Package 'ejscreen'

December 27, 2019

Title EJSCREEN Tools for US EPA Environmental Justice (EJ) Mapping and Screening

Description Data and tools related to the United States Environmental Protection Agency's screening and mapping tool for environmental justice, EJSCREEN. For any imported/suggested packages not on CRAN, see http://ejanalysis.github.io

Version 1.0.0

Date 2020-01-01

Imports analyze.stuff,
Hmisc,
ejanalysis,
data.table,
devtools

Suggested ACSdownload,
proxistat

Depends R (>= 3.1.2)

URL http://ejanalysis.github.io, https://github.com/ejanalysis/ejscreen, http://www.ejanalysis.com/

BugReports https://github.com/ejanalysis/ejscreen/issues

RoxygenNote 7.0.2

License MIT + file LICENSE

Repository GitHub

Author info@ejanalysis.com

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

NeedsCompilation no

LazyData true

Encoding UTF-8

R topics documented:

`addFIPScomponents` *Add col for each component of longer FIPS Given a data.frame with FIPS col that is the full state county tract blockgroup FIPS returns the data.frame with extra columns up front, with components of FIPS.*

Description

Add col for each component of longer FIPS

Given a data.frame with FIPS col that is the full state county tract blockgroup FIPS returns the data.frame with extra columns up front, with components of FIPS.

Usage

```
addFIPScomponents(bg)
```

Arguments

bg	Data.frame with a character column called FIPS

Value

Returns the whole data.frame with new columns in front: 'FIPS', 'FIPS.TRACT', 'FIPS.COUNTY', 'FIPS.ST', 'ST', 'statename', 'REGION'

---

bg17 *This is the 2017 version of the EJSCREEN dataset plus lat lon and countynames, etc., minus some cols and rows*

Description

Note the 2018 version of EJSCREEN (released late 2018) actually uses ACS2016, which is from 2012-2016 (released late 2017). Note the 2019 version of EJSCREEN (released late 2019) actually uses ACS2017, which is from 2013-2017 (released late 2018).

This data set is the EJSCREEN 2017 dataset from the ftp site but with fields renamed for easier use in the ejscreen package, and some columns dropped (svi6-related, and the 2 alternative versions of an EJ Index) and some fields added (lat lon for bg centroids, flagged if any of EJ indexes above 80th percentile in US), and state name and state abbrev and county name and FIPS for tract, county, state, minus a handful of rows (that had NA values in FIPS? These are left in the bg18 data for now.)

Format

data.frame

Details

Previously the data.frame was bg not bg17, and the file was called bg2017_plus_latlon_etc_minus_some_fields_minus_N

---

bg18 *The 2018 version of EJSCREEN data, plus lat lon, countynames, etc., minus some nonessential fields*

Description

Note the 2018 version of EJSCREEN (released late 2018) actually uses ACS2016, which is from 2012-2016 (released late 2017). Note the 2019 version of EJSCREEN (released late 2019) actually uses ACS2017, which is from 2013-2017 (released late 2018).

This data set is the EJSCREEN 2018 dataset from the ftp site but with fields renamed for easier use in the ejscreen package, and some columns dropped (svi6-related, and the 2 alternative versions of an EJ Index) and some fields added (lat lon for bg centroids, flagged if any of EJ indexes above 80th percentile in US), and state name and state abbrev and county name and FIPS for tract, county, state, BUT NOT REMOVING a handful of rows removed from the bg17 data (that had NA values in FIPS.ST)

Format

data.frame with 220,333 rows (block groups) and 118 columns

Details

bg18 was created for this package as follows:

```
bg18 <-ejscreen.download(yr = 2018,addflag = TRUE)
# OR IF ALREADY DOWNLOADED AND UNZIPPED,JUST DO THIS:
# bg18 <-ejscreen.download(justreadname = 'EJSCREEN_Full_USPR_2018.csv',addflag = TRUE)

# Starts by reading in 368 columns from csv, then has about 373 columns
# after ejscreen.download, which renames fields, drops an ID field,
# and adds fields called FIPS.TRACT, FIPS.COUNTY, FIPS.ST, countyname, flagged.

# Then for this package, got rid of some nonessential fields:
# (But note svi6 fields - which combine all 6 demog indicators not just 2 - were named in names.d)

bg18 <-bg18[ ,!grepl(pattern = 'svi6',x = names(bg18))]
bg18 <-bg18[ ,!grepl(pattern = 'pctile\\.text',x = names(bg18))]
bg18 <-bg18[ ,!grepl(pattern = 'EJ\\.PCT',x = names(bg18))]
bg18 <-bg18[ ,!grepl(pattern = 'EJ\\.BURDEN',x = names(bg18))]
bg18 <-bg18[ ,names(bg18) != 'ID_1']
bg18 <-bg18[ ,names(bg18) != 'Shape_Length']
# The ID_1 field is like FIPS but NA where some data missing

# Then added lat, lon fields for block group centroids, via bg.pts from proxistat package:
bg18 <-merge(bg18,bg.pts[ ,c('FIPS','lat')],by.x = 'FIPS',by.y = 'FIPS',all.x = TRUE,all.y
= FALSE)
bg18 <-merge(bg18,bg.pts[ ,c('FIPS','lon')],by.x = 'FIPS',by.y = 'FIPS',all.x = TRUE,all.y
= FALSE)

save(bg18,file = 'bg18.rdata')
plot(bg18$lon[bg18$lon < -50],bg18$lat[bg18$lon < -50],pch = '.')

# sum(is.na(bg18$FIPS.ST))
# [1] 13
```

---

bg19 *The 2019 version of EJSCREEN data, plus lat lon, countynames, etc., minus some nonessential fields*

Description

Note the 2018 version of EJSCREEN (released late 2018) actually uses ACS2016, which is from 2012-2016 (released late 2017). Note the 2019 version of EJSCREEN (released late 2019) actually uses ACS2017, which is from 2013-2017 (released late 2018). This data set is the EJSCREEN dataset from the ftp site but with fields renamed for easier use in the ejscreen package, and some columns dropped (svi6-related, and the 2 alternative versions of an EJ Index) and some fields added (lat lon for bg centroids, flagged if any of EJ indexes above 80th percentile in US), and state name and state abbrev and county name and FIPS for tract, county, state, BUT NOT REMOVING a handful of rows removed from the data (that had NA values in FIPS.ST) Also, it does not include the lookup tables of percentiles for USA, Regions, States, which are in the gdb.

Format

data.frame with 220,333 rows (block groups) and 118 columns (may vary by year)

Details

It was created for this package as follows:

```
require(ejscreen); require(ACSdownload)
require(proxistat) # for the lat lon of each block group
require(analyze.stuff); require(ejanalysis); require(readr)

bg <-ejscreen.download(yr = 2018,addflag = TRUE)
# OR IF ALREADY DOWNLOADED AND UNZIPPED (as with 2019 dataset),JUST DO THIS:
```

2019 version of EJSCREEN downloaded 2019-08-27 from public FTP site
as gdb format (zipped)
EJSCREEN_V2019.gdb.zip
then unzipped to
EJSCREEN_V2019.gdb
Opened EJSCREEN_V2019.gdb file in ESRI's ArcGIS

Opened attribute table EJSCREEN_Full
Exported all records as text format to a file named
EJSCREEN_Full_2019_Export_Output.csv

That has this format:

OBJECTID,ID,ACSTOTPOP,ACSIPOVBAS,ACSEDUCBAS,ACSTOTHH,ACSTOTHU,MINORPOP,MINORPCT,LC
1,010010201001,692,692,441,300,300,58,0.083815028901734,203,0.293352601156069,86,0.195011337868480,12,0.0
116.176110108974143,-39.018093266051089,0.278663068700000,49.377031606578001,0.788051737455781,91.0159
33.691071931602380,-11.315247047154775,37.844999999999828,25.308679213907656,0.054689306358381,0.03657
32.373991352595830,-10.872901604340603,36.365530465349963,24.319290394798056,0.052551344603107,0.03514
5736.431460780102498,-1926.597624426212860,6443.702624658421882,4309.198115399539347,9.31170899517113
91.552785422231196,-30.748276190523271,102.840751737979318,68.774305611141415,0.148613803089565,0.0993
10573.873229999833711,-3551.266878229423583,11877.574961157059079,7943.076615961609605,17.1641256664
0.000000000000000,-0.000000000000000,0.000000000000000,0.000000000000000,0.000000000000000,0.000000000
8.241713214253599,-2.768003977445550,9.257872151608664,6.191161751577060,0.013378427964752,0.00894676
9.898284614015102,-3.324368425498922,11.118689900335962,7.435575530889657,0.016067470954243,0.0107450
7.618576564102895,-2.558721673964495,8.557906102062088,5.723062499136828,0.012366916332460,0.00827032
4661.186377971540423,-1565.473354681477304,5235.885602941166326,3501.475733268755903,7.56630867477047
1161.544048564898276,-390.108034922326908,1304.756186065572138,872.550026833011771,1.885485818013833,
```
# bg <-ejscreen.download(justreadname = 'EJSCREEN_Full_2019_Export_Output.csv',addflag
= TRUE)

# Starts by reading in 368 columns from csv, then has about 373 columns
# after ejscreen.download, which renames fields, drops an ID field,
# and adds fields called FIPS.TRACT, FIPS.COUNTY, FIPS.ST, countyname, flagged.

# Then for this package, got rid of some nonessential fields:
# (But note svi6 fields - which combine all 6 demog indicators not just 2 - were named in names.d)
```

```
bg <-bg[ ,!grepl(pattern = 'svi6',x = names(bg))]
bg <-bg[ ,!grepl(pattern = 'pctile\\.text',x = names(bg))]
bg <-bg[ ,!grepl(pattern = 'EJ\\.PCT',x = names(bg))]
bg <-bg[ ,!grepl(pattern = 'EJ\\.BURDEN',x = names(bg))]
bg <-bg[ ,names(bg) != 'ID_1']
bg <-bg[ ,names(bg) != 'Shape_Length']
# The ID_1 field is like FIPS but NA where some data missing

# Then added lat, lon fields for block group centroids, via bg.pts from proxistat package:
bg <-merge(bg,bg.pts[ ,c('FIPS','lat')],by.x = 'FIPS',by.y = 'FIPS',all.x = TRUE,all.y
= FALSE)
bg <-merge(bg,bg.pts[ ,c('FIPS','lon')],by.x = 'FIPS',by.y = 'FIPS',all.x = TRUE,all.y
= FALSE)

plot(bg$lon[bg$lon < -50],bg$lat[bg$lon < -50],pch = '.')

# sum(is.na(bg$FIPS.ST))
# [1] 13

# THEN USE A NAME SPECIFIC TO THE YEAR:
bg19 <-bg
save(bg19,file = 'bg19.rdata')
```

`change.fieldnames.ejscreen.csv`

Change colnames of csv file on EJSCREEN FTP site to nicer colnames

---

Description

Just a wrapper to help easily change colnames used in csv file on EJSCREEN FTP site into friendlier, preferred colnames for work in R. Uses `change.fieldnames`

Usage

```
change.fieldnames.ejscreen.csv(mynames)
```

Arguments

`mynames`	A character vector of colnames from a data.frame, like names(mydf). No default.

Value

Returns a character vector of colnames, same length as input parameter

See Also

`ejscreenformulas` `ejscreen.acs.rename` `change.fieldnames`

Examples

```
## Not run:
gdbtable <- ejscreen.download()
names(gdbtable) <- change.fieldnames.ejscreen.csv(names(gdbtable))

## End(Not run)
```

`ejformula` *See formula(s) used for EJSCREEN variable(s)*

Description

Just a convenient way to look at the formula(s) used to create one or more variables in EJSCREEN.

