Package: geodimension (via r-universe)

November 2, 2026

Type Package

Title Definition of Geographic Dimensions

Version 2.0.0.9000

Description The geographic dimension plays a fundamental role in multidimensional systems. To define a geographic dimension in a star schema, we need a table with attributes corresponding to the levels of the dimension. Additionally, we will also need one or more geographic layers to represent the data using this dimension. The goal of this package is to support the definition of geographic dimensions from layers of geographic information related to each other. It makes it easy to define relationships between layers and obtain the necessary data from them.

License MIT + file LICENSE

URL https://josesamos.github.io/geodimension/, https://github.com/josesamos/geodimension

BugReports https://github.com/josesamos/geodimension/issues

Depends R (>= 2.10)

Imports dplyr, methods, sf, snakecase, tibble, tidyselect

Suggests knitr, pander, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

LazyData true

LazyDataCompression xz

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2024-01-09 07:14:33 UTC

RemoteUrl https://github.com/josesamos/geodimension

RemoteRef HEAD

RemoteSha bfe363f10b53176616698ec944a79bef1afd209d

Contents

add_geometry 2
add_level 4
check_key 5
complete_point_geometry 6
complete_relation_by_geography 7
coordinates_to_geometry 8
gd_es 10
gd_us 10
geodimension 11
geolevel 12
get_empty_geometry_instances 13
get_geometry 14
get_higher_level_names 15
get_level_data 16
get_level_data_geo 17
get_level_geometries 18
get_level_layer 19
get_level_names 20
get_unrelated_instances 21
relate_levels 22
select_levels 25
set_level_data 26
transform_crs 27
us_division 28

Index **29**

`add_geometry` *Add geometry to a level*

Description

A level can have several geometries (*point* or *polygon*). This function adds the geometry of the layer to the level.

Usage

```
add_geometry(gl, layer, layer_key, level_key)

## S3 method for class 'geolevel'
add_geometry(gl, layer = NULL, layer_key = NULL, level_key = NULL)
```

Arguments

gl	A geolevel object.
layer	A sf object.
layer_key	A vector of string.
level_key	A vector of string.

Details

The association of the geometry to the existing instances is done through join using the level and layer keys.

If none is indicated, by default the key defined in the level is considered.

Value

A geolevel.

See Also

geodimension, relate_levels

Other geolevel definition functions: check_key(), complete_point_geometry(), coordinates_to_geometry(), geolevel(), get_empty_geometry_instances(), get_geometry()

Examples

```
layer_us_state <-
  dplyr::inner_join(
    get_level_data_geo(gd_us, "state"),
    get_level_layer(gd_us, "state"),
    by = c("statefp", "division", "region", "stusps", "name")
  ) |>
  sf::st_as_sf()

us_state_point <-
  coordinates_to_geometry(layer_us_state,
                          lon_lat = c("intptlon", "intptlat"))

state <-
  geolevel(name = "state",
           layer = layer_us_state,
           key = "statefp",
           snake_case = TRUE) |>
  add_geometry(layer = us_state_point)
```

add_level *Add a level to a dimension*

Description

Once a level is part of the dimension, it can then be related to other levels of the dimension.

Usage

```
add_level(gd, level)

## S3 method for class 'geodimension'
add_level(gd, level = NULL)
```

Arguments

gd	A geodimension object.
level	A geolevel, level to add to the dimension.

Value

A geodimension.

See Also

geolevel, relate_levels, get_level_layer

Other geodimension definition functions: geodimension(), set_level_data(), transform_crs()

Examples

```
layer_us_place <- gd_us |>
  get_level_layer("place")

layer_us_county <-
  dplyr::inner_join(
    get_level_data_geo(gd_us, "county"),
    get_level_layer(gd_us, "county"),
    by = c("geoid", "statefp", "name", "type")
  ) |>
  sf::st_as_sf()

place <-
  geolevel(name = "place",
           layer = layer_us_place,
           attributes = c("statefp", "county_geoid", "name", "type"),
           key = "geoid")

county <-
  geolevel(
```

```
    name = "county",
    layer = layer_us_county,
    attributes = c("statefp", "name", "type"),
    key = "geoid"
  ) |>
  add_geometry(coordinates_to_geometry(layer_us_county,
                                       lon_lat = c("intptlon", "intptlat")))

gd_us <-
  geodimension(name = "gd_us",
               level = place) |>
  add_level(level = county)
```

---

check_key *Check key*

Description

Check if the specified set of attributes can be the key of the table.

