

Package ‘AIGIS’

April 17, 2009

Type Package

Title Areal Interpolation for GIS data

Version 1.0

Date 2009-03-06

Author Benjamin P. Bryant and Anthony Westerling

Maintainer Bryant <bryant@prgs.edu>

Depends R (>= 2.5.1), gpclib (>= 1.4)

Description AIGIS can be used to interpolate spatially associated data onto arbitrary target polygons which lack such data. Version 1.0 of the package is oriented toward convenient interpolation of specific US census data for California, but the tools provided should work for any combination of GIS data source and target polygon, provided appropriate care is taken. Future versions will be aimed at facilitating more general applications.

License GPL-3

Repository CRAN

Date/Publication 2009-03-09 19:14:32

R topics documented:

AIGIS-package	2
ab.areal.interp	3
ab.gbounds	6
ab.gcbounds	8
ab.gmu	9
ab.gstats	11
area2frac	12
arealw	14
bbox.mat	16
bgvals	17
cabgbbmat	18

cabggpc	19
cell.arealw	19
damrats	21
dp.interp	22
fpdemogpc	24
gc2gpc	25
gridgpc	26
gridinws	27
MASK	27
mult.arealw	28
polytobg	30
rbgarea	31
sab.areal.interp	32

Index	35
--------------	-----------

AIGIS-package	<i>Areal interpolation for GIS data</i>
---------------	---

Description

AIGIS can be used to interpolate spatially associated data onto arbitrary target polygons which lack such data. Version 1.0 of the package is oriented toward convenient interpolation of specific US census data for California, but the tools provided should work for any combination of GIS data source and target polygon, provided appropriate care is taken. Future versions will be aimed at facilitating more general applications.

Details

Package: AIGIS
 Type: Package
 Version: 1.0
 Date: 2009-03-06
 License: GPL-3

AIGIS was originally built for specific research on estimating losses associated with wildfires in California. The current state of the package very much reflects this, with most functions built around supporting the task of the researchers. This is of immediate use for the author and colleagues, and for those who may wish to verify, expand or explore the work described in the publication cited below. However, the tools provided by the package are generally applicable, provided the user has their own spatially associated dataset and target polygons. Below we give an overview of general interest tasks the package can accomplish, with reference to relevant commands.

Task 1: For polygon A and a list of polygons B, find what fraction of each polygon in B is contained inside A. This can be accomplished with the command `arealw`. Polygon A can also be specified by simply providing a bounding box, rather than a polygon, in which case `cell.arealw` may be used.

Task 2: Task 1, but applied to a collection of polygons A. This can be accomplished with `mult.arealw`.

Task 3: Given single or multiple ‘target’ polygons A, and spatially associated data for polygons B, estimate the quantities associated with polygons A based on the relative contributions from polygons B. This can be accomplished via an iterated applications of the `arealw` or `mult.arealw`, followed by `dp.interp`. Alternatively, this can be done with one function call to `sab.areal.interp` or `ab.areal.interp` (for single and multiple polygons respectively).

The package also contains specialized functions and data relevant to aggregating and bounding estimates for wildfire in 1/8 degree gridcells covering california. These are found mainly in `ab.gstats`. Use `ab.gmu` if not interested in bounding estimates.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu> and Anthony L. Westerling <awesterling@ucmerced.edu>
Maintainer: Benjamin Bryant

References

Westerling, A. L. and B. P. Bryant, 2008. Climate Change and Wildfire in California. *Climatic Change*, 87: s231-249.

For general overview of areal interpolation and terminology, see Sadahiro, Y. Accuracy of areal interpolation: A comparison of alternative methods. *Journal of Geographical Systems* 1(4): 323-346 (1999).

Examples

```
#See the details section above for noteworthy commands. Their helpfiles
#should contain illuminating example scripts.
```

ab.areal.interp *Area-based areal interpolation for multiple polygons*

Description

Given a list of target polygons, a list of zonal polygons, and a matrix of data associated with the zonal polygons, interpolate zonal polygon data to specified target polygons. As a wrapper of `mult.arealw` and `dp.interp`, it generates areal weights as an intermediate product, which can also be returned if desired. This allows other variables in the data record to be interpolated to the same polygons without having to redo the costly weight-calculating step.

Usage

```
ab.areal.interp(target.ind = "all", targetlist = fireppgc, trim = TRUE,
  recvar = 1, recvals = bgvals, zones = cabggpc, zbbmat = cabgbbmat,
  zoneareas = rbgarea, appdam = TRUE, dr = damrats, twokcensus = TRUE,
  digits = 6, nobounds = FALSE, keepwts = TRUE, verbose = 2)
```

Arguments

<code>target.ind</code>	A vector containing list indices of target polygons, or character “all”, which indicates the entire list is to be used.
<code>targetlist</code>	A list of possible target polygons
<code>trim</code>	A logical indicating whether areal weights should be stored compactly or in one long vector per target polygon. The default is <code>TRUE</code> , as for lengthy lists of target polygons available memory can become prohibitive.
<code>recvar</code>	An integer indicating which column of the record value matrix to use. That is, what variable is to be interpolated.
<code>recvals</code>	A matrix of record values containing the data to be interpolated, with rows corresponding to zones, and columns corresponding to different variables.
<code>zones</code>	List of zonal polygons from which data will be interpolated to the target.
<code>zbbmat</code>	“Zonal bounding box matrix.” Either a matrix of dimension <code>length(zones)</code> by 4, or character “none”. If “none,” an appropriate matrix is automatically computed by a call to <code>bbox.mat</code> . See ‘Details.’
<code>zoneareas</code>	A vector giving the area of each zone or character “none”, indicating they must be calculated inside the function.
<code>appdam</code>	CA Wildfire specific. A logical indicating whether or not to apply the empirically derived damage ratio.
<code>dr</code>	CA Wildfire specific. A vector giving precalculated damage ratios by block group. Only used if <code>appdam=TRUE</code> .
<code>twokcensus</code>	Logical indicating whether the data in question is block group level data for California from the 2000 US Census. If ‘TRUE’, the function automatically handles certain ideosyncracies associated with that data.
<code>digits</code>	Digits argument passed to <code>round</code> internally. Used in setting the tolerance for considering fractional overlap to be 100 percent. Due to numerical issues, overlap will not always be identically 1 when a zone is contained entirely within the target. There should be little reason to modify this away from the default value of six.
<code>nobounds</code>	Logical indicating whether or not to return the estimates without bounds. Default is ‘FALSE’, but there may be situations where having no bounds will be more convenient for handling data objects. See <code>Details</code> for how the bounds are generated.
<code>keepwts</code>	Whether or not to also return the areal weights calculated by <code>arealw</code> . See <code>Value</code> for details.
<code>verbose</code>	An integer indicating how frequently to print progress through the target polygons. Will print a message every time 10 to the <code>verbose</code> power polygons have been processed. Setting <code>verbose</code> negative results in no progress output.