Usage

```
ejformula(fieldname = "all", decreasing = NA, dropNA = TRUE, recursive = FALSE)
```

Arguments

`fieldname`	Optional, character vector specifying variable(s) in `ejscreenformulas$Rfieldname`, default is all `ejscreenformulas$Rfieldname` that are not NA values.
`decreasing`	Optional, passed to `sort` except default is not sorted (just the order that exists in `ejscreenformulas`)
`dropNA`	Be careful: Optional, default is TRUE. If TRUE, returns only formulas that are not NA values. If FALSE, and decreasing is not specified (sorting drops NA values here), returns vector the same length as `fieldname` (unless `recursive = TRUE`)
`recursive`	Optional, default is FALSE. If TRUE, returns also returns formula(s) for variable(s) found on right hand side of formula(s), i.e. those used to create specified variable(s)

Value

Character vector of the formula(s) used to calculate the specified variable, in `ejscreenformulas`

See Also

`ejscreenformulas`

Examples

```
ejformula('VSI.eo')
ejformula(c('pctmin', 'pctlowinc'))
ejformula('VSI.eo', recursive = TRUE)
ejformula()
```

ejscreen — *Tools for EJSCREEN, US EPA's Environmental Justice (EJ) Screening and Mapping Tool*

Description

This R package provides tools related to environmental justice (EJ) analysis, specifically related to the United States Environmental Protection Agency (EPA) screening and mapping/GIS tool called EJSCREEN. See http://www.epa.gov/ejscreen This package facilitates development of the EJSCREEN dataset, based on user-provided environmental indicators. The resulting dataset is a data.frame that contains data on demographics (e.g., percent of residents who are low-income) and user-provided local environmental indicators (e.g., an air quality index), and calculated indicators called EJ Indexes, which combine environmental and demographic indicators. The dataset also provides each key indicator as a national population-percentile that represents what percentage of the US population have equal or lower raw values for the given indicator. The dataset has one row per spatial location (e.g., Census block group).

Details

Key functions include

- ejscreen.download To download the raw data from the FTP site.
- ejscreen.create To create a dataset of demographic indicators, EJ indexes, etc. starting with your own environmental indicators and taking demographic raw data from the American Community Survey (ACS).
- ejscreen.lookuptables To create the file that shows percentiles for each indicator
- Various functions from the **ejanalysis** package are also relevant.

References

http://www.epa.gov/ejscreen
http://ejanalysis.github.io
http://www.ejanalysis.com/

ejscreen.acs.calc — *Create Calculated EJSCREEN Variables*

Description

Use specified formulas to create calculated, derived variables such as percent low income. Relies upon calc.fields from **analyze.stuff** package.

Usage

```
ejscreen.acs.calc(
  bg,
  folder = getwd(),
  keep.old,
  keep.new,
  formulafile,
  formulas
)
```

Arguments

bg	Data.frame of raw demographic data counts, and environmental indicators, for each block group, such as population or number of Hispanics.
folder	Default is getwd(). Specifies path for where to read from (if formulafile specified) and write to.
keep.old	Vector of variables names from names(bg), indicating which to return (retain, not drop). Default is to keep only the ones that match the list of default names in this code. Or this can be simply 'all' which means keep all input fields.
keep.new	Vector of variables names of new created variables, indicating which to return (retain, not drop). Default is to keep a specific list of fields (see source code). Or this can be simply 'all' which means keep all new fields.
formulafile	Name of optional csv file with column called formula, providing R syntax formulas as character fields. If not specified, function loads this as data(ejscreenformulas). Example of one formula: 'pctunder5 <- ifelse(pop==0,0, under5/pop)' Use a result of zero in cases where the denominator is zero, to avoid division by zero. For example, the formula `'pctmin <-ifelse(pop==0,0,as.numeric(mins ) / pop)'` indicates that percent minority is calculated as the ratio of number of minorities over total population of a block group, but is set to zero if the population is zero.
formulas	Options vector of formulas as character strings that contain R statements in the form "var1 <- var2 + var3" for example. Either formulafile or formulas can be specified (or neither) but not both (error). Formulas should be in the same format as a formulafile field or the contents of ejscreenformulas (via data(ejscreenformulas) or lazy loading like x <- ejscreenformulas).

Value

Returns a data.frame with some or all of input fields (those in keep.old), plus calculated new fields (those in keep.new).

Examples

```
set.seed(99)
envirodata=data.frame(FIPS=analyze.stuff::lead.zeroes(1:1000, 12),
  air=rlnorm(1000), water=rlnorm(1000)*5, stringsAsFactors=FALSE)
demogdata=data.frame(FIPS=analyze.stuff::lead.zeroes(1:1000, 12),
  pop=rnorm(n=1000, mean=1400, sd=200), mins=runif(1000, 0, 800),
  num2pov=runif(1000, 0,500), stringsAsFactors=FALSE)
demogdata$povknownratio <- demogdata$pop
x=ejscreen.acs.calc(bg=demogdata)
```

ejscreen.acs.rename *Rename Fields of ACS Data for Use in EJSCREEN*

Description

Start with raw counts from demographic survey data, and environmental data, and rename fields to use friendly variable names.

Usage

```
ejscreen.acs.rename(acsraw, folder = getwd(), formulafile)
```

Arguments

acsraw	Data.frame of raw data counts for each block group, such as population or number of Hispanics.
folder	Default is getwd(). Specifies path for where to read from (if formulafile specified) and write to.
formulafile	Default if this is blank is to use data(ejscreenformulas). Otherwise filename must be specified. If not specified, function loads this as data().

Value

Returns a data.frame with some or all of input fields, plus calculated new fields.

See Also

ejscreenformulas change.fieldnames.ejscreen.csv change.fieldnames

ejscreen.acsget *Download ACS tables EJSCREEN uses, but with more race ethnicity poverty details*

Description

Note the 2018 version of EJSCREEN (released late 2018) actually uses ACS2016, which is from 2012-2016 (released late 2017). Note the 2019 version of EJSCREEN (released late 2019) actually uses ACS2017, which is from 2013-2017 (released late 2018).

Helper function used by ejscreen.create, but can be used if one wants to obtain the more detailed relevant ACS data. The EJSCREEN-related ACS tables have more of the detailed fields than the demographic data on the EJSCREEN FTP site, because the detailed fields are used to calculate the ones retained for EJSCREEN, such as percent non-hispanic black alone, percent hispanic, percent poor (below 1x poverty line) not just percent low income (below 2x poverty line), etc.

Usage

```
ejscreen.acsget(
  end.year = "2017",
  tables = c("B01001", "B03002", "B15002", "C17002", "B25034"),
  base.path = getwd(),
  data.path = file.path(base.path, "acsdata"),
  output.path = file.path(base.path, "acsoutput"),
  vars = "all",
  sumlevel = "bg",
  write.files = TRUE,
  ...
)
```

Arguments

end.year	optional character year like 2017 specifying last of 5 years of ACS summary file
tables	Default is the ones needed for EJSCREEN - character vector list of Census data tables like B01001
base.path	optional, default is working directory; folder in which data.path and output.path subfolders are or will be created
data.path	see get.acs
output.path	see get.acs
vars	Default here is 'all' vars which is more than what ejscreen.create keeps. (or can be a vector of things like 'B01001')
sumlevel	Default here is just bg but see get.acs
write.files	Default here is TRUE but see get.acs
...	passed to get.acs

Value

list of data.frames, default is just block group not tracts, unlike results of get.acs

---

ejscreen.create *Create EJSCREEN Dataset from Environmental Indicators*

Description

Start with raw environmental indicator data, and create (or replicate) a full EJSCREEN dataset. The source code also contains comments with an outline of steps involved. Note that rather than using this function, one can instead download the gdb file from the EJSCREEN FTP site and open it in ESRI ArcGIS and export the attribute tables as text files in csv format.

Usage

```
ejscreen.create(
  e,
  acsraw,
  folder = getwd(),
  keep.old,
  formulas,
  mystates = "all",
  demogvarname0 = "VSI.eo",
  demogvarname1 = "VSI.svi6",
  wtsvarname = "pop",
  checkfips = TRUE,
  EJprefix0 = "EJ.DISPARITY",
  EJprefix1 = "EJ.BURDEN",
  EJprefix2 = "EJ.PCT",
  ejformulasfromcode = FALSE,
  ejtype = 1,
  demogvarname0suffix = "eo",
  demogvarname1suffix = "svi6",
  end.year,
  threshold = FALSE,
  cutoff = 0.8,
  thresholdfieldnames,
  ...
)
```

Arguments

e	Data.frame of raw data for environmental indicators, one row per block group, one column per indicator.
acsraw	Optional data.frame of raw demographic indicators. Downloaded if not provided as parameter.
folder	Optional, default is getwd(). Passed to get.acs if demog data must be downloaded. Passed to but not currently used by ejscreen.acs.rename which uses change.fieldnames in **analyze.stuff** package. Not currently passed to ejscreen.acs.calc which uses calc.fields in **analyze.stuff** package.
keep.old	optional vector of colnames from e that are to be used/returned. For nondefault colnames, this must be used.
formulas	optional, see ejscreen.acs.calc for details. Defaults are in ejscreenformulas$formula Note that if formulas is specified, ejformulasfromcode is ignored.
mystates	optional vector of 2-letter state abbreviations. Default is "all" which specifies all states plus DC (BUT NOT PR - we exclude PR so that calculating US percentiles works right)
demogvarname0	optional, default is 'VSI.eo' used as demographic indicator for EJ Indexes. Must be a colname in acsraw or created and kept by formulas.
demogvarname1	optional, default is 'VSI.svi6' used for alternative EJ Indexes. Must be a colname in acsraw or created and kept by formulas.
wtsvarname	optional, default is 'pop' used for weighted percentiles, etc. Must be a colname in acsraw or created and kept by formulas.

checkfips
: optional, default is TRUE. If TRUE, function checks to verify all FIPS codes appear to be valid US FIPS (correct number of characters, adding any leading zero needed, and checking the first five to ensure valid county). To use something other than actual US FIPS codes, set this to FALSE.

EJprefix0
: optional, default is 'EJ.DISPARITY' - specifies prefix for colnames of main EJ Indexes, with a period separating prefix from body of colname

EJprefix1
: optional, default is 'EJ.BURDEN' - specifies prefix for colnames of Alternative 1 version of EJ Indexes, with a period separating prefix from body of colname

EJprefix2
: optional, default is 'EJ.PCT' - specifies prefix for colnames of Alternative 2 version of EJ Indexes, with a period separating prefix from body of colname

ejformulasfromcode
: optional, default is FALSE. If TRUE, use EJ Index formulas built into this function instead of the EJ Index formulas in ejscreenformulas. The parameters such as demogvarname0 are only used if ejformulasfromcode=TRUE. Note that if formulas is specified, ejformulasfromcode is ignored.

ejtype
: optional, default is 1, defines which formula to use for ejindex if not using ejscreenformulas. See ej.indexes But note alt1 and alt2 still use type 5 and 6 ignoring ejtype.

demogvarname0suffix
: optional, default is 'eo' - specifies suffix for colnames of EJ Indexes based on demogvarname0, with a period separating body of colname from suffix

demogvarname1suffix
: optional, default is 'svi6' - specifies suffix for colnames of EJ Indexes based on demogvarname1, with a period separating body of colname from suffix

end.year
: optional to pass to get.acs (such as end.year='2013' – otherwise uses default year used by get.acs)

threshold
: optional, default is FALSE. Set to TRUE to add a column (called 'flagged') to results that is TRUE when one or more of certain percentiles (US EJ Index) in a block group (row) exceed cutoff. A field called flagged can also be added via flagged ejanalysis::flagged() or via ejscreen.download(addflag = TRUE)

cutoff
: optional, default is 0.80 (80th percentile). If threshold=TRUE, then cutoff defines the threshold against which percentiles are compared.

thresholdfieldnames
: optional, default is standard EJSCREEN EJ Indexes built into code. Otherwise, vector of character class fieldnames, specifying which fields to compare to cutoff if threshold=TRUE.

...
: optional extra parameters passed only to get.acs such as new.geo = FALSE, save.files = TRUE, write.files = TRUE

Details

**Note that if non-default fieldnames are used in e and/or acsraw, those must be specified in parameters including demogvarname0, demogvarname1, wtsvarname, keep.old (and could be reflected in prefix and suffix params as well).

Value

Returns a data.frame with full ejscreen dataset of environmental and demographics indicators, and EJ Indexes, as raw values, US percentiles, and text for popups. Output has one row per block group.

