Package 'geodimension'

November 27, 2020

Type Package

Title Definition of Geographic Dimensions

Version 1.0.0

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

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.1.1

Imports dplyr, readr, tibble, tidyr, tidyselect, generics, sf, magrittr, rlang, pander

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

Depends R (>= 2.10)

NeedsCompilation no

Author Jose Samos [aut, cre, cph] (<https://orcid.org/0000-0002-4457-3439>)

Maintainer Jose Samos <jsamos@ugr.es>

Repository CRAN

Date/Publication 2020-11-27 11:40:06 UTC

R topics documented:

`add_geometry` *Add geometry to a level*

Description

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

Usage

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

## 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 key and the layer key.

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

Value

A geolevel.

See Also

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

Examples

```
library(tidyr)
library(sf)

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

state <-
  geolevel(name = "state",
           layer = layer_us_state,
           key = c("geoid")) %>%
  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 = NULL)

## 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

Other level association functions: complete_relation_by_geography(), geodimension(), get_unrelated_instances(), relate_levels()

Examples

```
library(tidyr)
library(sf)

region <-
  geolevel(name = "region",
           layer = layer_us_region,
           key = c("geoid"))

division <-
  geolevel(name = "division",
           layer = layer_us_division,
           key = c("geoid"))

gd <-
  geodimension(name = "gd_us",
               level = region) %>%
  add_level(division)
```

---

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

Examples

```
library(sf)

is_key <- check_key(layer_us_region, key = c("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, use_intermediate_projected_crs = FALSE)

## S3 method for class 'geolevel'
complete_point_geometry(gl, use_intermediate_projected_crs = FALSE)
```

Arguments

gl	A geolevel object.
use_intermediate_projected_crs	A boolean.

Details

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

If the geometry of the CRS is not projected, it warns that the calculations may not be correct. A projected intermediate geometry can be used to perform the operation, indicating it by the boolean parameter.

Value

A `geolevel` object.

See Also

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

Examples

```
library(tidyr)
library(sf)

state <-
  geolevel(name = "state",
           layer = layer_us_state,
           key = c("geoid")) %>%
  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, you 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

For example, if the lower level has associated point and polygon geometries, only point geometry is considered to establish the initial relationship. Polygon geometry is also considered in this function.

It does not necessarily succeed trying to relate the instances.

Value

A geodimension object.

See Also

Other level association functions: add_level(), geodimension(), get_unrelated_instances(), relate_levels()

Examples

```
library(tidyr)
library(sf)

ui <- gd_us %>%
  get_unrelated_instances(lower_level_name = "state",
                          upper_level_name = "division")

gd <- gd_us %>%
  complete_relation_by_geography(lower_level_name = "state",
                          upper_level_name = "division")
```

`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 = NULL, crs = NULL)
```

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. If a CRS is not indicated, it considers the layer's CRS by default and, if it is not a layer, it considers 4326 CRS (WGS84).

Value

A `sf` object.

See Also

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

Examples

```
library(sf)

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

gd_us gd_us

Description

geodimension obtained from vector layers over USA.

Usage

```
gd_us
```

Format

A geodimension.

Details

It includes the levels city, county, state, region, division and nation.

Source

https://www.census.gov

geodimension geodimension *S3 class*

Description

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

Usage

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

Arguments

name	A string, name of the dimension.
level	A geolevel.

Value

A geodimension object.

See Also

Other level association functions: add_level(), complete_relation_by_geography(), get_unrelated_instances(), relate_levels()

Examples

```
library(tidyr)
library(sf)

region <-
  geolevel(name = "region",
           layer = layer_us_region,
           key = c("geoid"))

gd <-
  geodimension(name = "gd_us",
               level = region)
```

`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, and the subset of these that make up the natural key. If no attribute is indicated, all are considered. In any case, the attributes that make up the key must be indicated.

Usage

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

Arguments

`name`	A string, level name.
`layer`	A `sf` object.
`attributes`	A vector, selected attributes.
`key`	A vector, attributes that compose the key.

Details

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

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

Value

A geolevel object.

See Also

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

Examples

```
library(sf)

region <-
  geolevel(name = "region",
           layer = layer_us_region,
           key = c("geoid"))
```

---

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 = NULL)

## 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

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

Examples

```
library(tidyr)
library(sf)

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

state <-
  geolevel(name = "state",
           layer = layer_us_state,
           key = c("geoid")) %>%
  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 level definition functions: `add_geometry()`, `check_key()`, `complete_point_geometry()`, `coordinates_to_geometry()`, `geolevel()`, `get_empty_geometry_instances()`

Examples

```
library(sf)

geometry <- get_geometry(layer_us_region)
```

`get_higher_level_names` *Get higher level names*

Description

Get the names of levels included in the `geodimension` that are at a higher level than the indicated level. You can get only the direct levels or the levels reached by passing through other levels.

Usage

```
get_higher_level_names(gd, level_name = NULL, indirect_levels = FALSE)

## 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.

Details

The indicated level may inherit properties of the obtained levels.

