Package ‘areal’

May 21, 2019

Type Package
Title Areal Weighted Interpolation
Version 0.1.5
Description A pipeable, transparent implementation of areal weighted interpolation with support for interpolating multiple variables in a single function call. These tools provide a full-featured workflow for validation and estimation that fits into both modern data management (e.g. tidyverse) and spatial data (e.g. sf) frameworks.
Depends R (>= 3.3)
License GPL-3
URL https://github.com/slu-openGIS/areal
BugReports https://github.com/slu-openGIS/areal/issues
Encoding UTF-8
LazyData true
Imports dplyr, glue, purrr, rlang, sf
RoxygenNote 6.1.1
Suggests knitr, rmarkdown, testthat, covr
VignetteBuilder knitr
NeedsCompilation no
Author Christopher Prener [aut, cre] (<https://orcid.org/0000-0002-4310-9888>), Charlie Revord [aut]
Maintainer Christopher Prener <chris.prener@slu.edu>
Repository CRAN
Date/Publication 2019-05-21 07:30:07 UTC

R topics documented:

ar_stl_asthma .......................................................... 2
ar_stl_race ............................................................ 3
ar_stl_asthma

Asthma in St. Louis by Census Tract, 2017

Description

A simple features data set containing the geometry and asthma estimates from the Centers for Disease Control for St. Louis.

Usage

data(ar_stl_asthma)

Format

A data frame with 106 rows and 24 variables:

GEOID full GEOID string
STATEFP state FIPS code
COUNTYFP county FIPS code
TRACTCE tract FIPS code
NAMELSAD tract name
ALAND area of tract land, square meters
AWATER area of tract water, square meters
ASTHMA percent of residents with current asthma diagnosis, estimated
gometry simple features geometry

Source

Centers for Disease Control’s 500 Cities Data
**ar_stl_race**

**Examples**

```r
str(ar_stl_asthma)
head(ar_stl_asthma)
summary(ar_stl_asthma$ASTHMA)
```

---

**ar_stl_race**  
**Race in St. Louis by Census Tract, 2017**

---

**Description**

A simple features data set containing the geometry and associated attributes for the 2013-2017 American Community Survey estimates for race in St. Louis.

**Usage**

```r
data(ar_stl_race)
```

**Format**

A data frame with 106 rows and 24 variables:

- **GEOID** full GEOID string
- **STATEFP** state FIPS code
- **COUNTYFP** county FIPS code
- **TRACTCE** tract FIPS code
- **NAMELSAD** tract name
- **ALAND** area of tract land, square meters
- **AWATER** area of tract water, square meters
- **TOTAL_E** total population count, estimated
- **TOTAL_M** total population count, margin of error
- **WHITE_E** white population count, estimated
- **WHITE_M** white population count, margin of error
- **BLACK_E** black population count, estimated
- **BLACK_M** black population count, margin of error
- **AIAN_E** American Indian and Alaskan Native population count, estimated
- **AIAN_M** American Indian and Alaskan Native population count, margin of error
- **ASIAN_E** Asian population count, estimated
- **ASIAN_M** Asian population count, margin of error
- **NHPI_E** Native Hawaiian and Pacific Islander population count, estimated
- **NHPI_M** Native Hawaiian and Pacific Islander population count, margin of error
ar_stl_wards

<table>
<thead>
<tr>
<th>OBJECT_E</th>
<th>other race population count, estimated</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTHER_M</td>
<td>other race population count, margin of error</td>
</tr>
<tr>
<td>TWOPLUS_E</td>
<td>two or more races population count, estimated</td>
</tr>
<tr>
<td>TWOPLUS_M</td>
<td>two or more races population count, margin of error</td>
</tr>
</tbody>
</table>

geometry: simple features geometry

Source

tidycensus package

Examples

```r
str(ar_stl_race)
head(ar_stl_race)
summary(ar_stl_race$ALAND)
```

---

ar_stl_wards  Ward Boundaries in St. Louis, 2010

Description

A simple features data set containing the 2010 Ward boundaries, which are used as districts for Alderpersons who serve as elected representatives. The OBJECTID and AREA columns are included to simulate "real" data that may have superfluous or unclear columns.

