

# Package ‘ejanalysis’

December 4, 2015

**Type** Package

**Title** Tools for Environmental Justice Analysis

**Version** 0.1.0

**Date** 2015-03-14

**Author** info@ejanalysis.com

**Maintainer** ejanalysis.com <info@ejanalysis.com>

**Description** Tools that simplify some basic tasks in exploring and analyzing a dataset in a matrix or data.frame that contains data on demographics (e.g., counts of residents in poverty) and local environmental indicators (e.g., an air quality index), with one row per spatial location (e.g., Census block group). Key functions help to find relative risk or similar ratios of means in demographic groups, etc. The analyze.stuff, proxistat, ejanalysis, and countyhealthrankings packages, once made public can be installed from github using the devtools package: devtools::install\_github(c("`ejanalysis/analyze.stuff", "`ejanalysis/proxistat", "`ejanalysis/ejanalysis", "`ejanalysis/countyhealthrankings"))

**Depends** R (>= 3.1.2)

**License** MIT + file LICENSE

**Suggests** countyhealthrankings,  
data.table

**Imports** sp

**URL** <http://ejanalysis.github.io>  
<http://www.ejanalysis.com/>

## R topics documented:

assign.map.bins . . . . .	3
assign.pctiles . . . . .	4
assign.pctiles.alt2 . . . . .	6
clean.fips . . . . .	7
clean.fips1215 . . . . .	9
ej.added . . . . .	9
ej.indexes . . . . .	10
ejanalysis . . . . .	13
ejRRadded . . . . .	14

flagged . . . . .	14
flagged.by . . . . .	15
flagged.only.by . . . . .	16
geofips . . . . .	17
geotype . . . . .	18
get.county.info . . . . .	19
get.epa.region . . . . .	21
get.fips.bg . . . . .	22
get.fips.county . . . . .	22
get.fips.etc . . . . .	23
get.fips.st . . . . .	24
get.fips.tract . . . . .	25
get.name.county . . . . .	25
get.name.state . . . . .	26
get.pctile . . . . .	28
get.state.info . . . . .	29
KolmPollak . . . . .	31
lookup.pctile . . . . .	32
make.bin.cols . . . . .	33
make.bin.pctile.cols . . . . .	34
make.bin.pctile.cols.byzone . . . . .	36
make.pctile.cols . . . . .	36
make.pctile.cols.alt2 . . . . .	37
names.dvars . . . . .	39
names.ejvars . . . . .	39
names.evars . . . . .	40
pct.moe . . . . .	41
pop.cdf . . . . .	42
pop.cdf.density . . . . .	43
pop.cdf2 . . . . .	44
pop.ecdf . . . . .	46
rollup . . . . .	48
rollup.pct . . . . .	49
RR . . . . .	50
RR.cut.if.gone . . . . .	52
RR.if.address.top.x . . . . .	53
RR.means . . . . .	54
RR.plot . . . . .	56
RR.table . . . . .	57
RR.table.add . . . . .	58
RR.table.sort . . . . .	59
RR.table.view . . . . .	60
RR.table.vs.us . . . . .	61
rrf . . . . .	62
rrfv . . . . .	64
state.health.url . . . . .	65
sum.moe . . . . .	67
url.census . . . . .	68
url.censusblock . . . . .	69
url.open . . . . .	70
url.qf . . . . .	71
write.RR.tables . . . . .	72

---

assign.map.bins	<i>Assign Each Row (Place) to a Bin Based on Cutpoints</i>
-----------------	--

---

## Description

Takes a vector (not matrix or df) of values and returns a vector of bin numbers. For creating color-coded maps (choropleths), assign each place (e.g., each row of a single column) to a bin. Each bin represents one map color, and is defined by cutoff values.

## Usage

```
assign.map.bins(pctiles.vector, cutpoints = c((0:9)/10, 0.95, 1),
  labels = 1:11)
```

## Arguments

pctiles.vector	Vector of percentiles as decimal fractions, 0 to 1, one per place.
cutpoints	Optional vector of cutpoints defining edges of bins. Default is every 0.10 from 0 to 1.00, as well as 0.95
labels	vector of bin numbers, optional (default is 1 through 11, and NA values are put in bin 0).

## Details

The default bins 0-11 are defined as follows:

bin 0: PCTILE=NA

...

bin 9:  $0.80 \leq \text{PCTILE} < 0.90$

bin 10:  $0.90 \leq \text{PCTILE} < 0.95$

bin 11:  $0.95 \leq \text{PCTILE} \leq 1.00$

## Value

Returns a vector bin numbers.

## See Also

[make.bin.pctile.cols](#) to call functions below, converting columns of values to percentiles and then bins

[assign.pctiles](#) for one vector, assign (weighted) percentile (quantile) to each value within its zone (subset)

[assign.pctiles.alt2](#) as an alternative method, to replicate assign.pctiles, but not by zone

[get.pctile](#) to get (weighted) percentile of just 1+ values within given vector of values

[make.pctile.cols](#) for a data.frame, assign percentiles, return a same-sized df that is wtd.quantile of each value within its column

[make.pctile.cols.alt2](#) as an alternative method, to replicate make.pctile.cols

[assign.map.bins](#) for one vector (or data.frame) of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.

[make.bin.cols](#) for a data.frame of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[write.pctiles](#) to save file that is lookup table of percentiles for columns of a data.frame  
[write.pctiles.by.zone](#) to save file that is lookup table of percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[write.wtd.pctiles](#) to save file that is lookup table of weighted percentiles for columns of a data.frame  
[write.wtd.pctiles.by.zone](#) to save file that is lookup table of weighted percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[lookup.pctile](#) to look up current approx weighted percentiles in a lookup table that is already in global memory

## Examples

```
junk<-c(0,0.799,0.8, 0.8000001, 0.8999999,0.95,0.95001,1)
data.frame(pctile=junk, bin=assign.map.bins(junk))
# How it puts these in bins by default:
#      pctile bin  notes
#1  0.0000000    1
#2  0.7990000    8
#
#3  0.8000000    9 (0.80<=PCTILE<0.90 )
#4  0.8000001    9 (0.80<=PCTILE<0.90 )
#5  0.8999999    9 (0.80<=PCTILE<0.90 )
#
#6  0.9000000   10 (0.90<=PCTILE<0.95 )
#7  0.9001000   10 (0.90<=PCTILE<0.95 )
#8  0.9499990   10 (0.90<=PCTILE<0.95 )
#
#9  0.9500000   11 (0.95<=PCTILE<=1.00 ) I.E. THIS INCLUDES 100th percentile
#10 0.9500100   11 (0.95<=PCTILE<=1.00 ) I.E. THIS INCLUDES 100th percentile
#11 1.0000000   11 (0.95<=PCTILE<=1.00 ) I.E. THIS INCLUDES 100th percentile
```

---

assign.pctiles

*Assign percentiles to vector of values (weighted, by zone)*

---

## Description

For the vector, look at the distribution of values across all rows in a given zone (e.g., places in zone), and find what percentile a given value is at.

## Usage

```
assign.pctiles(values, weights = NULL, zone = NULL, na.rm = TRUE)
```

## Arguments

values	vector, required, with numeric values. To do this with a matrix, see <a href="#">make.pctile.cols</a>
weights	Optional, NULL by default (not fully tested), vector of weights for weighted percentiles (e.g., population weighted).
zone	Optional, NULL by default, defines subsets of rows, so a percentile is found among rows that are within a given zone only.

na.rm                    NOT IMPLEMENTED HERE. Logical, optional, TRUE by default. Should NA values (missing data) be removed first to get percentile of those with valid data. If FALSE, NA values are treated as being at the high percentiles.

## Details

Relies on the [wtd.Ecdf](#) function. Could also add parameter like in [rank\(\)](#), [na.last](#), defining [na.rm](#) but also where to rank NA values if included, etc. Default now is like [na.last=NA](#), but like [na.last='last'](#) if [na.rm=FALSE](#). Could also add parameter like in [rank\(\)](#), [ties.method](#), defining if ties get min, max, or mean of percentiles initially assigned to ties. Default for ties right now is like [ties.method=max](#) (which might not be what [assign.pctiles\(\)](#) does in fact).

## Value

Returns a numeric vector same size as [x](#), but if [zone](#) is specified, provides percentile with given [zone](#).

## See Also

[make.bin.pctile.cols](#) to call functions below, converting columns of values to percentiles and then bins  
[assign.pctiles](#) for one vector, assign (weighted) percentile (quantile) to each value within its [zone](#) (subset)  
[assign.pctiles.alt2](#) as an alternative method, to replicate [assign.pctiles](#), but not by [zone](#)  
[get.pctile](#) to get (weighted) percentile of just 1+ values within given vector of values  
[make.pctile.cols](#) for a data.frame, assign percentiles, return a same-sized df that is [wtd.quantile](#) of each value within its column  
[make.pctile.cols.alt2](#) as an alternative method, to replicate [make.pctile.cols](#)  
[assign.map.bins](#) for one vector (or data.frame) of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[make.bin.cols](#) for a data.frame of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[write.pctiles](#) to save file that is lookup table of percentiles for columns of a data.frame  
[write.pctiles.by.zone](#) to save file that is lookup table of percentiles for columns of a data.frame, for each geographic [zone](#) (subset of rows)  
[write.wtd.pctiles](#) to save file that is lookup table of weighted percentiles for columns of a data.frame  
[write.wtd.pctiles.by.zone](#) to save file that is lookup table of weighted percentiles for columns of a data.frame, for each geographic [zone](#) (subset of rows)  
[lookup.pctile](#) to look up current approx weighted percentiles in a lookup table that is already in global memory

## Examples

```
x <- c(30, 40, 50, 12,12,5,5,13,13,13,13,13,8,9,9,9,9,10:20,20,20,20,21:30); wts <- rep(c(2,3), length(x)/
cbind(wts, x, PCTILE=assign.pctiles(x,wts))
```

```
# PERCENTILE OF ALL, NOT JUST THOSE WITH VALID DATA, IF na.rm=FALSE, but then NA values preclude high percenti
x <- c(NA, NA, NA, NA,NA,NA,NA,NA,NA,NA,NA,13,13,8,9,9,9,9,10:20,20,20,20,21:30); wts <- rep(c(2,3), length(x)
cbind(wts, x, PCTILE.alt2=assign.pctiles.alt2(x, wts, na.rm=FALSE), pctile=assign.pctiles(x,wts))[order(x),
cbind(wts, x, PCTILE.alt2=assign.pctiles.alt2(x, wts, na.rm=TRUE), pctile=assign.pctiles(x,wts))[order(x),]
```

V=9

```

sum(wts[!is.na(x) & x <= V]) / sum(wts[!is.na(x)])

#A value (V) being at this PCTILE% means that (assuming na.rm=TRUE):

# V >= x for      PCTILE% of wts      (for non-NA x), so
# V < x   for 100% - PCTILE% of wts    (for non-NA x), or
# PCTILE% of all wts have V >= x (for non-NA x), so
# 100% - PCTILE% of all wts have V < x (for non-NA x).

x <- c(32, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 13, 13, 8, 9, 9, 9, 9, 10:20, 20, NA, 20, 21:30); wts <- rep(c(2, 3), length(x))
cbind(wts, x, PCTILE.alt2=assign.pctiles.alt2(x, wts, na.rm=FALSE), pctile=assign.pctiles(x, wts))[order(x), ]
cbind(wts, x, PCTILE.alt2=assign.pctiles.alt2(x, wts, na.rm=TRUE), pctile=assign.pctiles(x, wts))[order(x), ]

```

---

assign.pctiles.alt2	<i>Assign percentiles to values (alternative formula, and not by zone)</i>
---------------------	--

---

## Description

For each column look at the distribution of values across all rows, and find what percentile a given value is at.

## Usage

```
assign.pctiles.alt2(x, weights = NULL, na.rm = TRUE, zone = NULL)
```

## Arguments

x	vector or data.frame
weights	Optional, NULL by default (not fully tested), vector of weights for weighted percentiles (e.g., population weighted).
na.rm	Logical, optional, TRUE by default. Should NA values (missing data) be removed first to get percentile of those with valid data. If FALSE, NA values are treated as being at the high percentiles.
zone	Optional, NULL by default, *** not yet implemented here.

## Details

Assign percentile as cumulative sum of (the weights ranked by the value x). Then fixes ties. # Could also add parameter like in rank(), na.last, defining na.rm but also where to rank NA values if included, etc. Default now is like na.last=NA, but like na.last='last' if na.rm=FALSE Could also add parameter like in rank(), ties.method, defining if ties get min, max, or mean of percentiles initially assigned to ties. Default for ties right now is like ties.method=max (which might not be what assign.pctiles() does in fact).

## Value

Returns a numeric vector or data.frame same size as x.

## See Also

[make.bin.pctile.cols](#) to call functions below, converting columns of values to percentiles and then bins  
[assign.pctiles](#) for one vector, assign (weighted) percentile (quantile) to each value within its zone (subset)  
[assign.pctiles.alt2](#) as an alternative method, to replicate [assign.pctiles](#), but not by zone  
[get.pctile](#) to get (weighted) percentile of just 1+ values within given vector of values  
[make.pctile.cols](#) for a data.frame, assign percentiles, return a same-sized df that is wtd.quantile of each value within its column  
[make.pctile.cols.alt2](#) as an alternative method, to replicate [make.pctile.cols](#)  
[assign.map.bins](#) for one vector (or data.frame) of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[make.bin.cols](#) for a data.frame of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[write.pctiles](#) to save file that is lookup table of percentiles for columns of a data.frame  
[write.pctiles.by.zone](#) to save file that is lookup table of percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[write.wtd.pctiles](#) to save file that is lookup table of weighted percentiles for columns of a data.frame  
[write.wtd.pctiles.by.zone](#) to save file that is lookup table of weighted percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[lookup.pctile](#) to look up current approx weighted percentiles in a lookup table that is already in global memory

## Examples

```
x <- c(30, 40, 50, 12,12,5,5,13,13,13,13,13,8,9,9,9,9,9,10:20,20,20,20,20,21:30); weights <- rep(c(2,3), length(
cbind(weights, x, PCTILE=assign.pctiles.alt2(x,weights)))

# PERCENTILE OF ALL, NOT JUST THOSE WITH VALID DATA, IF na.rm=FALSE, but then NA values preclude high percentiles
x <- c(NA, NA, NA, NA,NA,NA,NA,NA,NA,NA,NA,13,13,8,9,9,9,9,9,10:20,20,20,20,20,21:30); weights <- rep(c(2,3), length(
cbind(weights, x, PCTILE.alt2=assign.pctiles.alt2(x, weights, na.rm=FALSE), pctile=assign.pctiles(x,weights),
cbind(weights, x, PCTILE.alt2=assign.pctiles.alt2(x, weights, na.rm=TRUE), pctile=assign.pctiles(x,weights)))

V=9
sum(weights[!is.na(x) & x <= V]) / sum(weights[!is.na(x)])

#A value (V) being at this PCTILE% means that (assuming na.rm=TRUE):

# V >= x for PCTILE% of weights (for non-NA x), so
# V < x for 100% - PCTILE% of weights (for non-NA x), or
# PCTILE% of all weights have V >= x (for non-NA x), so
# 100% - PCTILE% of all weights have V < x (for non-NA x).

x <- c(32, NA, NA, NA,NA,NA,NA,NA,NA,NA,NA,13,13,8,9,9,9,9,9,10:20,20,NA,20,21:30); weights <- rep(c(2,3), length(
cbind(weights, x, PCTILE.alt2=assign.pctiles.alt2(x, weights, na.rm=FALSE), pctile=assign.pctiles(x,weights),
cbind(weights, x, PCTILE.alt2=assign.pctiles.alt2(x, weights, na.rm=TRUE), pctile=assign.pctiles(x,weights)))
```

## Description

Clean up US Census FIPS (add any missing leading zeroes)

## Usage

```
clean.fips(fips)
```

## Arguments

fips	Vector of numeric or character class, required. Can be state FIPs as number or character, for example.
------	--

## Details

For information on FIPS codes, see <http://www.census.gov/geo/reference/ansi.html>, and also see <https://www.census.gov/geo/reference/geoidentifiers.html>

If FIPS provided is 1-2 digits long assume it is a State.

If FIPS provided is 3 digits long, it is a mistake and return NA.

If FIPS provided is 4-5 digits, assume it is a County.

If FIPS provided is 6-9 digits, it is a mistake and return NA.

If FIPS provided is 10 digits long, assume it is a tract missing a leading zero on the state portion (should have 11 characters).

If FIPS provided is 11 digits long, assume it is a tract (correctly 11 characters), not simply a block group FIPS missing a leading zero (block group FIPS would correctly would have 12 characters).

If FIPS provided is 12 digits long, assume it is a block group (correctly 12 characters).

If FIPS provided is 13 digits long, it is a mistake and return NA.

If FIPS provided is 14 OR 15 digits long, assume it is a block.

## Value

Returns vector of FIPS (all characters from 2-digit State code onwards as appropriate) as character with leading zeroes

## See Also

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

## Examples

```
 #(No example yet)
```



---

clean.fips1215	<i>Check and clean Census block group or block FIPS</i>
----------------	---

---

### Description

Check if valid Census block group or block FIPS, warn if not, add missing leading zero if inferred.

### Usage

```
clean.fips1215(fips)
```

### Arguments

fips	Vector of one or more elements, ideally character class, ideally 12 or 15 characters long (block group or block), required.
------	---

### Details

For information on FIPS codes, see <http://www.census.gov/geo/reference/ansi.html>, and also see <https://www.census.gov/geo/reference/geoidentifiers.html>

### Value

Returns a vector of one or more character elements, same lengths as fips, NA if NA input

### See Also

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

### Examples

```
samplefips <- c("011030001003", "011030001003001", 11030001003001, 35, 1, NA, 'invalidtext', '02')
clean.fips1215(samplefips)
clean.fips1215("011030001003001")
```

---

ej.added	<i>Contribution of each place to net excess people-points in demographic group</i>
----------	--

---

### Description

US total sum of net excess vulnerable \* E = net excess people-points and contribution of one place = what the EJ Index intends to indicate

### Usage

```
ej.added(e, d, p, vs = "nond", silent = TRUE)
```

**Arguments**

e	Environmental indicator
d	Demographic indicator
p	Population count (universe for d)
vs	Reference group, optional, 'nond' by default which means the reference group is everyone other than the d group (everyone else), but can also specify 'avg' in which case the overall average value including the d group is used as the reference value.
silent	Optional, TRUE by default, in which case more details are printed.