See Also

make.popup.d make.popup.e make.popup.ej ejscreen.lookuptables

Examples

```
## Not run:
set.seed(99)
envirodata=data.frame(FIPS=analyze.stuff::lead.zeroes(1:1000, 12),
  air=rlnorm(1000), water=rlnorm(1000)*5, stringsAsFactors=FALSE)
demogdata=data.frame(FIPS=analyze.stuff::lead.zeroes(1:1000, 12),
  pop=rnorm(n=1000, mean=1400, sd=200), mins=runif(1000, 0, 800),
  num2pov=runif(1000, 0,500), stringsAsFactors=FALSE)
demogdata$povknownratio <- demogdata$pop
# downloads ACS demographics and combines with user provided envirodata:
# bg1=ejscreen.create(envirodata, mystates=c('de','dc'))
# currently does not work for nonstandard colnames
#  unless keep.old used as follows (work in progress):
y=ejscreen.create(e=envirodata, acsraw=demogdata,
  keep.old = c(names(envirodata), names(demogdata)),
  demogvarname0 = 'pctmin', demogvarname1 = 'pctlowinc', wtsvarname = 'pop' )

## End(Not run)
```

---

ejscreen.download *Download the EJSCREEN Dataset for use in R*

Description

Download EJSCREEN dataset from FTP site, unzip if necessary, import to R as data.table, renaming fields with friendly colnames, optionally adding a flag field (see parameter called addflag).

Usage

```
ejscreen.download(
  folder = getwd(),
  yr = NULL,
  ftpurlbase = "ftp://newftp.epa.gov/EJSCREEN/",
  justreadname = NULL,
  addflag = FALSE,
  cutoff = 80,
  or.tied = TRUE
)
```

Arguments

folder	Optional path to folder (directory) where the file will be downloaded and unzipped. Default is current working directory.
yr	Default is latest available year found as a folder on the FTP site. That was 2018 as of 7/20/19, but would be updated to 2019 upon release of that version in late 2019. Default was 2017 as of 6/2018. Optional numeric year designating EJSCREEN version such as 2015, 2016, 2017, 2018, 2019.

ftpurlbase	Optional. Default is ftp://newftp.epa.gov/EJSCREEN/ and must have ending slash – for where to find the zipped data.
justreadname	Optional character file name - if specified, skips downloading and just tries to read csv found in folder.
addflag	Optional. Default is FALSE. If TRUE, it adds a field called flagged, which is TRUE if 1 or more of the EJ Indexes is at/above the cutoff US percentile.
cutoff	Optional. Default is 80. See addflag parameter.
or.tied	Optional. Default is TRUE, meaning at or above the cutoff. FALSE means above only. See addflag parameter.

Details

Not fully tested.
Each version of EJSCREEN uses updated environmental data and updated 5-year summary file estimates from the American Community Survey (ACS).
The 2015 version of EJSCREEN, released in mid 2015, was based on 2008-2012 ACS data, and was the first public version available for download.
The 2018 version of EJSCREEN, released in mid 2018, was based on 2012-2016 ACS 5-year summary file data that came out in Dec 2017.
The 2019 version of EJSCREEN, released in mid/late 2019, is based on 2013-2017 ACS 5-year data that came out in Dec 2018.
The 2014-2018 ACS 5-year data will be released starting December 19, 2019.

Value

Returns a data.frame with ejscreen dataset of environmental and demographics indicators, and EJ Indexes, as raw values, US percentiles, text for popups. Output has one row per block group.

Source

See http://www.epa.gov/ejscreen for more information, and see http://www.epa.gov/ejscreen/download-ejscreen-data or ftp://newftp.epa.gov/EJSCREEN for raw data.

See Also

ejscreen.create

Examples

```
 # bg18 <- ejscreen.download('~/Dropbox/EJSCREEN/R Analysis/2018 dataset EJSCREEN/')
## bg18 <- ejscreen.download('~/Dropbox/EJSCREEN/R Analysis/2018 dataset EJSCREEN/', justreadname = 'EJSCR
 # bg18 <- bg18[ , !grepl(pattern = 'pctile\.text', x = names(bg18))]
 # bg18 <- bg18[ , !grepl(pattern = 'svi6', x = names(bg18))]
 # setwd('~/Dropbox/EJSCREEN/R analysis/2018 dataset EJSCREEN')
 # save(bg18, file = 'bg18.rdata')
```

`ejscreen.lookuptables` *Create EJSCREEN Lookup Tables of Pop. Percentiles by Zone - WORK IN PROGRESS*

Description

*** Work in progress as of 2019. *** The Hmisc package provides the function called Hmisc::wtd.quantile(), but could recode to use analyze.stuff::wtd.pctiles ?

Start with raw environmental, demographic, and EJ indicator data, and write as csv files to disk a series of lookup tables that show population percentiles and mean values for each indicator.

Usage

```
ejscreen.lookuptables(
  x,
  weights = x$pop,
  cols,
  zonecols = c("ST", "REGION"),
  folder = getwd(),
  missingcode = NA
)
```

Arguments

`x`	Data.frame of indicators, one row per block group, one column per indicator.
`weights`	Weights for percentiles – Default is population count to provide population percentiles.
`cols`	Optional vector of colnames of x that need percentile lookup tables, or `all` which means all numeric fields in x. Default is a standard set of EJSCREEN fieldnames defined within this function (see source code).
`zonecols`	Optional. Must set to NULL if no zones wanted, because default is `c('ST','REGION')`, names of cols in x that contain zone codes, such as State names or Region numbers, used to create a lookup table file for each of the zonecols, with separate percentiles calculated within each zone.
`folder`	Default is `getwd()` - specifies where to save the csv files.
`missingcode`	Leave this unspecified if missing values are set to NA in the input data. Default is -9999999 (but if already NA then do not specify anything for this). The number or value in the input data that designates a missing value.

Details

Percentiles are calculated as exact values and then rounded down to the nearest 0-100 percentile. This calculates percentiles among only the non-NA values. In other words, people in places with missing data are excluded from the calculation. This means the percentile is the percent of people with valid data (i.e., not NA) who have a tied or lower value.

Value

Overall lookup table(s) as data.frame (but not zonal ones). Creates lookup tables saved as csv files to specified folder. One table for overall percentiles, and one for each of the zonecols (unless that is set to NULL).

Examples

```
## Not run:
# Try with a sample envt data set:
set.seed(99)
envirodata <- data.frame(FIPS=analyze.stuff::lead.zeroes(1:1000, 12),
  pm = runif(1000,5,20), o3 = runif(1000,3,50),
  air=rlnorm(1000), water=rlnorm(1000)*5, stringsAsFactors=FALSE)
demogdata <- data.frame(FIPS=analyze.stuff::lead.zeroes(1:1000, 12),
  pop = rlnorm(1000, meanlog = log(1000), sdlog = 1), stringsAsFactors=FALSE)
x <- ejscreen.lookuptables(envirodata, weights=demogdata$pop, cols='all', zonecols=NULL)
x

## End(Not run)
```

`ejscreen.rollup` *Aggregate EJSCREEN Dataset at Lower Resolution (e.g., Tracts)*

Description

Start with full EJSCREEN dataset at one resolution (typically block groups), and create aggregated data at a higher geographic scale (e.g., tracts or counties)

Usage

```
ejscreen.rollup(
  bg,
  fipsname = "FIPS.TRACT",
  scalename = "tracts",
  enames,
  folder = getwd(),
  sumnames,
  avgnames,
  wts,
  acsnames,
  ...
)
```

Arguments

`bg`	Data.frame of raw data for environmental and demographic counts, one row per block group typically, one column per indicator.
`fipsname`	Default is 'FIPS.TRACT' - specifies colname of unique ID field FIPS used to group by. Can be FIPS.TRACT, FIPS.COUNTY, FIPS.ST, or REGION in default dataset.
`scalename`	***Not used. Default is 'tracts' - specifies text to use in naming the saved file.
`enames`	Default is `names.e`, the colnames of raw envt indicators in bg
`folder`	***Not used. Optional, default is getwd().
`sumnames`	Default is a vector of colnames in bg, those which should be rolled up as sums (e.g., sum of all block group population counts in the tract)

avgnames	Default is a vector of colnames in bg, those which should be rolled up as weighted averages (e.g., pop wtd mean of air pollution level)
wts	Default is 'pop', the colname in bg specifying the field to use when calculating the weighted mean of all blockgroups in a tract, for example.
acsnames	Not used. Default is a vector of demographic colnames in bg, used in default ejscreen dataset (see code or ejscreenformulas)
...	Optional parameters to pass to ejscreen.create which uses formulas to create indicators from raw values.

Details

**default fieldnames are assumed for now. Uses ejscreen.create

Value

Returns a data.frame with ejscreen dataset of environmental and demographics indicators, and EJ Indexes, as raw values, US percentiles, but not text for popups. *** Output has one row per tract, county, state, or region, depending on what is specified.

See Also

ejscreen.create

Examples

```
## Not run:
 # load("~/Dropbox/EJSCREEN/R analysis/bg 2015-04-22 Rnames plus subgroups.RData")
 # Do this for each of several levels of resolution
 #
 fipsnames <- c('FIPS.TRACT', 'FIPS.COUNTY', 'FIPS.ST', 'REGION')
 scalenames <- c('tracts', 'counties', 'states', 'regions')
 # or just for tracts, say this:
 #   fipsnames <- 'FIPS.TRACT'; scalenames <- 'tracts'

 for (i in 1:length(fipsnames)) {

 ##################################### #
 # Specify resolution of interest
 fipsname <- fipsnames[i] # 'FIPS.TRACT'
 scalename <- scalenames[i] # 'tracts'

 ##################################### #
 # Get results, using the function
 myrollup <- ejscreen.rollup(bg = bg, fipsname = fipsname, scalename = scalename)

 ##################################### #
 # Save results
 save(myrollup, file = paste('EJSCREEN 2015', scalename, 'data.RData') )
 write.csv(myrollup, row.names = FALSE, file = paste('EJSCREEN 2015', scalename, 'data.csv'))

 }

## End(Not run)
```

`ejscreen.rollup.all` *Aggregate EJSCREEN Dataset at Lower Resolutions (e.g., Tracts and Counties)*

Description

Does what ejscreen.rollup does, but for more than one resolution - a batch of rollups done at once. Start with full EJSCREEN dataset at one resolution (typically block groups), and create aggregated data at higher geographic scales (e.g., tracts and counties)

Usage

```
ejscreen.rollup.all(
  bg,
  scalenames = c("tracts", "counties", "states", "regions"),
  fipsnames = c("FIPS.TRACT", "FIPS.COUNTY", "FIPS.ST", "REGION"),
  myfolder = getwd(),
  filenamebase = "EJSCREEN",
  filenames.R,
  filenames.csv,
  save.R = FALSE,
  save.csv = FALSE,
  assigning = FALSE,
  ...
)
```

Arguments

`bg`	Required, data.frame of raw data for environmental and demographic counts, one row per block group typically, one column per indicator.
`scalenames`	optional character vector of terms used to create filenames if saving files, default = c('tracts', 'counties', 'states', 'regions')
`fipsnames`	optional character vector of certain colnames in bg, used to select columns from bg to summarize by, default = c('FIPS.TRACT', 'FIPS.COUNTY', 'FIPS.ST', 'REGION'),
`myfolder`	optional folder path for saving files, default = getwd()
`filenamebase`	optional character element, default = 'EJSCREEN', used to construct filenames to save files if relevant.
`filenames.R`	optional vector of filenames, default has the word EJSCREEN and scalename .RData
`filenames.csv`	optional vector of filenames, default has the word EJSCREEN and scalename .csv
`save.R`	optional logical, default = FALSE, whether to save files as .RData
`save.csv`	optional logical, default = FALSE, whether to save files as .csv
`assigning`	optional logical, default = FALSE, whether to assign results to variable in calling environment, or just return list of data.frames as result.
`...`	Optional parameters to pass to `ejscreen.create` which uses formulas to create indicators from raw values.