Usage

```
check_key(table, key = NULL)
```

Arguments

table	A tibble object.
key	A vector, attributes that compose the key.

Details

The table can be a data table or a vector layer.

Value

A boolean.

See Also

Other geolevel definition functions: add_geometry(), complete_point_geometry(), coordinates_to_geometry(), geolevel(), get_empty_geometry_instances(), get_geometry()

Examples

```
layer_us_county <- get_level_layer(gd_us, "county")

is_key <- check_key(layer_us_county, key = c("statefp", "name"))
```

`complete_point_geometry` *Complete point geometry*

Description

In case of having the polygon geometry defined, it obtains the point geometry from it.

Usage

```
complete_point_geometry(gl)

## S3 method for class 'geolevel'
complete_point_geometry(gl)
```

Arguments

`gl`	A `geolevel` object.

Details

If the point geometry was already defined, if there are instances with this geometry empty, it completes them.

Value

A `geolevel` object.

See Also

`geodimension`, `relate_levels`

Other geolevel definition functions: `add_geometry()`, `check_key()`, `coordinates_to_geometry()`, `geolevel()`, `get_empty_geometry_instances()`, `get_geometry()`

Examples

```
layer_us_state <- get_level_layer(gd_us, "state")

state <-
  geolevel(name = "state",
           layer = layer_us_state,
           key = "statefp",
           snake_case = TRUE) |>
  complete_point_geometry()
```

`complete_relation_by_geography` *Complete relation by geography*

Description

Two levels can be related by attributes or by geography (if the upper level has polygon-type geometry). Once related, if there are unrelated instances, we can try to relate those instances using this function, which considers alternative geographic relationships.

Usage

```
complete_relation_by_geography(
  gd,
  lower_level_name = NULL,
  upper_level_name = NULL
)

## S3 method for class 'geodimension'
complete_relation_by_geography(
  gd,
  lower_level_name = NULL,
  upper_level_name = NULL
)
```

Arguments

`gd`	A `geodimension` object.
`lower_level_name`	A string, name of the lower level.
`upper_level_name`	A string, name of the upper lever.

Details

It does not necessarily succeed trying to relate instances.

Value

A `geodimension` object.

See Also

`geodimension`, `geolevel`

Other relationship between geolevels: `get_unrelated_instances()`, `relate_levels()`, `select_levels()`

Examples

```
layer_us_place <- gd_us |>
  get_level_layer("place")

layer_us_county <-
  dplyr::inner_join(
    get_level_data_geo(gd_us, "county"),
    get_level_layer(gd_us, "county"),
    by = c("geoid", "statefp", "name", "type")
  ) |>
  sf::st_as_sf()

place <-
  geolevel(name = "place",
           layer = layer_us_place,
           attributes = c("statefp", "county_geoid", "name", "type"),
           key = "geoid")

county <-
  geolevel(
    name = "county",
    layer = layer_us_county,
    attributes = c("statefp", "name", "type"),
    key = "geoid"
  ) |>
  add_geometry(coordinates_to_geometry(layer_us_county,
                                       lon_lat = c("intptlon", "intptlat")))

gd <-
  geodimension(name = "gd_us",
               level = place) |>
  add_level(level = county)

gd <- gd |>
  relate_levels(
    lower_level_name = "place",
    lower_level_attributes = "county_geoid",
    upper_level_name = "county"
  ) |>
  complete_relation_by_geography(
    lower_level_name = "place",
    upper_level_name = "county"
  )
```

`coordinates_to_geometry` *Transform coordinates to point geometry*

Description

From the coordinates defined in fields such as latitude and longitude, it returns a layer of points.

Usage

```
coordinates_to_geometry(table, lon_lat = c("intptlon", "intptlat"), crs = 4326)
```

Arguments

table	A tibble object.
lon_lat	A vector, name of longitude and latitude attributes.
crs	A coordinate reference system: integer with the EPSG code, or character with proj4string.

Details

If we start from a geographic layer, it initially transforms it into a table.