**Details**

WORK IN PROGRESS. \*\*\*\* e.g., presumes one setting for vs..... \*\*\*\*\* Directly calculate total number of excess people-points in a demographic subgroup, across all locations \*\*\*\* here by default defining "excess" as above what it would be if e in d group were same as e in nond group.\*\*\*\* where people-points are  $e \cdot p \cdot d$

e = environmental points or individual risk (vector of places)

p = population counts (vector of places)

d = demographic fraction that is in specified demographic group (vector of places)

For example, if e=cancer risk (individual risk) and p=pop and d= value returned is number of cases among lowincome individuals in excess of what it would be if their average risk was the same as that of nonlowincome individuals.

net excess vulnerable \* E = net excess people-points

vs can be 'avg' or 'nonD' (default) (not case sensitive),

for, respectively, excess cases relative to risk scenario where

avg d's e is set to that of avg person (all people including d and nonD),

or avg d's e is set to that of avg nonD person.

**Value**

Returns numeric vector

**Examples**

```
x=data.frame(pop=rep(1000, 10), pct=0.05+6*(1:10)/100, e= (10*(1:10))/100 )
y=ej.added(x$e, x$pct, x$pop, silent=FALSE)
```

---

ej.indexes

*Calculate environmental justice (EJ) index for each place*

---

**Description**

Create an index that combines environmental and demographic indicators for each Census unit (e.g., block group).

**Usage**

```
ej.indexes(env.df, demog, weights, us.demog, universe.us.demog, as.df = TRUE,
  prefix = "EJ.DISPARITY.", type = 1, na.rm = FALSE)
```

**Arguments**

env.df	Environmental indicators vector or numeric data.frame, one column per environmental factor, one row per place (e.g., block group).
demog	Demographic indicator(s) vector or data.frame, numeric fractions of population that is in specified demographic group (e.g., fraction below poverty line), one per place.
weights	Optional, and default is equal weighting. If us.demog and universe.us.demog are not specified, weights are used to find weighted mean of demog to use as area-wide overall average (e.g., population-weighted average percent Hispanic, for all block groups in USA). One weight per place.
us.demog	Optional number specifying overall area-wide value for demog (e.g., US percent Hispanic). Default is to calculate it as weighted mean of demog, where the weights are universe.us.demog if specified, or else just 'weights'. If the weights are population counts and demog is percent Hispanic, for example, us.demog is the percent of US population that is Hispanic.
universe.us.demog	Optional numeric vector. If specified and us.demog not specified, used instead of weights to get weighted mean of demog to find area-wide demog. This should be the actual denominator, or universe, that was used to create percent demog – universe.us.demog if specified should be a vector that has the count, for each place, of the denominator for finding the US overall percent and this may be slightly different than total population. For example if demog=places\$pctlwinc then true universe.us.demog=places\$povknownratio which is the count for whom poverty ratio is known in each place, which is <= pop.
prefix	Optional character string used as first part of each colname in results. Default is "EJ.DISPARITY."
type	Specifies type of EJ Index. Default is type=1. Several formulas are available: <ul style="list-style-type: none"> <li>• For type=1, <math>ej.indexes = weights * env.df * (demog - us.demog)</math> ## This is the EJ Index in EJSCREEN 2015. Note: us.demog could also be called d.avg.all, and ** note that na.rm is currently ignored for type=1</li> <li>• For type=1.5, <math>ej.indexes = weights * env.df * (demog - d.avg.all.elsewhere)</math> # for a place that is one of many this can be almost identical to type 1</li> <li>• For type=2.1, <math>ej.indexes = weights * demog * (env.df - e.avg.all)</math> # like type 1 but env and demog roles are swapped</li> <li>• For type=2, <math>ej.indexes = weights * demog * (env.df - e.avg.nond)</math></li> <li>• For type=2.5, <math>ej.indexes = weights * demog * (env.df - e.avg.nond.elsewhere)</math></li> <li>• For type=3, <math>ej.indexes = weights * ((demog * env.df) - (d.avg.all * e.avg.nond))</math></li> <li>• For type=3.5, <math>ej.indexes = weights * ((demog * env.df) - (d.avg.all.elsewhere * e.avg.nond.elsewhere))</math></li> </ul>

- For type 4 ,  $ej.indexes = weights * ((demog - d.avg.all) * (env.df - e.avg.nond))$
- For type=4.5,  $ej.indexes = weights * ((demog - d.avg.all.elsewhere) * (env.df - e.avg.nond.elsewhere))$
- For type=5 ,  $ej.indexes = weights * env.df * demog$  ## A "Population Risk" or "Burden" index = Number of cases among D group if e is individual risk, or just "people-points among D" if e is "points"
- For type=6,  $ej.indexes = env.df * demog$  ## A "percent-based" indicator = percent in group D times envt indicator.

where

- $us.demog$  = overall demog where avg person lives (pop wtd mean of demog). This may be almost exactly the same as  $d.avg.all.elsewhere$
- $d.avg.all$  = overall value for d as fraction of entire population (including the one place being analyzed).
- $d.avg.all.elsewhere$  = overall value for d as fraction of entire population other than the one place being analyzed.
- $e.avg.all$  = avg environmental indicator value for average person
- $e.avg.nond$  = avg environmental indicator value for average person who is not in the D-group, among all (including the one place being analyzed). This is typically the expected as opposed to observed value of e within group D, in the context of EJ analysis of disparity in e.
- $e.avg.nond.elsewhere$  = avg environmental indicator value for average person who is not in the D-group, among all except in the one place being analyzed.

## Details

Creates one EJ index column for each environmental indicator column, or if given a vector or single column of environmental indicators instead of data.frame, returns a vector or column. Each "place" can be a Census unit such as a State, County, zip code, tract, block group, block, for example (or even by individual if person-level data are available).

Note: For 1.5, 3.5, 4.5 need vector  $d.avg.all.elsewhere$ . calculated in each of those. \*\*\* But does not properly handle cases where  $us.demog$  or  $universe.us.demog$  specified because denominator is not same as pop \*\*\* and need to fix for when  $is.na(p)$  |  $is.na(demog)$  \*\*\*\*

## Value

Returns a numeric data.frame (or matrix if  $as.df=FALSE$ ) of EJ indexes, one per place per environmental indicator.

## Examples

```
statedat <- data.frame(state.x77)
hist(myej <- ej.indexes(env.df=statedat[, c('Life.Exp', 'Frost')], demog=statedat$HS.Grad/100, weights=sta
set.seed(999)
myej <- ej.indexes(env.df=rnorm(1000, 10, 3), demog=runif(1000, 0, 1), weights=runif(1000, 500, 5000))
myej
```

---

ejanalysis

*Tools for Environmental Justice (EJ) Analysis*

---

## Description

This R package provides tools for environmental justice (EJ) analysis, including metrics characterizing disparities between demographic groups, such as differences in environmental or other indicators measured for each Census unit such as Census blocks, block groups, tracts, zip codes, or counties.

These tools simplify some basic tasks in exploring and analyzing a dataset in a matrix or data.frame that contains data on demographics (e.g., counts of residents in poverty) and local environmental indicators (e.g., an air quality index), with one row per spatial location (e.g., Census block group). Key functions help to find relative risk or similar ratios of means in demographic groups, , etc.

## Details

Key functions include

- [RR](#)
- [pop.ecdf](#)
- [ej.indexes](#)
- [assign.pctiles](#)
- [make.bin.pctile.cols](#)
- [get.county.info](#)
- [state.health.url](#)

## References

<http://ejanalysis.github.io>  
<http://www.ejanalysis.com/>

ejRRadded

*Formulas for local contributions to EJ metrics***Description**

Formulas for pop counts, pop demog pct, risk, cases, RR, excess risk, excess cases (like EJ index sum), and how those are different under 4 alt scenarios like E of D is set to that of dref.

**Usage**

```
ejRRadded(E, D = 1, P = 1, na.rm = TRUE, n = 5, ...)
```

**Arguments**

E	Environmental indicator (e.g., risk, exposure, or any local indicator, but if it is individual risk then number of cases can be calculated rather than just people-points)
D	Demographic group per place as fraction of P (default is 1)
P	Population count per place (default is 1)

flagged

*Which rows have a value above cutoff in at least one column***Description**

Flags rows that have values above some cutoff in at least one column.

**Usage**

```
flagged(df, cutoff = 0.8, or.tied = TRUE)
```

**Arguments**

df	Data.frame with numeric values to be checked against the threshold.
cutoff	Number that is the threshold that must be (met or) exceeded for a row to be flagged. Optional, default is 0.80
or.tied	Logical, optional, default is TRUE, in which case a value equal to the cutoff also flags the row.

**Details**

The use of na.rm=TRUE in this function means it will always ignore NA values in a given place and take the max of the valid (non-NA) values instead of returning NA when there is an NA in that row

**Value**

Returns a logical vector or data.frame the shape of df

## Examples

```
set.seed(999)
places <- data.frame(p1=runif(10, 0,1), p2=c(NA, runif(9,0,1)), p3=runif(10,0,1))
pctilecols <- c('p1','p2', 'p3')
x <- flagged(places[, pctilecols], 0.80)
a <- cbind(any.over.0.8=x, round(places,2))
a[order(a[,1]),]
```

---

flagged.by	<i>flagged.by</i>
------------	-------------------

---

## Description

Flag which cells are at or above some cutoff.

## Usage

```
flagged.by(...)
```

## Arguments

x	Data.frame or matrix of numbers to be compared to cutoff value.
cutoff	The numeric threshold or cutoff to which numbers are compared. Default is arithmetic mean of row. Usually one number, but can be a vector of same length as number of rows, in which case each row can use a different cutoff.
or.tied	Logical. Default is FALSE, which means we check if number in x is greater than the cutoff (>). If TRUE, check if greater than or equal (>=).
below	Logical. Default is FALSE. If TRUE, uses > or >= cutoff. If FALSE, uses < or <= cutoff.

## Details

For a matrix with a few cols of related data, find which cells are at/above (or below) some cutoff. Returns a logical matrix, with TRUE for each cell that is at/above the cutoff. Can be used in EJ analysis as 1st step in identifying places (rows) where some indicator(s) is/are at/above a cutoff, threshold value.

## Value

Returns a logical matrix the same size as x.

## Note

Future work: these functions could have wts, na.rm, & allow cutoffs or benchmarks as a vector (not just 1 number), & have benchnames.

**See Also**

cols.above.which, another name for the exact same function.

cols.above.count or cols.above.pct to see, for each row, count or fraction of columns with numbers at/above/below cutoff.

flagged.only.by to find cells that are the only one in the row that is at/above/below the cutoff.

rows.above.count, rows.above.pct, rows.above.which

**Examples**

```
out <- flagged.by(x<-data.frame(a=1:10, b=rep(7,10), c=7:16), cutoff=7)
x; out # default is or.tied=FALSE
out <- flagged.by(data.frame(a=1:10, b=rep(7,10), c=7:16), cutoff=7, or.tied=TRUE, below=TRUE)
out
out <- flagged.by(data.frame(a=1:10, b=rep(7,10), c=7:16) ) # Compares each number in each row to the row's m
out
```

---

flagged.only.by	<i>flagged.only.by</i>
-----------------	------------------------

---

**Description**

Flag which cells are the only one in the row that is at/above a cutoff (find rows that meet only 1 of several criteria).

**Usage**

```
flagged.only.by(x, cutoff = 8, or.tied = FALSE, below = FALSE)
```

**Arguments**

x	Data.frame or matrix of numbers to be compared to cutoff value.
cutoff	The numeric threshold or cutoff to which numbers are compared. Default is 8! Usually one number, but can be a vector of same length as number of rows, in which case each row can use a different cutoff.
or.tied	Logical. Default is FALSE, which means we check if number in x is greater than the cutoff (>). If TRUE, check if greater than or equal (>=).
below	Logical. Default is FALSE. If TRUE, uses > or >= cutoff. If FALSE, uses < or <= cutoff.

**Details**

For a data.frame with a few cols of related data, find which cells are the only one in the row that is at/above some cutoff. This can find rows that meet only 1 of several criteria, for example. Returns a logical matrix or data.frame, with TRUE for each cell that meets the test. Can be used in EJ analysis in identifying places (rows) that were only flagged because one of the indicator(s) is at/above a cutoff, threshold value. For example, if there were four criteria to be met in flagging a location, this function identifies places that met only one of the criteria, and can show which one was met.

**Value**

Returns a logical matrix the same size as x.



**Note**

Future work: these functions could have wts, na.rm, & allow cutoffs or benchmarks as a vector (not just 1 number), & have benchnames.

**See Also**

flagged.by or cols.above.which to see which cells are at/above/below some cutoff  
 cols.above.count to see, for each row, how many columns are at/above some cutoff  
 cols.above.percent to see, for each row, what fraction of columns are at/above some cutoff

**Examples**

```
out <- flagged.only.by(x<-data.frame(a=1:10, b=rep(7,10), c=7:16), cutoff=7)
x; out # default is or.tied=FALSE
out <- flagged.only.by(data.frame(a=1:10, b=rep(7,10), c=7:16), cutoff=7, or.tied=TRUE, below=TRUE)
out
out <- flagged.only.by(data.frame(a=1:10, b=rep(7,10), c=7:16) ) # Compares each number in each row to the de
out
```

geofips

*Convert between FIPS / ANSI codes and Names of U.S. Geographies***Description**

Tries to interpret vector of one or more FIPS codes and/or names of geographies, and convert between them.

**Usage**

```
geofips(x, to)
```

**Arguments**

- |    |   |
|----|---|
| x  | Required vector of one or more numeric or character FIPS and/or names of geographic locations. Allowed types are State, County (or equivalent), tract, block group, and block. Names for tracts, blockgroups, and blocks are not provided or interpreted. FIPS codes here are all the relevant digits starting with the 2-character state FIPS, so county fips must be 4-5 digits or characters for example (leading zeroes are inferred where possible and included in outputs). See <a href="#">clean.fips</a> for details.   |
| to | Optional. When the to parameter is not specified, then x that appear to be fips are converted to name, and x that appear to be names are converted to fips. Can specify vector of types of geographies (state, etc.) to convert fips to. This is useful to find County or State names for a list of tract fips, for example. The to parameter can be fips (which converts names to any appropriate type of fips), name (which converts to appropriate type of name based on fips length), specific type of name including state, county, tract, bg (or blockgroup), or block, or specific type of fips, including fips.st, fips.county, fips.tract, fips.bg, fips.block. Names for tracts, blockgroups, and blocks are not provided or interpreted. Variations also work such as plural and case-insensitive. |

**Value**

Returns vector of same length as `x`, containing fips as character elements with any leading zeroes, and/or names of geographies.

**See Also**

[get.fips.st](#) and related functions noted there, [clean.fips](#), [get.state.info](#)

**Examples**

```
geofips(c('NY', 'Alabama', 1, 14, 'Montgomery County, Maryland', '01121', 1121, '060690006002', 60690006002,
  geofips(c('01121', 'Montgomery County, Maryland'), c('state', 'fips.st')
```

---

geotype	<i>Infer Type of Data as FIPS / ANSI Codes or Names of U.S. Geographies</i>
---------	---

---

**Description**

Tries to interpret vector of one or more FIPS codes and/or names of geographies.

**Usage**

```
geotype(x)
```

**Arguments**

<code>x</code>	Required vector of one or more numeric or character FIPS and/or names of geographic locations. Allowed types are State, County (or equivalent), tract, block group, and block. Names for tracts, blockgroups, and blocks are not provided or interpreted. FIPS codes here are all the relevant digits starting with the 2-character state FIPS, so county fips must be 4-5 digits or characters for example (leading zeroes are inferred where possible and included in outputs). See <a href="#">clean.fips</a> for details.
----------------	---

**Value**

\*\*\*TBD \*\*\* Returns \*\*\* types as character strings, and maybe cleaned values themselves? \*\*\*

**See Also**

[geofips](#), [get.fips.st](#) and related functions noted there, [clean.fips](#), [get.state.info](#)

**Examples**

```
# none yet
```

---

get.county.info	<i>Get info on US Counties</i>
-----------------	--------------------------------

---

## Description

Function that reports some or all of a table of data about queried (or all) US Counties (and county equivalents) for 1 or more counties. Query terms can be 5-digit FIPS, or 'countyname, statename' so 'Montgomery, MD' will not work. 'Montgomery County, Maryland' will work. Requested fields can include any of these: "ST", "countyname", "FIPS.COUNTY", "statename", "fullname"

## Usage

```
get.county.info(query, fields = "all", download = FALSE)
```

## Arguments

query	Vector of search terms. Can be county's 5-digit FIPS code(s) (as numbers or strings with numbers), and also could be 'countyname, statename' (fullname, exactly matching formats in countiesall\$fullname, but case insensitive).
fields	Character string optional defaults to 'all' but can specify 'countyname' 'ST' and/or 'FIPS.COUNTY'

## Details

Converted basic data to data, so now can also say data(counties, package='proxistat') or x <- countiesall via lazy loading.

Also, as of 3/2015, a list is here: [http://www2.census.gov/geo/docs/reference/codes/files/national\\_county.txt](http://www2.census.gov/geo/docs/reference/codes/files/national_county.txt)

```
help(county.names, package='choroplethr')
```

```
data(county.names)
```

compare that function to this one:

```
> length(county.names[,1])
```

```
[1] 3142
```

```
> length(get.county.info()[,1])
```

```
[1] 3235
```

```
> head(county.names)
```

```
county.name county.fips.character county.fips.numeric state.name state.abb state.fips.character state.fips.numeric
```

```
1 autauga 01001 1001 alabama AL 01 1
```

```
2 blount 01009 1009 alabama AL 01 1
```

```
3 monroe 01099 1099 alabama AL 01 1
```

```
4 washington 01129 1129 alabama AL 01 1
```

```
5 marshall 01095 1095 alabama AL 01 1
```

```
6 mobile 01097 1097 alabama AL 01 1
```

```
county.names[county.names$county.fips.character=='02185',]
```

```
county.name county.fips.character county.fips.numeric state.name state.abb state.fips.character state.fips.numeric
```

```
88 north slope 02185 2185 alaska AK 02 2
```

```
> head(get.county.info())
```

```
ST countyname FIPS.COUNTY statename fullname
```

```
1 AL Autauga County 01001 Alabama Autauga County, Alabama
```

```
2 AL Baldwin County 01003 Alabama Baldwin County, Alabama
```

```

3 AL Barbour County 01005 Alabama Barbour County, Alabama
4 AL Bibb County 01007 Alabama Bibb County, Alabama
5 AL Blount County 01009 Alabama Blount County, Alabama
6 AL Bullock County 01011 Alabama Bullock County, Alabama
get.county.info()[get.county.info()$FIPS.COUNTY=='02185',]
ST countyname FIPS.COUNTY statename fullname
85 AK North Slope Borough 02185 Alaska North Slope Borough, Alaska

```

```

State,State ANSI,County ANSI,County Name,ANSI CI
AL,01,001,Autauga County,H1
AL,01,003,Baldwin County,H1

```

Also see:

- [http://www2.census.gov/geo/docs/reference/codes/files/national\\_county.txt](http://www2.census.gov/geo/docs/reference/codes/files/national_county.txt)
- <http://www.census.gov/geo/reference/ansi.html>
- [http://www.census.gov/geo/reference/codes/files/national\\_county.txt](http://www.census.gov/geo/reference/codes/files/national_county.txt)
- <http://www.census.gov/geo/reference/docs/state.txt> for just state info
- State: [http://www.census.gov/geo/reference/ansi\\_statetables.html](http://www.census.gov/geo/reference/ansi_statetables.html)
- County: <http://www.census.gov/geo/www/codes/county/download.html>
- Note on definitions of is.usa, is.contiguous.us, etc.:
- [https://www.census.gov/geo/reference/gtc/gtc\\_usa.html](https://www.census.gov/geo/reference/gtc/gtc_usa.html)
- [https://www.census.gov/geo/reference/gtc/gtc\\_codes.html](https://www.census.gov/geo/reference/gtc/gtc_codes.html)
- [https://www.census.gov/geo/reference/gtc/gtc\\_island.html](https://www.census.gov/geo/reference/gtc/gtc_island.html)

Note this other possible list of abbreviations (not used) lacks US, PR, DC:

```
require(datasets); state.abb
```

Note another possible list of States, abbrev, FIPS which has island areas but not US total and not leading zeroes on FIPS:

```
require(acs)
```

```
print(fips.state)
```

## Value

Returns a data.frame or vector of results depending on fields selected. Returns a data.frame (if query has 2+ elements), 'QUERY' as first column, and then all or specified fields of information, covering matching counties, or NA if certain problems arise. If no query term, or fields not specified, then all information fields are returned: QUERY, ST, countyname, FIPS.COUNTY, statename, fullname

## See Also

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

## Examples

```

x <- get.county.info(); str(x); print(''); head(x)
get.county.info(c('05001','01005'), fields=c('countyname', 'ST'))

```

---

get.epa.region	<i>Identify EPA Region for given state</i>
----------------	--

---

## Description

Identify US Environmental Protection Agency region(s) containing given state(s)

## Usage

```
get.epa.region(state)
```

## Arguments

state	Vector of numeric or character class, required. Can be state FIPs as number or character, 2-character state abbreviation, or full state name
-------	--

## Details

For EPA Regions, see <http://www2.epa.gov/aboutepa#pane-4> or <http://www2.epa.gov/aboutepa/visiting-regional-office>. For information on FIPS codes, see <http://www.census.gov/geo/reference/ansi.html>, and also see <https://www.census.gov/geo/reference/geoidentifiers.html>

## Value

Returns a vector of numbers, same length as state

## See Also

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countynames, or region based on FIPS or name.

