINOMINALI_2020": Basi geografiche dei collegi elettorali plurinominali - Camera dei deputati • "CAMERA_CollegiUNINOMINALI_2020": Basi geografiche dei collegi elettorali uninominali - Camera dei deputati • "SENATO_CollegiPLURINOMINALI_2020": Basi geografiche dei collegi elettorali plurinominali - Senato della Repubblica • "SENATO_CollegiUNINOMINALI_2020": Basi geografiche dei collegi elettorali uninominali - Senato della Repubblica • "UT_Collegi2020": Basi geografiche delle unità territoriali che formano i collegi elettorali (comuni e aree sub-comunali, limitatamente ai comuni di Torino, Genova, Milano, Roma, Napoli e Palermo con territorio ripsrtito su più di un collegio). Geografia comunale vigente alla data della pubblicazione |
| `year` | Defaults to 2022 (latest available). Currently no other year accepted. |
| `no_check_certificate` | Logical, defaults to TRUE. Enable only if certificate issues, and if you are aware of the security implications. |

## Details

Column names metadata:

- COD_REG Codice della regione/circoscrizione elettorale del Senato della Repubblica
- DEN_REG Denominazione della regione amministrativa/circoscrizione elettorale Senato della Repubblica
- COD_PRO Codice della provincia
- DEN_P_CM Denominazione della provincia o città metropolitana
- COD_CM Codice della città metropolitana
- PRO_COM Codice del comune
- DEN_COM Denominazione del comune
- CAP_DEN Denominazione del capoluogo di provincia o città metropolitana
- POP_2011 Popolazione - Censimento 2011
- ASC_COD Codice concatenato comune e area sub-comunale
- ASC_COD1 Codice progressivo area sub-comunale
- ASC_COD2 Codice alfanumerico dell'area sub-comunale attribuito dal comune
- ASC_NOME Denominazione dell'area sub-comunale
- ASC_TIPO Tipologia di area-sub-comunale

- CIRC_COD Codice della circoscrizione elettorale della Camera dei deputati
- CIRC_DEN Denominazione della circoscrizione elettorale della Camera dei deputati
- CU20_COD Codice del collegio elettorale uninominale della Camera dei deputati
- CP20_COD Codice del collegio elettoraleplurinominale della Camera dei deputati
- SU20_COD Codice del collegio elettorale uninominale del Senato della Repubblica
- SP20_COD Codice del collegio elettorale plurinominale del Senato della Repubblica
- CU20_DEN Denominazione del collegio elettorale uninominale della Camera dei deputati
- CP20_DEN Denominazione del collegio elettorale plurinominale della Camera dei deputati
- SU20_DEN Denominazione del collegio elettorale uninominale del Senato della Repubblica
- SP20_DEN Denominazione del collegio elettorale plurinominale del Senato della Repubblica
- CU20_C1 Sigla del collegio elettorale uninominale della Camera dei deputati
- CP20_C1 Sigla del collegio elettorale plurinominale della Camera dei deputati
- SU20_C1 Sigla del collegio elettorale uninominale del Senato della Repubblica
- SP20_C1 Sigla del collegio elettorale plurinominale del Senato della Repubblica

## Examples

```
ll_set_folder(fs::path(fs::path_home_r(), "R"))
ll_get_electoral_districts_it()
ll_get_electoral_districts_it(name = "Lombardia 2")
ll_get_electoral_districts_it() %>% ggplot2::ggplot() + ggplot2::geom_sf() + ggplot2::labs(title = "Circoscrizion
ll_get_electoral_districts_it(level = "SENATO_CollegiUNINOMINALI_2020") %>% ggplot2::ggplot() + ggplot2::geom_sf
```

---

`ll_get_gadm` *Get administrative boundaries*

---

## Description

Source: https://gadm.org/

## Usage

```
ll_get_gadm(geo, level = 0, version = "4.1")
```

## Arguments

| | |
|---|---|
| `geo` | Three letter country codes. If a two letter country code is given, it will tentatively be converted to a three-letter country code. Check consistency. |
| `level` | Defaults to 0. Available labels, depending on data availability for the specific country, between 0 and 3. |
| `version` | Defaults to "4.0". Untested with others. |