The CRS of the new layer is indicated. By default, it considers 4326 (WGS84).

Value

A sf object.

See Also

Other geolevel definition functions: add_geometry(), check_key(), complete_point_geometry(), geolevel(), get_empty_geometry_instances(), get_geometry()

Examples

```
layer_us_county <-
  dplyr::inner_join(
    get_level_data_geo(gd_us, "county"),
    get_level_layer(gd_us, "county"),
    by = c("geoid", "statefp", "name", "type")
  ) |>
  sf::st_as_sf()

us_county_point <-
  coordinates_to_geometry(layer_us_county,
                          lon_lat = c("intptlon", "intptlat"))
```

gd_es gd_es

Description

geodimension obtained from vector layers over Spain

Usage

```
gd_es
```

Format

A geodimension.

Details

It includes the levels nucleus, municipality, agricultural region, province, autonomous community and country.

Source

https://centrodedescargas.cnig.es/CentroDescargas/, https://www.mapa.gob.es/es/cartografia-y-sig/ide/descargas/agricultura/default.aspx

gd_us gd_us

Description

geodimension obtained from vector layers over USA.

Usage

```
gd_us
```

Format

A geodimension.

Details

It includes the levels place, county, state, division, region and country.

Source

https://www.census.gov

geodimension geodimension *S3 class*

Description

A geodimension object is created. A geodimension allows us to relate levels. In addition to the name of the geodimension , a level has to be given.

Usage

```
geodimension(name = NULL, level = NULL, snake_case = FALSE)
```

Arguments

name	A string, name of the dimension.
level	A geolevel.
snake_case	A boolean, transform all names to snake_case.

Value

A geodimension object.

See Also

geolevel, relate_levels, get_level_layer

Other geodimension definition functions: add_level(), set_level_data(), transform_crs()

Examples

```
layer_us_place <- get_level_layer(gd_us, "place")

place <-
  geolevel(name = "place",
           layer = layer_us_place,
           key = "geoid")
gd <-
  geodimension(name = "gd_us",
               level = place)
```

geolevel geolevel *S3 class*

Description

A geolevel object is created from a given geographic layer. The attributes of the layer to be included in the level can be indicated; if no attribute is indicated, all are considered. The attributes that make up the key must be indicated.

Usage

```
geolevel(
  name = NULL,
  layer = NULL,
  attributes = NULL,
  key = NULL,
  snake_case = FALSE
)
```

Arguments

name	A string, level name.
layer	A tibble or sf object.
attributes	A string vector, selected attributes.
key	A string vector, attributes that compose the key.
snake_case	A boolean, transform all names to snake_case.

Details

A level can have two associated geometries (point or polygon). The geometry is obtained from the layer data.

We can also define a level from a tibble, which does not have any associated geometry. The geometry will be obtained from the relationships between levels that we define or from layers related to this data.

The name of the level is used later to reference it and relate it to other levels.

Value

A geolevel object.

See Also

geodimension, relate_levels

Other geolevel definition functions: add_geometry(), check_key(), complete_point_geometry(), coordinates_to_geometry(), get_empty_geometry_instances(), get_geometry()

Examples

```
layer_us_state <- get_level_layer(gd_us, "state")

state <-
  geolevel(name = "state",
           layer = layer_us_state,
           key = "statefp",
           snake_case = TRUE)
```

`get_empty_geometry_instances` *Get empty geometry instances*

Description

Get the instances of the data table that do not have associated geometry for the specified geometry type.

Usage

```
get_empty_geometry_instances(gl, geometry)

## S3 method for class 'geolevel'
get_empty_geometry_instances(gl, geometry = NULL)
```

Arguments

`gl`	A `geolevel` object.
`geometry`	A string, type of geometry of the layer.

Value

A `tibble`.

See Also

`geodimension`, `relate_levels`

Other geolevel definition functions: `add_geometry()`, `check_key()`, `complete_point_geometry()`, `coordinates_to_geometry()`, `geolevel()`, `get_geometry()`

Examples

```
layer_us_state <-
  dplyr::inner_join(
    get_level_data_geo(gd_us, "state"),
    get_level_layer(gd_us, "state"),
    by = c("statefp", "division", "region", "stusps", "name")
  ) |>
  sf::st_as_sf()

us_state_point <-
  coordinates_to_geometry(layer_us_state,
                          lon_lat = c("intptlon", "intptlat"))

state <-
  geolevel(name = "state",
           layer = layer_us_state,
           key = "statefp",
           snake_case = TRUE) |>
  add_geometry(layer = us_state_point)

empty_geometry_instances <- state |>
  get_empty_geometry_instances(geometry = "point")
```

`get_geometry` *Get geometry*

Description

Get the geometry of a layer, as it is interpreted to define a `geolevel` object.

