

# Package ‘geomultistar’

September 30, 2020

**Type** Package

**Title** Multidimensional Queries Enriched with Geographic Data

**Version** 1.0.0

**Description** Multidimensional systems allow complex queries to be carried out in an easy way. The geographical dimension, together with the temporal dimension, plays a fundamental role in multidimensional systems. Through this package, vector layers can be associated to the attributes of geographic dimensions, so that the results of multidimensional queries can be obtained directly as vector layers. The multidimensional structures on which we can define the queries can be created from a flat table or imported directly using functions from this package.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Language** en-GB

**LazyData** true

**RoxygenNote** 7.1.1

**Imports** dplyr, readr, tibble, tidyr, tidysselect, starschemar, generics, purrr, 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](mailto:jsamos@ugr.es)>

**Repository** CRAN

**Date/Publication** 2020-09-30 08:40:09 UTC

R topics documented:

|                                  |           |
|----------------------------------|-----------|
| add_dimension . . . . .          | 2         |
| add_facts . . . . .              | 4         |
| define_geoattribute . . . . .    | 6         |
| geomultistar . . . . .           | 8         |
| get_empty_geoinstances . . . . . | 8         |
| mrs_fact_age . . . . .           | 9         |
| mrs_fact_cause . . . . .         | 10        |
| mrs_when . . . . .               | 10        |
| mrs_where . . . . .              | 11        |
| mrs_who . . . . .                | 11        |
| multistar . . . . .              | 12        |
| relate_dimension . . . . .       | 12        |
| run_geoquery . . . . .           | 14        |
| uk_london_boroughs . . . . .     | 16        |
| usa_cities . . . . .             | 16        |
| usa_counties . . . . .           | 17        |
| usa_divisions . . . . .          | 18        |
| usa_nation . . . . .             | 19        |
| usa_regions . . . . .            | 19        |
| usa_states . . . . .             | 20        |
| <b>Index</b>                     | <b>21</b> |

---

|               |   |
|---------------|---|
| add_dimension | <i>Add a dimension table to a multistar</i> |
|---------------|---|

---

Description

To add a dimension table to a multistar object, we must indicate the name that we give to the dimension, the tibble that contains the data and the name of the attribute corresponding to the table primary key.

Usage

```
add_dimension(  
  ms,  
  dimension_name = NULL,  
  dimension_table = NULL,  
  dimension_key = NULL,  
  fact_name = NULL,  
  fact_key = NULL,  
  key_as_data = FALSE  
)  
  
## S3 method for class 'multistar'  
add_dimension(  
  ms,
```

```

ms,
dimension_name = NULL,
dimension_table = NULL,
dimension_key = NULL,
fact_name = NULL,
fact_key = NULL,
key_as_data = FALSE
)

```

## Arguments

|                              |   |
|------------------------------|---|
| <code>ms</code>              | A multistar object.   |
| <code>dimension_name</code>  | A string, name of dimension table.  |
| <code>dimension_table</code> | A tibble, dimension table.  |
| <code>dimension_key</code>   | A string, name of the dimension primary key.  |
| <code>fact_name</code>       | A string, name of fact table.   |
| <code>fact_key</code>        | A string, name of the dimension foreign key.  |
| <code>key_as_data</code>     | A boolean, define the primary key as an attribute of the dimension accessible in queries? |

## Details

We cannot add a dimension without defining a correspondence with one of the multistar's fact tables. We have to define the name of the fact table and the name of its foreign key. The referential integrity of the instances of the facts is checked.

The attribute that is used as the primary key will no longer be accessible for queries (its function is considered to be exclusively related to facts). If you want to use it for queries, it must be explicitly indicated by the boolean parameter `key_as_data`.

## Value

A multistar.