### Value

An `sf` object

### Examples

```
ll_get_gadm(geo = "UKR", level = 2)
```

---

## `ll_get_lau_eu` *Gets local administrative units from Eurostat's website*

---

### Description

Source: https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/lau#lau18

### Usage

```
ll_get_lau_eu(
  gisco_id = NULL,
  name = NULL,
  year = 2021,
  silent = FALSE,
  lau_sf = NULL,
  fallback = TRUE
)
```

### Arguments

| | |
|---|---|
| `gisco_id` | Gisco identifier of the relevant administrative unit. If given, takes precedence over name. |
| `name` | Name of the local administrative unit in the local language. Use gisco_id whenever possible, as names of local administrative units are not unique, e.g. there are 11 "Neuenkirchen" in the dataset. If both `name` and `gisco_id` are NULL, then it returns all municipalities. |
| `year` | Year of mapping, defaults to most recent (2021). Available starting with 2011. |
| `silent` | Defaults to FALSE. If TRUE, hides copyright notice. Useful e.g. when using this in reports or in loops. The copyright notice must still be shown where the final output is used. |
| `lau_sf` | sf object, exactly such as the one that would be returned by `ll_get_lau_eu()`. Used to speed-up computation when bulk processing. |

### Value

European LAU in sf format

## Examples

```
ll_set_folder("~/R/")
ll_get_lau_eu()
```

---

## ll_get_lau_nuts_concordance

*Gets correspondence tables between local administrative units and nuts from Eurostat's website*

---

### Description

Source: https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/lau

### Usage

```
ll_get_lau_nuts_concordance(lau_year = 2019, nuts_year = 2016, silent = FALSE)
```

### Arguments

| | |
|---|---|
| `lau_year` | Defaults to 2019. See `ll_lau_nuts_concordance_links` for details on available combinations. |
| `nuts_year` | Defaults to 2016. See `ll_lau_nuts_concordance_links` for details on available combinations. |
| `silent` | Defaults to FALSE. If TRUE, hides copyright notice. Useful e.g. when using this in reports or in loops. The copyright notice must still be shown where the final output is used. |

### Details

Warning: due to issues in the original data, nuts may not always correspond to the given year for all countries, e.g. in files with nuts 2016 one may find nuts 2013 for single country, e.g. Italy. Do check the sources for details and ensure complete matching.

### Value

A tibble with a correspondence table.

### Examples

```
ll_set_folder("~/R/")
ll_get_lau_nuts_concordance()
## Not run:
lau_with_nuts_df <- ll_get_lau_eu(year = 2018) %>%
  sf::st_drop_geometry() %>%
  filter(is.na(LAU_NAME) == FALSE) %>%
  dplyr::rename(gisco_id = GISCO_ID) %>%
  dplyr::left_join(
    y = ll_get_lau_nuts_concordance(
      lau_year = 2018,
      nuts_year = 2016
    ),
    by = "gisco_id"
  )

## End(Not run)
```

---

`ll_get_lau_pt` *Regions and provinces in Italy (high detail, CC-BY Istat)*

---

## Description

Source: https://dados.gov.pt/pt/datasets/freguesias-de-portugal/

## Usage

```
ll_get_lau_pt(
  id = NULL,
  name = NULL,
  year = 2017,
  level = "concelho",
  silent = FALSE
)
```

## Arguments

| | |
|---|---|
| `id` | A character vector composed of six digits. Corresponds to "dicofre". |
| `year` | Defaults to 2017 (latest and currently only available). |
| `level` | Defaults to "freguesia". Valid value include "freguesia", "concelho", "distrito", "des_simpli". |

## Examples

```
ll_set_folder(fs::path(fs::path_home_r(), "R"))
ll_get_lau_pt()
ll_get_lau_pt(name = "Porto")
```

## `ll_get_nuts_eu` *Gets NUTS as sf object from Eurostat's website*

### Description

Source: https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts#nuts16

### Usage

```
ll_get_nuts_eu(
  nuts_id = NULL,
  nuts_name = NULL,
  level = 3,
  resolution = 60,
  year = 2021,
  silent = FALSE
)
```

### Arguments

| | |
|---|---|
| `nuts_id` | NUTS id. Must correspond to the level given. e.g. "DE149" for the NUTS3 regionion of Sigmaringen in Germany. |
| `nuts_name` | Name of the NUTS region. Run `ll_get_nuts_eu()` to check valid values in `NUTS_NAME` column. |
| `level` | Defaults to 3, corresponding to nuts3. Available values are: 0, 1, 2, and 3. |
| `resolution` | Defaults to "60", for 1:60 Million. Available values: are 20, 10, 3, 1 (1 is highest quality available). |
| `year` | Defaults to 2021 Available values: 2021, 2016, 2013, 2010, 2006, 2003 |