Details

If not “none”, ‘zbbmat’ must take the form of a `length(zones)` by 4 matrix, where each row corresponds to the bounding latitude and longitude values for polygons in `zones`, the same format

returned by the function `bbox.mat`. The columns correspond to low longitude, high longitude, low latitude and high latitude respectively.

The lower bound estimates are generated by assuming any zone less than fully enclosed by the current target polygon contributes zero to the total value contained. The upper bound is generated by assuming that any zone with nonzero overlap contributes its entire value. In all cases, zones that are entirely enclosed contribute their entire value.

Value

The output under default settings is a list of two elements. The first element is a `length(target.ind)` by 3 matrix giving the expected value, lower bound and upper bound on enclosed value, for each target polygon specified. The second element is another list, each entry of which has the matrix of areal weights for the corresponding target polygon. In these matrices, the first column indexes those zones with nonzero overlap, and the second column gives those overlaps. These matrices can be passed to `dp.interp` to generate other interpolated values for the same target polygon, either individually, or using `lapply`.

If `nobounds=TRUE`, lower and upper bound are not returned, and the result is just a vector of length one.

If `keepwts=FALSE`, the function returns only the matrix or vector as above, and the areal weights are not returned.

If `trimout=FALSE`, the weights are returned as a vector of length equal to `nrow(recvals)`, rather than as a matrix containing specifying only those nonzero weights. See help for `arealw` for more info.

Thus, there can be six different possible output forms. The default should suffice in most cases.

Note

The bounds on the outputs when `appdam` is set to `TRUE` do not take into account uncertainty in the damage ratio. The absolute high would be to take the upper bound when not applying the damage ratio, and the absolute low is always zero.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>

Examples

```
data(bgvals)
data(cabgbbmat)
data(cabggpc)
data(damrats)
data(fpdemogpc)
data(rbgarea)
data(polytobg)

#Calculate structures enclosed by the Union and Cedar fires:
cedar <- ab.areal.interp(target.ind=c(1,3),targetlist=fpdemogpc,recvar=1,
  appdam=FALSE,keepwts=TRUE)
```

```
#check it out:
cedar[[1]]

#Using the weights already generated, calculate the total value enclosed for
#the Cedar fire:
dp.interp(cedar[[2]][[2]],recvar=2,appdam=FALSE,trimmedin=TRUE)

#Again using the weights already generated, calculate estimated houses
#destroyed, this time for both at the same time:
lapply(cedar[[2]],dp.interp,recvar=1,appdam=TRUE,trimmedin=TRUE)
```

ab.gbounds

Calculate bounds on hypothetical fires in CA 1/8 degree gridcells

Description

Calculate the upper and lower bounds on values that could be contained inside fire perimeters of arbitrary area in every single 1/8 degree gridcell in California. Assumes the fire occupied the highest and lowest density block groups respectively, and does (at present) not enforce contiguity requirements. Also, at present this function is not generalizable beyond the CA wildfire problem, and `area`, `units`, `recvar` and `appdam` are the only arguments that can be safely changed from the default.

Usage

```
ab.gbounds(gridgpcobj = gridgpc, gws = gridinws, area = 200, units = "ha",
  recvar = 1, recvals = bgvals, recarea = rbgarea, appdam = TRUE,
  dr = damrats, cap = TRUE, maskobj=MASK)
```

Arguments

<code>gridgpcobj</code>	A list object containing gridcells in <code>gpc</code> form as well as indexing vectors. See help for the data object <code>gridgpc</code> for exact form.
<code>gws</code>	Grid weights. A list of matrices giving zone indices and overlaps for each gridcell. For now <code>gridinws</code> is the appropriate companion object to <code>gridgpc</code> .
<code>area</code>	An numeric area expressed in one of five units, specified by <code>units</code> .
<code>units</code>	A character object indicating the units on <code>area</code> , one of: <ul style="list-style-type: none"> “ha” Hectares “sqkm” Square kilometers “sqmi” Square miles “acres” Acres “deg-area” Area expressed in lat-lon degree area
<code>recvar</code>	An integer indicating which column of the record value matrix to use. That is, what variable is to be bounded.

recvals	A matrix containing the data to be bounded, with rows corresponding to zones, and columns corresponding to different variables.
recarea	A vector giving the area of each polygon corresponding to the records in <code>recvals</code> , in units of lat-lon degree area. Unlike for the standard areal interpolation functions built around <code>arealw</code> , this must be provided and cannot be set to “none”.
appdam	CA Wildfire specific. A logical indicating whether or not to apply the empirically derived damage ratio.
dr	CA Wildfire specific. A vector giving precalculated damage ratios by block group.
cap	Logical passed to <code>area2frac</code> , indicating whether or not to automatically cap the answer at one, if the area called for is greater than the area of the cell.
maskobj	A matrix of ones and zeros noting which gridcells should be used and which omitted

Details

This is primarily a midlevel wrapper, which loops calls to `ab.gcbounds`. It itself is wrapped by `ab.gstats`, which uses its output in assembling the final array reporting expected values, and bounds for housing structures, housing structures damaged, and total property value damaged.

Value

Returns an array with dimensions `dim=nrow(MASK), ncol(MASK), 2`, where the `value[i,j]` is a length two vector whose first entry is the lower bound estimate for the gridcell identified by `MASK[i,j]`, and whose second entry is the upper bound estimate.

Note

If enough area within a gridcell is overlapping the border, the results will always be zero for the lower bound - thus fire sizes need to become reasonable large before the lower bound will be nonzero anywhere.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>

See Also

[ab.gcbounds](#)

Examples

```
data(bgvals)
data(damrats)
data(gridgpc)
data(gridinws)
data(MASK)
data(rbgarea)
```

```
#find upper and lower bounds of housing value threatened by 200 ha fires in all
#gridcells. (Will take a minute or two of calculating time.)

somebounds <- ab.gcbounds(gridgpcobj = gridgpc, gws = gridinws, area = 200,
  units = "ha", recvar = 2, recvals = bgvals, recarea = rbgarea,
  appdam = FALSE, dr = damrats, cap = TRUE)
```

ab.gcbounds	<i>Calculate the bounds on values contained in a random fire perimeter of given area</i>
-------------	--

Description

Given an 1/8 degree gridcell and an area representing the area contained inside a hypothetical fire perimeter, calculate the upper and lower bounds on values that could be contained inside that perimeter. Assumes the fire occupied the highest and lowest density block groups respectively, and (at present) does not enforce contiguity requirements.

Usage

```
ab.gcbounds(iwtmat, area = 0.125^2, recarea = rbgarea, recvar = 1,
  recvals = bgvals, appdam = FALSE, dr = damrats)
```

Arguments

iwtmat	Index Weight Matrix. A two column matrix of zone indices and weights such as that output by a call <code>arealw(target=thegridcell, zones=theappropriatezones, trimout=TRUE)</code> . More likely it will be an element of data object <code>gws</code> , which contains precalculated matrices for 1/8 degree gridcells as targets and the California block groups as the zones.
area	The area of the hypothetical fire, expressed in lat/lon degrees area.
recarea	A vector of zone areas, ordered by their index in a matrix of <code>recvals</code> .
recvar	An integer indicating which column of the record value matrix to use.
recvals	A matrix of record values containing the data to be bounded.
appdam	CA Wildfire specific. A logical indicating whether or not to apply the empirically derived damage ratio.
dr	CA Wildfire specific. A vector giving precalculated damage ratios by block group. Only used if <code>appdam=TRUE</code> .