## See Also

Other multistar functions: [add\\_facts\(\)](#), [multistar\(\)](#), [relate\\_dimension\(\)](#)

## Examples

```

library(tidyr)

ms <- multistar() %>%
  add_facts(
    fact_name = "mrs_age",
    fact_table = mrs_fact_age,
    measures = "n_deaths",
    nrow_agg = "count"
  )

```

```

) %>%
add_facts(
  fact_name = "mrs_cause",
  fact_table = mrs_fact_cause,
  measures = c("pneumonia_and_influenza_deaths", "other_deaths"),
  nrow_agg = "nrow_agg"
) %>%
add_dimension(
  dimension_name = "where",
  dimension_table = mrs_where,
  dimension_key = "where_pk",
  fact_name = "mrs_age",
  fact_key = "where_fk"
) %>%
add_dimension(
  dimension_name = "when",
  dimension_table = mrs_when,
  dimension_key = "when_pk",
  fact_name = "mrs_age",
  fact_key = "when_fk",
  key_as_data = TRUE
) %>%
add_dimension(
  dimension_name = "who",
  dimension_table = mrs_who,
  dimension_key = "who_pk",
  fact_name = "mrs_age",
  fact_key = "who_fk"
)

```

---

add\_facts

---

Add a fact table to a multistar

---

## Description

To add a fact table to a multistar object, we must indicate the name that we give to the facts, the tibble that contains the data and a vector of attribute names corresponding to the measures.

## Usage

```

add_facts(
  ms,
  fact_name = NULL,
  fact_table = NULL,
  measures = NULL,
  agg_functions = NULL,
  nrow_agg = "nrow_agg"
)

```

```
## S3 method for class 'multistar'
add_facts(
  ms,
  fact_name = NULL,
  fact_table = NULL,
  measures = NULL,
  agg_functions = NULL,
  nrow_agg = "nrow_agg"
)
```

### Arguments

|                            |  |
|----------------------------|--|
| <code>ms</code>            | A multistar object.  |
| <code>fact_name</code>     | A string, name of fact table.  |
| <code>fact_table</code>    | A tibble, fact table.  |
| <code>measures</code>      | A vector of measure names.   |
| <code>agg_functions</code> | A vector of aggregation function names. If none is indicated, the default is SUM. Additionally they can be MAX or MIN. |
| <code>nrow_agg</code>      | A string, measurement name for the number of rows aggregated. If it does not exist, it is added to the table.          |

### Details

Associated with each measurement, an aggregation function is required, which by default is SUM. It that can be SUM, MAX or MIN. Mean is not considered among the possible aggregation functions: The reason is that calculating the mean by considering subsets of data does not necessarily yield the mean of the total data.

An additional measurement, `nrow_agg`, corresponding to the number of aggregated rows is always added which, together with SUM, allows us to obtain the mean if needed. As the value of this parameter, you can specify an attribute of the table or the name that you want to assign to it (if it does not exist, it is added to the table).

### Value

A multistar.

### See Also

Other multistar functions: [add\\_dimension\(\)](#), [multistar\(\)](#), [relate\\_dimension\(\)](#)

### Examples

```
library(tidyr)

ms <- multistar() %>%
  add_facts(
    fact_name = "mrs_age",
```

```

    fact_table = mrs_fact_age,
    measures = "n_deaths",
    nrow_agg = "count"
  ) %>%
  add_facts(
    fact_name = "mrs_cause",
    fact_table = mrs_fact_cause,
    measures = c("pneumonia_and_influenza_deaths", "other_deaths"),
    nrow_agg = "nrow_agg"
  )

```

---

define\_geoattribute     *Define geographic attributes*

---

### Description

Defines a geographic attributes in two possible ways: Associates the instances of attributes of the geographic dimension with the instances of a geographic layer or defines it from the geometry of another previously defined geographic attribute. Multiple attributes can be specified in the attribute parameter.