Value

A vector of names.

See Also

Other information output functions: `get_level_data()`, `get_level_geometries()`, `get_level_layer()`, `get_level_names()`

Examples

```
library(tidyr)

ln <- gd_us %>%
  get_higher_level_names(level_name = "state",
                         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 = NULL, inherited = FALSE, add_prefix = TRUE)

## 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 in uppercase.

Value

A `tibble` object.

See Also

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

Examples

```
library(tidyr)

ld <- gd_us %>%
  get_level_data(level_name = "state",
                 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 = NULL)

## 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

Other information output functions: `get_higher_level_names()`, `get_level_data()`, `get_level_layer()`, `get_level_names()`

Examples

```
library(tidyr)

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, the subrogate key, inherited attributes.

Usage

```
get_level_layer(
  gd,
  level_name = NULL,
  geometry = NULL,
  only_key = FALSE,
  surrogate_key = FALSE,
  inherited = FALSE,
  add_prefix = TRUE
)

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

Arguments

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

Details

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

Value

A sf object.

See Also

Other information output functions: get_higher_level_names(), get_level_data(), get_level_geometries(), get_level_names()

Examples

```
library(tidyr)
library(sf)

ll <- gd_us %>%
  get_level_layer(level_name = "division",
                  only_key = TRUE,
                  surrogate_key = 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

Other information output functions: get_higher_level_names(), get_level_data(), get_level_geometries(), get_level_layer()

Examples

```
library(tidyr)

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

`get_unrelated_instances` *Get unrelated instances*

Description

Given two levels between which an explicit relationship is defined, it returns the lower-level instances that are not related to any higher-level instances.

Usage

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

## 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`.

See Also

Other level association functions: `add_level()`, `complete_relation_by_geography()`, `geodimension()`, `relate_levels()`

Examples

```
library(tidyr)
library(sf)

region <-
  geolevel(name = "region",
           layer = layer_us_region,
```

```
            key = c("geoid"))

division <-
  geolevel(name = "division",
           layer = layer_us_division,
           key = c("geoid"))

gd <-
  geodimension(name = "gd_us",
               level = region) %>%
  add_level(division)

gd <- gd %>%
  relate_levels(lower_level_name = "division",
                upper_level_name = "region",
                by_geography = TRUE)

ui <- gd %>%
  get_unrelated_instances(lower_level_name = "division",
                          upper_level_name = "region")
```

`layer_us_city` `layer_us_city`

Description

Point geometry layer, with data for US cities.

Usage

```
layer_us_city
```

Format

A `sf` object.

Source

https://www.census.gov

layer_us_county layer_us_county

Description

Polygon geometry layer, with data for US counties.

Usage

```
layer_us_county
```

Format

A sf object.

Source

https://www.census.gov

layer_us_division layer_us_division

Description

Polygon geometry layer, with data for US divisions.

Usage

```
layer_us_division
```

Format

A sf object.

Source

https://www.census.gov

layer_us_nation layer_us_nation

Description

Polygon geometry layer, with data for US nation.

Usage

```
layer_us_nation
```

Format

A sf object.

Source

https://www.census.gov

layer_us_region layer_us_region

Description

Polygon geometry layer, with data for US regions.

Usage

```
layer_us_region
```

Format

A sf object.

Source

https://www.census.gov

layer_us_state layer_us_state

Description

Polygon geometry layer, with data for US states.

Usage

```
layer_us_state
```

Format

A sf object.

Source

https://www.census.gov

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 = NULL,
  lower_level_attributes = NULL,
  upper_level_name = NULL,
  upper_level_key = NULL,
  by_geography = FALSE
)

## 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.

To use the geometric relationship, it must be explicitly indicated by the Boolean parameter.

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.

As a special case, if the top level has only one instance, it is not necessary to specify any attributes to define the relationship.

Value

A geodimension.

See Also

Other level association functions: add_level(), complete_relation_by_geography(), geodimension(), get_unrelated_instances()

Examples

```
library(tidyr)
library(sf)

region <-
  geolevel(name = "region",
           layer = layer_us_region,
           key = c("geoid"))

division <-
  geolevel(name = "division",
           layer = layer_us_division,
           key = c("geoid"))

gd <-
  geodimension(name = "gd_us",
```

```
                level = region) %>%
  add_level(division)

gd <- gd %>%
  relate_levels(lower_level_name = "division",
                upper_level_name = "region",
                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

Other configuration functions: `transform_crs()`

Examples

```
library(tidyr)

gds <- gd_us %>%
  select_levels(level_names = c("division", "region", "nation"))
```

`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

Other configuration functions: `select_levels()`

Index