Details

The bounds are calculated in an extremely conservative way (similar to, but distinct from the method used by `dp.interp`). In both cases, it orders block groups by density, and adds block group values in increasing or decreasing density order (for lower and upper bounds respectively) until the total area is reached. Furthermore, for the lower bound it is assumed that all zones crossing the target border have density zero (i.e., that their entire value is contained outside the cell). For the upper bound, it is assumed that all zones crossing the border have their entire value within the cell.

Value

A vector of length two with a lower bound and an upper bound on possible values for a fire contained inside the grid.

Author(s)

Benjamin P. Bryant, `<bryant@prgs.edu>`

Examples

```
data(bgvals)
data(damrats)
data(gridgpc)
data(gridinws)
data(rbgarea)

#What are the bounds on a 200 ha fire near San Diego?
#First tranlate 200 ha into lat-lon degree^2:
thearea <- .125^2*area2frac(cell=gridgpc[[1]][[2139]], area=200, units="ha",
  naive=TRUE)

ab.gcbounds(iwtmat=gridinws[[2139]], area=thearea)
```

ab.gmu

Find expected value for a random fires in all 1/8 degree gridcells

Description

For each 1/8 degree gridcell in California, estimate the expected value contained inside a fire perimeter of specified area. Operates by finding the total value inside the gridcell and multiplying by the appropriate fraction, determined by a call to `area2frac`.

Usage

```
ab.gmu(gridgpcobj = gridgpc, gws = gridinws, afracs = gfracs, recvar = 1,
  recvals = bgvals, appdam = FALSE, trimmedin = TRUE, dr = damrats,
  maskobj = MASK)
```

Arguments

<code>gridgpcobj</code>	A list object containing gridcells in <code>gpc</code> form as well as indexing vectors. See help for the data object <code>gridgpc</code> for exact form.
<code>gws</code>	Grid weights. A list of matrices giving zone indices and overlaps for each gridcell. For now <code>gridinws</code> is the appropriate companion object to <code>gridgpc</code> .
<code>afrac</code>	A precalculated vector giving the fraction of land area in each gridcell occupied by a fire of a given area.
<code>recvar</code>	An integer indicating which column of the record value matrix to use. That is, what variable is to be interpolated.
<code>recvals</code>	A matrix containing the data to be interpolated, with rows corresponding to zones, and columns corresponding to different variables.
<code>appdam</code>	CA Wildfire specific. A logical indicating whether or not to apply the empirically derived damage ratio.
<code>trimmedin</code>	Whether the overlap weights are provided in matrix form. Must be true for this version.
<code>dr</code>	CA Wildfire specific. A vector giving precalculated damage ratios by block group.
<code>maskobj</code>	A matrix of ones and zeros noting which gridcells should be used and which omitted

Value

Returns a matrix of with `nrow=nrow(MASK)` and `ncol=ncol(MASK)`, where entry `value[i,j]` corresponds to the expected value for a fire of given size in the gridcell corresponding to `MASK[i, j]`.

Author(s)

Benjamin P. Bryant, bryant@prgs.edu

Examples

```
data(bgvals)
data(damrats)
data(gridgpc)
data(gridinws)
data(MASK)

#Find expected values on houses enclosed by a 200 ha fire.
#First, create the (reusable) vector of area fractions:

gfrac <- sapply(gridgpc[[1]],area2frac, area=200,units="ha", cap=TRUE)

theresult <- ab.gmu(gridgpcobj = gridgpc, gws = gridinws, afrac = gfrac,
  appdam=FALSE, recvar = 1)
```

 ab.gstats

Get statistics for hypothetical fires in all gridcells

Description

This is the primary and all-inclusive function for the gridcell statistics half of the package. It wraps various supporting functions to return a 3-d array with x and y dimensions corresponding to the gridcells in MASK, and gives the expected value of housing structures enclosed, damaged, and value damaged for a fire of specified area in each gridcell, also providing upper and lower bounds for each of those values.

Usage

```
ab.gstats(area = 200, units = "ha", gridgpcobj = gridgpc, gws = gridinws,
  strind = 1, valind = 2, recvals = bgvals, recarea = rbgarea, dr = damrats,
  cap = TRUE)
```

Arguments

area	An numeric area for the fire, expressed in one of five units, specified by units.
units	A character object indicating the units on area, one of: “ha” Hectares “sqkm” Square kilometers “sqmi” Square miles “acres” Acres “deg-area” Area expressed in lat-lon degree area
gridgpcobj	A list object containing gridcells in gpc form as well as indexing vectors. See help for the data object <code>gridgpc</code> for exact form.
gws	Grid weights. A list of matrices giving zone indices and overlaps for each gridcell. For now <code>gridinws</code> is the appropriate companion object to <code>gridgpc</code> .
strind	An integer indicating which column of the record value matrix to use for structures, (or any variable you would like evaluated both with and without the application of the damage ratio).
valind	An integer indicating which column of the record value matrix to use for aggregate value (or any column you would like to have evaluated only with the damage ratio applied).
recvals	A matrix containing the data to be bounded, with rows corresponding to zones, and columns corresponding to different variables.
recarea	A vector giving the area of each polygon corresponding to the records in <code>recvals</code> , in units of lat-lon degree area. Unlike for the standard areal interpolation functions built around <code>arealw</code> , this must be provided and cannot be set to “none”.
dr	CA Wildfire specific. A vector giving precalculated damage ratios by block group.
cap	Logical passed to <code>area2frac</code> , indicating whether or not to automatically cap the answer at one, if the area called for is greater than the area of the cell.

Value

An array with `dim=c(93, 97, 9)`, the (currently) hardcoded size of the CA mask. The first three layers are the expected value for housing structures, housing structures damaged, and housing value damaged. The next three are the lower bounds, and the last three are the upper bounds on those values.

Note

Currently this function is rather inflexible. Future versions may feature the ability to specify a more extensive combination of input values (in terms of variables to use and whether to apply damages). For now that can be done by very simple modifications to the function, by altering the underlying calls to `ab.gbounds` and `ab.gmu`. Alternatively you can create customized data objects by combining the output from multiple calls to this function.

Author(s)

Benjamin P. Bryant, bryant@prgs.edu and Anthony Westerling, awesterling@ucmerced.edu

Examples

```
data(bgvals)
data(damrats)
data(gridgpc)
data(gridinws)
data(MASK)
data(rbgarea)

#Do the whole shebang for California gridcells with 200 ha fire:
#(will take several minutes)
twohstats <- ab.gstats(area = 200, units = "ha", gridgpcobj = gridgpc,
  gws = gridinws, strind = 1, valind = 2, recvals = bgvals, recarea = rbgarea,
  dr = damrats, cap = TRUE)
```

area2frac

Find the fraction of a rectangular polygon taken up by a given area

Description

Converts an area value into a fraction of gridcell coverage for use in creating random fire-within-gridcell statistics. Allows area to be given in several possible units. The cell can be something other than a rectangular polygon provided it is of class `gpc.poly`, but if so the fraction will be slightly (likely insignificantly) off, as the centroid will be incorrectly calculated. Also allows rectangle to be specified by vector of bounds rather than a `gpc.poly`.