Details

**default fieldnames are assumed for now. Uses ejscreen.create

Value

Returns a list of data.frames each like output of ejscreen.rollup, one per resolution (e.g., one for counties)

See Also

ejscreen.rollup

Examples

```
# (none)
```

---

ejscreen.rollup.save *Helper for ejscreen.rollup.all, to save files of results*

Description

Just saves csv and/or RData file(s)

Usage

```
ejscreen.rollup.save(
  myrollup,
  myfolder = getwd(),
  filenamebase = "EJSCREEN",
  scalename = c("tracts"),
  filename.R,
  filename.csv,
  save.R = TRUE,
  save.csv = TRUE
)
```

Arguments

myrollup	Required, data.frame results from ejscreen.rollup.all (just one scale at a time though)
myfolder	optional folder path for saving files, default = getwd()
filenamebase	optional character element, default = 'EJSCREEN', used to construct filenames to save files if relevant.
scalename	optional character term used to create filenames, default = c('tracts')
filename.R	optional filename, default has the word EJSCREEN and scalename .RData
filename.csv	optional filename, default has the word EJSCREEN and scalename .csv
save.R	optional logical, default = FALSE, whether to save files as .RData
save.csv	optional logical, default = FALSE, whether to save files as .csv

Value

Returns a 2 element vector with full paths of saved R and csv files (or NA instead of a path, if one of those is not saved)

See Also

ejscreen.rollup.all

---

ejscreenformulas *EJSCREEN 2015 Formulas and Fieldnames*

Description

This provides fieldnames and formulas required by the **ejscreen** package. Formulas can be viewed this way: sort(ejscreenformulas$formula)

Usage

```
data('ejscreenformulas')
```

Format

A data.frame:

> str(ejscreenformulas)
'data.frame': 470 obs. of 8 variables:

- $ gdbfieldname : chr NA NA NA NA ...
- $ Rfieldname : chr "ageunder5m" "age5to9m" "age10to14m" "age15to17m" ...
- $ acsfieldname : chr "B01001.003" "B01001.004" "B01001.005" "B01001.006" ...
- $ type : chr "ACS" "ACS" "ACS" "ACS" ...
- $ glossaryfieldname: chr NA NA NA NA ...
- $ formula : chr NA NA NA NA ...
- $ acsfieldnamelong : chr "Under 5 years|SEX BY AGE" "5 to 9 years|SEX BY AGE" "10 to 14 years|SEX BY AGE" "15 to 17 years|SEX BY AGE" ...
- $ universe : chr "Universe: Total population" "Universe: Total population" "Universe: Total population" "Universe: Total population" ...

Source

See related Technical Documentation at http://www.epa.gov/ejscreen

See Also

ejscreenformulasnoej names.evars names.dvars names.ejvars

`ejscreenformulasnoej` *EJSCREEN 2015 Formulas and Fieldnames Excluding EJ Index Formulas*

Description

This provides fieldnames and formulas required by the **ejscreen** package. Formulas can be viewed this way: sort(ejscreenformulas$formula) This excludes the EJ Index formulas for cases where those are to be calculated using code separately.

Usage

```
data('ejscreenformulasnoej')
```

Format

A data.frame:

> str(ejscreenformulas)
'data.frame': 470 obs. of 8 variables:

- $ gdbfieldname : chr NA NA NA NA ...
- $ Rfieldname : chr "ageunder5m" "age5to9m" "age10to14m" "age15to17m" ...
- $ acsfieldname : chr "B01001.003" "B01001.004" "B01001.005" "B01001.006" ...
- $ type : chr "ACS" "ACS" "ACS" "ACS" ...
- $ glossaryfieldname: chr NA NA NA NA ...
- $ formula : chr NA NA NA NA ...
- $ acsfieldnamelong : chr "Under 5 years|SEX BY AGE" "5 to 9 years|SEX BY AGE" "10 to 14 years|SEX BY AGE" "15 to 17 years|SEX BY AGE" ...
- $ universe : chr "Universe: Total population" "Universe: Total population" "Universe: Total population" "Universe: Total population" ...

Source

See related Technical Documentation at http://www.epa.gov/ejscreen

See Also

`ejscreenformulas` `names.evars` `names.dvars` `names.ejvars`

`ejscreensignifarray` *Specify Significant Digits for Each Column of EJSCREEN Indicators*

Description

Given a matrix or numeric data.frame, round each column to a specified column-specific number of significant digits. This function provides default values significant digits to use for an EJSCREEN environmental dataset. This is a wrapper for analyze.stuff::signifarray which is a wrapper that applies signif() to a matrix or data.frame.

Usage

```
ejscreensignifarray(dat, digits = "ejscreen")
```

Arguments

`dat`	Required, matrix or numeric data.frame with the values to be rounded.
`digits`	Optional, 'ejscreen' by default. Can be a vector as long as the number of columns in dat, where each elements specifies the number of significant digits to retain for numbers in the corresponding column of dat. If 'ejscreen' it specifies using the default settings described below in details, in which case all colnames(dat) must be among (but in any order) defaultcolnames below.

Details

Sig figs used if digits specified as 'ejscreen' are those stored in data(esigfigs)

Value

Returns dat, but with numbers rounded based on digits parameter.

See Also

`esigfigs` `signifarray` `signif`

Examples

```
ejscreensignifarray(data.frame(a=rnorm(10), b=rnorm(10), c=rnorm(10)), 1:3)
envirodata <- data.frame(matrix(rnorm(11*10), ncol=11))
  # data("names.evars"); names(envirodata) <- names.e
 names(envirodata) <- c("pm", "o3", "cancer", "resp", "dpm", "pctpre1960", "traffic.score",
  "proximity.npl", "proximity.rmp", "proximity.tsdf", "proximity.npdes")
ejscreensignifarray(envirodata)
```

`esigfigs` *How many signif digits to show*

Description

How many sig figs to show in showing environmental indicators in EJSCREEN?

Usage

```
data('esigfigs')
```

Format

A data.frame:

> str(esigfigs)
'data.frame': 12 obs. of 2 variables:
$ sigfigs: num 3 3 2 2 2 3 2 2 2 2 ...
$ evar : chr "pm" "o3" "cancer" "neuro" ...

sigfigs evar
3 pm
3 o3
2 cancer
2 neuro
2 resp
3 dpm
2 pctpre1960
2 traffic.score
2 proximity.npl
2 proximity.rmp
2 proximity.tsdf
2 proximity.npdes

Source

See related Technical Documentation at http://www.epa.gov/ejscreen

See Also

make.popup.e

lookupRegions19 *The EPA-Region-level 2019 version of the EJSCREEN percentile lookup table.*

Description

Note the 2018 version of EJSCREEN (released late 2018) actually uses ACS2016, which is from 2012-2016 (released late 2017). Note the 2019 version of EJSCREEN (released late 2019) actually uses ACS2017, which is from 2013-2017 (released late 2018). This is from the EJSCREEN dataset from the ftp site but with fields renamed for easier use in the ejscreen package. It can be used with for example ejanalysis::lookup.pctile(13, varname.in.lookup.table = 'pm', lookup = lookupUSA19) It shows what the cutpoints are for each variable at percentiles 0,1,2 through 99, 100. For example, if the traffic.score is 1000 in a given location, you can look where that falls in the percentiles and see that 81 lookup.pctile(1000, varname.in.lookup.table = 'traffic.score', lookup = lookupUSA19)

Details

It was created for this package as follows:

```
require(ejscreen)
require(analyze.stuff); require(ejanalysis); require(readr)
```

Get EJSCREEN geodatabase downloaded from public FTP site
as gdb format (zipped)
EJSCREEN_V2019.gdb.zip
then unzipped to
EJSCREEN_V2019.gdb
Opened EJSCREEN_V2019.gdb file in ESRI's ArcGIS

Opened attribute tables USA, Regions, and States
Exported all records as text format to files named
EJSCREEN_USA_2019_Export_Output.csv, etc.

```
lookupUSA19 <-readr::read_csv('USA_2019_Export_Output.csv')
lookupRegions19 <-readr::read_csv('Regions_2019_Export_Output.csv')
lookupStates19 <-readr::read_csv('States_2019_Export_Output.csv')

names(lookupStates19)
# [1] "OBJECTID" "REGION" "PCTILE" "MINORPCT" "LOWINCPCT" "LESSHSPCT" "LINGISOPCT" "UNDER5PCT"
"OVER64PCT" "PRE1960PCT" "VULEOPCT"
# [12] "VULSVI6PCT" "DSLPM" "CANCER" "RESP" "PTRAF" "PWDIS" "PNPL" "PRMP" "PTSDF" "OZONE"
"PM25"
# [23] "D_LDPNT_2" "LDPNT_D6" "LDPNT_B2" "LDPNT_B6" "LDPNT_P2" "LDPNT_P6" "D_DSLPM_2"
"DSLPM_D6" "DSLPM_B2" "DSLPM_B6" "DSLPM_P2"
# [34] "DSLPM_P6" "D_CANCR_2" "CANCR_D6" "CANCR_B2" "CANCR_B6" "CANCR_P2" "CANCR_P6"
"D_RESP_2" "RESP_D6" "RESP_B2" "RESP_B6"
# [45] "RESP_P2" "RESP_P6" "D_PTRAF_2" "PTRAF_D6" "PTRAF_B2" "PTRAF_B6" "PTRAF_P2" "PTRAF_P6"
"D_PWDIS_2" "PWDIS_D6" "PWDIS_B2"
# [56] "PWDIS_B6" "PWDIS_P2" "PWDIS_P6" "D_PNPL_2" "PNPL_D6" "PNPL_B2" "PNPL_B6" "PNPL_P2"
"PNPL_P6" "D_PRMP_2" "PRMP_D6"
# [67] "PRMP_B2" "PRMP_B6" "PRMP_P2" "PRMP_P6" "D_PTSDF_2" "PTSDF_D6" "PTSDF_B2" "PTSDF_B6"
```

```
"PTSDF_P2" "PTSDF_P6" "D_OZONE_2"
# [78] "OZONE_D6" "OZONE_B2" "OZONE_B6" "OZONE_P2" "OZONE_P6" "D_PM25_2" "PM25_D6" "PM25_B2"
"PM25_B6" "PM25_P2" "PM25_P6"

names(lookupUSA19) <-ejscreen::change.fieldnames.ejscreen.csv(names(lookupUSA19))
names(lookupRegions19) <-ejscreen::change.fieldnames.ejscreen.csv(names(lookupRegions19))
names(lookupStates19) <-ejscreen::change.fieldnames.ejscreen.csv(names(lookupStates19))

# c('','',names(lookupUSA19))
# [1] "" "" "OBJECTID" "REGION" "PCTILE" "pctmin"
# [7] "pctlowinc" "pctlths" "pctlingiso" "pctunder5" "pctover64" "pctpre1960"
# [13] "VSI.eo" "VSI.svi6" "dpm" "cancer" "resp" "traffic.score"
# [19] "proximity.npdes" "proximity.npl" "proximity.rmp" "proximity.tsdf" "o3" "pm"
# [25] "EJ.DISPARITY.pctpre1960.eo" "EJ.DISPARITY.pctpre1960.svi6" "EJ.BURDEN.pctpre1960.eo"
"EJ.BURDEN.pctpre1960.svi6" "EJ.PCT.pctpre1960.eo" "EJ.PCT.pctpre1960.svi6"
# [31] "EJ.DISPARITY.dpm.eo" "EJ.DISPARITY.dpm.svi6" "EJ.BURDEN.dpm.eo" "EJ.BURDEN.dpm.svi6"
"EJ.PCT.dpm.eo" "EJ.PCT.dpm.svi6"
# [37] "EJ.DISPARITY.cancer.eo" "EJ.DISPARITY.cancer.svi6" "EJ.BURDEN.cancer.eo" "EJ.BURDEN.cancer
"EJ.PCT.cancer.eo" "EJ.PCT.cancer.svi6"
# [43] "EJ.DISPARITY.resp.eo" "EJ.DISPARITY.resp.svi6" "EJ.BURDEN.resp.eo" "EJ.BURDEN.resp.svi6"
"EJ.PCT.resp.eo" "EJ.PCT.resp.svi6"
# [49] "EJ.DISPARITY.traffic.score.eo" "EJ.DISPARITY.traffic.score.svi6" "EJ.BURDEN.traffic.score.
"EJ.BURDEN.traffic.score.svi6" "EJ.PCT.traffic.score.eo" "EJ.PCT.traffic.score.svi6"
# [55] "EJ.DISPARITY.proximity.npdes.eo" "EJ.DISPARITY.proximity.npdes.svi6" "EJ.BURDEN.proximity.
"EJ.BURDEN.proximity.npdes.svi6" "EJ.PCT.proximity.npdes.eo" "EJ.PCT.proximity.npdes.svi6"
# [61] "EJ.DISPARITY.proximity.npl.eo" "EJ.DISPARITY.proximity.npl.svi6" "EJ.BURDEN.proximity.npl.
"EJ.BURDEN.proximity.npl.svi6" "EJ.PCT.proximity.npl.eo" "EJ.PCT.proximity.npl.svi6"
# [67] "EJ.DISPARITY.proximity.rmp.eo" "EJ.DISPARITY.proximity.rmp.svi6" "EJ.BURDEN.proximity.rmp.
"EJ.BURDEN.proximity.rmp.svi6" "EJ.PCT.proximity.rmp.eo" "EJ.PCT.proximity.rmp.svi6"
# [73] "EJ.DISPARITY.proximity.tsdf.eo" "EJ.DISPARITY.proximity.tsdf.svi6" "EJ.BURDEN.proximity.ts
"EJ.BURDEN.proximity.tsdf.svi6" "EJ.PCT.proximity.tsdf.eo" "EJ.PCT.proximity.tsdf.svi6"
# [79] "EJ.DISPARITY.o3.eo" "EJ.DISPARITY.o3.svi6" "EJ.BURDEN.o3.eo" "EJ.BURDEN.o3.svi6"
"EJ.PCT.o3.eo" "EJ.PCT.o3.svi6"
# [85] "EJ.DISPARITY.pm.eo" "EJ.DISPARITY.pm.svi6" "EJ.BURDEN.pm.eo" "EJ.BURDEN.pm.svi6"
"EJ.PCT.pm.eo" "EJ.PCT.pm.svi6"

lookupUSA19 <-as.data.frame(lookupUSA19)
lookupRegions19 <-as.data.frame(lookupRegions19)
lookupStates19 <-as.data.frame(lookupStates19)

# Then for this package, could get rid of some nonessential fields as follows:
# (But note svi6 fields - which combine all 6 demog indicators not just 2 - were named in names.d)

# x <- lookupStates19 #x <-x[ ,!grepl(pattern = 'svi6',x = names(x))]
#x <-x[ ,!grepl(pattern = 'pctile\\.text',x = names(x))]
#x <-x[ ,!grepl(pattern = 'EJ\\.PCT',x = names(x))]
#x <-x[ ,!grepl(pattern = 'EJ\\.BURDEN',x = names(x))]
# lookupStates19 <- x
save(lookupUSA19,file = 'lookupUSA19.rdata')
save(lookupRegions19,file = 'lookupRegions19.rdata')
save(lookupStates19,file = 'lookupStates19.rdata')
```