### Usage

```

define_geoattribute(
  gms,
  dimension = NULL,
  attribute = NULL,
  from_layer = NULL,
  by = NULL,
  from_attribute = NULL
)

## S3 method for class 'geomultistar'
define_geoattribute(
  gms,
  dimension = NULL,
  attribute = NULL,
  from_layer = NULL,
  by = NULL,
  from_attribute = NULL
)

```

### Arguments

|           |                           |
|-----------|---------------------------|
| gms       | A geomultistar object.    |
| dimension | A string, dimension name. |

|                             |  |
|-----------------------------|--|
| <code>attribute</code>      | A vector, attribute names.   |
| <code>from_layer</code>     | A sf object.   |
| <code>by</code>             | a vector of correspondence of attributes of the dimension with the sf layer structure. |
| <code>from_attribute</code> | A string, attribute name.  |

## Details

If defined from a layer (`from_layer` parameter), additionally the attributes used for the join between the tables (dimension and layer tables) must be indicated (`by` parameter).

If defined from another attribute, it should have a finer granularity, to obtain the result by grouping its instances.

If no value is indicated in the `attribute` parameter, it is defined for all those attributes of the dimension that do not have any previous definition, they are obtained from the attribute indicated in the `from_attribute` parameter.

## Value

A `geomultistar` object.

## See Also

Other geo functions: [geomultistar\(\)](#), [get\\_empty\\_geoinstances\(\)](#), [run\\_geoquery\(\)](#)

## Examples

```
library(tidyr)
library(starschema)
library(sf)

gms <- geomultistar(ms = ms_mrs, geodimension = "where") %>%
  define_geoattribute(
    attribute = "city",
    from_layer = usa_cities,
    by = c("city" = "city", "state" = "state")
  )

gms <- gms %>%
  define_geoattribute(attribute = c("region", "all_where"),
    from_attribute = "city")

gms <- gms %>%
  define_geoattribute(from_attribute = "city")

gms <- gms %>%
  define_geoattribute(attribute = "all_where",
    from_layer = usa_nation)
```

---

|              |                              |
|--------------|------------------------------|
| geomultistar | geomultistar <i>S3 class</i> |
|--------------|------------------------------|

---

### Description

A geomultistar object is created. Dimensions that contain geographic information are indicated.

### Usage

```
geomultistar(ms = NULL, geodimension = NULL)
```

### Arguments

|              |                              |
|--------------|------------------------------|
| ms           | A multistar structure.       |
| geodimension | A vector of dimension names. |

### Value

A geomultistar object.

### See Also

Other geo functions: [define\\_geoattribute\(\)](#), [get\\_empty\\_geoinstances\(\)](#), [run\\_geoquery\(\)](#)

### Examples

```
library(tidyr)
library(starschema)

ms_mrs <- ct_mrs %>%
  constellation_as_multistar()

gms <- geomultistar(ms = ms_mrs, geodimension = "where")
```

---

|                        |  |
|------------------------|--|
| get_empty_geoinstances | <i>Get empty instances of a geographic attribute</i> |
|------------------------|--|

---

### Description

Gets the instances of the given geographic attribute that do not have a geometry associated with them.



**Usage**

```
get_empty_geoinstances(gms, dimension = NULL, attribute = NULL)

## S3 method for class 'geomultistar'
get_empty_geoinstances(gms, dimension = NULL, attribute = NULL)
```

**Arguments**

**gms**                    A geomultistar object.  
**dimension**            A string, dimension name.  
**attribute**            A string, attribute name.

**Value**

A sf object.

**See Also**

Other geo functions: [define\\_geoattribute\(\)](#), [geomultistar\(\)](#), [run\\_geoquery\(\)](#)

**Examples**

```
library(tidyr)
library(starschema)
library(sf)

gms <- geomultistar(ms = ms_mrs, geodimension = "where") %>%
  define_geoattribute(
    attribute = "city",
    from_layer = usa_cities,
    by = c("city" = "city", "state" = "state")
  )

empty <- gms %>%
  get_empty_geoinstances(attribute = "city")
```

---

|              |                 |
|--------------|-----------------|
| mrs_fact_age | <i>Fact age</i> |
|--------------|-----------------|

---

**Description**

*Fact age* table of the Mortality Reporting System. Defined from *ms\_mrs*. Foreign keys have been renamed, only a *when* dimension has been considered, the type for the *when* dimension has been changed.