## Examples

```
myregions <- get.epa.region() # to see full list
myregions <- get.epa.region('DC') # for one state by 2-letter abbreviation
myregions <- get.epa.region(c('AK', 'NY', 'PR')) # for a vector of 2-letter abbreviations
myregions <- get.epa.region(c('alaska', 'new york', 'puerto rico')) # for a vector of state names
myregions <- get.epa.region(c('04', '36', '72')) # for a vector of state FIPS codes as characters with leading zeroes
myregions <- get.epa.region(c(4,36,72)) # for a vector of state FIPS codes as numeric with no leading zeroes
myregions <- get.epa.region(c('NY', 'Ohio')) # CANNOT MIX WAYS OF DEFINING STATES - PICK ONE FORMAT
```

---

get.fips.bg

*Get Census block group FIPS from block or block group FIPS*


---

### Description

Extract partial FIPS code from longer FIPS code

### Usage

```
get.fips.bg(fips)
```

### Arguments

fips                      Vector of one or more elements, character class, 12 or 15 characters long FIPS (block group or block), required.

### Details

For information on FIPS codes, see <http://www.census.gov/geo/reference/ansi.html>, and also see <https://www.census.gov/geo/reference/geoidentifiers.html>. It does not check to see if the codes are valid other than counting how many characters each has.

### Value

Returns a vector of one or more character elements, same lengths as fips

### See Also

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

### Examples

```
samplefips <- c("011030001003", "011030001003001", 11030001003001, 35, 1, NA, 'invalidtext', '02')
get.fips.bg(samplefips)
```

---

get.fips.county

*Get County FIPS from block or block group FIPS*


---

### Description

Extract partial FIPS code from longer FIPS code

### Usage

```
get.fips.county(fips)
```

**Arguments**

**fips** Vector of one or more elements, character class, 5, 11, 12 or 15 characters long after missing leading zeroes inferred (county, tract, block group or block), required.

**Details**

Each fips passed to the function is a FIPS code (see <http://www.census.gov/geo/reference/ansi.html>.) It does not check to see if the codes are valid other than counting how many characters each has.

**Value**

Returns a vector of one or more character elements, same lengths as fips NA where input cannot be converted to county/tract/block group/block length code, as with state fips as input.

**See Also**

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

**Examples**

```
samplefips <- c("011030001003", "011030001003001", 02610, 11030001003001, 35, 1, NA, 'invalidtext', '02')
get.fips.county(samplefips)
```

---

get.fips.etc

*Get tract, county, state, region info from US Census block group FIPS codes*


---

**Description**

Use US Census block group FIPS codes to get more information about each block group's overall location, such as State and County name. It does not check to see if the codes are valid other than counting how many characters each has.

**Usage**

```
get.fips.etc(fips)
```

**Arguments**

**fips** Vector of US Census block group FIPS codes (missing leading zeroes are added).

**Value**

Returns a data.frame with these fields: c('FIPS', 'FIPS.TRACT', 'FIPS.COUNTY', 'FIPS.ST', 'ST', 'countyname', 'statename', 'REGION') where FIPS is the input fips, the next few are the first few characters of fips corresponding to tract, county, or state, ST is the 2-letter state abbreviation, statename is state name, countyname is county name, and REGION is USEPA Region 1-10.

**Examples**

```
x=c("391670211002", "060730185143", "261079609003", 02, 02610, "400353734002", "371190030121", "250235022001")
get.fips.etc(x)
```

---

get.fips.st

*Get State FIPS from block or block group FIPS*


---

**Description**

Extract partial FIPS code from longer FIPS code

**Usage**

```
get.fips.st(fips)
```

**Arguments**

fips                      Vector of one or more elements, character class, 12 or 15 characters long (block group or block), required.

**Details**

For information on FIPS codes, see <http://www.census.gov/geo/reference/ansi.html>, and also see <https://www.census.gov/geo/reference/geoidentifiers.html>. It does not check to see if the codes are valid other than counting how many characters each has.

**Value**

Returns a vector of one or more character elements, same lengths as fips

**See Also**

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

**Examples**

```
samplefips <- c("011030001003", "011030001003001", 02610, 11030001003001, 35, 1, NA, 'invalidtext', '02')
get.fips.st(samplefips)
```



---

get.fips.tract	<i>Get Census tract FIPS from block or block group FIPS</i>
----------------	---

---

**Description**

Extract partial FIPS code from longer FIPS code

**Usage**

```
get.fips.tract(fips)
```

**Arguments**

fips	Vector of one or more elements, character class, 11, 12 or 15 characters long after missing leading zeroes inferred (tract, block group or block), required.
------	--

**Details**

For information on FIPS codes, see <http://www.census.gov/geo/reference/ansi.html>, and also see <https://www.census.gov/geo/reference/geoidentifiers.html>. It does not check to see if the codes are valid other than counting how many characters each has.

**Value**

Returns a vector of one or more character elements, same lengths as fips. NA where input cannot be converted to tract/block group/block length code, as with county or state fips as input.

**See Also**

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

**Examples**

```
samplefips <- c("011030001003", "011030001003001", 02610, 11030001003001, 35, 1, NA, 'invalidtext', '02')
get.fips.tract(samplefips)
```

---

get.name.county	<i>Extract county name from Census geo name field</i>
-----------------	---

---

**Description**

Parse text names of places from Census Bureau datasets like American Community Survey 5-year Summary File. Extracts partial (e.g., just state or county name) text name of a place from the full Census text name of a place.

**Usage**

```
get.name.county(placename)
```

## Arguments

placename      character vector, required, with text names of places from Census Bureau, such as 'Block Group 1, Census Tract 1, Aleutians East Borough, Alaska'

## Details

Inputs can be tracts or block groups as in the relevant ACS summary files, possibly others at some point.

For tracts, there are only 3 parts to placename (tract, county, state)

For block groups, there are 4 parts to placename (block group, tract, county, state)

Note that County names are not unique – the same County name may exist in 2+ States.

also see <http://www.census.gov/geo/reference/ansi.html>

See [http://www.census.gov/geo/reference/codes/files/national\\_county.txt](http://www.census.gov/geo/reference/codes/files/national_county.txt) but file has moved Note old code was in GET COUNTY NAMES FROM NHGIS DATASET.R

## Value

character vector of names

## See Also

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

## Examples

```
# Test data where some are block groups and some are tracts, as in file downloaded from Census FTP site for ACS
```

```
testnames <- c(
  'Block Group 1, Census Tract 1, Aleutians East Borough, Alaska',
  'Block Group 2, Census Tract 1, Aleutians East Borough, Alaska',
  'Block Group 1, Census Tract 2, Aleutians West Census Area, Alaska',
  'Block Group 2, Census Tract 2, Aleutians West Census Area, Alaska',
  'Block Group 1, Census Tract 2.01, Anchorage Municipality, Alaska',
  'Census Tract 1, Aleutians East Borough, Alaska',
  'Census Tract 2, Aleutians West Census Area, Alaska',
  'Census Tract 2.01, Anchorage Municipality, Alaska')
testnames <- rep(testnames, floor(280000/8))
```

```
mynames1 <- get.name.state(testnames)
head(mynames1, 20)
mynames2 <- get.name.county(testnames)
head(mynames2, 20)
```

---

get.name.state

*Extract state name from Census geo name field*

---

## Description

Parse text names of places from Census Bureau datasets like American Community Survey 5-year Summary File. Extracts partial (e.g., just state or county name) text name of a place from the full Census text name of a place.

**Usage**

```
get.name.state(placename)
```

**Arguments**

placename	character vector, required, with text names of places from Census Bureau, such as 'Block Group 1, Census Tract 1, Aleutians East Borough, Alaska'
-----------	---

**Details**

Inputs can be tracts or block groups as in the relevant ACS summary files, possibly others at some point.

For tracts, there are only 3 parts to placename (tract, county, state)

For block groups, there are 4 parts to placename (block group, tract, county, state)

Note that County names are not unique – the same County name may exist in 2+ States.

also see <http://www.census.gov/geo/reference/ansi.html>

See [http://www.census.gov/geo/reference/codes/files/national\\_county.txt](http://www.census.gov/geo/reference/codes/files/national_county.txt) but file has moved Note old code was in GET COUNTY NAMES FROM NHGIS DATASET.R

**Value**

character vector of names

**See Also**

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

**Examples**

```
# Test data where some are block groups and some are tracts, as in file downloaded from Census FTP site for ACS

testnames <- c(
  'Block Group 1, Census Tract 1, Aleutians East Borough, Alaska',
  'Block Group 2, Census Tract 1, Aleutians East Borough, Alaska',
  'Block Group 1, Census Tract 2, Aleutians West Census Area, Alaska',
  'Block Group 2, Census Tract 2, Aleutians West Census Area, Alaska',
  'Block Group 1, Census Tract 2.01, Anchorage Municipality, Alaska',
  'Census Tract 1, Aleutians East Borough, Alaska',
  'Census Tract 2, Aleutians West Census Area, Alaska',
  'Census Tract 2.01, Anchorage Municipality, Alaska')
testnames <- rep(testnames, floor(280000/8))

mynames1 <- get.name.state(testnames)
head(mynames1, 20)
mynames2 <- get.name.county(testnames)
head(mynames2, 20)
```

get.pctile

*Determine (Weighted) Percentiles***Description**

Given a vector of numbers (a dataset), and one or more specific values of interest, determine at what percentile(s) are those selected values. In other words, what fraction of all of the numbers (actually weighted based on cumulative sum of weights) are smaller than (or tied with) the selected value(s) of interest? Based on `analyze.stuff::pct.above` ([pct.above](#))

**Usage**

```
get.pctile(x, values, wts = 1, or.tied = TRUE, na.rm = TRUE)
```

**Arguments**

<code>x</code>	Vector of numeric values (the distribution within which percentiles are sought)
<code>values</code>	Required vector of one or more numbers for which percentiles are sought.
<code>wts</code>	Optional weights, default is 1 (equal weighting). Useful to find for example what percent of people have an <code>x</code> score (at or) below a given value.
<code>or.tied</code>	Default is TRUE which means percentile represents what fraction have <code>x</code> at or below value, not just below value.
<code>na.rm</code>	Logical, optional, TRUE by default. Should NA values (missing data) be removed first to get percentile of those with valid (nonmissing) data.

**See Also**

[make.bin.pctile.cols](#) to call functions below, converting columns of values to percentiles and then bins  
[assign.pctiles](#) for one vector, assign (weighted) percentile (quantile) to each value within its zone (subset)  
[assign.pctiles.alt2](#) as an alternative method, to replicate `assign.pctiles`, but not by zone  
[get.pctile](#) to get (weighted) percentile of just 1+ values within given vector of values  
[make.pctile.cols](#) for a data.frame, assign percentiles, return a same-sized df that is `wtd.quantile` of each value within its column  
[make.pctile.cols.alt2](#) as an alternative method, to replicate `make.pctile.cols`  
[assign.map.bins](#) for one vector (or data.frame) of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[make.bin.cols](#) for a data.frame of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[write.pctiles](#) to save file that is lookup table of percentiles for columns of a data.frame  
[write.pctiles.by.zone](#) to save file that is lookup table of percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[write.wtd.pctiles](#) to save file that is lookup table of weighted percentiles for columns of a data.frame  
[write.wtd.pctiles.by.zone](#) to save file that is lookup table of weighted percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[lookup.pctile](#) to look up current approx weighted percentiles in a lookup table that is already in global memory

## Examples

```
get.pctile(89:95, 1:100)
get.pctile(89:95, 1:100, c(rep(1,90), rep(10000,10)))
```

---

get.state.info	<i>Get information on U.S. State(s)</i>
----------------	---

---

## Description

Query information about States, from proxistat package data in data(lookup.states, package='proxistat')

## Usage

```
get.state.info(query, fields = "all")
```

## Arguments

query	vector of 1+ elements, which can be state FIPS code(s) (as numbers or strings with numbers), state name(s) (exactly matching formats here), or 2-letter state abbreviation(s) (case insensitive).
fields	vector of 1+ character string names of the fields available here: FIPS.ST, ST, statename, ftpname, REGION, is.usa.plus.pr, is.usa, is.state, is.contiguous.us, is.island.areas, and others (see below)

## Details

See **proxistat** package for data source (<http://ejanalysis.github.io/proxistat/>) For 1+ or all US States plus DC, PR, Island Areas (and USA overall for use in FTP URL):

EPA Region, FIPS, State name, abbreviation for State(s); based on any of these query methods:

State's FIPS, State's name, OR State's abbreviation, (i.e., FIPS.ST, statename, or ST).

Also see data in packages **acs** and **choroplethr**

Also see <http://www.census.gov/geo/reference/docs/state.txt> and <http://www.census.gov/geo/reference/ansi.html>

# Note on definitions of is.usa, is.contiguous.us, etc.:

[https://www.census.gov/geo/reference/gtc/gtc\\_usa.html](https://www.census.gov/geo/reference/gtc/gtc_usa.html)

[https://www.census.gov/geo/reference/gtc/gtc\\_codes.html](https://www.census.gov/geo/reference/gtc/gtc_codes.html)

[https://www.census.gov/geo/reference/gtc/gtc\\_island.html](https://www.census.gov/geo/reference/gtc/gtc_island.html)

[http://en.wikipedia.org/wiki/Contiguous\\_United\\_States](http://en.wikipedia.org/wiki/Contiguous_United_States)

Also note this other possible list of abbreviations (not used) lacks US, PR, DC:  
require(datasets); state.abb

Note another possible list of States, abbrev, FIPS  
which has island areas but not US total and not leading zeroes on FIPS:  
require(acs)

```
print(fips.state)
```

Note FIPS were also available here:

State: [http://www.census.gov/geo/reference/ansi\\_statetables.html](http://www.census.gov/geo/reference/ansi_statetables.html)

County: <http://www.census.gov/geo/www/codes/county/download.html>

Also see <https://www.census.gov/geo/reference/state-area.html> for info on state area and internal point

## Value

A data.frame (if query has 2+ elements), providing all or specified fields of information, covering matching states/dc/pr/island areas, a vector of the same type of information for just one place (if only 1 query term, i.e., one element in the query vector is provided), or NA if certain problems arise.

If no query term, or fields not specified, then all information fields are returned:

```
get.state.info()[1:2, ]
```

```
statename FIPS.ST ST ftnname REGION is.usa.plus.pr is.usa is.state is.contiguous.us
```

```
1 Alabama 01 AL Alabama 4 TRUE TRUE TRUE TRUE
```

```
2 Alaska 02 AK Alaska 10 TRUE TRUE TRUE FALSE
```

```
is.island.areas area.sqmi area.sqkm landarea.sqmi landarea.sqkm waterarea.sqmi waterarea.sqkm
```

```
1 FALSE 52420 135767 50645 131171 1775 4597
```

```
2 FALSE 665384 1723337 570641 1477953 94743 245383
```

```
inland.sqmi inland.sqkm coastal.sqmi coastal.sqkm greatlakes.sqmi greatlakes.sqkm
```

```
1 1058 2740 517 1340 0 0
```

```
2 19304 49997 26119 67647 0 0
```

```
territorial.sqmi territorial.sqkm lat lon
```

```
1 199 516 32.73963 -86.84346
```

```
2 49320 127739 63.34619 -152.83707
```

## See Also

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

## Examples

```
# data(lookup.states, package='proxistat')
# x <- get.state.info(); str(x); cat('\n'); x[ 1:2, ]
# get.state.info(c('alaska', 'north carolina', 'montana', "hawaii"), fields=c('ST', 'statename', 'REGION'))
# get.state.info('DC'); get.state.info('U.S. Virgin Islands'); get.state.info(4)
# get.state.info(c('New york', 'alaska', 'North Carolina', 'MONTANA', 'typo'))
# get.state.info(c('ny', 'DC', 'AK', 'mt', 'PR'))
```

```
# get.state.info( c(36, 36, 'ny', ' ny', 'ny ', 'California', 'DC','AK','mt', 'PR', '02', 2, 'North carolina
# get.state.info(1:80)
```

KolmPollak

*Kolm-Pollak Inequality Index adjustable for indexing inequality in bad outcomes*

## Description

This is a simple wrapper for the `ineq::Kolm` function, but it makes explicit the need to transform `x` first if `x` measures a harm or cost.