See Also

lookupUSA19 lookupRegions19 lookupStates19 [ejanalysis]lookup.pctile

Examples

```
lookup.pctile(1000, varname.in.lookup.table = 'traffic.score',  lookup = lookupUSA19)
    lookup.pctile(c(1000, 3000), varname.in.lookup.table = 'traffic.score',
      lookup = lookupStates19, zone = 'NY')
     # Those traffic scores are at the 62d and 83d percentiles within NY State (83 percent
      # of the NY State population had a traffic score lower than 3000).
   ## Not run:
    bg <- bg19[sample(1:NROW(bg19), 100), ]
   state.pctile.pm <- ejanalysis::lookup.pctile(myvector = bg$pm, varname.in.lookup.table = 'pm',
       lookup = lookupStates19, zone = bg$ST)
    plot(state.pctile.pm, bg$pctile.pm, pch = '.')
   text(state.pctile.pm, bg$pctile.pm, labels = paste(bg$ST, round(bg$pm,1)), cex = 0.8)
    abline(0,1)
    lookupStates19[lookupStates19$PCTILE == 'mean', c('REGION', 'pm')]
    lookupUSA19[lookupUSA19$PCTILE == 'mean', c('REGION', 'pm')]

## End(Not run)
```

`lookupStates19` *The State-level 2019 version of the EJSCREEN percentile lookup table.*

Description

Note the 2018 version of EJSCREEN (released late 2018) actually uses ACS2016, which is from 2012-2016 (released late 2017). Note the 2019 version of EJSCREEN (released late 2019) actually uses ACS2017, which is from 2013-2017 (released late 2018).

This is from the EJSCREEN dataset from the ftp site but with fields renamed for easier use in the ejscreen package. It can be used with for example ejanalysis::lookup.pctile(13, varname.in.lookup.table = 'pm', lookup = lookupUSA19) It shows what the cutpoints are for each variable at percentiles 0,1,2 through 99, 100. For example, if the traffic.score is 1000 in a given location, you can look where that falls in the percentiles and see that 81 lookup.pctile(1000, varname.in.lookup.table = 'traffic.score', lookup = lookupUSA19)

Details

It was created for this package as follows:

```
require(ejscreen)
require(analyze.stuff); require(ejanalysis); require(readr)
```

Get EJSCREEN geodatabase downloaded from public FTP site
as gdb format (zipped)
EJSCREEN_V2019.gdb.zip
then unzipped to
EJSCREEN_V2019.gdb
Opened EJSCREEN_V2019.gdb file in ESRI's ArcGIS

Opened attribute tables USA, Regions, and States
Exported all records as text format to files named
EJSCREEN_USA_2019_Export_Output.csv, etc.

```
lookupUSA19 <-readr::read_csv('USA_2019_Export_Output.csv')
lookupRegions19 <-readr::read_csv('Regions_2019_Export_Output.csv')
lookupStates19 <-readr::read_csv('States_2019_Export_Output.csv')

names(lookupStates19)
# [1] "OBJECTID" "REGION" "PCTILE" "MINORPCT" "LOWINCPCT" "LESSHSPCT" "LINGISOPCT" "UNDER5PCT"
"OVER64PCT" "PRE1960PCT" "VULEOPCT"
# [12] "VULSVI6PCT" "DSLPM" "CANCER" "RESP" "PTRAF" "PWDIS" "PNPL" "PRMP" "PTSDF" "OZONE"
"PM25"
# [23] "D_LDPNT_2" "LDPNT_D6" "LDPNT_B2" "LDPNT_B6" "LDPNT_P2" "LDPNT_P6" "D_DSLPM_2"
"DSLPM_D6" "DSLPM_B2" "DSLPM_B6" "DSLPM_P2"
# [34] "DSLPM_P6" "D_CANCR_2" "CANCR_D6" "CANCR_B2" "CANCR_B6" "CANCR_P2" "CANCR_P6"
"D_RESP_2" "RESP_D6" "RESP_B2" "RESP_B6"
# [45] "RESP_P2" "RESP_P6" "D_PTRAF_2" "PTRAF_D6" "PTRAF_B2" "PTRAF_B6" "PTRAF_P2" "PTRAF_P6"
"D_PWDIS_2" "PWDIS_D6" "PWDIS_B2"
# [56] "PWDIS_B6" "PWDIS_P2" "PWDIS_P6" "D_PNPL_2" "PNPL_D6" "PNPL_B2" "PNPL_B6" "PNPL_P2"
"PNPL_P6" "D_PRMP_2" "PRMP_D6"
# [67] "PRMP_B2" "PRMP_B6" "PRMP_P2" "PRMP_P6" "D_PTSDF_2" "PTSDF_D6" "PTSDF_B2" "PTSDF_B6"
"PTSDF_P2" "PTSDF_P6" "D_OZONE_2"
# [78] "OZONE_D6" "OZONE_B2" "OZONE_B6" "OZONE_P2" "OZONE_P6" "D_PM25_2" "PM25_D6" "PM25_B2"
"PM25_B6" "PM25_P2" "PM25_P6"

names(lookupUSA19) <-ejscreen::change.fieldnames.ejscreen.csv(names(lookupUSA19))
names(lookupRegions19) <-ejscreen::change.fieldnames.ejscreen.csv(names(lookupRegions19))
names(lookupStates19) <-ejscreen::change.fieldnames.ejscreen.csv(names(lookupStates19))

# c('','',names(lookupUSA19))
# [1] "" "" "OBJECTID" "REGION" "PCTILE" "pctmin"
# [7] "pctlowinc" "pctlths" "pctlingiso" "pctunder5" "pctover64" "pctpre1960"
# [13] "VSI.eo" "VSI.svi6" "dpm" "cancer" "resp" "traffic.score"
# [19] "proximity.npdes" "proximity.npl" "proximity.rmp" "proximity.tsdf" "o3" "pm"
# [25] "EJ.DISPARITY.pctpre1960.eo" "EJ.DISPARITY.pctpre1960.svi6" "EJ.BURDEN.pctpre1960.eo"
"EJ.BURDEN.pctpre1960.svi6" "EJ.PCT.pctpre1960.eo" "EJ.PCT.pctpre1960.svi6"
# [31] "EJ.DISPARITY.dpm.eo" "EJ.DISPARITY.dpm.svi6" "EJ.BURDEN.dpm.eo" "EJ.BURDEN.dpm.svi6"
"EJ.PCT.dpm.eo" "EJ.PCT.dpm.svi6"
# [37] "EJ.DISPARITY.cancer.eo" "EJ.DISPARITY.cancer.svi6" "EJ.BURDEN.cancer.eo" "EJ.BURDEN.cancer
"EJ.PCT.cancer.eo" "EJ.PCT.cancer.svi6"
# [43] "EJ.DISPARITY.resp.eo" "EJ.DISPARITY.resp.svi6" "EJ.BURDEN.resp.eo" "EJ.BURDEN.resp.svi6"
"EJ.PCT.resp.eo" "EJ.PCT.resp.svi6"
# [49] "EJ.DISPARITY.traffic.score.eo" "EJ.DISPARITY.traffic.score.svi6" "EJ.BURDEN.traffic.score.
"EJ.BURDEN.traffic.score.svi6" "EJ.PCT.traffic.score.eo" "EJ.PCT.traffic.score.svi6"
# [55] "EJ.DISPARITY.proximity.npdes.eo" "EJ.DISPARITY.proximity.npdes.svi6" "EJ.BURDEN.proximity.
"EJ.BURDEN.proximity.npdes.svi6" "EJ.PCT.proximity.npdes.eo" "EJ.PCT.proximity.npdes.svi6"
# [61] "EJ.DISPARITY.proximity.npl.eo" "EJ.DISPARITY.proximity.npl.svi6" "EJ.BURDEN.proximity.npl.
"EJ.BURDEN.proximity.npl.svi6" "EJ.PCT.proximity.npl.eo" "EJ.PCT.proximity.npl.svi6"
# [67] "EJ.DISPARITY.proximity.rmp.eo" "EJ.DISPARITY.proximity.rmp.svi6" "EJ.BURDEN.proximity.rmp.
"EJ.BURDEN.proximity.rmp.svi6" "EJ.PCT.proximity.rmp.eo" "EJ.PCT.proximity.rmp.svi6"
# [73] "EJ.DISPARITY.proximity.tsdf.eo" "EJ.DISPARITY.proximity.tsdf.svi6" "EJ.BURDEN.proximity.ts
```

```
"EJ.BURDEN.proximity.tsdf.svi6" "EJ.PCT.proximity.tsdf.eo" "EJ.PCT.proximity.tsdf.svi6"
# [79] "EJ.DISPARITY.o3.eo" "EJ.DISPARITY.o3.svi6" "EJ.BURDEN.o3.eo" "EJ.BURDEN.o3.svi6"
"EJ.PCT.o3.eo" "EJ.PCT.o3.svi6"
# [85] "EJ.DISPARITY.pm.eo" "EJ.DISPARITY.pm.svi6" "EJ.BURDEN.pm.eo" "EJ.BURDEN.pm.svi6"
"EJ.PCT.pm.eo" "EJ.PCT.pm.svi6"

lookupUSA19 <-as.data.frame(lookupUSA19)
lookupRegions19 <-as.data.frame(lookupRegions19)
lookupStates19 <-as.data.frame(lookupStates19)

# Then for this package, could get rid of some nonessential fields as follows:
# (But note svi6 fields - which combine all 6 demog indicators not just 2 - were named in names.d)

# x <- lookupStates19 #x <-x[ ,!grepl(pattern = 'svi6',x = names(x))]
#x <-x[ ,!grepl(pattern = 'pctile\\.text',x = names(x))]
#x <-x[ ,!grepl(pattern = 'EJ\\.PCT',x = names(x))]
#x <-x[ ,!grepl(pattern = 'EJ\\.BURDEN',x = names(x))]
# lookupStates19 <- x
save(lookupUSA19,file = 'lookupUSA19.rdata')
save(lookupRegions19,file = 'lookupRegions19.rdata')
save(lookupStates19,file = 'lookupStates19.rdata')
```