Usage

```
get_geometry(layer)
```

Arguments

`layer`	A `sf` object.

Details

It will only be valid if one of the three geometries is interpreted: *point*, *line* or *polygon*.

Value

A string.

See Also

Other geolevel definition functions: add_geometry(), check_key(), complete_point_geometry(), coordinates_to_geometry(), geolevel(), get_empty_geometry_instances()

Examples

```
layer_us_county <- get_level_layer(gd_us, "county")

geometry <- get_geometry(layer_us_county)
```

---

get_higher_level_names

Get higher level names

Description

Get the names of levels included in the geodimension that are related to the given level and are upper levels. We can get only the direct levels or the levels reached by passing through other levels.

Usage

```
get_higher_level_names(gd, level_name, indirect_levels)

## S3 method for class 'geodimension'
get_higher_level_names(gd, level_name = NULL, indirect_levels = FALSE)
```

Arguments

gd	A geodimension object.
level_name	A string.
indirect_levels	A boolean.

Value

A vector of names.

See Also

geodimension, geolevel

Other information gathering functions: get_level_data_geo(), get_level_data(), get_level_geometries(), get_level_layer(), get_level_names()

Examples

```
ln_1 <- gd_us |>
  get_higher_level_names(level_name = "place")

ln_2 <- gd_us |>
  get_higher_level_names(level_name = "place", indirect_levels = TRUE)
```

`get_level_data` *Get level data*

Description

Get the data table of a given level.

Usage

```
get_level_data(gd, level_name, inherited, add_prefix)

## S3 method for class 'geodimension'
get_level_data(gd, level_name = NULL, inherited = FALSE, add_prefix = TRUE)
```

Arguments

`gd`	A `geodimension` object.
`level_name`	A string.
`inherited`	A boolean.
`add_prefix`	A boolean.

Details

It allows selecting whether we want only the data defined locally in the level or also those that it inherits from other higher levels with which it is related.

In case of inheriting attributes from other levels, in the table, these can have as a prefix the name of the level.

Value

A `tibble` object.

See Also

`geodimension`, `geolevel`

Other information gathering functions: `get_higher_level_names()`, `get_level_data_geo()`, `get_level_geometries()`, `get_level_layer()`, `get_level_names()`

Examples

```
ld <- gd_us |>
  get_level_data(level_name = "county",
                 inherited = TRUE)
```

`get_level_data_geo` *Get level data with latitude and longitude*

Description

Get the data table of a given level with latitude and longitude.

Usage

```
get_level_data_geo(gd, level_name, inherited, add_prefix, lon_lat, crs)

## S3 method for class 'geodimension'
get_level_data_geo(
  gd,
  level_name = NULL,
  inherited = FALSE,
  add_prefix = TRUE,
  lon_lat = c("intptlon", "intptlat"),
  crs = 4326
)
```

Arguments

`gd`	A `geodimension` object.
`level_name`	A string.
`inherited`	A boolean.
`add_prefix`	A boolean.
`lon_lat`	A vector, name of longitude and latitude attributes.
`crs`	A coordinate reference system: integer with the EPSG code, or character with proj4string.

Details

It allows selecting whether we want only the data defined locally in the level or also those that it inherits from other higher levels with which it is related.

In case of inheriting attributes from other levels, in the table, these can have as a prefix the name of the level.

Additionally, we indicate the names of the fields where longitude and latitude will be stored, as well as the crs that is used, if they are different from the default values.

Value

A tibble object.

See Also

geodimension, geolevel

Other information gathering functions: get_higher_level_names(), get_level_data(), get_level_geometries(), get_level_layer(), get_level_names()

Examples

```
ld <- gd_us |>
  get_level_data_geo(level_name = "county",
                     inherited = TRUE)
```

---

get_level_geometries *Get level geometries*

Description

Gets the geometry types defined for a given level.