## Usage

```
KolmPollak(x, parameter = 1, na.rm = TRUE, bigbadx = FALSE)
```

## Arguments

<code>x</code>	Same as in <a href="#">Kolm</a> .
<code>parameter</code>	Same as in <a href="#">Kolm</a> .
<code>na.rm</code>	Same as in <a href="#">Kolm</a> .
<code>bigbadx</code>	Optional, default is FALSE. Setting <code>bigbadx = TRUE</code> means bigger <code>x</code> values are bad, as in the case of <code>x</code> measuring levels of risk to health, or exposure to a pollutant. If <code>bigbadx=TRUE</code> , the <code>x</code> values are first transformed into $0 - x$

## Details

The Kolm-Pollak inequality index is provided by the package **ineq**, but that index is only suitable for use with a variable where larger values are preferred, such as with income. The need for this type of transformation is noted by Maguire and Sheriff (2011) in <http://dx.doi.org/10.3390/ijerph8051707> Many variables represent negative outcomes, such as measures of risks to health, or costs. For those types of variables, the variable should first be transformed by multiplying it by negative one. This function does that when a parameter is set appropriately.

The typical KP index says inequality is **worse if some incomes are very low** (than if some are very high). The adjusted KP index says inequality is **worse if some costs or health risks are very high** (than if some are very low).

## Value

This returns what Kolm does, except if `bigbadx=TRUE` it first transforms `x`.

## Examples

```
# Typical KP index says
#   inequality is worse
#   if some incomes are very low
#   (than if some are very high):
KolmPollak(c(-5,0,1), bigbadx = FALSE) #[1] 2.577229
KolmPollak(c(-1,0,5), bigbadx = FALSE) #[1] 1.549793

# Adjusted KP index says
```

```
# inequality is worse
# if some costs or health risks are very high
# (than if some are very low):
KolmPollak(c(-1,0,5), bigbadx = TRUE) #[1] 2.577229
KolmPollak(c(-5,0,1), bigbadx = TRUE) #[1] 1.549793

set.seed(99)
x=rep(c(1,2,5),50)*rnorm(mean=1, sd=0.2, n=50*3)
ineq::Kolm(x)
KolmPollak(x)
KolmPollak(x, bigbadx = TRUE)
```

lookup.pctile

*Find approx wtd percentiles in lookup table that is in memory***Description**

This is used with `lookup` or `us`, a data.frame that is a lookup table created by `write.pctiles`, `write.pctiles.by.zone`, `write.wtd.pctiles`, or `write.wtd.pctiles.by.zone`. The data.frame `us` or `lookup` must have a field called "PCTILE" that contains quantiles/percentiles and other column(s) with values that fall at those percentiles. This function accepts lookup table (or uses one called `us` if that is in memory), and finds the number in the PCTILE column that corresponds to where a specified value (in `myvector`) appears in the column called `varname.in.lookup.table`. The function just looks for where the specified value fits between values in the lookup table and returns the approximate percentile as found in the PCTILE column.

**Usage**

```
lookup.pctile(myvector, varname.in.lookup.table, lookup, zone)
```

**Arguments**

<code>myvector</code>	Numeric vector, required. Values to look for in the lookup table.
<code>varname.in.lookup.table</code>	Character element, required. Name of column in lookup table to look in to find interval where a given element of <code>myvector</code> values is.
<code>lookup</code>	Either <code>lookup</code> must be specified, or a lookup table called <code>us</code> must already be in memory. This is the lookup table data.frame with a PCTILE column and column whose name is the value of <code>varname.in.lookup.table</code> .
<code>zone</code>	Character element, optional. If specified, must appear in a column called REGION within the lookup table. For example, it could be 'NY' for New York State.

**Value**

By default, returns numeric vector length of `myvector`.

**See Also**

`write.pctiles`, `write.pctiles.by.zone`, `write.wtd.pctiles`, and `write.wtd.pctiles.by.zone`



**Examples**

```
# places.etc$new.pctile.pm <- lookup.pctile(places.etc$countymean, "PM25")
```

---

make.bin.cols

---

*Create a Bin Numbers Column for each Percentiles Column*


---

**Description**

Simplifies creation of data.frame or matrix (or vector) with columns that specify bins by bin number, based on values (percentiles for example) and specified cutpoints. Each bin is defined by cutpoints.

**Usage**

```
make.bin.cols(pctile.df, as.df = TRUE, cutpoints = c((0:9)/10, 0.95, 1),
  labels = 1:11, prefix = "bin.")
```

**Arguments**

pctile.df	Data.frame (or matrix or vector), required. Typically percentiles as decimal fractions, 0 to 1, one per place.
as.df	Logical value, optional, TRUE by default. Defines whether results are data.frame or matrix.
cutpoints	Optional vector of cutpoints defining edges of bins. Default is every 0.10 from 0 through 1.00, as well as 0.95
labels	Vector of bin numbers (defining what is returned for values in those bins), optional (default is 1 through 11, and NA values are put in bin 0).
prefix	Optional character element, ".bin" by default, pasted as prefix to each column name of pctile.df, and used as names of returned columns.

**Details**

This is one way to prepare data to be mapped in a series of choropleths, for example (color-coded maps).

The default bins 0-11 are defined as follows:

bin 0: PCTILE=NA

...

bin 9: 0.80<=PCTILE<0.90

bin 10: 0.90<=PCTILE<0.95

bin 11: 0.95<=PCTILE<=1.00

**Value**

Returns a data.frame, matrix, or vector (depending on as.df) the same shape as pctile.df

**See Also**

[make.bin.pctile.cols](#) to call functions below, converting columns of values to percentiles and then bins  
[assign.pctiles](#) for one vector, assign (weighted) percentile (quantile) to each value within its zone (subset)  
[assign.pctiles.alt2](#) as an alternative method, to replicate [assign.pctiles](#), but not by zone  
[get.pctile](#) to get (weighted) percentile of just 1+ values within given vector of values  
[make.pctile.cols](#) for a data.frame, assign percentiles, return a same-sized df that is wtd.quantile of each value within its column  
[make.pctile.cols.alt2](#) as an alternative method, to replicate [make.pctile.cols](#)  
[assign.map.bins](#) for one vector (or data.frame) of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[make.bin.cols](#) for a data.frame of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[write.pctiles](#) to save file that is lookup table of percentiles for columns of a data.frame  
[write.pctiles.by.zone](#) to save file that is lookup table of percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[write.wtd.pctiles](#) to save file that is lookup table of weighted percentiles for columns of a data.frame  
[write.wtd.pctiles.by.zone](#) to save file that is lookup table of weighted percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[lookup.pctile](#) to look up current approx weighted percentiles in a lookup table that is already in global memory

**Examples**

```
# new.bin.cols <- make.bin.cols(places[ , names.e.pctile])
# new.bin.cols <- make.bin.cols( new.pctile.cols )
```

---

make.bin.pctile.cols    *Weighted Percentiles and Bin Numbers for Each Column*

---

**Description**

This function just combines [make.pctile.cols](#) and [make.bin.cols](#). Takes a data.frame of values and returns a data.frame (or matrix) of percentiles, showing the percentile of a value within all values in its column, as well as bin numbers, showing what bin each falls into, based on specified cutoffs defining bins.

**\*\* Work in progress/ not fully tested, e.g., need to test if all code below works with both as.df=TRUE and as.df=FALSE**

**Usage**

```
make.bin.pctile.cols(raw.data.frame, weights = 1, zone = NULL,
  as.df = TRUE, prefix.bin = "bin.", prefix.pctile = "pctile.",
  cutpoints = c((0:9)/10, 0.95, 1), labels = 1:11)
```

**Arguments**

raw.data.frame	Data.frame of values
weights	Optional Numeric vector of weights to create weighted percentiles, such as population-weighted quantiles. Unweighted if not specified. Vector same length as number of rows in data.frame.
zone	NULL by default, but if a vector is provided, it defines zones to group by, so percentiles are within a given zone only.
as.df	Optional logical TRUE by default, in which case matrix results are converted to data.frame
prefix.bin	Optional character element, default is 'bin.', provides text to paste to beginning of input data.frame column names to use as bin output column names.
prefix.pctile	Optional character element, default is 'pctile.', provides text to paste to beginning of input data.frame column names to use as pctile output column names.
cutpoints	Default is 1:11. see <a href="#">make.bin.cols</a>
labels	Default is c(0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 0.95, 1.00). see <a href="#">make.bin.cols</a>

**Value**

Returns a matrix or data.frame

**See Also**

[make.bin.pctile.cols](#) to call functions below, converting columns of values to percentiles and then bins  
[assign.pctiles](#) for one vector, assign (weighted) percentile (quantile) to each value within its zone (subset)  
[assign.pctiles.alt2](#) as an alternative method, to replicate [assign.pctiles](#), but not by zone  
[get.pctile](#) to get (weighted) percentile of just 1+ values within given vector of values  
[make.pctile.cols](#) for a data.frame, assign percentiles, return a same-sized df that is wtd.quantile of each value within its column  
[make.pctile.cols.alt2](#) as an alternative method, to replicate [make.pctile.cols](#)  
[assign.map.bins](#) for one vector (or data.frame) of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[make.bin.cols](#) for a data.frame of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[write.pctiles](#) to save file that is lookup table of percentiles for columns of a data.frame  
[write.pctiles.by.zone](#) to save file that is lookup table of percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[write.wtd.pctiles](#) to save file that is lookup table of weighted percentiles for columns of a data.frame  
[write.wtd.pctiles.by.zone](#) to save file that is lookup table of weighted percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[lookup.pctile](#) to look up current approx weighted percentiles in a lookup table that is already in global memory

---

```
make.bin.pctile.cols.byzone
```

*Percentile and bin number fields, by zone, such as percentiles within each State*

---

### Description

Get weighted percentiles and bin numbers, by subset of rows (zone), for each column of data. This can be used to calculate population-weighted percentiles within each State, for national data on Census block groups, for example.

### Usage

```
make.bin.pctile.cols.byzone(bg, zone, wtsvarname, keyvarname = "FIPS",
  datavarnames = gsub(keyvarname, "", names(bg)), ...)
```

### Arguments

bg	Data.frame of data field(s), weights (optional), and a unique ID (e.g., FIPS), one row per place if geographic data is used.
zone	Optional vector specifying subsets of rows (e.g. State name) within which percentiles are calculated
wtsvarname	Name of field in bg that has optional weights for weighted percentiles. Default is unweighted.
keyvarname	Name of field in bg that has unique ID per row (per place). Default is "FIPS"
datavarnames	Vector of names of fields in bg that have numeric data for which percentiles are calculated.
...	Other parameters passed to <a href="#">make.bin.pctile.cols</a> such as cutpoints and labels

### Value

Returns a data.frame with one percentiles field and one bin numbers field for each of the datavarnames

---

```
make.pctile.cols
```

*Make columns of (weighted) percentiles from columns of values*

---

### Description

Takes values (vector, matrix, or data.frame) and returns percentiles, showing the percentile of a value within all values in its column.

### Usage

```
make.pctile.cols(values, weights = 1, prefix = "pctile.", as.df = TRUE,
  zone = NULL)
```

**Arguments**

values	Data.frame, matrix, or vector of values. One column at a time is analyzed using <a href="#">assign.pctiles</a>
weights	Optional numeric vector of weights (default is unweighted) to create weighted percentiles, such as population-weighted quantiles. Unweighted if not specified. Vector same length as number of rows in values parameter (or elements if vector).
prefix	Optional character element, default is 'pctile.', provides text to paste to beginning of input data.frame column names to use as output column names.
as.df	Optional logical TRUE by default, in which case matrix results are converted to data.frame
zone	NULL by default, but if a vector is provided, it defines zones to group by, so percentiles are within a given zone only.

**Value**

Returns a matrix or data.frame

**See Also**

[make.bin.pctile.cols](#) to call functions below, converting columns of values to percentiles and then bins  
[assign.pctiles](#) for one vector, assign (weighted) percentile (quantile) to each value within its zone (subset)  
[assign.pctiles.alt2](#) as an alternative method, to replicate [assign.pctiles](#), but not by zone  
[get.pctile](#) to get (weighted) percentile of just 1+ values within given vector of values  
[make.pctile.cols](#) for a data.frame, assign percentiles, return a same-sized df that is wtd.quantile of each value within its column  
[make.pctile.cols.alt2](#) as an alternative method, to replicate [make.pctile.cols](#)  
[assign.map.bins](#) for one vector (or data.frame) of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[make.bin.cols](#) for a data.frame of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[write.pctiles](#) to save file that is lookup table of percentiles for columns of a data.frame  
[write.pctiles.by.zone](#) to save file that is lookup table of percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[write.wtd.pctiles](#) to save file that is lookup table of weighted percentiles for columns of a data.frame  
[write.wtd.pctiles.by.zone](#) to save file that is lookup table of weighted percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[lookup.pctile](#) to look up current approx weighted percentiles in a lookup table that is already in global memory

---

`make.pctile.cols.alt2` *Alternative way to make columns of (weighted) percentiles from columns of values*

---

## Description

Takes a data.frame of values and returns a data.frame of percentiles, showing the percentile of a value within all values in its column.

## Usage

```
make.pctile.cols.alt2(raw.data.frame, weights, as.df = TRUE, na.rm = TRUE,
  zone = NULL)
```

## Arguments

raw.data.frame	Data.frame of values
weights	Optional Numeric vector of weights to create weighted percentiles, such as population-weighted quantiles. Unweighted if not specified. Vector same length as number of rows in data.frame.
as.df	Optional logical TRUE by default, in which case matrix results are converted to data.frame
zone	NULL by default, but if a vector is provided, it defines zones to group by, so percentiles are within a given zone only.
prefix	Optional character element, default is 'pctile.', provides text to paste to beginning of input data.frame column names to use as output column names.

## Value

Returns a matrix or data.frame

## See Also

[make.bin.pctile.cols](#) to call functions below, converting columns of values to percentiles and then bins  
[assign.pctiles](#) for one vector, assign (weighted) percentile (quantile) to each value within its zone (subset)  
[assign.pctiles.alt2](#) as an alternative method, to replicate assign.pctiles, but not by zone  
[get.pctile](#) to get (weighted) percentile of just 1+ values within given vector of values  
[make.pctile.cols](#) for a data.frame, assign percentiles, return a same-sized df that is wtd.quantile of each value within its column  
[make.pctile.cols.alt2](#) as an alternative method, to replicate make.pctile.cols  
[assign.map.bins](#) for one vector (or data.frame) of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[make.bin.cols](#) for a data.frame of values (e.g. percentiles), return same-sized df that is bin number (map color bin) using preset breaks.  
[write.pctiles](#) to save file that is lookup table of percentiles for columns of a data.frame  
[write.pctiles.by.zone](#) to save file that is lookup table of percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[write.wtd.pctiles](#) to save file that is lookup table of weighted percentiles for columns of a data.frame  
[write.wtd.pctiles.by.zone](#) to save file that is lookup table of weighted percentiles for columns of a data.frame, for each geographic zone (subset of rows)  
[lookup.pctile](#) to look up current approx weighted percentiles in a lookup table that is already in global memory

---

names.dvars

*Fieldnames of demographic columns in eanalysis data*


---

### Description

This data set provides variables that hold the colnames of demographic fields in data.frames that may be used in the eanalysis package to make it easier to refer to them as a vector, e.g., mydf[, names.e]

### Usage

```
data('names.dvars')
```

### Format

A series of variables (each is a character vector of colnames):

- "names.d" (VSI.eo, VSI.svi6, pctmin, pctlowinc, pctlths, pctlingiso, pctunder5, pctover64)
- "names.d.bin"
- "names.d.eo"
- "names.d.eo.bin"
- "names.d.eo.pctile"
- "names.d.pctile"
- "names.d.subgroups"
- "names.d.subgroups.count"
- "names.d.subgroups.pct"
- "names.d.svi6"
- "names.d.svi6.bin"
- "names.d.svi6.pctile" #'
- "Dlist" (this one is like names.d, but as a list, not a vector)

### Source

Names developed for this package. No external data source.

---

names.ejvars

*Fieldnames of environmental justice indicator columns in eanalysis data*


---

### Description

This data set provides variables that hold the colnames of environmental indicator fields in data.frames that may be used in the eanalysis package to make it easier to refer to them as a vector, e.g., mydf[, names.ej]

**Usage**

```
data('names.ejvars')
```

**Format**

A series of variables (each is a character vector of colnames):

- "names.ej"
- "names.ej.bin"
- "names.ej.burden.eo"
- "names.ej.burden.eo.bin"
- "names.ej.burden.eo.pctile"
- "names.ej.burden.svi6"
- "names.ej.burden.svi6.bin"
- "names.ej.burden.svi6.pctile"
- "names.ej.pct.eo"
- "names.ej.pct.eo.bin"
- "names.ej.pct.eo.pctile"
- "names.ej.pct.svi6"
- "names.ej.pct.svi6.bin"
- "names.ej.pct.svi6.pctile"
- "names.ej.pctile"
- "names.ej.svi6"
- "names.ej.svi6.bin"
- "names.ej.svi6.pctile"
- "namesall.ej"
- "namesall.ej.bin"
- "namesall.ej.pctile"

**Source**

Names developed for this package. No external data source.

---

names.evars

*Fieldnames of environmental indicator columns in ejanalysis data*

---

**Description**

This data set provides variables that hold the colnames of environmental indicator fields in data.frames that may be used in the ejanalysis package to make it easier to refer to them as a vector, e.g., mydf[, names.e]

**Usage**

```
data('names.evars')
```



**Format**

A series of variables (each is a character vector of colnames):

- "names.e" (pm, o3, cancer, neuro, resp, dpm, pctpre1960, traffic.score, proximity.npl, proximity.rmp, proximity.tsdf, proximity.npdes)
- "names.e.bin"
- "names.e.pctile"
- "Elist" (this one is like names.e, but as a list, not a vector)

**Source**

Names developed for this package. No external data source.

---

pct.moe	<i>Margin of Error for a Percent (Ratio)</i>
---------	--

---

**Description**

Estimates the margin of error (MOE) that characterizes uncertainty, for a ratio (a/b), based on a, b, and their MOE values.

**Usage**

```
pct.moe(a, b, a.moe, b.moe)
```

**Arguments**

a	Numerators, as a vector
b	Denominators, as a vector (should be same length as a, or it is recycled)
a.moe	Margins of error for numerators, as a vector (should be same length as a).
b.moe	Margins of error for denominators, as a vector (should be same length as a, or it is recycled).

**Details**

This is based on US Census Bureau recommendations for working with American Community Survey summary file data. See <http://www.census.gov/programs-surveys/acs/guidance.html> and <http://www.census.gov/library/publications/2009/acs/researchers.html> For example, one can estimate MOE for percent poor in a block group, given estimates and margins of error for the count who are poor, and the count for whom poverty status is known.