**Usage**

```
mrs_fact_age
```

**Format**

A tibble.

**Source**

<https://CRAN.R-project.org/package=starschemar>

---

|                |                   |
|----------------|-------------------|
| mrs_fact_cause | <i>Fact cause</i> |
|----------------|-------------------|

---

**Description**

*Fact cause* table of the Mortality Reporting System. Defined from `ms_mrs`. Foreign keys have been renamed, only a *when* dimension has been considered, the type for the *when* dimension has been changed.

**Usage**

```
mrs_fact_cause
```

**Format**

A tibble.

**Source**

<https://CRAN.R-project.org/package=starschemar>

---

|          |                       |
|----------|-----------------------|
| mrs_when | <i>Dimension when</i> |
|----------|-----------------------|

---

**Description**

*When* dimension table of the Mortality Reporting System. Defined from `ms_mrs`. The primary key has been renamed and its type has been changed. The other attributes have also been renamed.

**Usage**

```
mrs_when
```

**Format**

A tibble.

**Source**

<https://CRAN.R-project.org/package=starschemar>

---

|           |                        |
|-----------|------------------------|
| mrs_where | <i>Dimension where</i> |
|-----------|------------------------|

---

**Description**

*Where* dimension table of the Mortality Reporting System. Defined from `ms_mrs`. The primary key has been renamed.

**Usage**

`mrs_where`

**Format**

A tibble.

**Source**

<https://CRAN.R-project.org/package=starschemar>

---

|         |                      |
|---------|----------------------|
| mrs_who | <i>Dimension who</i> |
|---------|----------------------|

---

**Description**

*Who* dimension table of the Mortality Reporting System. Defined from `ms_mrs`. The primary key has been renamed.

**Usage**

`mrs_who`

**Format**

A tibble.

**Source**

<https://CRAN.R-project.org/package=starschemar>

---

|           |                    |
|-----------|--------------------|
| multistar | multistar S3 class |
|-----------|--------------------|

---

**Description**

Creates an empty `multistar` object that allows you to import fact and dimension tables.

**Usage**

```
multistar()
```

**Value**

A `multistar` object.

**See Also**

Other `multistar` functions: [add\\_dimension\(\)](#), [add\\_facts\(\)](#), [relate\\_dimension\(\)](#)

**Examples**

```
ms <- multistar()
```

---

|                  |   |
|------------------|---|
| relate_dimension | Relate a dimension table to a fact table in a multistar |
|------------------|---|

---

**Description**

Adding a dimension to a `multistar` can only relate to a fact table. You can then relate to other fact tables in the `multistar` using this function. The name of the fact table and its foreign key must be indicated. The referential integrity of the instances of the facts is checked.

**Usage**

```
relate_dimension(ms, dimension_name = NULL, fact_name = NULL, fact_key = NULL)
```

```
## S3 method for class 'multistar'
```

```
relate_dimension(ms, dimension_name = NULL, fact_name = NULL, fact_key = NULL)
```

**Arguments**

|                             |  |
|-----------------------------|--|
| <code>ms</code>             | A <code>multistar</code> object.             |
| <code>dimension_name</code> | A string, name of dimension table.           |
| <code>fact_name</code>      | A string, name of fact table.                |
| <code>fact_key</code>       | A string, name of the dimension foreign key. |

**Value**

A multistar.