Usage

```
get_level_geometries(gd, level_name)

## S3 method for class 'geodimension'
get_level_geometries(gd, level_name = NULL)
```

Arguments

gd	A geodimension object.
level_name	A string.

Value

A vector of names.

See Also

geodimension, geolevel

Other information gathering functions: get_higher_level_names(), get_level_data_geo(), get_level_data(), get_level_layer(), get_level_names()

Examples

```
lg <- gd_us |>
  get_level_geometries(level_name = "state")
```

`get_level_layer` *Get level layer*

Description

Get a geographic layer associated with a level. We can select the geometry and, using boolean parameters, which attributes are included in the layer's table: only the attributes that make up the key and, if applied to a geodimension, inherited attributes to which the prefix of the level where they are defined can be added.

Usage

```
get_level_layer(gd, level_name, geometry, only_key, inherited, add_prefix)

## S3 method for class 'geodimension'
get_level_layer(
  gd,
  level_name = NULL,
  geometry = NULL,
  only_key = FALSE,
  inherited = FALSE,
  add_prefix = TRUE
)

## S3 method for class 'geolevel'
get_level_layer(
  gd,
  level_name = NULL,
  geometry = NULL,
  only_key = FALSE,
  inherited = FALSE,
  add_prefix = TRUE
)
```

Arguments

`gd`	A `geolevel` or `geodimension` object.
`level_name`	A string.
`geometry`	A string.
`only_key`	A boolean.
`inherited`	A boolean.
`add_prefix`	A boolean.

Value

A sf object.

See Also

geodimension, geolevel

Other information gathering functions: get_higher_level_names(), get_level_data_geo(), get_level_data(), get_level_geometries(), get_level_names()

Examples

```
layer_us_state <- get_level_layer(gd_us, "state")

state <-
  geolevel(name = "state",
           layer = layer_us_state,
           key = "statefp")

state_ll <- state |>
  get_level_layer("polygon")


county_ll <- gd_us |>
  get_level_layer(level_name = "county",
                  geometry = "polygon",
                  inherited = TRUE)
```

---

get_level_names *Get level names*

Description

Get the names of levels included in the geodimension.

Usage

```
get_level_names(gd)

## S3 method for class 'geodimension'
get_level_names(gd)
```

Arguments

gd	A geodimension object.

Value

A vector of names.

See Also

geodimension, geolevel

Other information gathering functions: get_higher_level_names(), get_level_data_geo(), get_level_data(), get_level_geometries(), get_level_layer()

Examples

```
ln <- gd_us |>
  get_level_names()
```

---

get_unrelated_instances

Get unrelated instances

Description

Given two previously related levels of a dimension, it obtains the instances of the lower level that have not been related to the upper level.

Usage

```
get_unrelated_instances(gd, lower_level_name, upper_level_name)

## S3 method for class 'geodimension'
get_unrelated_instances(gd, lower_level_name = NULL, upper_level_name = NULL)
```

Arguments

gd	A geodimension object.
lower_level_name	A string, name of the lower level.
upper_level_name	A string, name of the upper lever.

Value

A tibble, unrelated lower level instances.

See Also

geodimension, geolevel

Other relationship between geolevels: complete_relation_by_geography(), relate_levels(), select_levels()

Examples

```
layer_us_place <- gd_us |>
  get_level_layer("place")

layer_us_county <-
  dplyr::inner_join(
    get_level_data_geo(gd_us, "county"),
    get_level_layer(gd_us, "county"),
    by = c("geoid", "statefp", "name", "type")
  ) |>
  sf::st_as_sf()

place <-
  geolevel(name = "place",
           layer = layer_us_place,
           attributes = c("statefp", "county_geoid", "name", "type"),
           key = "geoid")

county <-
  geolevel(
    name = "county",
    layer = layer_us_county,
    attributes = c("statefp", "name", "type"),
    key = "geoid"
  ) |>
  add_geometry(coordinates_to_geometry(layer_us_county,
                                       lon_lat = c("intptlon", "intptlat")))

gd <-
  geodimension(name = "gd_us",
               level = place) |>
  add_level(level = county)

gd <- gd |>
  relate_levels(
    lower_level_name = "place",
    upper_level_name = "county",
    by_geography = TRUE
  )

ui <- gd |>
  get_unrelated_instances(
    lower_level_name = "place",
    upper_level_name = "county"
  )
```

`relate_levels` *Relate levels in a dimension*

Description

Definition of a direct relationship between two levels of the dimension: the lower level composes the higher level.