**Value**

Returns margin(s) of error same shape as parameter a

**See Also**

[sum.moe](#)

**Examples**

```
x <- pct.moe(15, 100, 3, 10)
```

---

pop.cdf	<i>Draw PDF (overlays histograms) comparing distributions of scores in selected demographic groups</i>
---------	--

---

### Description

Draws a histogram plot using [weighted.hist](#), overlaying distribution functions, one for each subgroup specified. Useful to compare 2 groups based on each groups entire pdf distribution of peoples scores, using data from small places like census block groups, based on having for each place the pop total and

### Usage

```
pop.cdf(scores, pcts, pops, allothers = TRUE, col = "red", main, weights,
...)
```

### Arguments

scores	Numeric vector (not data.frame currently), required. Values to analyze.
pcts	Numeric vector or data.frame, required. Same number of vector elements or data.frame rows as length of scores. Specifies the fraction of population that is in demographic group(s) of interest, one row per place, one group per column.
pops	Vector used to define weights as pop*pcts, and if allothers=TRUE, for pop*(1-pcts) for nongroup
allothers	Logical value, optional, TRUE by default. Whether to plot a series for everyone else, using 1-pct
col	Optional, default is 'red' to signify line color red for key demographic group. Can also be a vector of colors if pcts is a data.frame with one column per group, one color per group.
main	Required character specifying plot title
weights	Not used currently (see pop parameter)
...	other optional parameters to pass to <a href="#">weighted.hist()</a>

### Details

Notes:  
to compare zones,  
compare demog groups, (see parameter called group)  
compare multiple groups and/or multiple zones, like hisp vs others in us vs ca all on one graph  
see [weighted.hist](#) for options

### Value

Draws a plot

### See Also

[Ecdf](#) [RR pop.cdf](#) [pop.cdf2](#) [pop.ecdf](#) [pop.cdf.density](#)

## Examples

```
###
## Not run:
# pop.cdf( 31:35, c(0.10, 0.10, 0.40, 0, 0.20), 1001:1005 )

set.seed(99)
pctminsim=c(runif(7000,0,1), pmin(rlnorm(5000, meanlog=log(0.30), sdlog=1.7), 4)/4)
popsim= runif(12000, 500, 3000)
esim= rlnorm(12000, log(10), log(1.15)) + rnorm(12000, 1, 0.5) * pctminsim - 1
pop.cdf(esim, pctminsim, popsim, xlab='Tract air pollution levels',
  main = 'Air pollution levels among minorities (red bars) vs rest of US pop.')
# x1=wtd.mean(esim, weights = pctminsim * popsim)
# x2=wtd.mean(esim, weights = (1-pctminsim) * popsim)

#
# pop.cdf(bg$pm, bg$pctmin, bg$pop)
# pop.cdf(log10(places$traffic.score), places$pctmin, places$pop)
# pop.cdf(places$cancer, places$pctmin, places$pop, allothers=FALSE); pop.cdf(places$cancer, places$pctlingi
# Demog suscept for each REGION (can't see if use vs others)
pop.cdf(bg$traffic.score, bg$VSI.eo, bg$pop, log='x', subtitles=FALSE,
  group=bg$REGION, allothers=FALSE,
  xlab='Traffic score (log scale)', ylab='frequency in population', main='Distribution of scores by EPA

# Demog suscept (how to show vs others??), one panel per ENVT FACTOR (ie per col in scores df)
data('names.evars')
# NOT
pop.cdf(bg[, names.e], bg$VSI.eo, bg$pop, log='x', subtitles=FALSE,
  allothers=TRUE, ylab='frequency in population', main='Distribution of scores by EPA Region')

# log scale is useful & so are these labels passed to function
# in CA vs not CA
pop.cdf(bg$traffic.score, bg$ST=='CA', bg$pop,
  subtitles=FALSE,
  log='x', ylab='frequency in population', xlab='Traffic scores (log scale)',
  main='Distribution of scores in CA (red) vs rest of US')

# Flagged vs not (all D, all zones)
pop.cdf(bg$traffic.score, bg$flagged, bg$pop, log='x')

# D=Hispanics vs others, within CA zone only
pop.cdf(bg$traffic.score, bg$ST=='CA', bg$pop * bg$pcthisp, log='x')
# Demog suscept vs others, within CA only
pop.cdf(bg$traffic.score, bg$ST=='CA', bg$pop * bg$VSI.eo, log='x')

## End(Not run)
```

---

pop.cdf.density

---

*Overlay Two PDFs as Weighted Histogram-Like Density Plots*


---

## Description

Use plot(density()) to see overlay of two distribution functions.

**Usage**

```
pop.cdf.density(e, dcount, refcount, etxt, dtxt, brks = 10, ...)
```

**Arguments**

e	Environmental or other indicator values vector
dcount	Vector of weights for the demographic group of interest, such as population counts of Hispanics by Census tract.
refcount	Vector of weights for the reference group, such as population counts of individuals who are not Hispanic, by Census tract.
etxt	Character string to name e in graph
dtxt	Character string to name d in graph
brks	Default is 10. Passed as breaks param to plot function
...	other parameters passed to density

**Details**

Assumes you have weights for each and are comparing values in one group vs another.

**Value**

Creates a plot

**See Also**

[pop.cdf](#) [pop.cdf2](#) [pop.ecdf](#) [pop.cdf.density](#)

**Examples**

```
\donotrunc{
  pop.cdf.density(e = e, dcount = dcount, refcount = refcount, etxt = etxt, dtxt = dtxt,
}
```

---

pop.cdf2

*Overlay Two PDFs as Weighted Histograms*

---

**Description**

Use `plotrix::weighted.hist` to see overlay of two weighted histograms.

**Usage**

```
pop.cdf2(e, dcount, refcount, etxt, dtxt, brks = 10)
```

**Arguments**

e	Environmental or other indicator values vector
dcount	Vector of weights for the demographic group of interest, such as population counts of Hispanics by Census tract.
refcount	Vector of weights for the reference group, such as population counts of individuals who are not Hispanic, by Census tract.
etxt	Character string to name e in graph
dtxt	Character string to name d in graph
brks	Default is 10. Passed as breaks param to plot function

**Details**

Does not handle NA values well yet. Assumes you have weights for each and are comparing values in one group vs another.

**Value**

Creates a plot

**See Also**

[pop.cdf](#) [pop.cdf2](#) [pop.ecdf](#) [pop.cdf.density](#)

**Examples**

```
\donotrunc{
# can get a dataset for examples
load("~/../Dropbox/EJSCREEN/R analysis/bg 2015-04-22 Rnames plus subgroups.RData")
# # to do this manually
# require(plotrix)
# weighted.hist(bg$proximity.rmp, bg$pop*bg$pctmin,main='pop hist for pctmin of RMP score', xlim=c(0,1.8),fr
# weighted.hist(bg$proximity.rmp,bg$pop*(1-bg$pctmin), main='pop hist for pctmin of RMP score', xlim=c(0,1.8)
e <- bg$pm[!is.na(bg$pm)]
dcount <- bg$pop[!is.na(bg$pm)] * bg$pctmin[!is.na(bg$pm)]
refcount <- bg$pop[!is.na(bg$pm)] * (1 - bg$pctmin[!is.na(bg$pm)])
brks = 0:17
etxt = 'PM2.5'
dtxt = 'minorities'

#pop.cdf2(e, dcount, refcount, etxt, dtxt, brks)
pop.cdf2(e = e, dcount = dcount, refcount = refcount, etxt = etxt, dtxt = dtxt, brks = brks)
# *** BUT, why does it not look the same as ... *****
pop.cdf(scores = e, pcts = bg$pctmin, pops = bg$pop, breaks = brks, main='PM2.5 distribution within each group
}
```

---

pop.ecdf	<i>Draw an Ecdf plot comparing distributions of scores in selected demographic groups</i>
----------	---

---

### Description

Draws a plot using [Ecdf](#), overlaying cumulative distribution functions, one for each subgroup specified. Useful to compare 2 groups based on each groups entire pdf or cdf distribution of peoples scores, using data from small places like census block groups, based on having for each place the pop total and

### Usage

```
pop.ecdf(scores, pcts, pops, allothers = TRUE, col = "red", main = "",
         weights, subtitles = FALSE, ...)
```

### Arguments

scores	Numeric vector (or data.frame) required. Values to analyze. If data.frame, then each column is plotted in its own panel.
pcts	Numeric vector (or data.frame), required. Same number of vector elements or data.frame rows as length of scores vector (not sure what happens if pcts and scores are both data.frames). Specifies the fraction of population that is in demographic group(s) of interest, one row per place, one column per group.
pops	Vector used to define weights as pop*pcts, and if allothers=TRUE, for pop*(1-pcts) for nongroup
allothers	Logical value, optional, TRUE by default. Whether to plot a series for everyone else, using 1-pct
col	Optional, default is 'red' to signify line color red for key demographic group. Can also be a vector of colors if pcts is a data.frame with one column per group, one color per group.
main	Optional character specifying plot title, default is none
weights	Not used currently. See pops parameter
subtitles	Logical FALSE by default, which means extra info is not shown (see help on <a href="#">Ecdf</a> )
...	other optional parameters to pass to Ecdf

### Details

Notes:  
 to compare zones,  
 compare demog groups, (see parameter called group)  
 compare multiple groups and/or multiple zones, like hisp vs others in us vs ca all on one graph  
 see [Ecdf](#) for options & try passing a data.frame instead of just vector

### Value

draws a plot

**See Also**

[Ecdf RR pop.cdf pop.cdf2 pop.ecdf pop.cdf.density](#)

**Examples**

```
###
## Not run:
pop.ecdf( 31:35, c(0.10, 0.10, 0.40, 0, 0.20), 1001:1005 )

set.seed(99)
pctminsim=c(runif(7000,0,1), pmin(rlnorm(5000, meanlog=log(0.30), sdlog=1.7), 4)/4)
popsim= runif(12000, 500, 3000)
esim= rlnorm(12000, log(10), log(1.15)) + rnorm(12000, 1, 0.5) * pctminsim - 1
pop.ecdf(esim, pctminsim, popsim, xlab='Tract air pollution levels (vertical lines are group means)',
  main = 'Air pollution levels among minorities (red curve) vs rest of US pop.')
abline(v=wtd.mean(esim, weights = pctminsim * popsim), col='red')
abline(v=wtd.mean(esim, weights = (1-pctminsim) * popsim), col='black')

pop.ecdf(bg$pm, bg$pctmin, 1000, xlab='Tract air pollution levels (vertical lines are group means)',
  main = 'PM2.5 levels among minorities (red curve) vs rest of US pop.')
abline(v=wtd.mean(bg$pm, weights = bg$pctmin * bg$pop), col='red')
abline(v=wtd.mean(bg$pm, weights = (1-bg$pctmin) * bg$pop), col='black')

#pop.ecdf(dat$Murder, dat$Population * (dat$Illiteracy/100))
pop.ecdf(bg$pm, bg$pctmin, bg$pop,
  main = 'PM2.5 levels among minorities (red curve) vs rest of US population (vertical lines are group means)',
  abline(v=wtd.mean(bg$pm, weights = bg$pctmin * bg$pop), col='red')
  abline(v=wtd.mean(bg$pm, weights = (1-bg$pctmin) * bg$pop), col='black')

pop.ecdf(log10(places$traffic.score), places$pctmin, places$pop)
pop.ecdf(places$cancer, places$pctmin, places$pop, allothers=FALSE); pop.ecdf(places$cancer, places$pctmin, places$pop, allothers=TRUE)
# Demog suscept for each REGION (can't see if use vs others)
pop.ecdf(bg$traffic.score, bg$VSI.eo, bg$pop, log='x', subtitles=FALSE,
  group=bg$REGION, allothers=FALSE,
  xlab='Traffic score (log scale)', ylab='%ile of population', main='Distribution of scores by EPA Region')

# Demog suscept (how to show vs others??), one panel per ENVT FACTOR (ie per col in scores df)
data('names.evars')
pop.ecdf(bg[, names.e], bg$VSI.eo, bg$pop, log='x', subtitles=FALSE,
  allothers=TRUE, ylab='%ile of population', main='Distribution of scores by EPA Region')

# log scale is useful & so are these labels passed to function
# in CA vs not CA
pop.ecdf(bg$traffic.score, bg$ST=='CA', bg$pop,
  subtitles=FALSE,
  log='x', xlab='%ile of population', ylab='Traffic scores (log scale)',
  main='Distribution of scores in CA (red) vs rest of US')

# Flagged vs not (all D, all zones)
pop.ecdf(bg$traffic.score, bg$flagged, bg$pop, log='x')

# D=Hispanics vs others, within CA zone only
pop.ecdf(bg$traffic.score, bg$ST=='CA', bg$pop * bg$pctthisp, log='x')
# Demog suscept vs others, within CA only
pop.ecdf(bg$traffic.score, bg$ST=='CA', bg$pop * bg$VSI.eo, log='x')
```

```
## End(Not run)
```

---

rollup	<i>Aggregate multiple columns of values by group</i>
--------	--

---

## Description

aggregate over zones - \*\*\* work in progress – NOT DONE YET \*\*\*

## Usage

```
rollup(x, by, wts = NULL, FUN, prefix, na.rm = TRUE)
```

## Arguments

x	Dataset
wts	Weights, default is unweighted
FUN	Default is weighted mean
prefix	Default is 'wtd.mean.'
NA	Default is TRUE

## Value

by Vector defining groups

## Examples

```
# SLOW BUT SEEMS TO WORK SOMEWHAT SO FAR
# 1.Do rollup of most fields as wtd mean
#t2 <- rollup(bg[ , names.e], by=bg$FIPS.TRACT, wts=bg$pop)
#names(t2) <- gsub('by', 'FIPS.TRACT', names(t2))
# 2.Do rollup of pop and areas as sum not wtd.mean: # not sure aggregate preserves sort order that rollup creat
#tractpop <- aggregate(bg[ , c('pop', 'area', 'sqmi', 'sqkm')], by=list(bg$FIPS.TRACT), sum)
#names(tractpop) <- c('FIPS.TRACT', c('pop', 'sqmi', 'sqkm'))
# 3.Merge the wtd.mean fields and sum fields, sort results.
#t2 <- merge(t2, tractpop, by='FIPS.TRACT')
#rm(tractpop)
#t2 <- t2[ order(t2$FIPS.TRACT), ]
#
# other examples:
# tracts <- rollup(bg[ , names.e], by=bg$FIPS.TRACT, wts=bg$pop); names(tracts) <- gsub('by', 'FIPS.TRACT'
# counties <- rollup(bg[ , names.e], by=bg$FIPS.COUNTY, wts=bg$pop)
# states <- rollup(bg[ , names.d], by=bg$ST, wts=bg$pop)
# states2 <- rollup(bg[ , c('pm', 'o3')], by=bg$ST, wts=bg$pop)

# SPECIFY WHICH FIELDS TO ROLLUP VIA WTD AVG AND WHICH TO DO VIA SUM OVER US/REGION/COUNTY/STATE/TRACT

# source('myfunctions.R') # if not already available
# load('bg ... plus race eth subgrps ACS0812.RData') # if not already working with it

# Get the wtd.mean for all except pctile, bin (or raw EJ index?)
```



```
#avgnames <- names(bg)
#avgnames <- avgnames[!(grepl('^ptile.', avgnames))]
#avgnames <- avgnames[!(grepl('^bin.', avgnames))]
#avgnames <- avgnames[!(grepl('^EJ.', avgnames))]

# Get the sum for all the raw counts, and area
#sumnames <- c('area', 'pop', 'povknownratio', 'age25up', 'hhlds', 'builtunits', 'mins', 'lowinc', 'lths', 'lths', 'lths', 'lths')
#      'hispanic', 'nhaa', 'nhaiana', 'nhba', 'nhmulti', 'nhnhpia', 'nhotheralone', 'nhwa', 'nonmins', 'sum'
#sumnames <- c(sumnames, names(bg)[grepl('^EJ.', names(bg))])
#
# Get the rollups of wtd.mean cols
#us      <- rollup( bg[, avgnames], wts=bg$pop )
#regions <- rollup( bg[, avgnames], wts=bg$pop, by=bg$REGION)
#states  <- rollup( bg[, avgnames], wts=bg$pop, by=bg$FIPS.ST)
#counties <- rollup( bg[, avgnames], wts=bg$pop, by=bg$FIPS.COUNTY)
#tracts  <- rollup( bg[, avgnames], wts=bg$pop, by=bg$FIPS.TRACT)
#
# Get the rollups of summed cols
#us <- cbind(us, rollup( bg[, ]))
#
#counties <- rollup(bg[, c(names.e, names.d)], wts=bg$pop, by=bg$FIPS.COUNTY)
# doesn't work yet to use rollup??
# counties$pop <- rollup(bg$pop, by=bg$FIPS.COUNTY, FUN=function(x) sum(x, na.rm=TRUE))
#counties$pop <- aggregate(bg$pop, by=list(bg$FIPS.COUNTY), FUN=function(x) sum(x, na.rm=TRUE))
```

rollup.pct	Calculate $a/b$ for each Subset
------------	---------------------------------

## Description

Uses `data.table` package to quickly calculate ratio of a/b within each subset of a dataset (e.g., by zone). This will be superseded by `rollup` once that is completed.

## Usage

```
rollup.pct(a, b, zone)
```

## Arguments

a	Required numeric vector, numerator
b	Required numeric vector, denominator. Same length as a.
zone	Optional, vector to group by. Same length as a and b.

## Value

Returns a table with a/b calculated within each zone.

## Examples

```
pre1960=1:100; builtunits=rep(c(10, 100),50); zone=rep(c('NY', 'MA'),50)
rollup.pct(a,b,zone)
```

---

RR	<i>Relative Risk (RR) by demographic group by indicator based on Census data</i>
----	--

---

### Description

Finds the ratio of mean indicator value in one demographic subgroup to mean in everyone else, based on data for each spatial unit such as for block groups or tracts.

### Usage

```
RR(e, d, pop, dref, na.rm = TRUE)
```

### Arguments

e	Vector or data.frame or matrix with 1 or more environmental indicator(s) or health risk level (e.g., PM2.5 concentration to which this person or place is exposed), one row per Census unit and one column per indicator.
d	Vector or data.frame or matrix with 1 or more demog groups percentage (as fraction of 1, not 0-100!) of place that is selected demog group (e.g. percent Hispanic) (or d=1 or 0 per row if this is a vector of individuals)
pop	Vector of one row per location providing population count of place (or pop=1 if this is a vector of individuals), to convert d into a count since d is a fraction
dref	Optional vector specifying a reference group for RR calculation by providing what percentage (as fraction of 1, not 0-100!) of place that is individuals in the reference group (or dref= vector of ones and zeroes if this is a vector of individuals)
na.rm	Optional, logical, TRUE by default. Specify if NA values should be removed first.

### Details

This function requires, for each Census unit, demographic data on total population and percent in each demographic group, and some indicator(s) for each Census unit, such as health status, exposure estimates, or environmental health risk. For example, given population count, percent Hispanic, and ppm of ozone for each tract, this calculates the ratio of the population mean tract-level ozone concentration among Hispanics divided by the same value among all non-Hispanics. The result is a ratio of means for two demographic groups, or for each of several groups and indicators. Each e (for environmental indicator) or d (for demographic percentage) is specified as a vector over small places like Census blocks or block groups or even individuals (ideally) but then d would be a dummy=1 for selected group and 0 for people not in selected group \*\*\* note: this currently does not use rrf() & rrfv() but perhaps it would be faster if it did? but rrfv not tested for multiple demog groups\*\*\*

\*\* NEED TO VERIFY/TEST THIS: REMOVES PLACES WITH NA in any one or more of the values used (e, d, pop, dref) in numerators and denominators.