**See Also**

Other multistar functions: [add\\_dimension\(\)](#), [add\\_facts\(\)](#), [multistar\(\)](#)

**Examples**

```
library(tidyr)

ms <- multistar() %>%
  add_facts(
    fact_name = "mrs_age",
    fact_table = mrs_fact_age,
    measures = "n_deaths",
    nrow_agg = "count"
  ) %>%
  add_facts(
    fact_name = "mrs_cause",
    fact_table = mrs_fact_cause,
    measures = c("pneumonia_and_influenza_deaths", "other_deaths"),
    nrow_agg = "nrow_agg"
  ) %>%
  add_dimension(
    dimension_name = "where",
    dimension_table = mrs_where,
    dimension_key = "where_pk",
    fact_name = "mrs_age",
    fact_key = "where_fk"
  ) %>%
  add_dimension(
    dimension_name = "when",
    dimension_table = mrs_when,
    dimension_key = "when_pk",
    fact_name = "mrs_age",
    fact_key = "when_fk",
    key_as_data = TRUE
  ) %>%
  add_dimension(
    dimension_name = "who",
    dimension_table = mrs_who,
    dimension_key = "who_pk",
    fact_name = "mrs_age",
    fact_key = "who_fk"
  ) %>%
  relate_dimension(dimension_name = "where",
    fact_name = "mrs_cause",
    fact_key = "where_fk") %>%
  relate_dimension(dimension_name = "when",
    fact_name = "mrs_cause",
    fact_key = "when_fk")
```

---

run\_geoquery

---

*Get a geographic vector from a query*


---

## Description

After defining a query and geographic dimensions, run the query and select the geographic data associated with it to get a geographic data layer as the result.

## Usage

```
run_geoquery(
  dq,
  unify_by_grain = TRUE,
  fact = NULL,
  dimension = NULL,
  attribute = NULL
)

## S3 method for class 'dimensional_query'
run_geoquery(
  dq,
  unify_by_grain = TRUE,
  fact = NULL,
  dimension = NULL,
  attribute = NULL
)
```

## Arguments

|                |   |
|----------------|---|
| dq             | A dimensional_query object.                             |
| unify_by_grain | A boolean, unify facts with the same grain.             |
| fact           | A string, name of the fact.                             |
| dimension      | A string, name of the geographic dimension.             |
| attribute      | A string, name of the geographic attribute to consider. |

## Details

In the case of having several fact tables, as an option, we can indicate if we do not want to unify the facts in the case of having the same grain.

If the result only has one fact table, it is not necessary to provide its name. Nor is it necessary to indicate the name of the geographic dimension if there is only one available.

If no attribute is specified, the geographic attribute of the result with finer granularity is selected.

**Value**

A sf object.

**See Also**

Other geo functions: [define\\_geoattribute\(\)](#), [geomultistar\(\)](#), [get\\_empty\\_geoinstances\(\)](#)

**Examples**

```
library(tidyr)
library(starschema)
library(sf)

gms <- geomultistar(ms = ms_mrs, geodimension = "where") %>%
  define_geoattribute(
    attribute = "city",
    from_layer = usa_cities,
    by = c("city" = "city", "state" = "state")
  ) %>%
  define_geoattribute(
    attribute = "state",
    from_layer = usa_states,
    by = c("state" = "state")
  ) %>%
  define_geoattribute(attribute = "region",
                      from_attribute = "state") %>%
  define_geoattribute(attribute = "all_where",
                      from_layer = usa_nation)

gdq <- dimensional_query(gms) %>%
  select_dimension(name = "where",
                  attributes = c("state", "city")) %>%
  select_dimension(name = "when",
                  attributes = c("when_happened_year", "when_happened_week")) %>%
  select_fact(
    name = "mrs_age",
    measures = c("n_deaths")
  ) %>%
  select_fact(name = "mrs_cause",
              measures = c("pneumonia_and_influenza_deaths", "other_deaths")) %>%
  filter_dimension(name = "when", when_happened_week <= "03") %>%
  filter_dimension(name = "where", state == "MA")

sf <- gdq %>%
  run_geoquery()
```

---

|                    |                           |
|--------------------|---------------------------|
| uk_london_boroughs | <i>UK London Boroughs</i> |
|--------------------|---------------------------|

---

### Description

From the original dataset, some fields have been selected and renamed.