Usage

```
area2frac(cell, area=200, units="ha", cap=TRUE, naive=TRUE, zones = cabggpc,
          zbbmat = cabgbbmat, twokcensus=TRUE, polytobgobj=polytobg)
```

Arguments

cell	A (probably rectangular) polygon of class <code>gpc.poly</code> or a vector of bounds ordered as min long, max long, min lat, max lat. Units must be in degrees latitude and longitude.
area	An numeric area expressed in one of five units, specified by <code>units</code> .
units	A character object, one of: “ha” Hectares “sqkm” Square kilometers “sqmi” Square miles “acres” acres “deg-area” Area expressed in lat-lon degree area
cap	Logical indicating whether or not to automatically cap the answer at one, if the area called for is greater than the area of the cell.
naive	Whether or not to assume the entire area occupied by the cell should be considered in the denominator. If you are interested in land area only, and the cell contains water, the resulting area fraction will be deflated. If you know this is not the case, the naive option is computationally much faster, and none of the remaining arguments are used.
zones	A list of polygons covering the real land area to be counted, if <code>naive=FALSE</code> .
zbbmat	“Zonal bounding box matrix.” Either a matrix of dimension <code>length(zones)</code> by 4, or character “none”. If “none,” an appropriate matrix is automatically computed by a call to <code>bbox.mat</code> . See ‘Details.’
twokcensus	Logical indicating whether the data in question is block group level data for California from the 2000 US Census. If ‘TRUE’, the function automatically handles certain ideosyncracies associated with that data.
polytobgobj	A link between polygon indices and block group row indices.

Details

If not “none”, ‘zbbmat’ must take the form of a `length(zones)` by 4 matrix, where each row corresponds to the bounding latitude and longitude values for polygons in ‘zones’, the same format returned by the function `bbox.mat`. The columns correspond to low longitude, high longitude, low latitude and high latitude respectively.

Value

A numeric value indicating the fraction of the rectangular cell taken up by a shape with the designated area.

Note

The primary use of this function is accounting for area expressed in degrees being dependent on latitude. It is useless if the coordinates are not expressed in degrees latitude.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>

Examples

```
#how much of an 1/8 degree cell does a 200 ha fire occupy,
#at a San Diego latitude:

data(cabgbbmat)
data(cabggpc)
data(polytobg)

area2frac(cell= c(-117.625,-117.5,33.375,33.5), naive=TRUE)

#what if we account for the fact that part of the cell covers the ocean:
thecell <- gc2gpc(c(-117.625,-117.5,33.375,33.5))

area2frac(cell=thecell,naive=FALSE)
```

arealw

Calculate areal weights for a single target polygon

Description

Given an arbitrary target polygon and a list of zonal polygons, this function finds the fraction of each zonal polygon that lies inside the target polygon. The weights can then be used by other functions to interpolate zonal vales to the target polygon. All polygons must be of class `gpc.poly`.

Usage

```
arealw(target, trimout = TRUE, zones = cabggpc, zbbmat = cabgbbmat,
       zoneareas = rbgarea, twokcensus = TRUE, digits = 6, polytobgobj=polytobg)
```

Arguments

target	An arbitrary polygon to which values will (presumably) be interpolated.
trimout	Logical indicating output format. See Value for details.
zones	List of zonal polygons for which fractional overlap is to be determined.
zbbmat	“Zonal bounding box matrix.” Either a matrix of dimension <code>length(zones)</code> by 4, or character “none”. If “none,” an appropriate matrix is automatically computed by a call to <code>bbox.mat</code> . See ‘Details.’

zoneareas	A vector giving the area of each zone, or character “none”, indicating they must be calculated inside the function.
twokcensus	Logical indicating whether the data in question is block group level data for California from the 2000 US Census. If ‘TRUE’, the function automatically handles certain ideosyncracies associated with that data.
digits	Digits argument passed to <code>round</code> internally. Used in setting the tolerance for considering fractional overlap to be 100 percent. Due to numerical issues, overlap will not always be identically 1 even when a zone is contained entirely within the target. There should be little reason to modify this away from the default value of six.
polytobgobj	A link between polygon indices and block group row indices.

Details

If not “none”, ‘zbbmat’ must take the form of a `length(zones)` by 4 matrix, where each row corresponds to the bounding latitude and longitude values for polygons in ‘zones’, the same format returned by the function `bbox.mat`. The columns correspond to low longitude, high longitude, low latitude and high latitude respectively.

Value

If `trimout` equals ‘FALSE’, the function returns a vector with length equal to the number of zones, and each entry corresponding to the fractional overlap for that zone. Since a large number of entries are likely to be zero, this is a wasteful storage method, but may be valuable as a convenient format with which to perform certain manual operations on the data.

Setting `trimout` equal to ‘TRUE’ returns an ‘n’ by 2 matrix, where ‘n’ is the number of zones with nonzero overlap. The first column contains the indices fo those zones, and the second column contains the actual overlap weights.

Note

The US Census 2000 data for California contains a mismatch between the number of block groups and the number of block group polygons. The code handles this provided `twokcensus` is set to ‘TRUE’, but for all other datasets it is assumed that the number of datarecords is equal two the number of zones, and that the indexes match as well. If not, one or the other needs to be sorted appropriately.

Author(s)

Benjamin P. Bryant, bryant@prgs.edu

References

For general overview of areal interpolation and terminology, see Sadahiro, Y. Accuracy of areal interpolation: A comparison of alternative methods. *Journal of Geographical Systems* 1(4): 323-346 (1999).

See Also

[mult.arealw](#)

Examples

```
data(cabgbbmat)
data(cabggpc)
data(fpdemogpc)
data(rbgarea)
data(polytobg)

#run arealw on the Cedar fire using CA block groups with trimmed output:
moh <- arealw(fpdemogpc[[3]],trimout=TRUE)

#run it with long output:
mohbig <- arealw(fpdemogpc[[3]],trimout=FALSE)

#reconstruct the untrimmed output from the trimmed output:
mohmadebig <- rep(0,length(mohbig))
mohmadebig[moh[,1]] <- moh[,2]

#should be true:
identical(mohbig, mohmadebig)
```

bbox.mat

Create a matrix of bounding box data for a list of polygons

Description

Given a list of polygons of class `gpc.poly`, this creates a matrix where each row contains the bounding box data for corresponding polygon.

Usage

```
bbox.mat(zones = cabggpc)
```

Arguments

`zones` The list of polygons from which to extract bounding box information.

