# areal: An R package for areal weighted interpolation

**Christopher G. Prener**[1] **and Charles K. Revord**[1]

**1** Department of Sociology and Anthropology, Saint Louis University, St. Louis, MO, USA

**DOI:** 10.21105/joss.01221

**Software**

- Review
- Repository
- Archive

**Submitted:** 18 January 2019
**Published:** 19 May 2019

## Introduction

Population data are often available at various levels of aggregation, such as census tracts or block groups in the United States or output areas in the United Kingdom. These units are often drawn for convenience and not because they represent some commonly accepted area in residents' lived experiences. Despite this, researchers often use them as proxies for neighborhoods since they come with readily available demographic data. Researchers who do wish to move past these proxies and produce population estimates for other geographies typically resort to developing their own, often manual, implementations of the areal weighted interpolation workflow (Qiu, Zhang, & Zhou, 2012). The lack of reliable and easy to use software that implements this technique has therefore served as an additional barrier to using local neighborhood boundaries in research.

Areal weighted interpolation is the most accessible technique for estimating population values for overlapping but incongruent areas, such as imputing neighborhood populations from census tracts. Unlike interpolation techniques, such as inverse distance weighted interpolation, that are appropriate for point data, areal weighted interpolation is designed specifically for working with data already aggregated to some set of polygon spatial features (N. S.-N. Lam, 1983). Estimating population values from these data to an overlapping but incongruent set of polygon features is known as the *polygon overlay problem* (Goodchild, 1978), with the original data known as the "source" data and the overlapping set of features known as the "target" data (Markoff & Shapiro, 1973). The estimation process varies based on whether data are spatially *extensive* (i.e. count data) or spatially *intensive* (i.e. ratios, percentages, means, or medians; Goodchild & Lam, 1980).

In estimating values for the target features, areal weighted interpolation makes a significant assumption that individuals are evenly spread throughout the source features (Qiu et al., 2012). This assumption often breaks down in practice. For example, a census tract with a large park or a neighborhood that has a significant commercial area alongside residential housing would violate this assumption. While more complex methods do exist to compensate for violating this assumption, there are few tools to implement them, and the strategies have mainly remained the focus of academics instead of GIS practitioners (Langford, 2007; Qiu et al., 2012).

## The areal R package

The `areal` package aims to reduce the barriers to the implementation of areal interpolation in spatial researchers' work. By implementing the process in `R`, we aim to improve the reproducibility of the interpolation process, which is often done manually using point-and-click desktop GIS software. We also aim to provide additional functionality that is not currently available in existing approaches found in the sf package (Pebesma, 2018), for example the `st_interpolate_aw()` function, while providing support for modern data management (e.g. `tidyverse`) and spatial data (e.g. `sf`) frameworks.

Therefore, we provide functions that:

1. validate data suitability for areal interpolation,
2. allow for frictionless, iterative interpolation of both spatially extensive (e.g., count) and intensive (e.g., ratio) data, and
3. provide a manual workflow for implementing the interpolation process as well as a single omnibus function.

The validation process checks for five conditions that must be met prior to interpolation:

1. Are both objects `sf` objects?
2. Do both objects share the same coordinate system?
3. Is that coordinate system planar?
4. Do the given variables exist in the source data?
5. Will interpolation overwrite columns in the target data?

This validation process can be run independently by end users via the `ar_validate()` function and is also run initially when `aw_interpolate()` is called. The validation process is aimed at helping users new to both `R` and areal interpolation ensure they have arranged their data correctly.

Unlike existing approaches, the `areal` package offers the ability to interpolate multiple variables in a single function call and provides a wider set of options for making these estimates. For spatially extensive interpolations, we offer a formula that matches the existing functionality in `sf`, which is based on the total area of the original source feature (specified with `weight = "total"`). Alternatively, we offer a second formula that is more suitable for data where there should be complete overlap between the source and target data, but there is not. Such incongruity could be due to data quality issues or variation in how different sources represent particular geographies. This uses a sum of the source feature areas remaining after the data are intersected as part of the spatial weight calculation process (specified with `weight = "sum"`).