### Value

NUTS in sf format

### Examples

```
ll_get_nuts_eu()
```

## `ll_get_nuts_it` *Regions and provinces in Italy (high detail, CC-BY Istat)*

### Description

2019 / WGS84 UTM32N

### Usage

```
ll_get_nuts_it(
  name = NULL,
  level = 2,
  year = 2023,
  resolution = "low",
  silent = FALSE,
  no_check_certificate = FALSE
)
```

### Arguments

| | |
|---|---|
| `level` | Defaults to "2", i.e. regioni. Available: "3" (i.e. province), and "lau", local administrative units. |
| `year` | Defaults to 2023 (latest available). |
| `resolution` | Defaults to "low". Valid values are either "low" or "high". |
| `no_check_certificate` | Logical, defaults to FALSE. Enable only if certificate issues, and if you are aware of the security implications. |

### Examples

```
ll_set_folder(fs::path(fs::path_home_r(), "R"))
ll_get_nuts_it()
ll_get_nuts_it(name = "Rimini", level = 3)
```

## `ll_get_nuts_us` *Get US counties*

### Description

Source: https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.html

### Usage

```
ll_get_nuts_us(level = "county", resolution = "500k", year = 2018)
```

## Arguments

| | |
|---|---|
| `level` | Defaults to "county". Available options are: "cd116" (for congressional districts of the 116th Congress) |
| `resolution` | Defaults to "500k", max available resolution. Available options are: "5m" and "20m" |
| `year` | Defaults to 2018 |

## Examples

```
ll_get_nuts_us(level = "county", resolution = "500k", year = 2018)
```

---

`ll_get_population_grid` *Get EU 1km population grid*

---

## Description

Source: https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/population-distribution-demography/geostat
More details: https://ec.europa.eu/eurostat/statistics-explained/index.php/Population_grids

## Usage

```
ll_get_population_grid(
  year = 2018,
  match_sf = NULL,
  match_name = NULL,
  match_country = NULL,
  join = sf::st_intersects,
  silent = FALSE,
  population_grid_sf = NULL
)
```

## Arguments

| | |
|---|---|
| `year` | Defaults to 2018. Currently, the EU population grid is available only for the year 2006, 2011, and 2018. |
| `match_sf` | An sf object to be matched with the population grid. If not given, full grid is returned. |
| `match_name` | A name to be used for local caching. It is the responsibility of the user to keept it consistent. If not given, data are not cached locally. |
| `match_country` | Defaults to NULL. If given, used to speed up processing. |
| `join` | The function to use for filtering. Defaults to sf::st_intersects. Alternative includes the likes of sf::st_within, sf::st_touches, etc. |
| `population_grid_sf` | Defaults to NULL. If given, it uses this one as population grid of reference. Useful to bulk process items, as it removes the need for re-loading the grid from local storage at each iteration. |

## Value

An sf object with the population grid.

---

`ll_get_population_grid_hr`
*Get High Resolution Population Density Maps + Demographic Estimates*

---

## Description

Source: https://data.humdata.org/organization/facebook Details on methodology: https://dataforgood.fb.com/docs/methodolo
high-resolution-population-density-maps-demographic-estimates/

## Usage

```
ll_get_population_grid_hr(
  geo,
  match_sf = NULL,
  match_name = NULL,
  population_grid_sf = NULL,
  join = sf::st_intersects,
  file_format = "CSV",
  dataset = "population|general",
  source_url = NULL,
  silent = FALSE
)
```

## Arguments

| | |
|---|---|
| `geo` | A twe letter country code, such as "IT" for Italy and "DE" for Germany |
| `match_sf` | An sf object to me matched with the population grid. If not given, full grid is returned. |
| `match_name` | A name to be used for local caching if a subset of the grid is used. It is the responsibility of the user to keept it consistent. If not given, data are not cached locally. |
| `file_format` | Defaults to "CSV". Other available formats include "GeoTIFF", "JSON", "zip", "GDAL Virtual Format". Currently only CSV supported. |
| `dataset` | Defaults to "population". Beginning of the name of the dataset. For alternatives, see e.g. `population_grid_hr_metadata %>% dplyr::filter(country_code=="IT") %>% dplyr::distinct(name)`. Currently only tested with default value. |
| `source_url` | A direct link to the zipped version of the csv file in the original database, if automatic download with the country code does not work. For example, for Italy this would be "https://data.humdata.org/dataset/0eb77b21-06be-42c8-9245-2edaff79952f/resource/1e96f272-7d86-4108-b4ca-5a951a8b11a0/download/population_ita_2019-07-01.csv.zip" |
| `silent` | |