Value

A matrix of dimension `length(zones)` by 4. The columns correspond to low longitude, high longitude, low latitude and high latitude respectively.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>

See Also

[arealw](#)

Examples

```
data(cabgbbmat)
data(cabggpc)

moh <- bbox.mat(cabggpc)

identical(cabgbbmat, moh)
```

bgvals

A data frame containing values of interest for CA block groups

Description

A dataframe that contains both raw values from the original US Census summary file 3 for California, as well as two estimated values. Data is at the block group level. Identifying info about the block group of row 'i' can be identified by referencing `polytobg[polytobg[,1]==i,]`.

Usage

```
data(bgvals)
```

Format

A data frame with 22133 observations on the following 9 variables.

h.str Estimated number of housing structures in the block group.

agg.val Estimated total housing value inside the block group.

pop Total population.

h.tot Total number of housing units.

h.vac Total number of vacant housing units.

h.occ Total number of occupied housing units.

h.oo Total number of owner occupied housing units.

h.ro Total number of renter occupied housing units.

oo.val Total reported value of owner-occupied housing units.

Details

The estimated values for the column 'h.str' were calculated as follows: Census tables provided data on number of housing units by units contained in structure. These numbers were weighted by the inverse of units in structure to arrive at a total estimate of housing values. The column 'agg.val' was the sum of census-provided data on owner occupied housing values, with an estimate for aggregate rental housing values. Rental housing values were estimated by assuming equal value as owner-occupied units, normalized by housing units per structure.

Source

Census 2000 Summary File 3: http://factfinder.census.gov/servlet/DatasetMainPageServlet?_program=DEC&_submenuId=datasets_1&_lang=en

Examples

```
data(bgvals)
```

cabgbbmat

Matrix of bounding box data for California block groups

Description

A 22195 by 4 matrix containing bounding box data for the US Census block group polygons contained in 'cabgpc'. Note this is for the polygons, and not the actual block groups (see Details).

Usage

```
data(cabgbbmat)
```

Format

See description above.

Details

This object used by other functions to provide a fast filter for determining which block groups need to be considered as potentially overlapping with a target polygon. For example, if the the low edge of the bounding box is higher than the high edge of the bounding box for a target polygon, it is impossible for them to overlap. Similarly for the other three sides.

As mentioned elsewhere, the mapping of GIS records provided by the census is not 1 to 1 with the number of block groups. This file is specific to the polygons provided, not the block groups.

References

The polygons from which these bounding boxes were extracted are found here: <http://www.census.gov/geo/www/cob/>

Examples

```
data(cabgbbmat)
```

```
cabggpc
```

California block group polygons in gpc.poly format

Description

A list consisting of California block group polygons, converted from Shapefile to class `gpc.poly`, to take advantage of the polygon intersecting capabilities offered by `gpplib`.

Usage

```
data(cabggpc)
```

Format

A list of length 22195, where each entry is a polygon of class `gpc.poly`.

References

The original polygons which were converted are available here: <http://www.census.gov/geo/www/cob/>

Examples

```
data(cabggpc)
```

```
cell.arealw
```

Find areal weights for a gridcell specified by its bounding box

Description

Given the bounds of a rectangular region, and the zones of interest, find areal weights for that region as though it was a target polygon. Basically just a sequential wrapper of `gc2gpc` and `arealw`.

Usage

```
cell.arealw(bounds, trimout = TRUE, zones = cabggpc, zbbmat = cabgbbmat,
  zoneareas = rbgarea, twokcensus = TRUE, digits = 6)
```

Arguments

<code>bounds</code>	A vector of length four, whose entries are (in order): Minimum longitude, maximum longitude, minimum latitude, maximum latitude.
<code>trimout</code>	Logical indicating output format. See <code>Value</code> for details.
<code>zones</code>	List of zonal polygons for which fractional overlap is to be determined.
<code>zbbmat</code>	“Zonal bounding box matrix.” Either a matrix of dimension <code>length(zones)</code> by 4, or character “none”. If “none,” an appropriate matrix is automatically computed by a call to <code>bbox.mat</code> . See ‘Details.’
<code>zoneareas</code>	A vector giving the area of each zone, or character “none”, indicating they must be calculated inside the function.
<code>twokcensus</code>	Logical indicating whether the data in question is block group level data for California from the 2000 US Census. If ‘TRUE’, the function automatically handles certain idiosyncracies associated with that data.
<code>digits</code>	Digits argument passed to <code>round</code> internally. Used in setting the tolerance for considering fractional overlap to be 100 percent. Due to numerical issues, overlap will not always be identically 1 even when a zone is contained entirely within the target. There should be little reason to modify this away from the default value of six.

Details

If not “none”, ‘zbbmat’ must take the form of a `length(zones)` by 4 matrix, where each row corresponds to the bounding latitude and longitude values for polygons in ‘zones’, the same format returned by the function `bbox.mat`. The columns correspond to low longitude, high longitude, low latitude and high latitude respectively.

Value

If `trimout` equals ‘FALSE’, the function returns a vector with length equal to the number of zones, and each entry corresponding to the fractional overlap for that zone. Since a large number of entries are likely to be zero, this is a wasteful storage method, but may be valuable as a convenient format with which to perform certain manual operations on the data.

Setting `trimout` equal to ‘TRUE’ returns an ‘n’ by 2 matrix, where ‘n’ is the number of zones with nonzero overlap. The first column contains the indices for those zones, and the second column contains the actual overlap weights.

Note

The US Census 2000 data for California contains a mismatch between the number of block groups and the number of block group polygons. The code handles this provided `twokcensus` is set to ‘TRUE’, but for all other datasets it is assumed that the number of data records is equal to the number of zones, and that the indexes match as well. If not, one or the other needs to be sorted appropriately.

Author(s)

Benjamin P. Bryant, bryant@prgs.edu

References

For general overview of areal interpolation and terminology, see Sadahiro, Y. Accuracy of areal interpolation: A comparison of alternative methods. *Journal of Geographical Systems* 1(4): 323-346 (1999).

Examples

```
data(cabgbbmat)
data(cabggpc)
data(rbgarea)
data(polytobg)

#An eight degree cell near the Orange/San Diego county border:
moh <- cell.arealw(bounds=c(-117.625,-117.5,33.375,33.5),trimout=TRUE)
```

damrats

Precomputed wildfire damage ratios for CA block groups

Description

Based on very approximate empirical analysis, gives the fraction of structures in a block group that would, on average, be damaged if that block group were contained in the perimeter of a wildfire. Note this will be zero for blocks above a certain housing density.

Usage

```
data(damrats)
```

Format

A vector of length 22133 (equal to the number of block groups). Entry [i] corresponds to block group [i] as referenced in 'bgvals' and identified in `polytobg[polytobg[,1]==i,]`.

Details

The estimates were derived by linking data from the archived National Interagency Coordination Center (NICC) Incident Management Situation Reports (<http://cidi.org/wildfire/>) with census data for the number of homes in a polygon, with GIS data on fire perimeters for fires in the NICC archives. They are highly uncertain, with an extremely low R squared.