\*\* Note also that THIS REMOVES NA VALUES FOR one e factor and not for another, so results can use different places & people for different e factors

**Value**

numeric results as vector or data.frame

**See Also**

- [ej.indexes](#) for local contribution to a variety of overall disparity metrics such as excess risk
- [RR](#) to calculate overall disparity metric as relative risk (RR), ratio of mean environmental indicator values across demographic groups
- [RR.table](#) to create 3-D table of RR values, by demographic group by environmental indicator by zone
- [RR.table.sort](#) to sort existing RR table
- [RR.table.add](#) to add zone(s) to existing RR table
- [write.RR.tables](#) to write a file with a table or RR by indicator by group
- [ej.added](#) to find EJ Index as local contribution to sum of EJ Indexes
- [RR.cut.if.gone](#) to find local contribution to RR
- [RR.if.address.top.x](#) to find how much RR would change if top-ranked places had different conditions
- [rrfv](#) for obsolete attempt to vectorize rrf
- [rrf](#) for older simpler function for RR of one indicator for one demographic group
- [pop.ecdf](#) to compare plots of cumulative frequency distribution of indicator values by group

**Examples**

```
mydat <- structure(list(state = structure(c(1L, 2L, 3L, 4L, 5L, 6L, 7L,
8L, 10L, 11L, 12L, 13L, 14L, 15L, 16L, 17L, 18L, 19L, 20L, 21L,
22L, 23L, 24L, 25L, 26L, 27L, 28L, 29L, 30L, 31L, 32L, 33L, 34L,
35L, 36L, 37L, 38L, 39L, 41L, 42L, 43L, 44L, 45L, 46L, 47L, 48L,
49L, 50L, 51L, 52L), .Label = c("Alabama", "Alaska", "Arizona",
"Arkansas", "California", "Colorado", "Connecticut", "Delaware",
"District of Columbia", "Florida", "Georgia", "Hawaii", "Idaho",
"Illinois", "Indiana", "Iowa", "Kansas", "Kentucky", "Louisiana",
"Maine", "Maryland", "Massachusetts", "Michigan", "Minnesota",
"Mississippi", "Missouri", "Montana", "Nebraska", "Nevada", "New Hampshire",
"New Jersey", "New Mexico", "New York", "North Carolina", "North Dakota",
"Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Puerto Rico",
"Rhode Island", "South Carolina", "South Dakota", "Tennessee",
"Texas", "Utah", "Vermont", "Virginia", "Washington", "West Virginia",
"Wisconsin", "Wyoming"), class = "factor"), pcthisp = c(0.0381527239296627,
0.056769492321473, 0.296826116572835, 0.0635169313105461, 0.375728960493789,
0.206327251656949, 0.134422275491411, 0.0813548250199138, 0.2249082771481,
0.0878682317249484, 0.0901734019211436, 0.111947738331921, 0.158094676641822,
0.0599941716405598, 0.0495552961203499, 0.104741084665558, 0.0301921562004411,
0.0425162900517816, 0.0129720920573869, 0.0816325860392955, 0.0960897601513277,
0.0442948677533508, 0.0470583828855611, 0.0264973278249911, 0.0354626134972627,
0.0292535716628734, 0.0914761950105784, 0.265451497002445, 0.0283535007142456,
0.177132117215957, 0.463472498001496, 0.176607017430808, 0.0834317084560556,
0.0209906647364226, 0.0307719359181436, 0.0883052970054721, 0.117261303415395,
0.0568442805511265, 0.124769233546578, 0.0503041778042313, 0.0279186292931113,
0.0454229079840698, 0.376044616311455, 0.129379195461843, 0.0151111594281676,
0.0788112971314249, 0.111945098129999, 0.0119028512046327, 0.0589405823830593,
0.0893971780534219), pop = c(3615, 365, 2212, 2110, 21198, 2541,
```

```

3100, 579, 8277, 4931, 868, 813, 11197, 5313, 2861, 2280, 3387,
3806, 1058, 4122, 5814, 9111, 3921, 2341, 4767, 746, 1544, 590,
812, 7333, 1144, 18076, 5441, 637, 10735, 2715, 2284, 11860,
931, 2816, 681, 4173, 12237, 1203, 472, 4981, 3559, 1799, 4589,
376), murder = c(15.1, 11.3, 7.8, 10.1, 10.3, 6.8, 3.1, 6.2,
10.7, 13.9, 6.2, 5.3, 10.3, 7.1, 2.3, 4.5, 10.6, 13.2, 2.7, 8.5,
3.3, 11.1, 2.3, 12.5, 9.3, 5, 2.9, 11.5, 3.3, 5.2, 9.7, 10.9,
11.1, 1.4, 7.4, 6.4, 4.2, 6.1, 2.4, 11.6, 1.7, 11, 12.2, 4.5,
5.5, 9.5, 4.3, 6.7, 3, 6.9), area = c(50708, 566432, 113417,
51945, 156361, 103766, 4862, 1982, 54090, 58073, 6425, 82677,
55748, 36097, 55941, 81787, 39650, 44930, 30920, 9891, 7826,
56817, 79289, 47296, 68995, 145587, 76483, 109889, 9027, 7521,
121412, 47831, 48798, 69273, 40975, 68782, 96184, 44966, 1049,
30225, 75955, 41328, 262134, 82096, 9267, 39780, 66570, 24070,
54464, 97203), temp = c(62.8, 26.6, 60.3, 60.4, 59.4, 45.1, 49,
55.3, 70.7, 63.5, 70, 44.4, 51.8, 51.7, 47.8, 54.3, 55.6, 66.4,
41, 54.2, 47.9, 44.4, 41.2, 63.4, 54.5, 42.7, 48.8, 49.9, 43.8,
52.7, 53.4, 45.4, 59, 40.4, 50.7, 59.6, 48.4, 48.8, 50.1, 62.4,
45.2, 57.6, 64.8, 48.6, 42.9, 55.1, 48.3, 51.8, 43.1, 42)), .Names = c("state",
"pcthis", "pop", "murder", "area", "temp"), class = "data.frame", row.names = c(NA,
-50L))

```

```

RR(mydat$area, mydat$pcthis, mydat$pop)
# Avg Hispanic lives in a State that is 69 percent larger than
  that of avg. non-Hispanic

RR(mydat$pcthis, mydat$pcthis, mydat$pop)
# Avg Hispanic lives in a State that has a much higher percent Hispanic than
  do non-Hispanics

#cbind(RR=RR(data.frame(d1=bg$pcthis, d2=1-bg$pcthis), bg$pcthis, bg$pop))
#RR(bg[, names.e], bg$pctlowinc, bg$pop)
#sapply(bg[, names.d], function(z) RR(bg[, names.e], z, bg$pop) )

```

RR.cut.if.gone

*How much is overall RR reduced if a given place did not exist?***Description**

As with RR function, calculates RR as ratio of means in one demographic group vs reference, based on Census demographic data and environmental indicator data on each place. Then this finds how much smaller RR would be if the given place did not exist.

**Usage**

```
RR.cut.if.gone(e, d, pop, dref, na.rm = TRUE)
```

**Arguments**

e	environmental indicator value
d	demog group as fraction of pop
pop	pop count
dref	reference demog group as fraction
na.rm	TRUE by default, should NA values be removed first

## Examples

```
# x=RR.cut.if.gone(bg[ , names.e[8]], bg$pctlowinc, bg$pop)
# summary(x*1000)
mydat=data.frame(AQI=99:101, pctlowinc=c(0.20,0.30,0.40), pop=rep(1000,3))
RR(mydat$AQI, mydat$pctlowinc, mydat$pop)
RR.cut.if.gone(e=mydat$AQI, d=mydat$pctlowinc, pop=mydat$pop)
```

---

RR.if.address.top.x	<i>Analyze how much RR would change if top-ranked places were addressed - ** NOT WORKING/ IN PROGRESS</i>
---------------------	---

---

## Description

Function to analyze data on demographic and environmental (e) indicators by place (e.g., Census block group) to get stats on what percent of population or places could account for all risk ratio (RR, or ratio of mean e in one demog group vs others), and what is risk ratio if you reduce e (environmental indicator) by using some multiplier on e, in top x percent of places

## Usage

```
RR.if.address.top.x(rank.by.df, e.df, d.pct, popcounts, d.pct.us,
  or.tied = TRUE, if.multiply.e.by = 0, zones = NULL, mycuts = c(50, 80,
  90:100), silent = TRUE)
```

## Arguments

rank.by.df	Data.frame of indicators to rank places by, when defining top places
e.df	Environmental indicators data.frame, one row per place, required.
d.pct	Demographic percentage, as fraction, defining what fraction of population in each place (row) is in demographic group of interest. Required.
popcounts	Numeric vector of counts of total population in each place
d.pct.us	xxxxx
or.tied	Logical value, optional, TRUE by default, in which case ties of ranking variable with a cutoff value (value >= cutoff) are included in places within that bin.
if.multiply.e.by	Optional, 0 by default. Specifies the number that environmental indicator values would be multiplied by in the scenario where some places are addressed. Zero means those top-ranked places would have the environmental indicator set to zero, while 0.9 would mean and 10 percent cut in the environmental indicator value.
zones	Subsets of places such as States
mycuts	optional vector of cutoff values to analyze. Default is c(50,80,90:100)
silent	optional logical, default is TRUE, while FALSE means more information is printed

## Details

The effects of one place on overall RR is related to an ej.index, which here is a metric describing one place's contribution to an overall metric of disparity. RR is one overall metric of disparity, the ratio of mean environmental indicator value in one demographic group over the mean in the reference group. If  $RR = E/e$ , where  $E$ =avg environmental indicator or risk in key demographic group and  $e$ = in reference group, another metric of disparity is the excess individual risk, or  $E-e$ . The excess population risk or excess cases would be  $(E-e) * p * d$  where  $p$ =total population and  $d$ =fraction that is in key demographic group. Various counterfactuals could be used here for scenario defining what it means to address the top places and what is used to define top places.

## Value

Returns a list of results:

1. rrs data.frame, one column per environmental indicator, one row per cutoff value
2. rrs2 data.frame, Relative risks 2
3. state.tables A list
4. worst.as.pct Worst as percent, vector as long as number of environmental indicators
5. worst.as.pct.of.bgs Worst as percent of places (e.g., block groups)

## See Also

[RR](#) and [RR.if.address.top.x](#) and [ej.indexes](#) and [ej.added](#)

## Examples

```
###
```

---

RR.means	<i>Relative Risk (RR) components - Means in demographic group and in rest of pop, based on Census data</i>
----------	--

---

## Description

Finds the mean indicator value in one demographic subgroup and mean in everyone else, based on data for each spatial unit such as for block groups or tracts.

## Usage

```
RR.means(e, d, pop, dref, na.rm = TRUE)
```

## Arguments

- |     |   |
|-----|---|
| e   | Vector or data.frame or matrix with 1 or more environmental indicator(s) or health risk level (e.g., PM2.5 concentration to which this person or place is exposed), one row per Census unit and one column per indicator. |
| d   | Vector or data.frame or matrix with 1 or more demog groups percentage (as fraction of 1, not 0-100!) of place that is selected demog group (e.g. percent Hispanic) (or d=1 or 0 if this is a vector of individuals)       |
| pop | Vector of one row per location providing population count of place (or pop=1 if this is a vector of individuals), to convert d into a count since d is a fraction   |

## Details

This function requires, for each Census unit, demographic data on total population and percent in each demographic group, and some indicator(s) for each Census unit, such as health status, exposure estimates, or environmental health risk. For example, given population count, percent Hispanic, and ppm of ozone for each tract, this calculates the population mean tract-level ozone concentration among Hispanics and the same value among all non-Hispanics. The result is a table of means for a demographic subset, or for each of several groups and indicators. Each e (for environmental indicator) or d (for demographic percentage) is specified as a vector over small places like Census blocks or block groups or even individuals (ideally) but then d would be a dummy=1 for selected group and 0 for people not in selected group NOTE: could NA values cause a problem here?

## Value

numeric results as vector or data.frame

## See Also

- [ej.indexes](#) for local contribution to a variety of overall disparity metrics such as excess risk
- [RR](#) to calculate overall disparity metric as relative risk (RR), ratio of mean environmental indicator values across demographic groups
- [RR.table](#) to create 3-D table of RR values, by demographic group by environmental indicator by zone
- [RR.table.sort](#) to sort existing RR table
- [RR.table.add](#) to add zone(s) to existing RR table
- [write.RR.tables](#) to write a file with a table or RR by indicator by group
- [ej.added](#) to find EJ Index as local contribution to sum of EJ Indexes
- [RR.cut.if.gone](#) to find local contribution to RR
- [RR.if.address.top.x](#) to find how much RR would change if top-ranked places had different conditions
- [rrfv](#) for obsolete attempt to vectorize rrf
- [rrf](#) for older simpler function for RR of one indicator for one demographic group
- [pop.ecdf](#) to compare plots of cumulative frequency distribution of indicator values by group

## Examples

```
# RR.means( bg$proximity.rmp, bg$pcthisp, bg$pop)
# RR.means(bg[ , names.e], bg$pcthisp, bg$pop)
# RR.means( bg$proximity.rmp, cbind(bg[ , names.d.subgroups],1), bg$pop)
mydat <- structure(list(state = structure(c(1L, 2L, 3L, 4L, 5L, 6L, 7L,
8L, 10L, 11L, 12L, 13L, 14L, 15L, 16L, 17L, 18L, 19L, 20L, 21L,
22L, 23L, 24L, 25L, 26L, 27L, 28L, 29L, 30L, 31L, 32L, 33L, 34L,
35L, 36L, 37L, 38L, 39L, 41L, 42L, 43L, 44L, 45L, 46L, 47L, 48L,
49L, 50L, 51L, 52L), .Label = c("Alabama", "Alaska", "Arizona",
"Arkansas", "California", "Colorado", "Connecticut", "Delaware",
"District of Columbia", "Florida", "Georgia", "Hawaii", "Idaho",
"Illinois", "Indiana", "Iowa", "Kansas", "Kentucky", "Louisiana",
"Maine", "Maryland", "Massachusetts", "Michigan", "Minnesota",
"Mississippi", "Missouri", "Montana", "Nebraska", "Nevada", "New Hampshire",
"New Jersey", "New Mexico", "New York", "North Carolina", "North Dakota",
"Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Puerto Rico",
```

```
"Rhode Island", "South Carolina", "South Dakota", "Tennessee",
"Texas", "Utah", "Vermont", "Virginia", "Washington", "West Virginia",
"Wisconsin", "Wyoming"), class = "factor"), pcthisp = c(0.0381527239296627,
0.056769492321473, 0.296826116572835, 0.0635169313105461, 0.375728960493789,
0.206327251656949, 0.134422275491411, 0.0813548250199138, 0.2249082771481,
0.0878682317249484, 0.0901734019211436, 0.111947738331921, 0.158094676641822,
0.0599941716405598, 0.0495552961203499, 0.104741084665558, 0.0301921562004411,
0.0425162900517816, 0.0129720920573869, 0.0816325860392955, 0.0960897601513277,
0.0442948677533508, 0.0470583828855611, 0.0264973278249911, 0.0354626134972627,
0.0292535716628734, 0.0914761950105784, 0.265451497002445, 0.0283535007142456,
0.177132117215957, 0.463472498001496, 0.176607017430808, 0.0834317084560556,
0.0209906647364226, 0.0307719359181436, 0.0883052970054721, 0.117261303415395,
0.0568442805511265, 0.124769233546578, 0.0503041778042313, 0.0279186292931113,
0.0454229079840698, 0.376044616311455, 0.129379195461843, 0.0151111594281676,
0.0788112971314249, 0.111945098129999, 0.0119028512046327, 0.0589405823830593,
0.0893971780534219), pop = c(3615, 365, 2212, 2110, 21198, 2541,
3100, 579, 8277, 4931, 868, 813, 11197, 5313, 2861, 2280, 3387,
3806, 1058, 4122, 5814, 9111, 3921, 2341, 4767, 746, 1544, 590,
812, 7333, 1144, 18076, 5441, 637, 10735, 2715, 2284, 11860,
931, 2816, 681, 4173, 12237, 1203, 472, 4981, 3559, 1799, 4589,
376), murder = c(15.1, 11.3, 7.8, 10.1, 10.3, 6.8, 3.1, 6.2,
10.7, 13.9, 6.2, 5.3, 10.3, 7.1, 2.3, 4.5, 10.6, 13.2, 2.7, 8.5,
3.3, 11.1, 2.3, 12.5, 9.3, 5, 2.9, 11.5, 3.3, 5.2, 9.7, 10.9,
11.1, 1.4, 7.4, 6.4, 4.2, 6.1, 2.4, 11.6, 1.7, 11, 12.2, 4.5,
5.5, 9.5, 4.3, 6.7, 3, 6.9), area = c(50708, 566432, 113417,
51945, 156361, 103766, 4862, 1982, 54090, 58073, 6425, 82677,
55748, 36097, 55941, 81787, 39650, 44930, 30920, 9891, 7826,
56817, 79289, 47296, 68995, 145587, 76483, 109889, 9027, 7521,
121412, 47831, 48798, 69273, 40975, 68782, 96184, 44966, 1049,
30225, 75955, 41328, 262134, 82096, 9267, 39780, 66570, 24070,
54464, 97203), temp = c(62.8, 26.6, 60.3, 60.4, 59.4, 45.1, 49,
55.3, 70.7, 63.5, 70, 44.4, 51.8, 51.7, 47.8, 54.3, 55.6, 66.4,
41, 54.2, 47.9, 44.4, 41.2, 63.4, 54.5, 42.7, 48.8, 49.9, 43.8,
52.7, 53.4, 45.4, 59, 40.4, 50.7, 59.6, 48.4, 48.8, 50.1, 62.4,
45.2, 57.6, 64.8, 48.6, 42.9, 55.1, 48.3, 51.8, 43.1, 42)), .Names = c("state",
"pcthisp", "pop", "murder", "area", "temp"), class = "data.frame", row.names = c(NA,
-50L))
```

```
RR.means(mydat$area, mydat$pcthisp, mydat$pop)
```

```
RR.means(mydat$pcthisp, mydat$pcthisp, mydat$pop)
# CHECK FORMATS OF OUTPUTS: *** WORK IN PROGRESS
#cbind(mymeans=RR.means(data.frame(d1=bg$pcthisp, d2=1-bg$pcthisp), bg$pcthisp, bg$pop))
#RR.means(bg[, names.e], bg$pctlowinc, bg$pop)
#sapply(bg[, names.d], function(z) RR.means(bg[, names.e], z, bg$pop) )
```