Our package also provides a manual approach for calculating estimates alongside the primary function `aw_interpolate()`. This is important for users who need to unpack the interpolation workflow and diagnose data issues that occur during interpolation as well as programmers who want to use a portion of the workflow in their software. Finally, `areal` provides both spatial (`sf` object) and a-spatial (`tibble` object) options for output.

## A Short Example

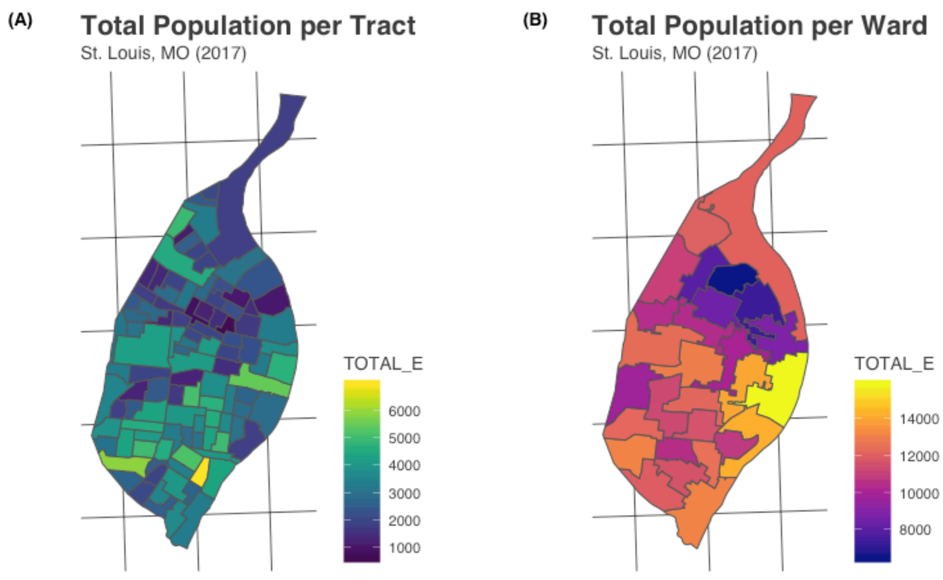
Once data have been prepared for interpolation, meaning they share the same projected coordinate system and there are no conflicts between variable names, interpolations can be calculated with a single function `aw_interpolate()`. This functionality is illustrated here using source and target data built into the `areal` package:

```
> aw_interpolate(ar_stl_wards, tid = WARD, source = ar_stl_race,
+   sid = GEOID, weight = "sum", output = "tibble",
+   extensive = c("TOTAL_E", "WHITE_E", "BLACK_E"))
# A tibble: 28 x 6
   OBJECTID  WARD        AREA TOTAL_E WHITE_E BLACK_E
      <dbl> <int>       <dbl>   <dbl>   <dbl>   <dbl>
 1        1     1   46138761.   7992.    153.   7779.
 2        2     2  267817711.  12145.   1323.  10639.
 3        3     3   66291644.   7344.    591.   6635.
 4        4     4   53210707.   8458.    160.   8203.
 5        5     5   60462396.   8783.   1526.   7056.
 6        6     6   64337271.  14050.   5840.   7439.
 7        7     7  101268146.  15840.   8220.   6629.
 8        8     8   45966410.  12188.   7604.   3796.
 9        9     9   73993891.  14217.   6838.   6413.
10       10    10   62915358.  11239.   8703.   1667.
# … with 18 more rows
```

If a `sf` object is desired, changing the `output` argument in the example above to `"sf"` will provide not only the estimated values but the relevant geometric data as well.

Spatially intensive interpolations, which are used by analysts for data that are ratios or percentage values, can be calculated by replacing the `extensive` argument with `intensive`. Alternatively, both arguments and corresponding vectors of variable names can be provided to interpolate both spatially extensive and intensive data simultaneously.