Examples

```
data(damrats)
```

dp.interp

*Areal interpolation using precalculated weights***Description**

Given areal weights such as those returned by `arealw` or `mult.arealw`, calculates the approximate value enclosed in target polygons, as well as extremely conservative bounds on the estimates generated. At present makes the estimate makes the strong assumption that data of interest is uniformly distributed within the zonal polygons. The bounds assume the most pathological distribution, as appropriate.

Usage

```
dp.interp(wtmat, recvar = 1, recvals = bgvals, appdam = FALSE,
          trimmedin = FALSE, nobounds = FALSE, dr = damrats)
```

Arguments

<code>wtmat</code>	A matrix or vector containing at least areal weights. Accepts several forms of input. See <i>Details</i> .
<code>recvar</code>	An integer indicating which column of the record value matrix to use. That is, what variable is to be interpolated.
<code>recvals</code>	A matrix containing the data to be interpolated, with rows corresponding to zones, and columns corresponding to different variables.
<code>appdam</code>	CA Wildfire specific. A logical indicating whether or not to apply the empirically derived damage ratio.
<code>trimmedin</code>	A logical indicating whether or not <code>wtmat</code> was generated using the <code>trimout</code> option of <code>arealw</code> or <code>mult.arealw</code> . If so, it should be a two column matrix indicating non-zero zones and their weights. See the <code>arealw</code> help for details.
<code>nobounds</code>	Logical indicating whether or not to return the estimates without bounds. Default is 'FALSE', but there may be situations where having <code>nobounds</code> will be more convenient for handling data objects. See <i>Details</i> for how the bounds are generated.
<code>dr</code>	CA Wildfire specific. A vector giving precalculated damage ratios by block group.

Details

The input may be of several forms: Either a matrix such as that outputted by `arealw` when called with `trimout=TRUE`, or a vector of weights for one target polygon (the output of `arealw` with `trimout=FALSE`), or as a matrix, each column of which is a vector of weights for one target polygon (the output of `mult.arealw` with `trimout=FALSE`).

The lower bound estimate is generated by assuming any zone less than fully enclosed by the target polygon contributes zero to the total value contained. The upper bound is generated by assuming that any zone with nonzero overlap contributes its entire value. In all cases, zones that are entirely enclosed contribute their entire value.

Value

The most common output will be an 'n' by 3 matrix, though either a vector or matrix can be returned, depending on the arguments passed:

If passed weights for target polygons with `nobounds=TRUE`, the results will be a vector listing the estimate for each polygon.

Lastly, if `nobounds=FALSE` (the default), the result is an 'n' by 3 matrix, with 'n' equal to the number of target polygons. The columns correspond to expected value, lower bound and upper bound respectively.

Note that if the weights are passed in trimmed form (using `trimmedin=TRUE`), then only values for one polygon can be calculated. To quickly calculate for multiple polygons, simply run `lapply` on a list of polygon weights, such as that returned by `mult.arealw`.

Note

The bounds on the outputs when `appdam` is set to `TRUE` do not take into account uncertainty in the damage ratio. The absolute high would be to take the upper bound when not applying the damage ratio, and the absolute low is always zero.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>, and Anthony Westerling, <awesterling@ucmerced.edu>

References

Westerling, A. L. and B. P. Bryant, 2008. Climate Change and Wildfire in California. *Climatic Change*, 87: s231-249.

See Also

`arealw`, `mult.arealw`

Examples

```
data(bgvals)
data(cabgbbmat)
data(cabggpc)
data(damrats)
data(fpdemogpc)
data(rbgarea)
data(polytobg)

#Get areal weights in various forms:

fire1 <- arealw(fpdemogpc[[1]],trimout=TRUE) #single trimmed matrix
fire3 <- arealw(fpdemogpc[[3]],trimout=FALSE) #single untrimmed vector

#length 2 list of trimmed matrices.
both <- mult.arealw(target.ind=c(1,3), targetlist = fpdemogpc, trimout=TRUE)
```

```
#n by 2 matrix of cbound vectors:
bothfat <- mult.arealw(target.ind=c(1,3),targetlist=fpdemogpc,trimout=FALSE)

#Give input in the form of:

#One 'trim' matrix:
dp.interp(fire1,trimmedin=TRUE)

#One full length vector:
dp.interp(fire3,trimmedin=FALSE)

#A list of 'trim' matrices:
lapply(both, dp.interp, trimmedin=TRUE)

#A matrix of bound untrimmed vectors:
dp.interp(bothfat, trimmedin=FALSE)
```

fpdemogpc

A sample of California wildfire polygons

Description

This is a sample of four California wildfire polygons taken from a much larger dataset of 15491 wildfires that occurred in California. Included to allow demonstration of various package functions while keeping the size of required package data to a minimum.

Usage

```
data(fpdemogpc)
```

Format

A list of length 4, containing four polygons of format `gpc.poly`.

Details

Includes four fires: Union, Piru, Cedar and Border.

Source

Contact Anthony L. Westerling for the original datafile from which this list samples: <http://ulmo.ucmerced.edu/~westerling/in>

Examples

```
data(fpdemogpc)
```

`gc2gpc`*A function to create a rectangular polygon from a vector of bounds*

Description

Given a vector indicating minimum and maximum latitude and longitudes, creates a rectangular polygon of class `gpc.poly`, which can then be used by other areal weighting functions. This is simply a very thin wrapper for `as(..., \dQuote{gpc.poly})`, and use primarily as a convenient internal function.

Usage

```
gc2gpc(bounds)
```

Arguments

`bounds` A vector of length four, whose entries are (in order): Minimum longitude, maximum longitude, minimum latitude, maximum latitude.

Value

A rectangular polygon object of class `gpc.poly`.

Author(s)

Benjamin P. Bryant, bryant@prgs.edu

See Also

[cell.arealw](#)

Examples

```
poly1 <- gc2gpc(c(0,1,0,1))
poly2 <- gc2gpc(c(.3,1.1,.25,1.02))

plot(poly1)
plot(poly2,add=TRUE)
plot(intersect(poly1,poly2), poly.args=list(col="blue"),add=TRUE)

## The function is currently defined as
function(bounds) {

  minglon <- bounds[1]
  maxglon <- bounds[2]
  minglat <- bounds[3]
  maxglat <- bounds[4]
```

```

gpoly <- cbind(c(minglon, maxglon, maxglat, minglon), c(minglat,
  minglat, maxglat, maxglat))

gpoly <- as(gpoly, "gpc.poly")

return(gpoly)
}

```

gridgpc

Gridcells for California in polygon form, with indexing for climate mask

Description

A list of three elements, containing a list of gpc.polys, and two indexing vectors.

Usage

```
data(gridinws)
```

Format

The first component is itself a list. Each element of that inner list is a gpc.poly corresponding to an 1/8 degree gridcell in California. For storage efficiency, these were extracted from a longer list containing gridcells spanning the entire bounding box for California.

To allow reprojection onto the same reference grid used by ‘MASK’, the original indices from that longer list are retained as a vector comprising the second component of gridgpc. That is, gridgpc[[2]][i] gives the original index of polygon gridgpc[[1]][[i]] in the list of gridcells covering the entire california block group. By applying the appropriate formula (given elsewhere), this index can then be converted back to the i,j coordinates of the ‘MASK’ object.

