Package ‘areal’

July 23, 2020

Type Package

Title Areal Weighted Interpolation

Version 0.1.7

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 (>= 3.4)

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 7.1.1

Suggests knitr, rmarkdown, testthat, covr

VignetteBuilder knitr

NeedsCompilation no

Author Christopher Prener [aut, cre] (<https://orcid.org/0000-0002-4310-9888>), Charlie Revord [aut], Branson Fox [aut] (<https://orcid.org/0000-0002-4361-2811>)

Maintainer Christopher Prener <chris.prener@slu.edu>

Repository CRAN

Date/Publication 2020-07-23 17:10:02 UTC
\textbf{R topics documented:}

\begin{verbatim}
ar_stl_asthma .......................................................... 2
ar_stl_race ......................................................... 3
ar_stl_wards .......................................................... 4
ar_stl_wardsClipped ............................................... 5
ar_tessellate .......................................................... 6
ar_validate ............................................................ 6
aw_aggregate .......................................................... 7
aw_calculate .......................................................... 8
aw_interpolate ......................................................... 9
aw_intersect .......................................................... 10
aw_preview_weights .................................................. 11
aw_total ............................................................... 12
aw_verify ............................................................. 13
aw_weight ............................................................ 14
\end{verbatim}

\textbf{Index} 15

\begin{verbatim}
ar_stl_asthma  Asthma in St. Louis by Census Tract, 2017
\end{verbatim}

\textbf{Description}

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

\textbf{Usage}

\begin{verbatim}
data(ar_stl_asthma)
\end{verbatim}

\textbf{Format}

A data frame with 106 rows and 24 variables:

\begin{verbatim}
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
geometry simple features geometry
\end{verbatim}
AR_STL_RACE

Source
Centers for Disease Control’s 500 Cities Data

Examples

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

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
data(ar_stl_race)

Format
A data frame with 106 rows and 24 variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEOID</td>
<td>full GEOID string</td>
</tr>
<tr>
<td>STATEFP</td>
<td>state FIPS code</td>
</tr>
<tr>
<td>COUNTYFP</td>
<td>county FIPS code</td>
</tr>
<tr>
<td>TRACTCE</td>
<td>tract FIPS code</td>
</tr>
<tr>
<td>NAMELSAD</td>
<td>tract name</td>
</tr>
<tr>
<td>ALAND</td>
<td>area of tract land, square meters</td>
</tr>
<tr>
<td>A_WATER</td>
<td>area of tract water, square meters</td>
</tr>
<tr>
<td>TOTAL_E</td>
<td>total population count, estimated</td>
</tr>
<tr>
<td>TOTAL_M</td>
<td>total population count, margin of error</td>
</tr>
<tr>
<td>WHITE_E</td>
<td>white population count, estimated</td>
</tr>
<tr>
<td>WHITE_M</td>
<td>white population count, margin of error</td>
</tr>
<tr>
<td>BLACK_E</td>
<td>black population count, estimated</td>
</tr>
<tr>
<td>BLACK_M</td>
<td>black population count, margin of error</td>
</tr>
<tr>
<td>AIAN_E</td>
<td>American Indian and Alaskan native population</td>
</tr>
<tr>
<td>AIAN_M</td>
<td>American Indian and Alaskan native population,</td>
</tr>
<tr>
<td>ASIAN_E</td>
<td>Asian population count, estimated</td>
</tr>
<tr>
<td>ASIAN_M</td>
<td>Asian population count, margin of error</td>
</tr>
</tbody>
</table>
NHPI_E  native hawaiian and pacific islander populaton count, estimated
NHPI_M  native hawaiian and pacific islander populaton count, margin of error
OTHER_E  other race populaton count, estimated
OTHER_M  other race populaton count, margin of error
TWOPLUS_E  two or more races populaton count, estimated
TWOPLUS_M  two or more races populaton count, margin of error
geometry  simple features geometry

Source
tidycensus package

Examples
str(ar_stl_race)
head(ar_stl_race)
syntax(ar_stl_race$ALAND)

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
gometry  simple features geometry

Source
City of St. Louis
ar_stl_wardsClipped

Examples

str(ar_stl_wards)
head(ar_stl_wards)
summary(ar_stl_wards$AREA)

ar_stl_wardsClipped  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)
ar_tessellate  
Create Tessellations From SF Object

Description

Create Tessellations From SF Object

Usage

ar_tessellate(.data, shape = "square", size = 1)

Arguments

.data  
An object of class sf to tessellate from

shape  
One of 'square' or 'hexagon', the shape to make tessellations from

size  
Numeric multiplier for size of tessellations, default is one kilometer

Value

A sf object

Examples

ar_tessellate(ar_stl_wards)

ar_tessellate(ar_stl_wards, shape = "hexagon", size = .75)

ar_validate  
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)
\textbf{aw_aggregate}

\textit{Aggregate Estimates Based on Target ID}

**Description**

\texttt{aw_aggregate} sums the new estimates produced by \texttt{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 \texttt{aw_weight}.

**Usage**

\texttt{aw_aggregate(.data, target, tid, interVar, newVar)}

**Arguments**

- \texttt{.data} A given intersected dataset
- \texttt{target} A \texttt{sf} object that data should be interpolated to
- \texttt{tid} A unique identification number within target
- \texttt{interVar} A variable containing an interpolated value created by \texttt{aw_calculate}
- \texttt{newVar} Optional; a new field name to store the interpolated value in. If not specified, the \texttt{interVar} argument will be used as the new field name.
aw_calculate

Value

A sf object with the interpolated value added to it.

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") %>%
  aw_calculate(value = "TOTAL_E", areaWeight = "areaWeight") -> intersect

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

Description

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

Usage

```r
aw_calculate(.data, value, areaWeight, newVar)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.data</td>
<td>A given intersected dataset</td>
</tr>
<tr>
<td>value</td>
<td>A column within source to be interpolated</td>
</tr>
<tr>
<td>areaWeight</td>
<td>The name of the variable containing area weight per feature</td>
</tr>
<tr>
<td>newVar</td>
<td>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.</td>
</tr>
</tbody>
</table>

Value

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

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

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
everwhere 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. pop-
ulation counts); optional if intensive is specified

intensive A vector of quoted variable names to be treated as spatially intensive (e.g. pop-
ulation 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")
```

---

### aw_intersect

**Intersect Source and Target Data**

**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

```r
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

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

aw_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")
```
$\textbf{aw\_verify}$  

Verify Correct Extensive-Sum Interpolation

**Description**

Verify Correct Extensive-Sum Interpolation

**Usage**

$\texttt{aw\_verify(source, sourceValue, result, resultValue)}$

**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**

$\texttt{aw\_verify}$ 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**

```r
result <- aw_interpolate(ar_stlwards, 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)
```
aw_weight  

*Calculate Areal Weight*

**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**

```
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**

```
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

* datasets
  ar_stl_asthma, 2
  ar_stl_race, 3
  ar_stlwards, 4
  ar_stlwardsClipped, 5
  ar_stl_wards, 2
  ar_stl_race, 3
  ar_stl_wards, 4
  ar_stl_wardsClipped, 5
  ar_tessellate, 6
  ar_validate, 6
  aw_aggregate, 7
  aw_calculate, 7, 8
  aw_interpolate, 9
  aw_intersect, 10, 12, 14
  aw_preview_weights, 11
  aw_total, 12
  aw_verify, 13
  aw_weight, 7, 8, 14, 14
  c, 7, 10