Usage

data(ar_stl_wards)

Format

A data frame with 28 rows and 4 variables:

- **OBJECTID**: Artifact from ESRI data creation
- **WARD**: Ward number
- **AREA**: area of each ward
- **geometry**: simple features geometry

Source

City of St. Louis

Examples

```r
str(ar_stl_wards)
head(ar_stl_wards)
summary(ar_stl_wards$AREA)
```
Clipped Ward Boundaries in St. Louis, 2010

Description
A simple features data set containing the 2010 Ward boundaries, which are used as districts for Alderpersons who serve as elected representatives. This version of the ward boundary has been modified so that the wards only extend to the Mississippi River shoreline.

Usage
data(ar_stl_wardsClipped)

Format
A data frame with 28 rows and 2 variables:

WARD  Ward number
geometry  simple features geometry

Source
City of St. Louis

Examples
str(ar_stl_wardsClipped)
head(ar_stl_wardsClipped)

Validating Data for Interpolation

Description
ar_validate executes a series of logic tests for sf object status, shared coordinates between source and target data, appropriate project, and absence of variable name conflicts.

Usage
ar_validate(source, target, varList, method = "aw", verbose = FALSE)
Arguments

source | A sf object with data to be interpolated

target | A sf object that data should be interpolated to

varList | A vector of variable names to be added to the target object

method | The areal interpolation method validation is being performed for. This should be set to "aw". Additional functionality will be added as the package adds new interpolation techniques.

verbose | A logical scalar; if TRUE, a tibble with test results is returned

Value

If verbose is FALSE, a logical scalar is returned that is TRUE if all tests are passed and FALSE if one or more tests is failed. If verbose is TRUE, a tibble with detailed test results is returned.

See Also
c

Examples

ar_validate(source = ar_stl_asthma, target = ar_stl_wards, varList = "ASTHMA")

ar_validate(source = ar_stl_asthma, target = ar_stl_wards, varList = "ASTHMA", verbose = TRUE)

aw_aggregate | Aggregate Estimates Based on Target ID

Description

aw_aggregate sums the new estimates produced by aw_calculate based on the target id. These are then joined with the target data. This is the fourth step in the interpolation process after aw_weight.

Usage

aw_aggregate(.data, target, tid, interVar, newVar)

Arguments

.data | A given intersected dataset

target | A sf object that data should be interpolated to

tid | A unique identification number within target

interVar | A variable containing an interpolated value created by aw_calculate

newVar | Optional; a new field name to store the interpolated value in. If not specified, the interVar argument will be used as the new field name.
Value

A sf object with the interpolated value added to it.

Examples

library(dplyr)

race <- select(ar_stl_race, GEOID, TOTAL_E)
wards <- select(ar_stlwards, WARD)

wards %>%
  aw_intersect(source = race, areaVar = "area") %>%
  aw_total(source = race, id = GEOID, areaVar = "area", totalVar = "totalArea",
           weight = "sum", type = "extensive") %>%
  aw_weight(areaVar = "area", totalVar = "totalArea", areaWeight = "areaWeight") %>%
  aw_calculate(value = "TOTAL_E", areaWeight = "areaWeight") -> intersect

aw_aggregate(intersect, target = wards, tid = WARD, interVar = TOTAL_E)

----------

aw_calculate  Calculate Estimated Population

Description

aw_calculate multiplies the given value by the area weight. This is the fourth step in the interpolation process after aw_weight.

Usage

aw_calculate(.data, value, areaWeight, newVar)

Arguments

.data A given intersected dataset
value A column within source to be interpolated
areaWeight The name of the variable containing area weight per feature
newVar Optional; a new field name to store the interpolated value in. If not specified, the value argument will be used as the new field name.

