

# Package: areal (via r-universe)

October 9, 2024

**Type** Package

**Title** Areal Weighted Interpolation

**Version** 0.1.8.9000

**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://chris-prener.github.io/areal/>

**BugReports** <https://github.com/chris-prener/areal/issues>

**Encoding** UTF-8

**LazyData** true

**Imports** dplyr, glue, purrr, rlang, sf

**RoxygenNote** 7.1.2

**Suggests** knitr, rmarkdown, testthat, covr

**VignetteBuilder** knitr

**Repository** <https://chris-prener.r-universe.dev>

**RemoteUrl** <https://github.com/chris-prener/areal>

**RemoteRef** HEAD

**RemoteSha** 07bda84887d9f2272babe91c2ccc1e438f27f162

## Contents

ar_stl_asthma . . . . .	2
ar_stl_race . . . . .	3
ar_stl_wards . . . . .	4

ar_stl_wardsClipped . . . . .	5
ar_tessellate . . . . .	5
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

<b>Index</b>	<b>15</b>
--------------	-----------

---

ar_stl_asthma	<i>Asthma in St. Louis by Census Tract, 2017</i>
---------------	--

---

## 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

**geometry** simple features geometry

## Source

Centers for Disease Control's 500 Cities Data

**Examples**

```
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**

```
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 populaton count, estimated

**TOTAL\_M** total populaton count, margin of error

**WHITE\_E** white populaton count, estimated

**WHITE\_M** white populaton count, margin of error

**BLACK\_E** black populaton count, estimated

**BLACK\_M** black populaton count, margin of error

**AIAN\_E** american indian and alskan native populaton count, estimated

**AIAN\_M** american indian and alskan native populaton count, margin of error

**ASIAN\_E** asian populaton count, estimated

**ASIAN\_M** asian populaton count, margin of error

**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)
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

```
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)
```

**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 <a href="#">aw_calculate</a>
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_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)
```

---

aw_calculate	<i>Calculate Estimated Population</i>
--------------	---------------------------------------

---

### 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

### 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	<i>Interpolate Values</i>
----------------	---------------------------

---

**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 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

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

```
aw_interpolate(ar_stl_wards, 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

```
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")
```

---

aw\_preview\_weights      *Preview Areal Weights*

---

## 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

<code>.data</code>	A sf object that data should be interpolated to (this is referred to as the target elsewhere in the package).
<code>tid</code>	A unique identification number within target
<code>source</code>	A sf object with data to be interpolated
<code>sid</code>	A unique identification number within source
<code>type</code>	One of either "extensive" (if the data are spatitally 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

```
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

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

### Arguments

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

### Value

A sf object with the intersected data and new total area 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") -> intersect

aw_total(intersect, source = race, id = GEOID, areaVar = "area",
         totalVar = "totalArea", weight = "sum", type = "extensive")
```

---

aw_verify	<i>Verify Correct Extensive-Sum Interpolation</i>
-----------	---

---

**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)
```

---

`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

<code>.data</code>	A sf object that has been intersected using <a href="#">aw_intersect</a>
<code>areaVar</code>	The name of the variable measuring a feature's area
<code>totalVar</code>	The name of the variable containing total area field by source id
<code>areaWeight</code>	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\_stl\_wards, 4
- ar\_stl\_wardsClipped, 5

  

- ar\_stl\_asthma, 2
- ar\_stl\_race, 3
- ar\_stl\_wards, 4
- ar\_stl\_wardsClipped, 5
- ar\_tessellate, 5
- 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, 6, 9