See Also

lookupUSA19 lookupRegions19 lookupStates19 [ejanalysis]lookup.pctile

Examples

```
lookup.pctile(1000, varname.in.lookup.table = 'traffic.score',  lookup = lookupUSA19)
    lookup.pctile(c(1000, 3000), varname.in.lookup.table = 'traffic.score',
      lookup = lookupStates19, zone = 'NY')
     # Those traffic scores are at the 62d and 83d percentiles within NY State (83 percent
      # of the NY State population had a traffic score lower than 3000).
   ## Not run:
    bg <- bg19[sample(1:NROW(bg19), 100), ]
   state.pctile.pm <- ejanalysis::lookup.pctile(myvector = bg$pm, varname.in.lookup.table = 'pm',
       lookup = lookupStates19, zone = bg$ST)
    plot(state.pctile.pm, bg$pctile.pm, pch = '.')
    text(state.pctile.pm, bg$pctile.pm, labels = paste(bg$ST, round(bg$pm,1)), cex = 0.8)
    abline(0,1)
    lookupStates19[lookupStates19$PCTILE == 'mean', c('REGION', 'pm')]
    lookupUSA19[lookupUSA19$PCTILE == 'mean', c('REGION', 'pm')]

## End(Not run)
```

---

lookupUSA19 *The nationwide 2019 version of the EJSCREEN percentile lookup table.*

Description

Note the 2018 version of EJSCREEN (released late 2018) actually uses ACS2016, which is from 2012-2016 (released late 2017). Note the 2019 version of EJSCREEN (released late 2019) actually uses ACS2017, which is from 2013-2017 (released late 2018). This is from the EJSCREEN dataset from the ftp site but with fields renamed for easier use in the ejscreen package. It can be used with for example ejanalysis::lookup.pctile(13, varname.in.lookup.table = 'pm', lookup = lookupUSA19) It shows what the cutpoints are for each variable at percentiles 0,1,2 through 99, 100. For example, if the traffic.score is 1000 in a given location, you can look where that falls in the percentiles and see that 81 lookup.pctile(1000, varname.in.lookup.table = 'traffic.score', lookup = lookupUSA19)

Details

It was created for this package as follows:

```
require(ejscreen)
require(analyze.stuff); require(ejanalysis); require(readr)
```

Get EJSCREEN geodatabase downloaded from public FTP site
as gdb format (zipped)
EJSCREEN_V2019.gdb.zip
then unzipped to
EJSCREEN_V2019.gdb
Opened EJSCREEN_V2019.gdb file in ESRI's ArcGIS

Opened attribute tables USA, Regions, and States
Exported all records as text format to files named
EJSCREEN_USA_2019_Export_Output.csv, etc.

```
lookupUSA19 <-readr::read_csv('USA_2019_Export_Output.csv')
lookupRegions19 <-readr::read_csv('Regions_2019_Export_Output.csv')
lookupStates19 <-readr::read_csv('States_2019_Export_Output.csv')

names(lookupStates19)
# [1] "OBJECTID" "REGION" "PCTILE" "MINORPCT" "LOWINCPCT" "LESSHSPCT" "LINGISOPCT" "UNDER5PCT"
"OVER64PCT" "PRE1960PCT" "VULEOPCT"
# [12] "VULSVI6PCT" "DSLPM" "CANCER" "RESP" "PTRAF" "PWDIS" "PNPL" "PRMP" "PTSDF" "OZONE"
"PM25"
# [23] "D_LDPNT_2" "LDPNT_D6" "LDPNT_B2" "LDPNT_B6" "LDPNT_P2" "LDPNT_P6" "D_DSLPM_2"
"DSLPM_D6" "DSLPM_B2" "DSLPM_B6" "DSLPM_P2"
# [34] "DSLPM_P6" "D_CANCR_2" "CANCR_D6" "CANCR_B2" "CANCR_B6" "CANCR_P2" "CANCR_P6"
"D_RESP_2" "RESP_D6" "RESP_B2" "RESP_B6"
# [45] "RESP_P2" "RESP_P6" "D_PTRAF_2" "PTRAF_D6" "PTRAF_B2" "PTRAF_B6" "PTRAF_P2" "PTRAF_P6"
"D_PWDIS_2" "PWDIS_D6" "PWDIS_B2"
# [56] "PWDIS_B6" "PWDIS_P2" "PWDIS_P6" "D_PNPL_2" "PNPL_D6" "PNPL_B2" "PNPL_B6" "PNPL_P2"
"PNPL_P6" "D_PRMP_2" "PRMP_D6"
# [67] "PRMP_B2" "PRMP_B6" "PRMP_P2" "PRMP_P6" "D_PTSDF_2" "PTSDF_D6" "PTSDF_B2" "PTSDF_B6"
"PTSDF_P2" "PTSDF_P6" "D_OZONE_2"
# [78] "OZONE_D6" "OZONE_B2" "OZONE_B6" "OZONE_P2" "OZONE_P6" "D_PM25_2" "PM25_D6" "PM25_B2"
"PM25_B6" "PM25_P2" "PM25_P6"

names(lookupUSA19) <-ejscreen::change.fieldnames.ejscreen.csv(names(lookupUSA19))
names(lookupRegions19) <-ejscreen::change.fieldnames.ejscreen.csv(names(lookupRegions19))
```

```
names(lookupStates19) <-ejscreen::change.fieldnames.ejscreen.csv(names(lookupStates19))

# c('','',names(lookupUSA19))
# [1] "" "" "OBJECTID" "REGION" "PCTILE" "pctmin"
# [7] "pctlowinc" "pctlths" "pctlingiso" "pctunder5" "pctover64" "pctpre1960"
# [13] "VSI.eo" "VSI.svi6" "dpm" "cancer" "resp" "traffic.score"
# [19] "proximity.npdes" "proximity.npl" "proximity.rmp" "proximity.tsdf" "o3" "pm"
# [25] "EJ.DISPARITY.pctpre1960.eo" "EJ.DISPARITY.pctpre1960.svi6" "EJ.BURDEN.pctpre1960.eo"
"EJ.BURDEN.pctpre1960.svi6" "EJ.PCT.pctpre1960.eo" "EJ.PCT.pctpre1960.svi6"
# [31] "EJ.DISPARITY.dpm.eo" "EJ.DISPARITY.dpm.svi6" "EJ.BURDEN.dpm.eo" "EJ.BURDEN.dpm.svi6"
"EJ.PCT.dpm.eo" "EJ.PCT.dpm.svi6"
# [37] "EJ.DISPARITY.cancer.eo" "EJ.DISPARITY.cancer.svi6" "EJ.BURDEN.cancer.eo" "EJ.BURDEN.cancer
"EJ.PCT.cancer.eo" "EJ.PCT.cancer.svi6"
# [43] "EJ.DISPARITY.resp.eo" "EJ.DISPARITY.resp.svi6" "EJ.BURDEN.resp.eo" "EJ.BURDEN.resp.svi6"
"EJ.PCT.resp.eo" "EJ.PCT.resp.svi6"
# [49] "EJ.DISPARITY.traffic.score.eo" "EJ.DISPARITY.traffic.score.svi6" "EJ.BURDEN.traffic.score.
"EJ.BURDEN.traffic.score.svi6" "EJ.PCT.traffic.score.eo" "EJ.PCT.traffic.score.svi6"
# [55] "EJ.DISPARITY.proximity.npdes.eo" "EJ.DISPARITY.proximity.npdes.svi6" "EJ.BURDEN.proximity.
"EJ.BURDEN.proximity.npdes.svi6" "EJ.PCT.proximity.npdes.eo" "EJ.PCT.proximity.npdes.svi6"
# [61] "EJ.DISPARITY.proximity.npl.eo" "EJ.DISPARITY.proximity.npl.svi6" "EJ.BURDEN.proximity.npl.
"EJ.BURDEN.proximity.npl.svi6" "EJ.PCT.proximity.npl.eo" "EJ.PCT.proximity.npl.svi6"
# [67] "EJ.DISPARITY.proximity.rmp.eo" "EJ.DISPARITY.proximity.rmp.svi6" "EJ.BURDEN.proximity.rmp.
"EJ.BURDEN.proximity.rmp.svi6" "EJ.PCT.proximity.rmp.eo" "EJ.PCT.proximity.rmp.svi6"
# [73] "EJ.DISPARITY.proximity.tsdf.eo" "EJ.DISPARITY.proximity.tsdf.svi6" "EJ.BURDEN.proximity.ts
"EJ.BURDEN.proximity.tsdf.svi6" "EJ.PCT.proximity.tsdf.eo" "EJ.PCT.proximity.tsdf.svi6"
# [79] "EJ.DISPARITY.o3.eo" "EJ.DISPARITY.o3.svi6" "EJ.BURDEN.o3.eo" "EJ.BURDEN.o3.svi6"
"EJ.PCT.o3.eo" "EJ.PCT.o3.svi6"
# [85] "EJ.DISPARITY.pm.eo" "EJ.DISPARITY.pm.svi6" "EJ.BURDEN.pm.eo" "EJ.BURDEN.pm.svi6"
"EJ.PCT.pm.eo" "EJ.PCT.pm.svi6"

lookupUSA19 <-as.data.frame(lookupUSA19)
lookupRegions19 <-as.data.frame(lookupRegions19)
lookupStates19 <-as.data.frame(lookupStates19)

# Then for this package, could get rid of some nonessential fields as follows:
# (But note svi6 fields - which combine all 6 demog indicators not just 2 - were named in names.d)

# x <- lookupStates19 #x <-x[ ,!grepl(pattern = 'svi6',x = names(x))]
#x <-x[ ,!grepl(pattern = 'pctile\\.text',x = names(x))]
#x <-x[ ,!grepl(pattern = 'EJ\\.PCT',x = names(x))]
#x <-x[ ,!grepl(pattern = 'EJ\\.BURDEN',x = names(x))]
# lookupStates19 <- x
save(lookupUSA19,file = 'lookupUSA19.rdata')
save(lookupRegions19,file = 'lookupRegions19.rdata')
save(lookupStates19,file = 'lookupStates19.rdata')
```

See Also

lookupUSA19 lookupRegions19 lookupStates19 [ejanalysis]lookup.pctile

Examples

```
lookup.pctile(1000, varname.in.lookup.table = 'traffic.score',  lookup = lookupUSA19)
    lookup.pctile(c(1000, 3000), varname.in.lookup.table = 'traffic.score',
      lookup = lookupStates19, zone = 'NY')
     # Those traffic scores are at the 62d and 83d percentiles within NY State (83 percent
      # of the NY State population had a traffic score lower than 3000).
   ## Not run:
    bg <- bg19[sample(1:NROW(bg19), 100), ]
   state.pctile.pm <- ejanalysis::lookup.pctile(myvector = bg$pm, varname.in.lookup.table = 'pm',
       lookup = lookupStates19, zone = bg$ST)
    plot(state.pctile.pm, bg$pctile.pm, pch = '.')
   text(state.pctile.pm, bg$pctile.pm, labels = paste(bg$ST, round(bg$pm,1)), cex = 0.8)
    abline(0,1)
    lookupStates19[lookupStates19$PCTILE == 'mean', c('REGION', 'pm')]
    lookupUSA19[lookupUSA19$PCTILE == 'mean', c('REGION', 'pm')]

## End(Not run)
```

`make.popup.d` *Make text to be shown in popups on Demographic data map*

Description

Takes raw values and what percentiles they are at, and presents those as a text field to be used as the text in a popup window on a map

Usage

```
make.popup.d(d, pctile, prefix = "pctile.text.", basenames)
```

Arguments

`d`	raw demographic values, 0-1 (such as 0.3345 where roughly 33 percent of the local population is under age 5)
`pctile`	required integers 0 to 100, representing the percentile(s) at which the raw value(s) fall(s).
`prefix`	optional, default is 'pctile.text.' This is a text string specifying the first part of the desired resulting fieldname in outputs.
`basenames`	optional, default is colnames(d). Defines colname(s) of outputs, which are the prefix plus this.

Details

Note d should be a (vector? or) data.frame of exact demographic percentages from 0 to 1, not 0 to 100 BUT pctile should be INTEGER 0 to 100, NOT 0 to 1! Because that is how EJSCREEN data are stored In EJSCREEN, there are three types of pctile.text fields: E (text varies), D, EJ: 'pctile.text.cancer' "55 lifetime risk per million (91 'pctile.text.pctmin' "13 'pctile.text.EJ.DISPARITY.cancer.eo' "36

Value

Returns character vector or data.frame, same shape as first input parameter.