Lastly, to avoid needlessly generating data for gridcells that are masked out, the third component indexes only those gridcells that are not masked out by MASK. Thus, a for loop through the elements of gridgpc[[3]] will reach all non-masked gridcell in California.

Examples

```
data(gridgpc)
```

 gridinws

A list of index-weight matrices for the grid and California block groups

Description

A list of 2871 elements corresponding to 1/8 degree gridcells in California. Each element of the list is a matrix that contains the block group indices and weights (fraction of area overlap) for one gridcell, in form that can be input to `dp.interp`. The cells themselves are stored in `gridgpc`.

Usage

```
data(gridinws)
```

Format

Each element of the list is a matrix, each row of which corresponds to a block group, with the first column giving the block group identifier, and the second giving the area fraction of that block group contained inside the gridcell `gridgpc[[1]][[i]]`.

Examples

```
data(gridinws)
```

 MASK

A matrix indicating whether CA gridcells do or do not have fire estimates

Description

A matrix whose entries correspond to 1/8 degree gridcells referenced against the (approximately) outer bounds of California.

Usage

```
data(MASK)
```

Format

A matrix with 93 rows and 97 columns, with each entry a 0 or 1.

Details

Estimated wildfire probabilities are available for 1/8 degree gridcells in the western United States. However, estimated data is not available for every gridcell, either because of missing data required for the model, or the gridcell comprises an regions for which wildfire is not relevant, such as an entirely urban environment or water feature (eg, the Pacific Ocean).

Entry [1,1] corresponds to the eighth degree gridcell centered at -124.5625 longitude and 31.9375 latitude. Entry [93,97] corresponds to the gridcell centered at -113.0625 longitude and 43.9375 latitude.

Examples

```
data (MASK)
image (MASK)
```

mult.arealw

Areal weights for multiple target polygons

Description

Essentially a wrapper for `arealw`, allowing the user to retrieve areal weights for multiple target polygons over the same list of zonal polygons. Permits calculations for all elements of a list of target polygons, or a those specified by vector indexing elements in that list. Depending on the calling options, returns the result as a list or matrix.

Usage

```
mult.arealw(target.ind = "all", targetlist = fireppgc, trimout = TRUE,
           zones = cabggpc, zbbmat = cabgbbmat, zoneareas = rbgarea, twokcensus = TRUE,
           digits = 6, verbose = 2)
```

Arguments

<code>target.ind</code>	A vector containing list indices of target polygons, or character "all", which indicates the entire list is to be used.
<code>targetlist</code>	A list of possible target polygons
<code>trimout</code>	Logical indicating output format. See <code>Value</code> for details.
<code>zones</code>	List of zonal polygons for which fractional overlap is to be determined.
<code>zbbmat</code>	"Zonal bounding box matrix." Either a matrix of dimension <code>length(zones)</code> by 4, or character "none". If "none," an appropriate matrix is automatically computed by a call to <code>bbox.mat</code> . See 'Details.'
<code>zoneareas</code>	A vector giving the area of each zone or character "none", indicating they must be calculated inside the function.
<code>twokcensus</code>	Logical indicating whether the data in question is block group level data for California from the 2000 US Census. If 'TRUE', the function automatically handles certain idiosyncracies associated with that data.

digits	Digits argument passed to <code>round</code> internally. Used in setting the tolerance for considering fractional overlap to be 100 percent. Due to numerical issues, overlap will not always be identically 1 even when a zone is contained entirely within the target. There should be little reason to modify this away from the default value of six.
verbose	An integer indicating how frequently to print progress through the target polygons. Will print a message every time 10 to the <code>verbose</code> target polygons have been processed. Setting <code>verbose</code> negative results in no progress output.

Details

If not “none”, ‘zbbmat’ must take the form of a `length(zones)` by 4 matrix, where each row corresponds to the bounding latitude and longitude values for polygons in ‘zones’, the same format returned by the function `bbox.mat`. The columns correspond to low longitude, high longitude, low latitude and high latitude respectively.

Value

If `trimout` equals ‘FALSE’, the function returns a matrix with number of rows equal to the number of zones, and columns equal to the number of target polygons specified by ‘target.ind’. Entry [i,j] then corresponds to the fractional overlap for of zone i in target polygon j. Since a large number of entries are likely to be zero, this is a wasteful storage method, but may be valuable as a convenient format with which to perform certain manual operations on the data.

Setting `trimout` equal to ‘TRUE’ returns a list of length equal to `length(target.ind)`. Each element of the list is an ‘n’ by 2 matrix, where ‘n’ is the number of zones with nonzero overlap for that the target polygon associated with that list element. The first column contains the indices for those zones with overlap, and the second column contains the actual overlap weights.

Whatever object returned also has the ‘call’ attribute, allowing one to identify target polygons by their original index in the original unsubsetted polygon list.

Note

The US Census 2000 data for California contains a mismatch between the number of block groups and the number of block group polygons. The code handles this provided `twokcensus` is set to ‘TRUE’, but for all other datasets it is assumed that the number of datarecords is equal two the number of zones, and that the indexes match as well. If not, one or the other needs to be sorted appropriately.

Author(s)

Benjamin P. Bryant, <bryant@prgs.edu>

References

For general overview of areal interpolation and terminology, see Sadahiro, Y. Accuracy of areal interpolation: A comparison of alternative methods. *Journal of Geographical Systems* 1(4): 323-346 (1999).

See Also[mult.arealw](#)**Examples**

```

data(cabgbbmat)
data(cabggpc)
data(fpdemogpc)
data(rbgarea)
data(polytobg)

#Get two areal weights independently using arealw:

fire1 <- arealw(fpdemogpc[[1]],trimout=TRUE)
fire3 <- arealw(fpdemogpc[[3]],trimout=TRUE)

#Stick them together and see if they match the output of mult.arealw, which
#does them at the same time:

stucktogether <- list(fire1,fire3)

both <- mult.arealw(target.ind=c(1,3), targetlist = fpdemogpc, trimout=TRUE)

attributes(stucktogether) <- NULL
attributes(both) <- NULL

#should be true:
identical(both, stucktogether)

```

polytobg

Link CA block group polygons to block group data records

Description

Each row corresponds to a block group polygon, and contains information to identify the corresponding block group, by index in [bgvals](#) as well as Logical Record Number, County ID, Block, and Group number.

Usage

```
data(polytobg)
```

Format

A data frame with 22195 observations on the following 5 variables.

bg-index The index in [bgvals](#) of the corresponding block group

LRN The logical record number - a unique identifier from the summary file 3

county A numeric county identifier

block The block number

group The group number within the block

References

US Census Summary File 3 Documentation <http://www.census.gov/prod/cen2000/doc/sf3.pdf>

Examples

```
data (polytobg)
```

rbgarea	<i>California census block group areas</i>
---------	--

Description

A vector of areas for California block groups, given in lat/lon degree-area, and thus latitude specific. Primarily for internal function use.