Value

An intersected file of class sf with a new field of interest recalculated with area weight
aw_interpolate

Examples

```r
library(dplyr)

race <- select(ar_stl_race, GEOID, TOTAL_E)
wards <- select(ar_stl_wards, WARD)

wards %>%
  aw_intersect(source = race, areaVar = "area") %>%
  aw_total(source = race, id = GEOID, areaVar = "area", totalVar = "totalArea",
           weight = "sum", type = "extensive") %>%
  aw_weight(areaVar = "area", totalVar = "totalArea", areaWeight = "areaWeight") -> intersect

aw_calculate(intersect, value = "TOTAL_E", areaWeight = "areaWeight")
```

## aw_interpolate

### Interpolate Values

### Description

This is the core function within the package for areal weighted interpolation. It validates both data sources before interpolating one or more listed values from the source data into the target data.

### Usage

```r
aw_interpolate(.data, tid, source, sid, weight = "sum", output = "sf", extensive, intensive)
```

### Arguments

- `.data` A `sf` object that data should be interpolated to (this is referred to as the target elsewhere in the package).
- `.tid` A unique identification number within target
- `.source` A `sf` object with data to be interpolated
- `.sid` A unique identification number within source
- `.weight` For "extensive" interpolations, should be either "total" or "sum". For "intensive" interpolations, should be "sum". For mixed interpolations, this will only impact the calculation of the extensive variables.
- `.output` One of either "sf" or "tibble"
- `.extensive` A vector of quoted variable names to be treated as spatially extensive (e.g. population counts); optional if intensive is specified
- `.intensive` A vector of quoted variable names to be treated as spatially intensive (e.g. population density); optional if extensive is specified
Details

Areal weighted interpolation can be used for generating demographic estimates for overlapping but incongruent polygon features. It assumes that individual members of a population are evenly dispersed within the source features (an assumption not likely to hold in the real world). It also functions best when data are in a projected coordinate system, like the UTM coordinate system.

Value

A `sf` object or a `tibble` with the value or values interpolated into the target data.

See Also

c

Examples

```r
aw_interpolate(ar_stlwards, tid = WARD, source = ar_stl_race, sid = GEOID, weight = "sum", output = "sf", extensive = "TOTAL_E")

aw_interpolate(ar_stlwards, tid = WARD, source = ar_stl_asthma, sid = GEOID, weight = "sum", output = "tibble", intensive = "ASTHMA")
```

Description

`aw_intersect` intersects the source and target datasets and computes a new area field for the intersected data using the units associated with whatever project the data are currently in. This is the first step in the interpolation process after data validation and subsetting.

Usage

```r
aw_intersect(.data, source, areaVar)
```

Arguments

- `.data`: A `sf` object that data should be interpolated to.
- `source`: A `sf` object with data to be interpolated.
- `areaVar`: The name of the new area variable to be calculated.

Value

A `sf` object with the intersected data and new area field.
Examples

library(dplyr)

race <- select(ar_stl_race, GEOID, TOTAL_E)
wards <- select(ar_stl_wards, WARD)

aw_intersect(wards, source = race, areaVar = "area")

Description

Provides a preview of the weight options for areal weighted interpolation. This can be useful for selecting the final specification for aw_interpolate without having to construct a pipeline of all of the subfunctions manually.

Usage

aw_preview_weights(.data, tid, source, sid, type)

Arguments

.data A sf object that data should be interpolated to (this is referred to as the target elsewhere in the package).

 tid A unique identification number within target

 source A sf object with data to be interpolated

 sid A unique identification number within source

 type One of either "extensive" (if the data are spatially extensive e.g. population counts), "intensive" (if the data are spatially intensive e.g. population density), or "mixed" (if the data include both extensive and intensive values). If "extensive", the sum is returned for the interpolated value. If "intensive", the mean is returned for the interpolated value. If "mixed", vectors named "extensive" and "intensive" containing the relevant variable names should be specified in the dots.