Usage

```
relate_levels(
  gd,
  lower_level_name,
  lower_level_attributes,
  upper_level_name,
  upper_level_key,
  by_geography
)

## S3 method for class 'geodimension'
relate_levels(
  gd,
  lower_level_name = NULL,
  lower_level_attributes = NULL,
  upper_level_name = NULL,
  upper_level_key = NULL,
  by_geography = FALSE
)
```

Arguments

gd	A geodimension object.
lower_level_name	A string, name of the lower level.
lower_level_attributes	A vector of attribute names.
upper_level_name	A string, name of the upper lever.
upper_level_key	A vector of attribute names.
by_geography	A boolean.

Details

The relationship may exist by having attributes with common values or by their geographic attributes. In the latter case, the geometry of the upper level must be of the polygon type.

If no top-level attributes are indicated, the attributes that make up the key are considered by default, only the corresponding attributes of the lower level have to be indicated.

To use the geometric relationship, it must be explicitly indicated by the Boolean parameter. In this case, the attributes of the lower level must not exist in the table, they will be added with the values of the key of the upper level, according to the established relationship. If lower level attribute names are not provided, they will be generated from the upper level key names, adding a prefix.

Value

A geodimension.

See Also

geodimension, geolevel

Other relationship between geolevels: complete_relation_by_geography(), get_unrelated_instances(), select_levels()

Examples

```
layer_us_place <- gd_us |>
  get_level_layer("place")

layer_us_county <-
  dplyr::inner_join(
    get_level_data_geo(gd_us, "county"),
    get_level_layer(gd_us, "county"),
    by = c("geoid", "statefp", "name", "type")
  ) |>
  sf::st_as_sf()

place <-
  geolevel(name = "place",
           layer = layer_us_place,
           attributes = c("statefp", "county_geoid", "name", "type"),
           key = "geoid")

county <-
  geolevel(
    name = "county",
    layer = layer_us_county,
    attributes = c("statefp", "name", "type"),
    key = "geoid"
  ) |>
  add_geometry(coordinates_to_geometry(layer_us_county,
                                       lon_lat = c("intptlon", "intptlat")))

gd <-
  geodimension(name = "gd_us",
               level = place) |>
  add_level(level = county)

gd <- gd |>
  relate_levels(
    lower_level_name = "place",
    lower_level_attributes = "county_geoid",
    upper_level_name = "county"
  )

gd_2 <- gd |>
```

```
relate_levels(
  lower_level_name = "place",
  upper_level_name = "county",
  by_geography = TRUE
)
```

`select_levels` *Select levels*

Description

Select a subset of the levels of the dimension so that the rest of the levels no longer belong to it.

Usage

```
select_levels(gd, level_names = NULL)

## S3 method for class 'geodimension'
select_levels(gd, level_names = NULL)
```

Arguments

`gd`	A `geodimension` object.
`level_names`	A vector of names.

Value

A `geodimension` object.

See Also

`geodimension`, `geolevel`

Other relationship between geolevels: `complete_relation_by_geography()`, `get_unrelated_instances()`, `relate_levels()`

Examples

```
gd_us_2 <- gd_us |>
  select_levels(level_names = c("state", "county", "place", "region"))
```

`set_level_data` *Set level data*

Description

Set the data table of a given level.

Usage

```
set_level_data(gd, level_name, data)

## S3 method for class 'geodimension'
set_level_data(gd, level_name = NULL, data = NULL)
```

Arguments

`gd`	A `geodimension` object.
`level_name`	A string.
`data`	A `tibble` object.

Details

We can get the table, filter or transform the data and redefine the level table.

It is checked that the attributes that have been used in the relationships remain in the table.

Value

A `geodimension` object.

See Also

`geolevel`, `get_level_data`

Other geodimension definition functions: `add_level()`, `geodimension()`, `transform_crs()`

Examples

```
ld <- gd_us |>
  get_level_data(level_name = "county",
                 inherited = TRUE)

gd_us <- gd_us |>
  set_level_data(level_name = "county",
                 data = ld)
```

`transform_crs` *Transform CRS*

Description

Transform the CRS of all the layers included in the dimension to the one indicated.