Usage

```
data (rbgarea)
```

Format

A vector of length 22133, with each entry corresponding to those in `bgvals`.

Source

For census block groups containing only one GIS record, this value is direct. For census block groups corresponding to multiple GIS records, this value is the sum of individual GIS record values.

Examples

```
data (rbgarea)
```

sab.areal.interp *Single polygon area-based areal interpolation*

Description

Given a single target polygon, a list of zonal polygons, and a matrix of data associated with the zonal polygons, interpolate zonal polygon data to the target polygon. As a wrapper of `arealw` and `dp.interp`, it generates areal weights as an intermediate product, which can also be returned if desired. This allows other variables in the data record to be interpolated to the same polygon without having to redo the costly weight-calculating step.

Usage

```
sab.areal.interp(target, recvar = 1, recvals = bgvals, appdam = FALSE,
  zones = cabggpc, zbbmat= cabgbbmat, zoneareas = rbgarea, twokcensus = TRUE,
  digits = 6, keepwts = TRUE, dr=damrats, nobounds=FALSE, trimout=TRUE)
```

Arguments

<code>target</code>	A target polygon of class <code>gpc.poly</code> .
<code>recvar</code>	An integer indicating which column of the record value matrix to use. That is, what variable is to be interpolated.
<code>recvals</code>	A matrix of record values containing the data to be interpolated, with rows corresponding to zones, and columns corresponding to different variables.
<code>appdam</code>	CA Wildfire specific. A logical indicating whether or not to apply the empirically derived damage ratio.
<code>zones</code>	List of zonal polygons from which data will be interpolated to the target.
<code>zbbmat</code>	“Zonal bounding box matrix.” Either a matrix of dimension <code>length(zones)</code> by 4, or character “none”. If “none,” an appropriate matrix is automatically computed by a call to <code>bbox.mat</code> . See ‘Details.’
<code>zoneareas</code>	A vector giving the area of each zone or character “none”, indicating they must be calculated inside the function.
<code>twokcensus</code>	Logical indicating whether the data in question is block group level data for California from the 2000 US Census. If ‘TRUE’, the function automatically handles certain ideosyncracies associated with that data.
<code>digits</code>	Digits argument passed to <code>round</code> internally. Used in setting the tolerance for considering fractional overlap to be 100 percent. Due to numerical issues, overlap will not always be identically 1 when a zone is contained entirely within the target. There should be little reason to modify this away from the default value of six.
<code>keepwts</code>	Whether or not to also return the areal weights calculated by <code>arealw</code> . See Value for details.
<code>dr</code>	CA Wildfire specific. A vector giving precalculated damage ratios by block group. Only used if <code>appdam=TRUE</code> .

nobounds	Logical indicating whether or not to return the estimates without bounds. Default is 'FALSE', but there may be situations where having no bounds will be more convenient for handling data objects. See Details for how the bounds are generated.
trimout	Logical indicating output format if <code>keepwts=TRUE</code> . See Value for details.

Details

If not "none", 'zbbmat' must take the form of a `length(zones)` by 4 matrix, where each row corresponds to the bounding latitude and longitude values for polygons in 'zones', the same format returned by the function [bbox.mat](#). The columns correspond to low longitude, high longitude, low latitude and high latitude respectively.

The lower bound estimate is generated by assuming any zone less than fully enclosed by the target polygon contributes zero to the total value contained. The upper bound is generated by assuming that any zone with nonzero overlap contributes its entire value. In all cases, zones that are entirely enclosed contribute their entire value.

Value

The output under default settings is a list of two elements. The first element is a 1 by 3 matrix giving the expected value, lower bound and upper bound on enclosed value. The second element is a matrix with the first column indexing those zones with nonzero overlap, and the second column gives those overlaps. This matrix can be passed to `dp.interp` to generate other interpolated values for the same target polygon.

If `nobounds=TRUE`, lower and upper bound are not returned, and the result is just a vector of length one.

If `keepwts=FALSE`, the function returns only the matrix or vector as above, and the areal weights are not returned.

If `trimout=FALSE`, the weights are returned as a vector of length equal to `nrow(recvals)`, rather than as a matrix containing specifying only those nonzero weights. See help for [arealw](#) for more info.

Thus, there can be six different possible output forms. The default should suffice in most cases.

Note

The bounds on the outputs when `appdam` is set to `TRUE` do not take into account uncertainty in the damage ratio. The absolute high would be to take the upper bound when not applying the damage ratio, and the absolute low is always zero.

Author(s)

Benjamin P. Bryant, bryant@prgs.edu, and Anthony Westerling, awesterling@ucmerced.edu

See Also

[dp.interp](#), [arealw](#), [mult.arealw](#)

Examples

```
data(bgvals)
data(cabgbbmat)
data(cabggpc)
data(damrats)
data(fpdemogpc)
data(rbgarea)
data(polytobg)

#Calculate structures enclosed by the Cedar fire:
cedar <- sab.areal.interp(fpdemogpc[[3]], recvar=1, appdam=FALSE, keepwts=TRUE)
cedar[[1]]

#Using the weights already generated, calculate total value enclosed:
dp.interp(cedar[[2]], recvar=2, appdam=FALSE, trimmedin=TRUE)

#Again using the weights already generated, calculate estimated houses
#destroyed:
dp.interp(cedar[[2]], recvar=1, appdam=TRUE, trimmedin=TRUE)
```

Index

*Topic **datasets**

bgvals, [17](#)
cabgbbmat, [18](#)
cabggpc, [19](#)
damrats, [21](#)
fpdemogpc, [24](#)
gridgpc, [26](#)
gridinws, [27](#)
MASK, [27](#)
polytobg, [30](#)
rbgarea, [31](#)

*Topic **package**

AIGIS-package, [2](#)

*Topic **spatial**

ab.areal.interp, [3](#)
ab.gbounds, [6](#)
ab.gcbounds, [8](#)
ab.gmu, [9](#)
ab.gstats, [10](#)
area2frac, [12](#)
arealw, [14](#)
bbox.mat, [16](#)
cell.arealw, [19](#)
dp.interp, [22](#)
gc2gpc, [25](#)
mult.arealw, [28](#)
sab.areal.interp, [32](#)

ab.areal.interp, [3](#)
ab.gbounds, [6](#)
ab.gcbounds, [7, 8](#)
ab.gmu, [9](#)
ab.gstats, [10](#)
AIGIS (*AIGIS-package*), [2](#)
AIGIS-package, [2](#)
area2frac, [12](#)
arealw, [14, 16, 23, 33](#)

bbox.mat, [4, 13, 15, 16, 20, 29, 33](#)
bgvals, [17, 30](#)

cabgbbmat, [18](#)
cabggpc, [19](#)
cell.arealw, [19, 25](#)
damrats, [21](#)
dp.interp, [22, 33](#)
fpdemogpc, [24](#)
gc2gpc, [25](#)
gridgpc, [26](#)
gridinws, [27](#)
MASK, [27](#)
mult.arealw, [15, 23, 28, 30, 33](#)
polytobg, [30](#)
rbgarea, [31](#)
sab.areal.interp, [32](#)