### Usage

```
uk_london_boroughs
```

### Format

A sf.

### Details

Since not so much detail is needed, the geometry has been simplified 20 m.

### Source

<https://data.london.gov.uk/dataset/statistical-gis-boundary-files-london>

### Examples

```
# Read by:
#
# filepath <- "data/London_Borough_Excluding_MHW/London_Borough_Excluding_MHW.shp"
# uk_london_boroughs <- st_read(filepath) %>%
#   dplyr::select(
#     borough = NAME,
#     gss_code = GSS_CODE
#   ) %>%
#   st_simplify(dTolerance = 20)
```

---

|            |                         |
|------------|-------------------------|
| usa_cities | <i>USA Cities, 2014</i> |
|------------|-------------------------|

---

### Description

From the original dataset, some fields have been selected and renamed, and only includes the Mortality Reporting System cities.

### Usage

```
usa_cities
```



**Format**

A sf.

**Source**

<https://earthworks.stanford.edu/catalog/stanford-bx729wr3020>

**Examples**

```
# Read by:
#
# filepath <- "data/citiesx010g/citiesx010g.shp"
# usa_cities <- st_read(filepath) %>%
#   dplyr::select(
#     gnis_id = GNIS_ID,
#     ansi_code = ANSICODE,
#     city = NAME,
#     state = STATE,
#     county = COUNTY,
#     latitude = LATITUDE,
#     longitude = LONGITUDE,
#     elev_m = ELEV_IN_M
#   )
```

---

usa\_counties

*USA Counties, 2018*

---

**Description**

From the original dataset, some fields have been selected and renamed, and only includes the Mortality Reporting System counties.

**Usage**

```
usa_counties
```

**Format**

A sf.

**Details**

Some counties appear with the same repeated name within the same state, they are the following: Baltimore, MD; Richmond, VA; St. Louis, MO. Since they are accessed by name (county and state), those of the same name within the state have been grouped together.

**Source**

[https://www2.census.gov/geo/tiger/GENZ2018/shp/cb\\_2018\\_us\\_county\\_20m.zip](https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_county_20m.zip)

**Examples**

```
# Read by:
#
# filepath <- "data/cb_2018_us_county_20m/cb_2018_us_county_20m.shp"
# usa_counties <- st_read(filepath) %>%
#   dplyr::select(
#     geo_id = GEOID,
#     state_fp = STATEFP,
#     county_fp = COUNTYFP,
#     county = NAME
#   )
# states <- sf::st_drop_geometry(usa_states[, c("geo_id", "state")])
# usa_counties <- usa_counties %>%
#   dplyr::left_join(states, by = c("state_fp" = "geo_id"))
```

---

usa\_divisions

*USA Divisions, 2018*

---

**Description**

From the original dataset, some fields have been selected and renamed.

**Usage**

```
usa_divisions
```

**Format**

A sf.

**Source**

[https://www2.census.gov/geo/tiger/GENZ2018/shp/cb\\_2018\\_us\\_division\\_20m.zip](https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_division_20m.zip)

**Examples**

```
# Read by:
#
# filepath <- "data/cb_2018_us_division_20m/cb_2018_us_division_20m.shp"
# usa_divisions <- st_read(filepath) %>%
#   dplyr::select(geo_id = GEOID,
#                 division = NAME)
```

---

|            |                         |
|------------|-------------------------|
| usa_nation | <i>USA Nation, 2018</i> |
|------------|-------------------------|

---

**Description**

From the original dataset, some fields have been selected and renamed.

**Usage**

```
usa_nation
```

**Format**

A sf.