---

RR.plot

---

*Draw lineplot comparing RR values by group*


---

## Description

Draws a plot using relative risk information, one line per group

## Usage

```
RR.plot(RRS, d, e, zone = "NY")
```



**Arguments**

RRS	Table as from <a href="#">RRS</a> function, of relative risk by group, risk type, and zone (3 dimensions).
d	Demographic group names to be found as names of first dim of RRS
e	Environmental factor or risk type names to be found as names of second dim of RRS
zone	Zone name to be found among names for third dim of RRS. Default is "NY"

**Value**

draws a plot

**See Also**

[RR](#)

**Examples**

```
#
####
```

---

RR.table

---

*Table of Relative Risk results by zone by group by envt risk factor*


---

**Description**

Make table of Relative Risk results by zone by group by envt risk factor

**Usage**

```
RR.table(mydat, Enames, Dnames, popcolname, Zcolname, testing = FALSE,
         digits = 4)
```

**Arguments**

mydat	Data.frame of input data, one row per geographic unit such as US Census block groups, or tracts.
Enames	names of columns with environmental risk factor data
Dnames	names of columns with percent (fraction) of population that is in each given demographic group
popcolname	name of column with total population count
Zcolname	name of column with name of zone such as US State name
testing	default is FALSE

**Value**

Compiles RR values in array of 3 dimensions: RRS[Dnames, Enames, Zcolnames] Returns a matrix with one demographic group per row, one environmental risk indicator per column, and third dimension for which zone (e.g., which US State)

See Also

- [ej.indexes](#) for local contribution to a variety of overall disparity metrics such as excess risk
- [RR](#) to calculate overall disparity metric as relative risk (RR), ratio of mean environmental indicator values across demographic groups
- [RR.table](#) to create 3-D table of RR values, by demographic group by environmental indicator by zone
- [RR.table.sort](#) to sort existing RR table
- [RR.table.add](#) to add zone(s) to existing RR table
- [write.RR.tables](#) to write a file with a table or RR by indicator by group
- [ej.added](#) to find EJ Index as local contribution to sum of EJ Indexes
- [RR.cut.if.gone](#) to find local contribution to RR
- [RR.if.address.top.x](#) to find how much RR would change if top-ranked places had different conditions
- [rrfv](#) for obsolete attempt to vectorize rrf
- [rrf](#) for older simpler function for RR of one indicator for one demographic group
- [pop.ecdf](#) to compare plots of cumulative frequency distribution of indicator values by group

Examples

```
RRS.US <- RR.table(mydat=bg, Enames=names.e, Dnames=c(names.d, names.d.subgroups.pct), popcolname='pop')
RRS.ST <- RR.table(mydat=bg, Enames=names.e, Dnames=c(names.d, names.d.subgroups.pct), popcolname='pop', Zc
RRS <- RR.table.add(RRS.ST, RRS.US)
RRS[ 'pctlowinc', , ]
RRS[ , , 'CA']
RRS[ , 'pm', ]
RRS.REGION <- RR.table(mydat=bg, Enames=names.e, Dnames=c(names.d, names.d.subgroups.pct), popcolname='pop'
RRS2 <- RR.table.add(RRS, RRS.REGION)
RRS2[ , , '8']
```

---

RR.table.add	Merge tables of Relative Risk results
--------------	---------------------------------------

---

Description

Merge table of Relative Risk results for some zones with table for USA overall

Usage

```
RR.table.add(rrs1, rrs2, zones2)
```

Arguments

rrs1	First table from RR.table()
rrs2	Another table from RR.table()
zones2	Zones in rrs2, as vector, default is dimnames(rrs2)[[3]]

Value

Returns a new array

See Also

- [ej.indexes](#) for local contribution to a variety of overall disparity metrics such as excess risk
- [RR](#) to calculate overall disparity metric as relative risk (RR), ratio of mean environmental indicator values across demographic groups
- [RR.table](#) to create 3-D table of RR values, by demographic group by environmental indicator by zone
- [RR.table.sort](#) to sort existing RR table
- [RR.table.add](#) to add zone(s) to existing RR table
- [write.RR.tables](#) to write a file with a table or RR by indicator by group
- [ej.added](#) to find EJ Index as local contribution to sum of EJ Indexes
- [RR.cut.if.gone](#) to find local contribution to RR
- [RR.if.address.top.x](#) to find how much RR would change if top-ranked places had different conditions
- [rrfv](#) for obsolete attempt to vectorize rrf
- [rrf](#) for older simpler function for RR of one indicator for one demographic group
- [pop.ecdf](#) to compare plots of cumulative frequency distribution of indicator values by group

Examples

```
RRS.US <- RR.table(mydat=bg, Enames=names.e, Dnames=c(names.d, names.d.subgroups.pct), popcolname='pop')
RRS.ST <- RR.table(mydat=bg, Enames=names.e, Dnames=c(names.d, names.d.subgroups.pct), popcolname='pop', Z
RRS <- RR.table.add(RRS.ST, RRS.US)
RRS[ 'pctlowinc', , ]
RRS[ , , 'CA']
RRS[ , 'pm', ]
RR.table.sort(RRS)

RRS.REGION <- RR.table(mydat=bg, Enames=names.e, Dnames=c(names.d, names.d.subgroups.pct), popcolname='pop
RRS2 <- RR.table.add(RRS, RRS.REGION)
RRS2[ , , '8']
```

---

RR.table.sort	<i>Sort table of Relative Risk results by zone by group by envt risk factor</i>
---------------	---

---

Description

Sort table of Relative Risk results by zone by group by envt risk factor

Usage

```
RR.table.sort(x, decreasing = TRUE)
```

Arguments

x	Array of RR values from <a href="#">RR</a> in array of 3 dimensions: x[Dnames, Enames, Zcolnames]
decreasing	default is TRUE, defines how to sort second and third dimensions (envt risk and demog group)

Value

Sorted version of x array of 3 dimensions: RRS[Dnames, Enames, Zcolnames] Returns an array with one demographic group per row, one environmental risk indicator per column, and third dimension for which zone (e.g., which US State)

See Also

- [ej.indexes](#) for local contribution to a variety of overall disparity metrics such as excess risk
- [RR](#) to calculate overall disparity metric as relative risk (RR), ratio of mean environmental indicator values across demographic groups
- [RR.table](#) to create 3-D table of RR values, by demographic group by environmental indicator by zone
- [RR.table.sort](#) to sort existing RR table
- [RR.table.add](#) to add zone(s) to existing RR table
- [write.RR.tables](#) to write a file with a table or RR by indicator by group
- [ej.added](#) to find EJ Index as local contribution to sum of EJ Indexes
- [RR.cut.if.gone](#) to find local contribution to RR
- [RR.if.address.top.x](#) to find how much RR would change if top-ranked places had different conditions
- [rrfv](#) for obsolete attempt to vectorize rrf
- [rrf](#) for older simpler function for RR of one indicator for one demographic group
- [pop.ecdf](#) to compare plots of cumulative frequency distribution of indicator values by group

Examples

```
RRS.US <- RR.table(mydat=bg, Enames=names.e, Dnames=c(names.d, names.d.subgroups.pct), popcolname='pop')
RRS.ST <- RR.table(mydat=bg, Enames=names.e, Dnames=c(names.d, names.d.subgroups.pct), popcolname='pop', Zc
RRS <- RR.table.add(RRS.ST, RRS.US)
RRS[ 'pctlowinc', , ]
RRS[ , , 'CA']
RRS[ , 'pm', ]

RR.table.sort(RRS)

RRS.REGION <- RR.table(mydat=bg, Enames=names.e, Dnames=c(names.d, names.d.subgroups.pct), popcolname='pop'
RRS2 <- RR.table.add(RRS, RRS.REGION)
RRS2[ , , '8']
```

---

RR.table.view	<i>Print Relative Risks (RR) info by demog group by risk type by zone</i>
---------------	---

---

Description

Takes zone-specific RRS values and prints them for viewing.

Usage

```
RR.table.view(RRS, d, e, zone)
```

**Arguments**

RRS	A three dimensional array that is created by <a href="#">RR</a> and has format like RRS[d, e, zone]
d	Vector of names of demographic groups that must be subset of first dimension of RRS
e	Vector of names of environmental risk indicators that must be subset of second dimension of RRS
zone	Vector of zone names that must be subset of names of third dimension of RRS

**Value**

prints RRS[d, e, zone]

**See Also**

- [ej.indexes](#) for local contribution to a variety of overall disparity metrics such as excess risk
- [RR](#) to calculate overall disparity metric as relative risk (RR), ratio of mean environmental indicator values across demographic groups
- [RR.table](#) to create 3-D table of RR values, by demographic group by environmental indicator by zone
- [RR.table.sort](#) to sort existing RR table
- [RR.table.add](#) to add zone(s) to existing RR table
- [write.RR.tables](#) to write a file with a table or RR by indicator by group
- [ej.added](#) to find EJ Index as local contribution to sum of EJ Indexes
- [RR.cut.if.gone](#) to find local contribution to RR
- [RR.if.address.top.x](#) to find how much RR would change if top-ranked places had different conditions
- [rrfv](#) for obsolete attempt to vectorize rrf
- [rrf](#) for older simpler function for RR of one indicator for one demographic group
- [pop.ecdf](#) to compare plots of cumulative frequency distribution of indicator values by group

---

RR.table.vs.us

*Relative Risks (RR) in zones as ratios to US values*


---

**Description**

Takes zone-specific RRS values and divides each value by the corresponding value for the USA overall.

**Usage**

```
RR.table.vs.us(RRS, d, e, zone)
```

**Arguments**

RRS	A three dimensional array that is created by <a href="#">RR</a> and has format like RRS[d, e, zone] but where zone must include the name USA
d	Vector of names of demographic groups that must be subset of first dimension of RRS
e	Vector of names of environmental risk indicators that must be subset of second dimension of RRS
zone	Vector of zone names that must be subset of names of third dimension of RRS

**Value**

numeric results same shape as RRS[d, e, zone]

**See Also**

- [ej.indexes](#) for local contribution to a variety of overall disparity metrics such as excess risk
- [RR](#) to calculate overall disparity metric as relative risk (RR), ratio of mean environmental indicator values across demographic groups
- [RR.table](#) to create 3-D table of RR values, by demographic group by environmental indicator by zone
- [RR.table.sort](#) to sort existing RR table
- [RR.table.add](#) to add zone(s) to existing RR table
- [write.RR.tables](#) to write a file with a table of RR by indicator by group
- [ej.added](#) to find EJ Index as local contribution to sum of EJ Indexes
- [RR.cut.if.gone](#) to find local contribution to RR
- [RR.if.address.top.x](#) to find how much RR would change if top-ranked places had different conditions
- [rrfv](#) for obsolete attempt to vectorize rrf
- [rrf](#) for older simpler function for RR of one indicator for one demographic group
- [pop.ecdf](#) to compare plots of cumulative frequency distribution of indicator values by group

---

rrf

*RR for one environmental indicator, one demographic group*


---

**Description**

**\*\*Probably obsolete given [RR](#).**

Inputs are vectors over small places like Census blocks or block groups or possibly individuals (ideally) but then d would be a dummy=1 for selected group and 0 for people not in selected group

**Usage**

```
rrf(e, d, pop, na.rm = TRUE)
```

## Arguments

<code>e</code>	Environmental indicator vector, one number per place, required. <code>e</code> is environmental index or health risk level (e.g., PM2.5 concentration to which this person or place is exposed)
<code>d</code>	Demographic indicator vector, required, one number per place, fraction of population that is in group of interest <code>d</code> is percent of place that is selected demog group (e.g. percent Hispanic) (or <code>d=1</code> or 0 if this is a vector of individuals)
<code>pop</code>	Population total per place, required, of which <code>d</code> is a fraction. <code>pop</code> is population count of place (or <code>pop=1</code> if this is a vector of individuals)
<code>na.rm</code>	Logical optional TRUE by default in which case NA values (missing values) are removed first. If FALSE, any NA value in <code>pop</code> , <code>e</code> , or <code>d</code> would make result NA.

## Value

Returns one number

## See Also

- [ej.indexes](#) for local contribution to a variety of overall disparity metrics such as excess risk
- [RR](#) to calculate overall disparity metric as relative risk (RR), ratio of mean environmental indicator values across demographic groups
- [RR.table](#) to create 3-D table of RR values, by demographic group by environmental indicator by zone
- [RR.table.sort](#) to sort existing RR table
- [RR.table.add](#) to add zone(s) to existing RR table
- [write.RR.tables](#) to write a file with a table or RR by indicator by group
- [ej.added](#) to find EJ Index as local contribution to sum of EJ Indexes
- [RR.cut.if.gone](#) to find local contribution to RR
- [RR.if.address.top.x](#) to find how much RR would change if top-ranked places had different conditions
- [rrfv](#) for obsolete attempt to vectorize `rrf`
- `rrf` for older simpler function for RR of one indicator for one demographic group
- [pop.ecdf](#) to compare plots of cumulative frequency distribution of indicator values by group

## Examples

```
#
# rrf(places$pm, places[, , unlist(Dlist)[1]], places$pop)
```

rrfv

*Vectorized version of rrf(relative risk) - \*\*not tested, possibly obsolete***Description**

Probably obsolete given [RR](#). Provides Relative Risk (RR) by demographic group by indicator based on Census data Inputs are vectors over small places like Census blocks or block groups or possibly individuals (ideally) but then d would be a dummy=1 for selected group and 0 for people not in selected group. For one environmental indicator, multiple demographic groups.

**Usage**

```
rrfv(e, d, pop, na.rm = TRUE)
```

**Arguments**

e	Environmental indicator vector, one per place, required. e is environmental index or health risk level (e.g., PM2.5 concentration to which this person or place is exposed)
d	Demographic indicator vector (not matrix/df?), required, one per place, fraction of population that is in group of interest d is percent of place that is selected demog group (e.g. percent Hispanic) (or d=1 or 0 if this is a vector of individuals)
pop	Population total per place, required, of which d is a fraction. pop is population count of place (or pop=1 if this is a vector of individuals)
na.rm	Logical optional TRUE by default in which case NA values (missing values) are removed first. If FALSE, any NA value in pop, e, or d would make result NA.

**Value**

Returns numeric vector if one demographic group? \*\* check

**See Also**

- [ej.indexes](#) for local contribution to a variety of overall disparity metrics such as excess risk
- [RR](#) to calculate overall disparity metric as relative risk (RR), ratio of mean environmental indicator values across demographic groups
- [RR.table](#) to create 3-D table of RR values, by demographic group by environmental indicator by zone
- [RR.table.sort](#) to sort existing RR table
- [RR.table.add](#) to add zone(s) to existing RR table
- [write.RR.tables](#) to write a file with a table or RR by indicator by group
- [ej.added](#) to find EJ Index as local contribution to sum of EJ Indexes
- [RR.cut.if.gone](#) to find local contribution to RR
- [RR.if.address.top.x](#) to find how much RR would change if top-ranked places had different conditions
- [rrfv](#) for obsolete attempt to vectorize rrf
- [rrf](#) for older simpler function for RR of one indicator for one demographic group
- [pop.ecdf](#) to compare plots of cumulative frequency distribution of indicator values by group



## Examples

```
#
# rrfv(places$pm, places[, unlist(Dlist)], places$pop)
```

---

state.health.url	<i>Get URL(s) with State health indicator data from RWJF - ** url scheme obsolete now so needs to be redone</i>
------------------	---

---

## Description

Robert Woods Johnson Foundation provides health indicator data by state. This function provides a basic interface to those webpages.

## Usage

```
state.health.url(ST = NA, scope = "state", ind = 66, dist = 23,
  char = 87, time = 3, viz = "bar", fstate = NA, locs = NA,
  cmp = "stcmp", open.browser = FALSE)
```

## Arguments

ST	State abbreviation, FIPS code, or name as character vector
scope	Character, default is "state" view in results, but can be "national" view as well
ind	Health Indicator code number. Number, default is 66. Possible values: <ul style="list-style-type: none"> <li>• 6=Cancer Incidence: Incidence of breast, cervical, lung and colorectal cancer per 100,000 population; age adjusted</li> <li>• 7=Cancer Incidence by Race: Incidence of breast, cervical, lung and colorectal cancer per 100,000 population; age adjusted</li> <li>• 10=Chronic Disease Prevalence: asthma/CVD/diabetes,</li> <li>• 15=Limited Activity: Average number of days in the previous 30 days when a person indicates their activities are limited due to mental or physical health difficulties,</li> <li>• 22=Tobacco taxes: State cigarette excise tax rate (\$)</li> <li>• 25=Income Inequality (Gini Coefficient)</li> <li>• 31=life expectancy= Life expectancy at birth: number of years that a new-born is expected to live if current mortality rates continue to apply</li> <li>• 44=Public health funding: Per capita state public health funding</li> <li>• 45=Poor/Fair Health: Self-reported health status: percent of adults reporting fair or poor health</li> <li>• 65=Premature death: Premature deaths: Average number of years of potential life lost prior to age 75 per 100,000 population</li> <li>• 66=Premature Death by Race/Ethnicity: Premature deaths: Average number of years of potential life lost prior to age 75 per 100,000 population</li> </ul>
dist	default is 23. unclear what this controls. 23 or 29 or 0 or 19 possible. dist/char were 0/0 for US totals not by race, and were 19/58 for same by race.
char	default is 87. unclear what this controls. 91=?, 119=?, 121=? 58? others possible.

time	Code for which years are covered by the data. default is 3, which means 2009-2010. Possible values: <ul style="list-style-type: none"> <li>• 3=2009-2010</li> <li>• 22=2010-2011</li> <li>• 23=2011-2012</li> <li>• 5=2000</li> <li>• 10=2005</li> <li>• 11=2006</li> <li>• 12=2007</li> <li>• 13=2008</li> <li>• 1=2009</li> <li>• 14=2010</li> <li>• 4=2011</li> <li>• 24=2012</li> <li>• OTHERS?</li> </ul>
viz	Type of visualization of data. Default is "bar" and can be line or bar or table.
fstate	State number. default is NA
locs	Locations to compare. default is NA
cmp	Comparisons to view. Default is "stcmp" to see 2+ states compared (or state vs US). Can also see breakdown for one state if set to "brkdwn"
open.browser	Logical, default is FALSE. Should URL be opened by launching browser.