Usage

```
transform_crs(gd, crs = NULL)

## S3 method for class 'geodimension'
transform_crs(gd, crs = NULL)
```

Arguments

gd	A `geodimension` object.
crs	A coordinate reference system: integer with the EPSG code, or character with proj4string.

Value

A `geodimension`.

See Also

`geolevel`, `relate_levels`, `get_level_layer`

Other geodimension definition functions: `add_level()`, `geodimension()`, `set_level_data()`

Examples

```
layer_us_place <- gd_us |>
  get_level_layer("place")

layer_us_county <-
  dplyr::inner_join(
    get_level_data_geo(gd_us, "county"),
    get_level_layer(gd_us, "county"),
    by = c("geoid", "statefp", "name", "type")
  ) |>
  sf::st_as_sf()

place <-
  geolevel(name = "place",
           layer = layer_us_place,
           attributes = c("statefp", "county_geoid", "name", "type"),
           key = "geoid")
```

```
county <-
  geolevel(
    name = "county",
    layer = layer_us_county,
    attributes = c("statefp", "name", "type"),
    key = "geoid"
  ) |>
  add_geometry(coordinates_to_geometry(layer_us_county,
                                       lon_lat = c("intptlon", "intptlat")))

gd_us <-
  geodimension(name = "gd_us",
               level = place) |>
  add_level(level = county) |>
  transform_crs(crs = 3857)
```

---

us_division us_division

Description

US Divisions and Regions (code and name).

Usage

```
us_division
```

Format

A tibble object.

Index

∗ **datasets**
  gd_es, 10
  gd_us, 10
  us_division, 28
∗ **geodimension definition functions**
  add_level, 4
  geodimension, 11
  set_level_data, 26
  transform_crs, 27
∗ **geolevel definition functions**
  add_geometry, 2
  check_key, 5
  complete_point_geometry, 6
  coordinates_to_geometry, 8
  geolevel, 12
  get_empty_geometry_instances, 13
  get_geometry, 14
∗ **information gathering functions**
  get_higher_level_names, 15
  get_level_data, 16
  get_level_data_geo, 17
  get_level_geometries, 18
  get_level_layer, 19
  get_level_names, 20
∗ **relationship between geolevels**
  complete_relation_by_geography, 7
  get_unrelated_instances, 21
  relate_levels, 22
  select_levels, 25

add_geometry, 2, *5, 6*, *9*, *12, 13*, *15*
add_level, 4, *11*, *26, 27*

check_key, *3*, 5, *6*, *9*, *12, 13*, *15*
complete_point_geometry, *3*, *5*, 6, *9*, *12, 13*, *15*
complete_relation_by_geography, 7, *21*, *24, 25*
coordinates_to_geometry, *3*, *5, 6*, 8, *12, 13*, *15*

gd_es, 10
gd_us, 10
geodimension, *3, 4*, *6, 7*, 11, *12, 13*, *15, 16*, *18*, *20, 21*, *24–27*
geolevel, *3–7*, *9*, *11*, 12, *13*, *15, 16*, *18*, *20, 21*, *24–27*
get_empty_geometry_instances, *3*, *5, 6*, *9*, *12*, 13, *15*
get_geometry, *3*, *5, 6*, *9*, *12, 13*, 14
get_higher_level_names, 15, *16*, *18*, *20, 21*
get_level_data, *15*, 16, *18*, *20, 21*, *26*
get_level_data_geo, *15, 16*, 17, *18*, *20, 21*
get_level_geometries, *15, 16*, 18, *18*, *20, 21*
get_level_layer, *4*, *11*, *15, 16*, *18*, 19, *21*, *27*
get_level_names, *15, 16*, *18*, 20, *20*
get_unrelated_instances, *7*, 21, *24, 25*

relate_levels, *3, 4*, *6, 7*, *11–13*, *21*, 22, *25*, *27*

select_levels, *7*, *21*, *24*, 25
set_level_data, *4*, *11*, 26, *27*

transform_crs, *4*, *11*, *26*, 27

us_division, 28