## `ll_get_world` *Get countries as an sf object*

### Description

Get countries as an sf object

### Usage

```
ll_get_world(resolution = "60", year = 2020, name = NULL)
```

### Arguments

| | |
|---|---|
| `resolution` | Defaults to "60", for 1:60 Million. Available values: are 20, 10, 3, 1 (1 is highest quality available)- |
| `year` | Defaults to 2020. Available values: 2020, 2016, 2013, 2010, 2006, 2001 |

## `ll_match` *Matches a data frame with longitude and latitude to an sf object*

### Description

Matches a data frame with longitude and latitude to an sf object

### Usage

```
ll_match(
  data,
  longitude = 1,
  latitude = 2,
  join = sf::st_intersects,
  sample = NULL,
  match = longlat2map::ll_get_world()
)
```

### Arguments

| | |
|---|---|
| `data` | A data frame or tibble with a column for longitude and one for latitude or an onject of the sf class. If an sf object is given, the longitude and latitude parameters are ignored. |
| `longitude` | The exact column name or the column index (e.g. 1 if first column) for longitude. Defaults to 1. |
| `latitude` | The exact column name or the column index (e.g. 1 if first column) for latitude. Defaults to 2. |

| | |
|---|---|
| `join` | A function of the sf class determining the type of join. Defaults to `sf::st_intersects`. Check `?sf::st_join` for alternatives. |
| `sample` | Defaults to NULL. If given, it runs the matching with only a subset of the original dataframe. Suggested for testing in particular when working with big datasets. |
| `match` | An sf object to be matched with the given dataframe, defaults to `longlat2map::ll_get_world()`. This package facilitate obtaining alternative reference maps with functions such as `longlat2map::ll_get_nuts_eu()` and `longlat2map::ll_get_nuts_us()` |

## Value

An sf object with CRS 4326.

---

`ll_osm_countries` *Countries and geographic entities for which shapefiles are made availabile by Geofabrik*

---

## Description

A dataset with all names of countries, continents, as included in the Geofabrik database. They are used to download files with `ll_osm_download()`

## Usage

```
ll_osm_countries
```

## Format

A tibble

**continent** Name of the continent

**country** Name of the country

**link** Link to shapefiles in a tibble

## Details

Links to shapefiles are stored as tibbles. Unnest to see them, e.g. `ll_osm_countries %>% tidyr::unnest(link)` or for a single country: `ll_osm_countries %>% dplyr::filter(country == "italy") %>% tidyr::unnest(link)`

## Source

http://download.geofabrik.de/

## `ll_osm_download` *Download OSM data for whole countries from Geofabrik.*

### Description

N.B. Names do not always correspond to official name of countries and may include different geographic entities. For a full list of available "countries" as made available by Geofabrik, see the internal dataset `ll_osm_countries`. Be considered in downloading files.

### Usage

```
ll_osm_download(countries, overwrite = FALSE, wget = FALSE)
```

### Arguments

| | |
|---|---|
| `countries` | One or more country names. For details on available country names see the dataset included in this package: `ll_osm_countries` |
| `overwrite` | Logical, defaults to FALSE. If true, downloads new files even if already present. |
| `wget` | Logical, defaults to FALSE. If TRUE, it downloads files with wget (if available), otherwise uses default method. Setting wget to TRUE may contribute to prevent download timeouts; notice that apparent freeze of the download progress in the console are common, and mostly the download is just continuing in the background (for reference, check file size in folder.) |

### Value

Used only for its side effects (downloads osm data).

### Examples

```
## Not run:
ll_osm_download(countries = "Romania")
ll_osm_download(countries = c("chile", "colombia"))

## End(Not run)
```

## `ll_osm_download_it` *Download OSM data in geopackage format for regions, provinces, and municipalities in Italy.*

### Description

See `ll_osm_it_gpkg` for all available files.