See Also

make.popup.d make.popup.e make.popup.ej pctileAsText

Examples

```
# inputs are test0 and test1, and desired output is like test2
  # (except note how prefix is added to each basename)
test0 <- structure(list(
  VSI.eo = c(0.185525372063833, 0.174428104575163, 0.485647788983707),
  pctmin = c(0.131656804733727, 0.111928104575163, 0.671062839410395),
  other = c(NA, NA, 0.02)),
  .Names = c("VSI.eo", "pctmin", "other"),
  row.names = c(NA, 3L), class = "data.frame")
 test0
# VSI.eo    pctmin other
# 1 0.1855254 0.1316568    NA
# 2 0.1744281 0.1119281    NA
# 3 0.4856478 0.6710628  0.02

 test1 <- structure(list(
   pctile.VSI.eo = c(27.1991395138354, 24.6836238179206, 72.382419748292),
   pctile.pctmin = c(30.2662374847936, 26.761078397073, 78.2620665123235),
   other = c(NA, NA, 4)),
   .Names = c("pctile.VSI.eo", "pctile.pctmin", "other"),
   row.names = c(NA, 3L), class = "data.frame")
 test1
#   pctile.VSI.eo pctile.pctmin other
# 1      27.19914      30.26624    NA
# 2      24.68362      26.76108    NA
# 3      72.38242      78.26207     4

 test2 <- structure(list(
   pctile.text.VSI.eo = c("19% (27%ile)", "17% (24%ile)","49% (72%ile)"),
   pctile.text.pctmin = c("13% (30%ile)", "11% (26%ile)", "67% (78%ile)"),
   other = c(NA, NA, 4)),
   .Names = c("pctile.text.VSI.eo","pctile.text.pctmin", "other"),
   row.names = c(NA, 3L), class = "data.frame")
 test2
#   pctile.text.VSI.eo pctile.text.pctmin other
# 1       19% (27%ile)       13% (30%ile)    NA
# 2       17% (24%ile)       11% (26%ile)    NA
# 3       49% (72%ile)       67% (78%ile)     4

make.popup.d(test0, test1)
 #   pctile.text.VSI.eo pctile.text.pctmin pctile.text.other
 # 1       19% (27%ile)       13% (30%ile)              <NA>
 # 2       17% (24%ile)       11% (26%ile)              <NA>
 # 3       49% (72%ile)       67% (78%ile)        2% (4%ile)
```

---

make.popup.e *Make text to be shown in popups on Envt data map*

Description

Takes raw values and what percentiles they are at, and presents those as a text field to be used as the text in a popup window on a map

Usage

```
make.popup.e(e, pctile, prefix = "pctile.text.", basenames, units, sigfigs)
```

Arguments

e	raw environmental indicator values for various locations
pctile	required integers 0 to 100, representing the percentile(s) at which the raw value(s) fall(s).
prefix	optional, default is 'pctile.text.' This is a text string specifying the first part of the desired resulting fieldname in outputs.
basenames	optional, default is colnames(e). Defines colname(s) of outputs, which are the prefix plus this.
units	optional character vector with one per column of e, default is the units used for the latest (2016) version of EJSCREEN environmental indicators, such as 'ppb' and 'ug/m3' – function will try to use units appropriate to basenames, looking in data(popupunits), and use '' (blank) if no match is found.
sigfigs	optional, numeric vector with one per col of e, defining number of significant digits to show in popup, defaulting to rules in EJSCREEN latest (2016) version, or just 2 for basenames not found in data(esigfigs).

Details

Could edit code to NOT put in the units when value is NA?
Could edit code to handle cases like only one row, matrix not df?
Could fix to use only one space when no units

EJSCREEN as of 2015 used 85 pctile.text. fields, for popup text, like "pctile.text.EJ.DISPARITY.pm.eo"

```
names(bg2)[grepl('pctile.text',names(bg2) )]
length( bg2[1,grepl('pctile.text',names(bg2) )] )
# [1] 85 \cr\cr
```

In EJSCREEN, there are three types of pctile.text fields: E (text varies), D, EJ:

```
'pctile.text.cancer' "55 lifetime risk per million (91 'pctile.text.pctmin' "13 'pctile.text.EJ.DISP
"36 } For E popups,text includes units:\cr (neuro was only in 2015 version,not later versions
of EJSCREEN)\cr\cr \code{ names.e.pctile[names.e.pctile != 'pctile.neuro']\cr # [1]
"pctile.pm" "pctile.o3" "pctile.cancer" \cr # [4] "pctile.resp" "pctile.dpm" "pctile.pctpre1960"
\cr # [7] "pctile.traffic.score" "pctile.proximity.npl" "pctile.proximity.rmp" \cr #
[10] "pctile.proximity.tsdf" "pctile.proximity.npdes"\cr\cr } # NOTE HOW UNITS ARE PART
OF THE POPUP,AND IT USES SPECIAL ROUNDING RULES \cr # #' # Stored in data('popunits') #
colnames are evar and units \cr\cr \code{ t(bg2[1,gsub('pctile','pctile.text',names.e.pctile[names.
!= 'pctile.neuro'])]) \cr # # pctile.text.pm "10.4 ug/m3 (76 # pctile.text.o3 "42.8 ppb
(22%ile)" \cr # pctile.text.cancer "55 lifetime risk per million (91%ile)" \cr # pctile.text.resp
"2.1 (72%ile)" \cr # pctile.text.dpm "0.401 ug/m3 (24%ile)" \cr # pctile.text.pctpre1960
"0.4 = fraction pre-1960 (68%ile)" \cr # pctile.text.traffic.score "23 daily vehicles/meters
distance (28%ile)" \cr # pctile.text.proximity.npl "0.071 sites/km distance (55%ile)"
\cr # pctile.text.proximity.rmp "0.085 facilities/km distance (21%ile)" \cr # pctile.text.proximity.
"0 facilities/km distance (26%ile)" \cr # pctile.text.proximity.npdes "0.25 facilities/km
```

```
distance (70%ile)" \cr # \cr t(bg2[125:126,gsub('pctile','pctile.text',names.e.pctile[names.e.pctil
!= 'pctile.neuro'])])
# 125 126 \cr # pctile.text.pm "8.37 ug/m3 (27%ile)" NA \cr # pctile.text.o3 "41.7 ppb (19%ile)"
NA \cr # pctile.text.cancer "36 lifetime risk per million (37%ile)" NA \cr # pctile.text.resp
"1.4 (37%ile)" NA \cr # pctile.text.dpm "0.275 ug/m3 (13%ile)" NA \cr # pctile.text.pctpre1960
"0.055 = fraction pre-1960 (27%ile)" "0 = fraction pre-1960 (10%ile)" \cr # pctile.text.traffic.score
"1.7 daily vehicles/meters distance (6%ile)" "0 daily vehicles/meters distance (2%ile)"
\cr # pctile.text.proximity.npl "0.056 sites/km distance (47%ile)" "0 sites/km distance
(16%ile)" \cr # pctile.text.proximity.rmp "0.046 facilities/km distance (7%ile)" "0 facilities/km
distance (1%ile)" \cr # pctile.text.proximity.tsdf "0 facilities/km distance (26%ile)"
"0 facilities/km distance (26%ile)" \cr # pctile.text.proximity.npdes "0.067 facilities/km
distance (16%ile)" "0 facilities/km distance (1%ile)" \cr # \cr # single result,e.g.:
"24% (36%ile)" \cr
```

Value

Returns character vector or data.frame, same shape as first input parameter.

See Also

esigfigs make.popup.d make.popup.e make.popup.ej pctileAsText

Examples

```
    #   Example: inputs are test0 and test1, and desired output is like test2
    #   (except note how prefix is added to each basename)

  test0 <- structure(list(
    e1 = c(0.185525372063833, 0.174428104575163, 0.485647788983707),
    e2 = c(0.131656804733727, 0.111928104575163, 0.671062839410395),
    other = c(NA, NA, 0.02)),
    .Names = c("e1", "e2", "other"),
    row.names = c(NA, 3L), class = "data.frame")
  test0

   test1 <- structure(list(
     pctile.e1 = c(27.1991395138354, 24.6836238179206, 72.382419748292),
     pctile.e2 = c(30.2662374847936, 26.761078397073, 78.2620665123235),
     other = c(NA, NA, 4)),
     .Names = c("pctile.e1", "pctile.e2", "other"),
     row.names = c(NA, 3L), class = "data.frame")
   test1

  test2 <- structure(list(
    pctile.text.e1 = c("19 (27%ile)", "17 (24%ile)","49 (72%ile)"),
    pctile.text.e2 = c("13 (30%ile)", "11 (26%ile)", "67 (78%ile)"),
    other = c(NA, NA, 4)),
    .Names = c("pctile.text.e1","pctile.text.e2", "other"),
    row.names = c(NA, 3L), class = "data.frame")
  test2

make.popup.e(test0, test1)
```

make.popup.ej *Make text to be shown in popups on EJ map*

Description

Takes percentiles (unlike make.popup.d or make.popup.e, which need raw values too), and presents those as a text field to be used as the text in a popup window on a map.

Usage

```
make.popup.ej(pctile, prefix = "pctile.text.", basenames)
```

Arguments

pctile	required integers 0 to 100
prefix	optional, default is 'pctile.text.' This is a text string specifying the first part of the desired resulting fieldname in outputs.
basenames	optional, default is 'pctile.xxx' where xxx is colnames(pctile). Defines colname(s) of outputs, which are the prefix plus this.

Details

Note pctile should be a (vector? or) data.frame of percentiles as INTEGER 0 to 100, NOT 0 to 1! Because that is how EJSCREEN data are stored. Might add code to handle cases like only one row, matrix not df, etc? Assume normal EJSCREEN pctile cols here would be like pctile.EJ.DISPARITY.pm.eo and then output popup col would be like pctile.text.EJ.DISPARITY.pm.eo In EJSCREEN, there are three types of pctile.text fields: E (text varies), D, EJ: 'pctile.text.cancer' "55 lifetime risk per million (91 'pctile.text.pctmin' "13 'pctile.text.EJ.DISPARITY.cancer.eo' "36

Value

Returns character vector or data.frame, same shape as pctile.

See Also

make.popup.d make.popup.e make.popup.ej pctileAsText

Examples

```
test1 <- structure(list(
 pctile.EJ.DISPARITY.pm.eo = c(43.1816682334032, 27.4198086017171, 71.7852110581344, NA),
 pctile.EJ.DISPARITY.o3.eo = c(47.1675935028896, 33.9578650432096, 69.7501760334948, NA)),
  .Names = c("pctile.EJ.DISPARITY.pm.eo", "pctile.EJ.DISPARITY.o3.eo"),
  row.names = c(1L, 2L, 3L, 126L), class = "data.frame")
test1
#    pctile.EJ.DISPARITY.pm.eo pctile.EJ.DISPARITY.o3.eo
#1                    43.18167                  47.16759
#2                    27.41981                  33.95787
#3                    71.78521                  69.75018
#126                        NA                        NA

test2 <- structure(list(
```

```
  pctile.text.EJ.DISPARITY.pm.eo = c("43%ile", "27%ile", "71%ile", NA),
  pctile.text.EJ.DISPARITY.o3.eo = c("47%ile", "33%ile", "69%ile", NA)),
  .Names = c("pctile.text.EJ.DISPARITY.pm.eo", "pctile.text.EJ.DISPARITY.o3.eo"),
  row.names = c(1L, 2L, 3L, 126L), class = "data.frame")
test2
#    pctile.text.EJ.DISPARITY.pm.eo pctile.text.EJ.DISPARITY.o3.eo
#1                           43%ile                         47%ile
#2                           27%ile                         33%ile
#3                           71%ile                         69%ile
#126                           <NA>                           <NA>

make.popup.ej(test1)
#  pctile.text.EJ.DISPARITY.pm.eo pctile.text.EJ.DISPARITY.o3.eo
#1                          43%ile                         47%ile
#2                          27%ile                         33%ile
#3                          71%ile                         69%ile
#4                            <NA>                           <NA>
```

`names.dvars` *Fieldnames of demographic columns in ejscreen package data*

Description

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

Usage

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

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.

See Also

ejscreenformulas names.evars names.dvars names.ejvars

`names.e.nice` *Nicer names for envt fields in ejscreen data*

Description

This data set provides nicer names for the ejscreen environmental indicator variables. These can be used to label graphs, for example.