## Details

NEWER URL SCHEME NOW... E.G. <http://www.rwjf.org/en/how-we-work/rel/research-features/rwjf-datahub/national.html#q/scope/national/ind/10/dist/0/char/0/time/3/viz/map/cmp/brkdwn>

see also related percent insured data here: e.g., <http://datacenter.shadac.org/profile/70#2/alabama/percent,moe,count/a/hide> DEFAULT IF NO PARAMETERS USED: state.health.url()

url created:

<http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub/national.html#q/scope/state/ind/66/dist/23/>

url that it resolves to on website:

<http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub/national.html#q/scope/state/ind/66/dist/23/>

# state.health.url("MD")

# url created:

# <http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub/national.html#q/scope/state/ind/66/dist/23/char/87/time/14/viz/bar/fstate/21/locs/21/cmp/stcmp>

# url that it resolves to on website should be

# <http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub/national.html#q/scope/state/ind/66/dist/23/char/87/time/3/viz/bar/fstate/21/locs/21,52/cmp/stcmp>

# URL scheme for linking to Resources for health data by state or by county:

#State-level data from SHADAC: <http://www.shadac.org/>

#State-level data from RWJF: <http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub.html>

\*\*\*\*\* see entire US clickable map of states # `shell.exec('http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub/national.html#q/scope/national/ind/31/dist/29/char/119/time/13/viz/map/fstate/2/locs/2,52/cmp/brkdw') # Use this URL format to figure out API or state-specific set of links to nice state reports. Public health, premature deaths, MD vs US, 2009-2010 http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub/national.html#q/scope/state/ind/66/dist/23/char/91/time/3/viz/bar/fstate/21/locs/21,52/cmp/stcmp Public health, life expectancy, MD (state number 21) vs US http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub/national.html#q/scope/state/ind/31/dist/29/char/119/time/14/viz/line/fstate/21/locs/21,52/cmp/brkdw http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub/national.html#q/scope/state/ind/31/dist/29/char/119/time/14/viz/bar/fstate/21/locs/21,52/cmp/stcmp Cancer by race in arizona (defaults to 2008-2009): http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub/national.html#q/scope/state/ind/7/dist/22/char/85/time/18/viz/bar/fstate/3/locs/3,52/cmp/stcmp`

### Value

URL(s) character vector, default: <http://www.rwjf.org/en/research-publications/research-features/rwjf-datahub/national.html#q/scope/state/ind/66/dist/23/char/87/time/3/viz/bar/fstate/33/locs/33,52,9/cmp/stcmp>

### Examples

```
#
shell.exec(state.health.url('CA'))
```

---

sum.moe

*Margin of Error for a Sum*

---

### Description

Estimates the margin of error (MOE) that characterizes uncertainty, for a sum of estimates, based on their MOE values.

### Usage

```
## S3 method for class 'moe'
sum(estimates, moes, include0 = FALSE)
```

### Arguments

<code>estimates</code>	A matrix or data.frame of numbers that are the estimates to be added across columns. Each row represents another place such as a Census block group, where the sum of all columns is calculated once for each place.
<code>moes</code>	A matrix of data.frame like estimates parameter, but with MOE values for the corresponding counts provided in estimates.
<code>include0</code>	Default is FALSE. If TRUE, based MOE on all data, even where estimate is zero, which gives MOEs that create conservatively wide confidence intervals.

Details

This is based on US Census Bureau recommendations for working with American Community Survey summary file data. This also works for differences (subtraction), not just sums. For example, one can estimate MOE for number of people with less than high school education in a block group, given estimates and margins of error for the count whose educational attainment is no school, first grade, second grade... 11th grade.

Value

Returns margin(s) of error same shape as parameter estimates

See Also

[pct.moe](#)

Examples

```
povknownratio <- c(500, 2000, 1500); povknownratio.m <- c(100, 300, 250)
pov2plus <- c(300, 1000, 1400); pov2plus.m <- c(100, 300, 200)
lowinc <- povknownratio - pov2plus
lowinc.m <- sum.moe(cbind(povknownratio, pov2plus), cbind(povknownratio.m, pov2plus.m))
cbind(lowinc=lowinc, MOE=lowinc.m, PCTMOE=round(lowinc.m/lowinc,2))
```

---

url.census	<i>See webpage with data on Census unit(s) from American Fact Finder</i>
------------	--

---

Description

DRAFT CODE to see webpage with table of Census Bureau data via AFF WITHOUT NEEDING API KEY

Usage

```
url.census(fips, censustable = "P2", censusfile = "DEC/10_PL",
  launch = TRUE)
```

Arguments

fips	vector of FIPS
censustable	'P2' by default but can be another table code. e.g., see <a href="http://factfinder.census.gov/faces/affhelp/jsf/pages/metadata.xhtml?lang=en&amp;type=dataset&amp;id=dataset.en.DEC_10_SF1">http://factfinder.census.gov/faces/affhelp/jsf/pages/metadata.xhtml?lang=en&amp;type=dataset&amp;id=dataset.en.DEC_10_SF1</a>
censusfile	'DEC/10_PL' by default. Also see 'DEC_10_SF1' for example.
launch	TRUE by default, whether to open page in web browser

## Details

For information on FIPS codes, see <http://www.census.gov/geo/reference/ansi.html>, and also see <https://www.census.gov/geo/reference/geoidentifiers.html>.

For links to AFF, see [http://factfinder2.census.gov/files/AFF\\_deep\\_linking\\_guide.pdf](http://factfinder2.census.gov/files/AFF_deep_linking_guide.pdf)  
e.g. to get 1 block census 2010 pop, for block fips 360610127001000 : [http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10\\_SF1/P1/1000000US360610127001000](http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10_SF1/P1/1000000US360610127001000)

e.g. to get 1 block census 2010 RACE/ETH/NHWA, for block fips 360610127002001 : [http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10\\_PL/P2/1000000US360610127002001](http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10_PL/P2/1000000US360610127002001)

Notice the second one gets P2 from 10\_PL not 10\_SF1, because P2 in SF1 means something different !!!

# TO GET RACE/ETHNICITY CENSUS 2010 COUNTS ON ONE BLOCK:

# block fips 360610127002001

# [http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10\\_PL/P2/1000000US360610127002001](http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10_PL/P2/1000000US360610127002001)

# TO GET RACE/ETHNICITY CENSUS 2010 COUNTS ON TWO BLOCKS:

# [http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10\\_PL/P2/1000000US360610127002001|1000000US360610127002000](http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10_PL/P2/1000000US360610127002001|1000000US360610127002000)

## Value

Can open a webpage. Returns vector of URL(s) as character

## See Also

[url.censusblock](#) and [urls.countyhealthrankings](#) (see <http://ejanalysis.github.io/countyhealthrankings>)

## Examples

```
myfips <- '360610127002001'
url.census(myfips, launch=FALSE)
myfips <- c('360610127002001', '360610127002000')
url.census(myfips, launch=FALSE)
```

---

url.censusblock

*See webpage with data on Census block(s)*

---

## Description

DRAFT CODE TO see table webpage of BLOCK DATA VIA AFF WITHOUT NEEDING API KEY

## Usage

```
url.censusblock(fips, censustable = "P2", censusfile = "DEC/10_PL",
  launch = TRUE)
```

## Arguments

fips	vector of FIPS
censustable	'P2' by default but can be another table code. e.g., see <a href="http://factfinder.census.gov/faces/affhelp/jsf/pages/metadata.xhtml?lang=en&amp;type=dataset&amp;id=dataset.en.DEC_10_SF1">http://factfinder.census.gov/faces/affhelp/jsf/pages/metadata.xhtml?lang=en&amp;type=dataset&amp;id=dataset.en.DEC_10_SF1</a>
censusfile	'DEC/10_PL' by default. Also see 'DEC_10_SF1' for example.
launch	TRUE by default, whether to open page in web browser

## Details

For information on FIPS codes, see <http://www.census.gov/geo/reference/ansi.html>, and also see <https://www.census.gov/geo/reference/geoidentifiers.html>.

For links to AFF, see [http://factfinder2.census.gov/files/AFF\\_deep\\_linking\\_guide.pdf](http://factfinder2.census.gov/files/AFF_deep_linking_guide.pdf)  
e.g. to get 1 block census 2010 pop, for block fips 360610127001000 : [http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10\\_SF1/P1/1000000US360610127001000](http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10_SF1/P1/1000000US360610127001000)

e.g. to get 1 block census 2010 RACE/ETH/NHWA, for block fips 360610127002001 : [http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10\\_PL/P2/1000000US360610127002001](http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10_PL/P2/1000000US360610127002001)

Notice the second one gets P2 from 10\_PL not 10\_SF1, because P2 in SF1 means something different !!!

# TO GET RACE/ETHNICITY CENSUS 2010 COUNTS ON ONE BLOCK:

# block fips 360610127002001

# [http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10\\_PL/P2/1000000US360610127002001](http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10_PL/P2/1000000US360610127002001)

# TO GET RACE/ETHNICITY CENSUS 2010 COUNTS ON TWO BLOCKS:

# [http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10\\_PL/P2/1000000US360610127002001|1000000US360610127002000](http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10_PL/P2/1000000US360610127002001|1000000US360610127002000)

## Value

Can open a webpage. Returns vector of URL(s) as character

## See Also

[urls.countyhealthrankings](#) (see <http://ejanalysis.github.io/countyhealthrankings>)

## Examples

```
myfips <- '360610127002001'
url.censusblock(myfips, launch=FALSE)
myfips <- c('360610127002001', '360610127002000')
url.censusblock(myfips, launch=FALSE)
```

---

url.open

*Launch a web browser to go to a URL, like browseURL does*

---

## Description

Just the same as [browseURL](#)

**Usage**

```
url.open(myurl)
```

**Arguments**

myurl                      required character element, URL to open

**Value**

Just launches browser to open URL

**See Also**

[browseURL](#), and [shell.exec](#) which this uses

---

url.qf	<i>US Census Quickfacts Webpage URL</i>
--------	---

---

**Description**

Get URL for webpage that provides basic demographic information from United States Census Bureau.

**Usage**

```
url.qf(fips = "", launch = TRUE)
```

**Arguments**

launch                      TRUE by default, whether to open page in web browser (max=1st 3 URLs opened)

FIPS                        Vector of numeric or character class, required. Can be state FIPs as number or character, for example.

**Details**

For information on FIPS codes, see <http://www.census.gov/geo/reference/ansi.html>, and also see <https://www.census.gov/geo/reference/geoidentifiers.html>

#####

If FIPS provided is 10 digits long, assume it is a tract missing a leading zero on the state portion (should have 11 characters).

If FIPS provided is 11 digits long, assume it is a tract (correctly 11 characters), not simply a block group FIPS missing a leading zero (block group FIPS would correctly would have 12 characters).

If FIPS provided is 12 digits long, assume it is a block group (correctly 12 characters).

If FIPS provided is 13 digits long, it is a block group.

If FIPS provided is 14 OR 15 digits long, assume it is a block. But that will not work in this function.

If FIPS is none of the above, return a default URL

```
# NOTES:
#
# URL FORMATS FOR QUICKFACTS REPORT ON A COUNTY FROM CENSUS QUICK-
# FACTS SITE:
#
# A WHOLE STATE: # http://quickfacts.census.gov/qfd/states/01000.html # WHERE
# 01000 = STATE 2-DIGIT FIPS, PLUS 3 ZEROES # A SINGLE COUNTY: # e.g., urlhttp://quickfacts.census.gov/qfd/states/XX/YYYYY.html # WHERE XX = STATE
# URL = # http://quickfacts.census.gov/qfd/states/XX/YYYYY.html # WHERE XX = STATE
# 2-DIGIT FIPS # WHERE YYYYY = STATE 2-DIGIT FIPS AND COUNTY 3-DIGIT FIPS = 5
# DIGITS TOTAL
```

### Value

Returns table of FIPS, geographic scale, and URL

### See Also

[clean.fips1215](#), [get.fips.bg](#), [get.fips.tract](#), [get.fips.county](#), [get.fips.st](#) to get partial FIPS from longer FIPS, or [get.name.county](#), [get.name.state](#) to extract name from longer Census name, or [get.state.info](#), [get.county.info](#), [get.epa.region](#) to look up info such as FIPS, state abbreviation, statename, countyname, or region based on FIPS or name.

### Examples

```
url.qf( c( '011030001003001', '011030001003', '01103000100', '01005', 1,
          c(8:10), 99999) )
\donotrun{
url.qf( c( '011030001003001', '011030001003', '01103000100', '01005', 1,
          ejanalysis::get.state.info()[ , 'FIPS.ST'], 99999) )
}
```

---

write.RR.tables

*Save RR tables to disk, one per zone.*

---

### Description

This function breaks up a 3-dimensional table of RR values (e.g., by demographic group, by environmental indicator, by geographic zone), and saves data for each zone to a 2-dimensional table in a file. Requires table in format provided by [RR](#) and related functions.

### Usage

```
write.RR.tables(my.RR.table, folder = getwd())
```

### Arguments

my.RR.table	Required RR table from function such as <a href="#">RR</a>
folder	Optional directory, default is current working directory. Specifies where to save files.



**Value**

Returns the file names as a character vector, after saving them locally. Saves one file per zone, with rownames and header. Format is one demographic group per row and one environmental indicator per column, plus max per row and max per column

**See Also**

- [ej.indexes](#) for local contribution to a variety of overall disparity metrics such as excess risk
- [RR](#) to calculate overall disparity metric as relative risk (RR), ratio of mean environmental indicator values across demographic groups
- [RR.table](#) to create 3-D table of RR values, by demographic group by environmental indicator by zone
- [RR.table.sort](#) to sort existing RR table
- [RR.table.add](#) to add zone(s) to existing RR table
- [write.RR.tables](#) to write a file with a table or RR by indicator by group
- [ej.added](#) to find EJ Index as local contribution to sum of EJ Indexes
- [RR.cut.if.gone](#) to find local contribution to RR
- [RR.if.address.top.x](#) to find how much RR would change if top-ranked places had different conditions
- [rrfv](#) for obsolete attempt to vectorize rrf
- [rrf](#) for older simpler function for RR of one indicator for one demographic group
- [pop.ecdf](#) to compare plots of cumulative frequency distribution of indicator values by group

**Examples**

```
# RRS.REGION <- RR.table(mydat=bg, Enames=names.e, Dnames=c(names.d, names.d.subgroups.pct), popcolname='pop')
# write.RR.tables(RRS.REGION)
```

# Index

## \*Topic **EJ**

- flagged.by, 15
- flagged.only.by, 16
- RR.table, 57
- RR.table.add, 58
- RR.table.sort, 59

## \*Topic **datasets**

- names.dvars, 39
- names.ejvars, 39
- names.evars, 40

assign.map.bins, 3, 3, 5, 7, 28, 34, 35, 37, 38  
assign.pctiles, 3, 4, 5, 7, 13, 28, 34, 35, 37, 38

assign.pctiles.alt2, 3, 5, 6, 7, 28, 34, 35, 37, 38

browseURL, 70, 71

clean.fips, 7, 17, 18  
clean.fips1215, 8, 9, 9, 20–27, 30, 72

demographic-variables (names.dvars), 39  
Dlist (names.dvars), 39

Ecdf, 42, 46, 47  
EJ-variable-names (names.ejvars), 39  
ej.added, 9, 51, 54, 55, 58–64, 73  
ej.indexes, 10, 13, 51, 54, 55, 58–64, 73  
ejanalysis, 13  
ejanalysis-package (ejanalysis), 13  
ejRRadded, 14  
Elist (names.evars), 40  
environmental-variable-names  
(names.evars), 40

flagged, 14  
flagged.by, 15  
flagged.only.by, 16

geofips, 17, 18  
geotype, 18  
get.county.info, 8, 9, 13, 19, 20–27, 30, 72  
get.epa.region, 8, 9, 20, 21, 21, 22–27, 30, 72

get.fips.bg, 8, 9, 20–22, 22, 23–27, 30, 72  
get.fips.county, 8, 9, 20–22, 22, 23–27, 30, 72  
get.fips.etc, 23  
get.fips.st, 8, 9, 18, 20–24, 24, 25–27, 30, 72  
get.fips.tract, 8, 9, 20–25, 25, 26, 27, 30, 72  
get.name.county, 8, 9, 20–25, 25, 26, 27, 30, 72  
get.name.state, 8, 9, 20–26, 26, 27, 30, 72  
get.pctile, 3, 5, 7, 28, 28, 34, 35, 37, 38  
get.state.info, 8, 9, 18, 20–27, 29, 30, 72

Kolm, 31  
KolmPollak, 31

lookup.pctile, 4, 5, 7, 28, 32, 34, 35, 37, 38

make.bin.cols, 4, 5, 7, 28, 33, 34, 35, 37, 38  
make.bin.pctile.cols, 3, 5, 7, 13, 28, 34, 34, 35–38  
make.bin.pctile.cols.byzone, 36  
make.pctile.cols, 3–5, 7, 28, 34, 35, 36, 37, 38  
make.pctile.cols.alt2, 3, 5, 7, 28, 34, 35, 37, 37, 38

names.d (names.dvars), 39  
names.dvars, 39  
names.e (names.evars), 40  
names.ej (names.ejvars), 39  
names.ejvars, 39  
names.evars, 40

pct.above, 28  
pct.moe, 41, 68  
pop.cdf, 42, 42, 44, 45, 47  
pop.cdf.density, 42, 43, 44, 45, 47  
pop.cdf2, 42, 44, 44, 45, 47  
pop.ecdf, 13, 42, 44, 45, 46, 47, 51, 55, 58–64, 73

rollup, 48, 49  
rollup.pct, 49

RR, [13](#), [42](#), [47](#), [50](#), [51](#), [54](#), [55](#), [57–64](#), [72](#), [73](#)  
RR.cut.if.gone, [51](#), [52](#), [55](#), [58–64](#), [73](#)  
RR.if.address.top.x, [51](#), [53](#), [54](#), [55](#), [58–64](#),  
[73](#)  
RR.means, [54](#)  
RR.plot, [56](#)  
RR.table, [51](#), [55](#), [57](#), [58–64](#), [73](#)  
RR.table.add, [51](#), [55](#), [58](#), [58](#), [59–64](#), [73](#)  
RR.table.sort, [51](#), [55](#), [58](#), [59](#), [59](#), [60–64](#), [73](#)  
RR.table.view, [60](#)  
RR.table.vs.us, [61](#)  
rrf, [51](#), [55](#), [58–62](#), [62](#), [63](#), [64](#), [73](#)  
rrfv, [51](#), [55](#), [58–64](#), [64](#), [73](#)  
RRS, [57](#)  
  
shell.exec, [71](#)  
state.health.url, [13](#), [65](#)  
sum.moe, [41](#), [67](#)  
  
url.census, [68](#)  
url.censusblock, [69](#), [69](#)  
url.open, [70](#)  
url.qf, [71](#)  
urls.countyhealthrankings, [69](#), [70](#)  
  
weighted.hist, [42](#)  
write.pctiles, [4](#), [5](#), [7](#), [28](#), [32](#), [34](#), [35](#), [37](#), [38](#)  
write.pctiles.by.zone, [4](#), [5](#), [7](#), [28](#), [32](#), [34](#),  
[35](#), [37](#), [38](#)  
write.RR.tables, [51](#), [55](#), [58–64](#), [72](#), [73](#)  
write.wtd.pctiles, [4](#), [5](#), [7](#), [28](#), [32](#), [34](#), [35](#),  
[37](#), [38](#)  
write.wtd.pctiles.by.zone, [4](#), [5](#), [7](#), [28](#), [32](#),  
[34](#), [35](#), [37](#), [38](#)  
wtd.Ecdf, [5](#)