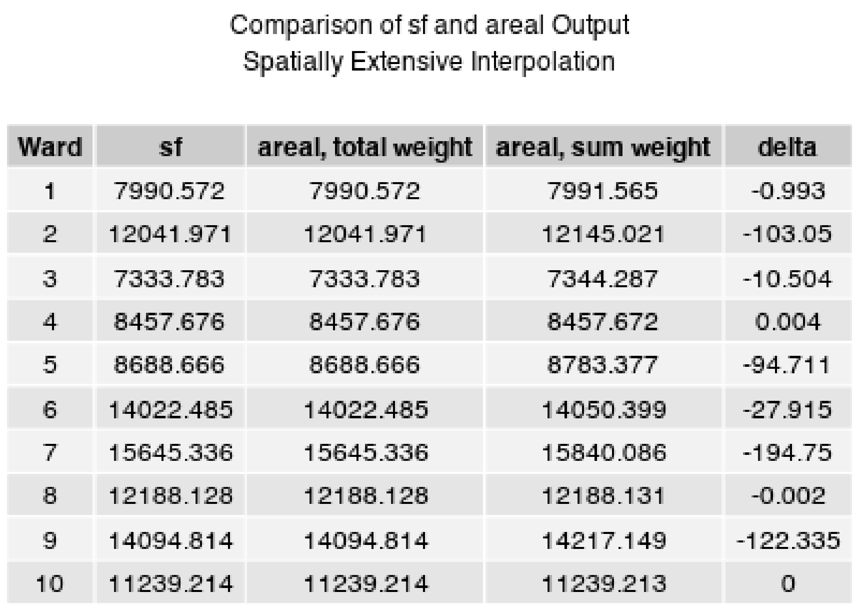
The source data from the above data along with one of the variables, total population (`TOTAL_E`), are mapped in panel A below at the census tract level. The resulting data for the total population variable interpolated to the target political ward data have been mapped in panel B:



Additional examples and vignette illustrations of `areal`'s functionality are available on the package's website.

## A Quick Comparison with sf

Since `sf` offers similar functionality, three comparisons are made and presented here. Code for these comparisons is available in an appendix within the `areal` GitHub repository. The first compares spatially extensive interpolations calculated in both `areal` and `sf`. The first ten features are shown here for comparison:



Comparison of sf and areal Output
Spatially Extensive Interpolation

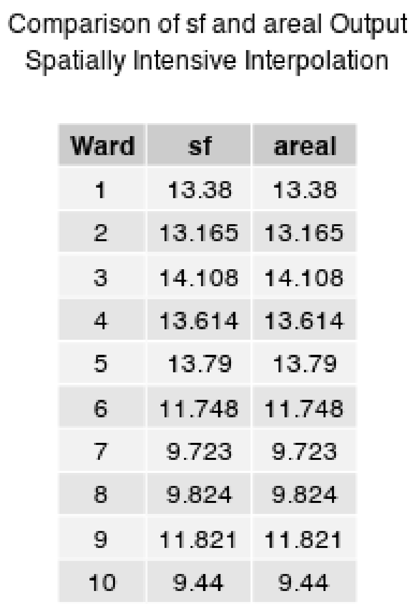
| Ward | sf | areal, total weight | areal, sum weight | delta |
|---|---|---|---|---|
| 1 | 7990.572 | 7990.572 | 7991.565 | -0.993 |
| 2 | 12041.971 | 12041.971 | 12145.021 | -103.05 |
| 3 | 7333.783 | 7333.783 | 7344.287 | -10.504 |
| 4 | 8457.676 | 8457.676 | 8457.672 | 0.004 |
| 5 | 8688.666 | 8688.666 | 8783.377 | -94.711 |
| 6 | 14022.485 | 14022.485 | 14050.399 | -27.915 |
| 7 | 15645.336 | 15645.336 | 15840.086 | -194.75 |
| 8 | 12188.128 | 12188.128 | 12188.131 | -0.002 |
| 9 | 14094.814 | 14094.814 | 14217.149 | -122.335 |
| 10 | 11239.214 | 11239.214 | 11239.213 | 0 |

Both `st_interpolate_aw()` and `aw_interpolate()` with the `weight = "total"` produce identical results, which is expected.

The above comparison also shows the difference between using `"total"` and `"sum"` for the `weight` argument in `aw_interpolate()`. If we expect that our source and target data should overlap completely (but perhaps do not because of data quality issues), the `"sum"` approach will allocate all individuals into target features whereas `"total"`will not, instead allocating only a proportion of individuals relative to the overlap between the source and target features. In the example data provided in the `areal` package, this is the case - the Census tracts do not perfectly map onto the extent of the wards, and so the `"total"` approach is inappropriate.