### Usage

```
ll_osm_download_it(
  level = "comuni",
  name = NULL,
  code = NULL,
  overwrite = FALSE,
  wget = FALSE,
  quiet = FALSE
)
```

### Arguments

| | |
|---|---|
| `level` | One of "regioni", "provincie", "comuni". Defaults to "comuni". |
| `name` | Name of geographic entity. Check `ll_osm_it_gpkg` or `ll_get_nuts_it()` for valid names. |
| `code` | Used in alternative to name. Check `ll_osm_it_gpkg` or `ll_get_nuts_it()` for valid values. |
| `wget` | Logical, defaults to FALSE. If TRUE, it downloads files with wget (if available), otherwise uses default method. Setting wget to TRUE may contribute to prevent download timeouts; notice that apparent freeze of the download progress in the console are common, and mostly the download is just continuing in the background (for reference, check file size in folder.) |
| `quiet` | Logical, defaults to FALSE. If TRUE no messages about download advancement are printed. |

### Value

Used only for its side effects (downloads osm data).

### Examples

```
## Not run:
ll_osm_download_it(level = "comuni", name = "Trento")

## End(Not run)
```

---

## `ll_osm_extract_it` *Extract OSM data for regions, provinces, and municipalities in Italy.*

---

### Description

See `ll_osm_it_gpkg` for all available files.

### Usage

```
ll_osm_extract_it(
  level = "comuni",
  name = NULL,
  code = NULL,
  layer = "lines",
  quiet = FALSE
)
```

### Arguments

| | |
|---|---|
| `level` | One of "regioni", "provincie", "comuni". Defaults to "comuni". |
| `name` | Name of geographic entity. Check `ll_osm_it_gpkg` or `ll_get_nuts_it()` for valid names. |
| `code` | Used in alternative to name. Check `ll_osm_it_gpkg` or `ll_get_nuts_it()` for valid values. |
| `layer` | Defaults to "lines". Must be one of "points", "lines", "multilinestrings", "multipolygons", or "other_relations" |
| `quiet` | Logical, defaults to FALSE. If TRUE, supresses messages generated when reading the geopackage file. |

### Value

An sf object.

### Examples

```
## Not run:
ll_osm_extract_it(level = "comuni", name = "Trento")

## End(Not run)
```

---

## `ll_osm_extract_roads` *Extract from zip shape files of roads from previously downloaded*

---

### Description

Extract from zip shape files of roads from previously downloaded

### Usage

```
ll_osm_extract_roads(countries, download_if_missing = TRUE, overwrite = FALSE)
```

### Arguments

| | |
|---|---|
| `countries` | The name of one or more geographic entities from files typically previously downloaded with `ll_osm_download()` |
| `download_if_missing` | Logical, defaults to TRUE. If TRUE, downloads country files with `ll_osm_download()` if they are not available locally. |
| `overwrite` | Logical, defaults to FALSE. If TRUE, extracts files from zip even if folder already existing. |

### Value

Nothing, used for its side effects (extracts shapefiles from country-level zip files)

### Examples

```
## Not run:
ll_extract_roads(countries = "Romania")

## End(Not run)
```

---

## `ll_osm_get_lau_streets` *Get all streets available in OpenStreetMap located in given local administrative unit.*

---

### Description

Relies on the output of `ll_get_lau_eu()` for the boundaries of local administrative units.

### Usage

```
ll_osm_get_lau_streets(
  gisco_id,
  country = NULL,
  unnamed_streets = TRUE,
  lau_boundary_sf = NULL,
  streets_sf = NULL,
  country_code_type = "eurostat",
  year = 2021,
  fallback = TRUE
)
```

## Arguments

| | |
|---|---|
| `gisco_id` | Gisco identifier. |
| `country` | Name of country as included in Geofabrik's datasets, does not always match common country names or geography. For details on available country names see the dataset included in this package: `ll_osm_countries` |
| `unnamed_streets` | Defaults to TRUE. If FALSE, it drops all streets with missing "name" or missing "fclass". |
| `lau_boundary_sf` | Defaults to NULL. If given, used to speed up processing. Must be an `sf` object such as the ones output by by `ll_get_lau_eu()`. |
| `streets_sf` | Defaults to NULL. If given, used to speed up processing. Must be an `sf` object such as the ones output by `ll_osm_get_roads()`. |
| `country_code_type` | Defaults to "eurostat". An alternative common value is "iso2c". See `countrycode::codelist` for a list of available codes. |
| `year` | Year of LAU boundaries, defaults to most recent (2021), passed to `ll_get_lau_eu()`. Available starting with 2011. |
| `fallback` | Logical, defaults to TRUE. If a `gisco_id` does not match an entity in `ll_get_lau_eu()`, try alternatives for the boundaries based on the country code, including `ll_get_nuts_eu()`, `ll_get_gadm()`, and `ll_get_adm_ocha()`. |