Usage

```
data('names.e.nice')
```

Format

character vector

Details

Defaults to the latest (2016) version

See Also

ejscreenformulas names.evars names.dvars names.ejvars

`names.ejvars` *Fieldnames of environmental justice indicator columns in ejscreen package data*

Description

This data set provides variables that hold the colnames of environmental indicator fields in data.frames that may be used in the ejscreen 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"

And names.ej in turn is this, for example:

- [1] "EJ.DISPARITY.pm.eo"
- [2] "EJ.DISPARITY.o3.eo"
- [3] "EJ.DISPARITY.cancer.eo"
- [4] "EJ.DISPARITY.resp.eo"
- [5] "EJ.DISPARITY.dpm.eo"
- [6] "EJ.DISPARITY.pctpre1960.eo"
- [7] "EJ.DISPARITY.traffic.score.eo"
- [8] "EJ.DISPARITY.proximity.npl.eo"
- [9] "EJ.DISPARITY.proximity.rmp.eo"
- [10] "EJ.DISPARITY.proximity.tsdf.eo"
- [11] "EJ.DISPARITY.proximity.npdes.eo"

Details

This should have the latest (2016) version. Also see `names.ejvars16`). The 2015 version had neuro-related indicators in it, and is now in `names.ejvars15`.

Source

Names developed for this package. No external data source.

See Also

ejscreenformulas names.evars names.dvars names.ejvars

`names.ejvars15` *2015 Fieldnames of environmental justice indicator columns in ejscreen package data*

Description

This data set provides variables that hold the colnames of environmental indicator fields in data.frames that may be used in the ejscreen package to make it easier to refer to them as a vector, e.g., mydf[, names.ej15]

Usage

```
data('names.ejvars15')
```

Format

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

- "names.ej15"
- "names.ej.bin15"
- "names.ej.burden.eo15"
- "names.ej.burden.eo.bin15"
- "names.ej.burden.eo.pctile15"
- "names.ej.burden.svi615"
- "names.ej.burden.svi6.bin15"
- "names.ej.burden.svi6.pctile15"
- "names.ej.pct.eo15"
- "names.ej.pct.eo.bin15"
- "names.ej.pct.eo.pctile15"
- "names.ej.pct.svi615"
- "names.ej.pct.svi6.bin15"
- "names.ej.pct.svi6.pctile15"
- "names.ej.pctile15"
- "names.ej.svi615"
- "names.ej.svi6.bin15"
- "names.ej.svi6.pctile15"
- "namesall.ej15"
- "namesall.ej.bin15"

- "namesall.ej.pctile15"

And names.ej15 in turn is this, for example:

- [1] "EJ.DISPARITY.pm.eo"
- [2] "EJ.DISPARITY.o3.eo"
- [3] "EJ.DISPARITY.cancer.eo"
- [4] "EJ.DISPARITY.neuro.eo" Note neuro items are only in 2015 version
- [5] "EJ.DISPARITY.resp.eo"
- [6] "EJ.DISPARITY.dpm.eo"
- [7] "EJ.DISPARITY.pctpre1960.eo"
- [8] "EJ.DISPARITY.traffic.score.eo"
- [9] "EJ.DISPARITY.proximity.npl.eo"
- [10] "EJ.DISPARITY.proximity.rmp.eo"
- [11] "EJ.DISPARITY.proximity.tsdf.eo"
- [12] "EJ.DISPARITY.proximity.npdes.eo"

Details

This is the 2015 (obsolete) version. The 2015 version had neuro-related indicators in it, and is now in `names.ejvars15`.

Source

Names developed for this package. No external data source.

See Also

`ejscreenformulas` `names.evars` `names.dvars` `names.ejvars`

`names.ejvars16` *Fieldnames of environmental justice indicator columns in ejscreen package data*

Description

This data set provides variables that hold the colnames of environmental indicator fields in data.frames that may be used in the ejscreen 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"

And names.ej in turn is this, for example:

- [1] "EJ.DISPARITY.pm.eo"
- [2] "EJ.DISPARITY.o3.eo"
- [3] "EJ.DISPARITY.cancer.eo"
- [4] "EJ.DISPARITY.resp.eo"
- [5] "EJ.DISPARITY.dpm.eo"
- [6] "EJ.DISPARITY.pctpre1960.eo"
- [7] "EJ.DISPARITY.traffic.score.eo"
- [8] "EJ.DISPARITY.proximity.npl.eo"
- [9] "EJ.DISPARITY.proximity.rmp.eo"
- [10] "EJ.DISPARITY.proximity.tsdf.eo"
- [11] "EJ.DISPARITY.proximity.npdes.eo"

Details

This is the 2016 version

Source

Names developed for this package. No external data source.

See Also

ejscreenformulas names.evars names.dvars names.ejvars names.ejvars15

`names.evars` *Fieldnames of environmental indicator columns in ejscreen package data*

Description

This data set provides variables that hold the colnames of environmental indicator fields in data.frames that may be used in the ejscreen 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). For the latest (2016) version of EJSCREEN:

- "names.e" (pm, o3, cancer, 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)

For 2015 version of EJSCREEN it was:

- "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)

Details

NOTE: This used to provide the 2015 version's list, which had "neuro" in it, but now defaults to the latest (2016) version

Source

Names developed for this package. No external data source.

See Also

names.e.nice ejscreenformulas names.dvars names.ejvars

`names.evars15` *2015 Fieldnames of environmental indicator columns in ejscreen package data*

Description

This data set provides variables that hold the colnames of environmental indicator fields in data.frames that may be used in the ejscreen package to make it easier to refer to them as a vector, e.g., mydf[, names.e15]

Usage

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

Format

A series of variables (each is a character vector of colnames). The 2016 version of EJSCREEN was:

- "names.e" (pm, o3, cancer, 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)

For 2015 version of EJSCREEN it was changed so names include 15, to distinguish:

- "names.e15" (pm, o3, cancer, neuro, resp, dpm, pctpre1960, traffic.score, proximity.npl, proximity.rmp, proximity.tsdf, proximity.npdes)
- "names.e.bin15"
- "names.e.pctile15"
- "Elist15" (this one is like names.e, but as a list, not a vector)

Details

NOTE: This is to provide the 2015 version, which had "neuro" in it

Source

Names developed for this package. No external data source.

See Also

`ejscreenformulas` `names.evars` `names.dvars` `names.ejvars`

`names.evars16` *Fieldnames of environmental indicator columns in ejscreen package data*

Description

This data set provides variables that hold the colnames of environmental indicator fields in data.frames that may be used in the ejscreen 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). For 2016 version of EJSCREEN:

- "names.e" (pm, o3, cancer, 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)

For 2015 version of EJSCREEN it was:

- "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)

Details

This is the 2016 version

Source

Names developed for this package. No external data source.

See Also

`names.e.nice` `ejscreenformulas` `names.dvars` `names.ejvars`

pctileAsText *Utility function in showing a percentile as popup text*

Description

Converts numeric percentiles (0-100) into character (text) that converts 95.3124 to '95

Usage

```
pctileAsText(x)
```

Arguments

x	vector or data.frame of numeric values 0 to 100 (not 0 to 1), representing percentiles from EJSCREEN dataset

Value

Returns matrix/vector of same shape as x if x was data.frame/vector

Examples

```
 ## Not run:
 (bg2[ 125:126, c('pctile.pctmin', 'pctile.EJ.DISPARITY.pm.eo') ])
 (bg2[ 125:126, c('pctile.text.pctmin', 'pctile.text.EJ.DISPARITY.pm.eo') ])
 pctileAsText(bg2[ 125:126, c('pctile.pctmin', 'pctile.EJ.DISPARITY.pm.eo') ])

## End(Not run)
```

popupunits *Units of measurement for environmental indicators*

Description

Table indicating what units to use, such as ug/m3, in showing environmental indicators in EJSCREEN, as shown in popup windows on maps

Usage

```
data('popupunits')
```

Format

A data.frame:

```
> str(popupunits)
'data.frame': 11 obs. of 2 variables:
$ evar : chr "pm" "o3" "cancer" ...
$ units: chr "ug/m3" "ppb" "lifetime risk per million" "" ...
> popupunits
```

```
evar units
1 pm ug/m3
2 o3 ppb
3 cancer lifetime risk per million
4 resp
5 dpm ug/m3
6 pctpre1960 = fraction pre-1960
7 traffic.score daily vehicles/meters distance
8 proximity.npl sites/km distance
9 proximity.rmp facilities/km distance
10 proximity.tsdf facilities/km distance
11 proximity.npdes facilities/km distance
```

Source

See related Technical Documentation at http://www.epa.gov/ejscreen

See Also

make.popup.e names.e

---

ustotals *Get US Totals and Percentages Overall for EJSCREEN Fields*

Description

This function simply takes a data.frame of EJSCREEN demographic data and returns the total count or overall US percentage for various fields, by using the appropriate denominator (universe) to calculate any given percentage. For example, PCTLOWINC.US equals sum(lowinc) / sum(povknownratio), not sum(lowinc) / sum(pop). This function is hard-coded to use specified field names referring to EJSCREEN variables. This function is not needed to create an EJSCREEN dataset, but is convenient if one wants US summary values.

Usage

```
ustotals(bg)
```

Arguments

bg	Must be a data.frame that has the following colnames: • pop, • lowinc, • mins, • under5, • over64, • lths, • lingiso, • pre1960, • hisp,

- nhwa,
- nhba,
- nhaiana,
- nhaa,
- nhnhpia,
- nhotheralone,
- nhmulti,
- povknownratio,
- age25up,
- hhlds,
- builtunits

Value

Returns a named list of US totals and percentages (as fractions 0-100) (e.g., POP.US=xxxx, etc.):

- POP.US,
- LOWINC.US,
- MINS.US,
- UNDER5.US,
- OVER64.US,
- LTHS.US,
- LINGISO.US,
- PRE1960.US,
- HISP.US,
- NHWA.US,
- NHBA.US,
- NHAIANA.US,
- NHAA.US,
- NHNHPIA.US,
- NHOTHERALONE.US,
- NHMULTI.US,
- PCTLOWINC.US,
- PCTMIN.US,
- PCTUNDER5.US,
- PCTOVER64.US,
- PCTLTHS.US,
- PCTLINGISO.US,
- PCTPRE1960.US,
- PCTHISP.US,
- PCTNHWA.US,
- PCTNHBA.US,

- PCTNHAIANA.US,
- PCTNHAA.US,
- PCTNHNHPIA.US,
- PCTNHOTHERALONE.US,
- PCTNHMULTI.US

Examples

```
# tots <- ustotals(bg)
tots <- list(POP.US = 314107084,
LOWINC.US = 105773407, MINS.US = 116947592,
UNDER5.US = 19973711, OVER64.US = 43177961,
LTHS.US = 28587748, LINGISO.US = 5275272,
PRE1960.US = 39159200,
HISP.US = 53070096,
NHWA.US = 197159492, NHBA.US = 38460598,
NHAIANA.US = 2082768, NHAA.US = 15536209,
NHNHPIA.US = 493155, NHOTHERALONE.US = 611881,
NHMULTI.US = 6692885,
PCTLOWINC.US = 0.345409177890786, PCTMIN.US = 0.372317588354677,
PCTUNDER5.US = 0.0635888587600272, PCTOVER64.US = 0.137462550828685,
PCTLTHS.US = 0.136746758570279, PCTLINGISO.US = 0.0453938768598784,
PCTPRE1960.US = 0.295004484408374,
PCTHISP.US = 0.168955425405178,
PCTNHWA.US = 0.627682411645323, PCTNHBA.US = 0.122444223512005,
PCTNHAIANA.US = 0.00663075780869686, PCTNHAA.US = 0.0494615046631677,
PCTNHNHPIA.US = 0.00157002189737306, PCTNHOTHERALONE.US = 0.00194800127462264,
PCTNHMULTI.US = 0.0213076537936343)

 # Display as a nice table with two columns, rounded numbers, rownames and colnames
tots <- round(cbind(unlist(tots)), 2)
totrownames <- rownames(tots)[1:16]
tots <- cbind(tots[1:16], c(1, tots[17:31]))
rownames(tots) <- totrownames
colnames(tots) <- c('count', 'pct')
tots

usapprox <- data.frame(
 pop=rep(1419.767,217739),lowinc=464.4692,mins=515.4554,under5=92.48634,
 over64=186.7899,lths=134.0128,lingiso=24.68058, pre1960=183.3237,hisp=232.1370,
 nhwa=904.3119,nhba=173.5408,nhaiana=9.418460, nhaa=67.47893,nhnhpia=2.204764,
 nhotheralone=2.829952,nhmulti=27.84555, povknownratio=1383.92,age25up=938.4447,
 hhlds=529.1969,builtunits=604.5883
 )
cbind( ustotals(usapprox))
```

Index