The differences are highlighted in the `delta` column in the above figure. With these data, the `"total"` approach yields generally smaller results that the `"sum"` approach, with an average difference of -30.781. The largest difference between the `"total"` and `"sum"` approaches was -194.750 individuals. The added functionality within `areal` therefore provides a potentially more accurate set of estimations in similar cases where data should overlap, but do not.

The second comparison is with spatially intensive interpolations. For this approach, both `st_interpolate_aw()` and `aw_interpolate()` should yield identical results. As before, the first ten features are shown here for comparison:



Comparison of sf and areal Output
Spatially Intensive Interpolation

| Ward | sf | areal |
|---|---|---|
| 1 | 13.38 | 13.38 |
| 2 | 13.165 | 13.165 |
| 3 | 14.108 | 14.108 |
| 4 | 13.614 | 13.614 |
| 5 | 13.79 | 13.79 |
| 6 | 11.748 | 11.748 |
| 7 | 9.723 | 9.723 |
| 8 | 9.824 | 9.824 |
| 9 | 11.821 | 11.821 |
| 10 | 9.44 | 9.44 |

As with the extensive interpolations where `weight = "total"`, both `st_interpolate_aw()` and `aw_interpolate()` produce the expected identical results.

### Notes on Performance

Finally, a comparison of the speed at which both the `sf` and `areal` packages calculated these values is included in the appendix as well. For the extensive interpolations, `aw_interpolate()` performed 28 milliseconds slower on the sample data provided in the package, with a mean execution time of 279.429 milliseconds. Similarly, the intensive interpolations were 20 milliseconds slower when using `areal`, which had a mean execution time of 267.878 milliseconds. Both comparisons were made using the `R` package `microbenchmark` (Mersmann, 2018).

One particular performance concern to note are intersections (a step in the interpolation process) that return geometry collections. These will cause `st_interpolate_aw()` function from `sf` to error. The `aw_interpolate()` function will not error, but instead will correctly address these geometry collections before proceeding with the interpolation. However, the correction process can take significantly longer than it would a similarly sized data set that does not require this additional step.

The appendix also contains an example of interpolating population from Missouri's 115 counties into its 4,506 Census block groups. When evaluated with `microbenchmark`, the average length of time to complete this process was 17.689 seconds.

## Availability

`areal` is open source software made available under the GNU GPL-3 license. It can be installed through CRAN (Prener & Revord, 2019) using: `install.packages("areal")`. `areal` can also be installed from its GitHub repository using the `remotes` package: `remotes::install_github("slu-openGIS/areal")`.

## Acknowledgements

The authors wish to thank their colleagues, J.S. Onésimo Sandoval, Ph.D. and Taylor Harris Braswell, M.A., for reviewing an initial draft of this manuscript.

## References

Goodchild, M. F. (1978). Statistical aspects of the polygon overlay problem. *Harvard Papers on Geographic Information Systems.*

Goodchild, M. F., & Lam, N. S. N. (1980). *Areal interpolation: A variant of the traditional spatial problem.* Department of Geography, University of Western Ontario London, ON, Canada.

Lam, N. S.-N. (1983). Spatial interpolation methods: A review. *The American Cartographer*, *10*(2), 129–150. doi:10.1559/152304083783914958

Langford, M. (2007). Rapid facilitation of dasymetric-based population interpolation by means of raster pixel maps. *Computers, Environment and Urban Systems*, *31*(1), 19–32. doi:10.1016/j.compenvurbsys.2005.07.005

Markoff, J., & Shapiro, G. (1973). The linkage of data describing overlapping geographical units. *Historical Methods Newsletter*, *7*(1), 34–46. doi:10.1080/00182494.1973.10112670

Mersmann, O. (2018). *Microbenchmark: Accurate timing functions.* Retrieved from https://CRAN.R-project.org/package=microbenchmark

Pebesma, E. (2018). Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, *10*(1), 439–446. doi:10.32614/RJ-2018-009

Prener, C., & Revord, C. (2019). Areal: Areal weighted interpolation. doi:10.5281/zenodo.2603540

Qiu, F., Zhang, C., & Zhou, Y. (2012). The development of an areal interpolation arcgis extension and a comparative study. *GIScience & Remote Sensing*, *49*(5), 644–663. doi:10.2747/1548-1603.49.5.644