Value

A tibble with the areal weights that would be used for interpolation if type is either "extensive" or "intensive". If it is mixed, two tibbles (one for "extensive" and one for "intensive") are returned as a list.
**Examples**

```r
apaw_preview_weights(ar_stl_wards, tid = "WARD", source = ar_stl_race, sid = "GEOID",
type = "extensive")
apaw_preview_weights(ar_stl_wards, tid = "WARD", source = ar_stl_asthma, sid = "GEOID",
type = "intensive")
```

**aw_total**

*Calculate Total Area*

**Description**

`aw_total` produces a new total area field that contains the total area by `source` id. This is the second step in the interpolation process after `aw_intersect`.

**Usage**

```r
aw_total(.data, source, id, areaVar, totalVar, type, weight)
```

**Arguments**

- `.data` A `sf` object that has been intersected using `aw_intersect`
- `source` A `sf` object with data to be interpolated
- `id` A unique identification number
- `areaVar` The name of the variable measuring a feature's area, which is created as part of `aw_intersect`
- `totalVar` The name of a new total area field to be calculated
- `type` One of "intensive" or "extensive"
- `weight` One of "sum" or "total"

**Value**

A `sf` object with the intersected data and new total area field.

**Examples**

```r
library(dplyr)

race <- select(ar_stl_race, GEOID, TOTAL_E)
wards <- select(ar_stl_wards, WARD)

wards %>%
  aw_intersect(source = race, areaVar = "area") -> intersect

aw_total(intersect, source = race, id = GEOID, areaVar = "area",
totalVar = "totalArea", weight = "sum", type = "extensive")
```
**aw_verify**  
*Verify Correct Extensive-Sum Interpolation*

**Description**
Verify Correct Extensive-Sum Interpolation

**Usage**

```
aw_verify(source, sourceValue, result, resultValue)
```

**Arguments**

- `source`: A sf object with data to be interpolated
- `sourceValue`: A column within `source` to be interpolated
- `result`: A sf object with interpolated data
- `resultValue`: A column within `result` with the interpolated values

**Details**

`aw_verify` ensures that the sum of the resulting interpolated value is equal to the sum of the original source value. This functionality only works for interpolations that are extensive and use the sum approach to calculating areal weights.

**Value**

A logical scalar; if TRUE, these two values are equal.

**Examples**

```
result <- aw_interpolate(ar_stl_wards, tid = WARD, source = ar_stl_race, sid = GEOID,
                         weight = "sum", output = "tibble", extensive = "TOTAL_E")

aw_verify(source = ar_stl_race, sourceValue = TOTAL_E, result = result, resultValue = TOTAL_E)
```
**Description**

`aw_weight` creates an area weight field by dividing the area field by the total area field. This is the third step in the interpolation process after `aw_weight`.

**Usage**

```r
aw_weight(.data, areaVar, totalVar, areaWeight)
```

**Arguments**

- `.data` A sf object that has been intersected using `aw_intersect`
- `areaVar` The name of the variable measuring a feature’s area
- `totalVar` The name of the variable containing total area field by source id
- `areaWeight` The name of a new area weight field to be calculated

**Value**

A sf object with the intersected data and new area weight field.

**Examples**

```r
library(dplyr)

race <- select(ar_stl_race, GEOID, TOTAL_E)
wards <- select(ar_stl_wards, WARD)

wards %>%
  aw_intersect(source = race, areaVar = "area") %>%
  aw_total(source = race, id = GEOID, areaVar = "area", totalVar = "totalArea",
            weight = "sum", type = "extensive") -> intersect

aw_weight(intersect, areaVar = "area", totalVar = "totalArea", areaWeight = "areaWeight")
```
Index

*Topic datasets
   ar_stl.asthma, 2
   ar_stl.race, 3
   ar_stl.wards, 4
   ar_stl.wardsClipped, 5

ar_stl.asthma, 2
ar_stl.race, 3
ar_stl.wards, 4
ar_stl.wardsClipped, 5
ar_validate, 5
aw_aggregate, 6
aw_calculate, 6, 7
aw_interpolate, 8
aw_intersect, 9, 11, 13
aw_preview_weights, 10
aw_total, 11
aw_verify, 12
aw_weight, 6, 7, 13, 13

c, 6, 9