## Value

An `sf` objects with all streets of a given LAU based on OpenStreetMap

## Examples

```
## Not run:
ll_osm_get_lau_streets(gisco_id = "IT_022205", unnamed_streets = FALSE)

# or if country name does not match

ll_osm_get_lau_streets(gisco_id = "EL_01020204", country = "greece")

## End(Not run)
```

---

## `ll_osm_get_nuts_streets`

*Get all streets available in OpenStreetMap located in given NUTS.*

---

## Description

Relies on the output of `ll_get_nuts_eu()` for the boundaries of NUTS.

## Usage

```
ll_osm_get_nuts_streets(
  nuts_id,
  level = 3,
  resolution = 1,
  country = NULL,
  unnamed_streets = TRUE,
  nuts_boundary_sf = NULL,
  streets_sf = NULL,
  country_code_type = "eurostat",
  year = 2021
)
```

## Arguments

| | |
|---|---|
| `nuts_id` | NUTS region identifier. |
| `country` | Name of country as included in Geofabrik's datasets, does not always match common country names or geography. For details on available country names see the dataset included in this package: `ll_osm_countries` |
| `unnamed_streets` | Defaults to TRUE. If FALSE, it drops all streets with missing "name" or missing "fclass". |
| `streets_sf` | Defaults to NULL. If given, used to speed up processing. Must be an `sf` object such as the ones output by `ll_osm_get_roads()`. |
| `country_code_type` | Defaults to "eurostat". An alternative common value is "iso2c". See `countrycode::codelist` for a list of available codes. |
| `year` | Year of LAU boundaries, defaults to most recent (2021), passed to `ll_get_lau_eu()`. Available starting with 2011. |

## Value

An `sf` objects with all streets of a given NUTS regions based on OpenStreetMap

## Examples

```
## Not run:
ll_osm_get_nuts_streets(nuts_id = "PT16D", country = "portugal")

## End(Not run)
```

## `ll_osm_get_roads` *Extract shape files of roads from previously downloaded*

### Description

Extract shape files of roads from previously downloaded

### Usage

```
ll_osm_get_roads(country, silent = FALSE)
```

### Arguments

| | |
|---|---|
| `country` | The name of one or more geographic entities from files typically previously downloaded with `ll_osm_download()` |
| `silent` | Defaults to FALSE. If TRUE, hides copyright notice. Useful e.g. when using this in reports or in loops. The copyright notice must still be shown where the final output is used. |

### Value

All roads in a country by OpenStreetMap.

### Examples

```
## Not run:
ll_osm_get_roads(country = "Romania")

## End(Not run)
```

## `ll_osm_it_gpkg` *Geographic entities in Italy for which geopackage files are availabile*

### Description

A dataset with all names of geographic entities available for direct download as geopackage files They are used to download files with `ll_osm_download_it()`

### Usage

```
ll_osm_it_gpkg
```

### Format

A list of tibbles

### Source

https://osmit-estratti.wmcloud.org/

---

## `ll_set_folder` *Set folder for caching data*

---

### Description

Set folder for caching data

### Usage

```
ll_set_folder(path = NULL)
```

### Arguments

| | |
|---|---|
| `path` | A path to a location. If the folder does not exist, it will be created. |

### Value

The path to the caching folder, if previously set.

---

## `population_grid_hr_metadata` *A data frame with links to High Resolution Population Density Maps distributed by Facebook on HDX*

---

### Description

It is used to download files with `ll_get_population_grid_hr()`

### Usage

```
population_grid_hr_metadata
```

### Format

A tibble

**country** Name of the country in English

**country_code** Two letter code as used by eurostat, see also `countrycode::codelist$eurostat`

**download_ulr** Link to zipped dataset

**url** Link to page describing the dataset

### Source

https://data.humdata.org/

# Index