**Source**

[https://www2.census.gov/geo/tiger/GENZ2018/shp/cb\\_2018\\_us\\_nation\\_20m.zip](https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_nation_20m.zip)

**Examples**

```
# Read by:
#
# filepath <- "data/cb_2018_us_nation_20m/cb_2018_us_nation_20m.shp"
# usa_nation <- st_read(filepath) %>%
#   dplyr::select(geo_id = GEOID,
#                 name = NAME)
```

---

|             |                          |
|-------------|--------------------------|
| usa_regions | <i>USA Regions, 2018</i> |
|-------------|--------------------------|

---

**Description**

From the original dataset, some fields have been selected and renamed.

**Usage**

```
usa_regions
```

**Format**

A sf.

**Source**

[https://www2.census.gov/geo/tiger/GENZ2018/shp/cb\\_2018\\_us\\_region\\_20m.zip](https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_region_20m.zip)

## Examples

```
# Read by:
#
# filepath <- "data/cb_2018_us_region_20m/cb_2018_us_region_20m.shp"
# usa_regions <- st_read(filepath) %>%
#   dplyr::select(geo_id = GEOID,
#                 region = NAME)
```

---

|            |                         |
|------------|-------------------------|
| usa_states | <i>USA States, 2018</i> |
|------------|-------------------------|

---

## Description

From the original dataset, some fields have been selected and renamed, and only includes the Mortality Reporting System states.

## Usage

```
usa_states
```

## Format

A sf.

## Source

[https://www2.census.gov/geo/tiger/GENZ2018/shp/cb\\_2018\\_us\\_state\\_20m.zip](https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_state_20m.zip)

## Examples

```
# Read by:
#
# filepath <- "data/cb_2018_us_state_20m/cb_2018_us_state_20m.shp"
# usa_states <- st_read(filepath) %>%
#   dplyr::select(geo_id = GEOID,
#                 state = STUSPS,
#                 state_name = NAME)
```

# Index

## \* datasets

- mrs\_fact\_age, [9](#)
- mrs\_fact\_cause, [10](#)
- mrs\_when, [10](#)
- mrs\_where, [11](#)
- mrs\_who, [11](#)
- uk\_london\_boroughs, [16](#)
- usa\_cities, [16](#)
- usa\_counties, [17](#)
- usa\_divisions, [18](#)
- usa\_nation, [19](#)
- usa\_regions, [19](#)
- usa\_states, [20](#)

## \* geo functions

- define\_geoattribute, [6](#)
- geomultistar, [8](#)
- get\_empty\_geoinstances, [8](#)
- run\_geoquery, [14](#)

## \* multistar functions

- add\_dimension, [2](#)
- add\_facts, [4](#)
- multistar, [12](#)
- relate\_dimension, [12](#)

[add\\_dimension](#), [2](#), [5](#), [12](#), [13](#)

[add\\_facts](#), [3](#), [4](#), [12](#), [13](#)

[define\\_geoattribute](#), [6](#), [8](#), [9](#), [15](#)

[geomultistar](#), [7](#), [8](#), [9](#), [15](#)

[get\\_empty\\_geoinstances](#), [7](#), [8](#), [8](#), [15](#)

[mrs\\_fact\\_age](#), [9](#)

[mrs\\_fact\\_cause](#), [10](#)

[mrs\\_when](#), [10](#)

[mrs\\_where](#), [11](#)

[mrs\\_who](#), [11](#)

[multistar](#), [3](#), [5](#), [12](#), [13](#)

[relate\\_dimension](#), [3](#), [5](#), [12](#), [12](#)

[run\\_geoquery](#), [7–9](#), [14](#)

[uk\\_london\\_boroughs](#), [16](#)

[usa\\_cities](#), [16](#)

[usa\\_counties](#), [17](#)

[usa\\_divisions](#), [18](#)

[usa\\_nation](#), [19](#)

[usa\\_regions](#), [19](#)

[usa\\_states](#), [20](#)