Package 'SeerMapper'

October 12, 2022

Type Package

Version 1.2.5

Date 2021-01-08

Title A Quick Way to Map U.S. Rates and Data of U.S. States, Counties, Census Tracts, or Seer Registries using 2000 and 2010 U.S. Census Boundaries

Author Jim Pearson <jbpearson353@gmail.com>

Maintainer Joe Zou <zouj@imsweb.com>

Imports graphics, maptools, RColorBrewer, rgdal, sp, stats, stringr, methods, SeerMapperRegs, SeerMapperEast, SeerMapperWest, SeerMapper2010Regs, SeerMapper2010East, SeerMapper2010West

Depends R (>= 3.6.0)

Suggests

Description Provides an easy way to map seer registry area rate data on a U.S. map. The U.S. data may be mapped at the state or state/county. U.S. Seer registry data may be mapped at the Seer registry area, Seer Registry area/county or Seer Registry area/county/census tract level. The function uses a calculated default categorization breakpoint list for 5 categories, when not breakpoint list is provided or the number of categories is specified by the user. A user provided break point list is limited to containing 5 values. The number of calculated categories may be from 3 to 11. The user provide the p-value for each area and request hatching over any areas with a p-value > 0.05. Other types of comparisons can be specified. If states or state/counties are used, the area identifier is the U.S. FIPS codes for states and counties, 2 digits and 5 digits respectfully. If the data is for U.S. Seer Registry areas, the Seer Registry area identifier used to link the data to the geographical area is a Seer Registry area abbreviation. See documentation for the list of acceptable Seer Registry area names and abbreviations. The state boundaries are overlaid all rate maps. The package contains the boundary data for state, county, Seer Registry areas, and census tracts for counties within a Seer registry area. The package support boundary data from the 2000 and 2010 U.S. Census. The SeerMapper package version contains the U.S. Census 2000 and 2010 boundary data for the regional, state, Seer Registry, and county levels. Six supplement packages

contain the census tract boundary data.
Copyrighted 2014, 2015, 2016, 2017, 2018 and 2019 by Pearson and Pickle.

License GPL (>= 2)

LazyData no

LazyLoad no

BuildResaveData yes

ByteCompile yes

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-12 21:00:17 UTC

R topics documented:

`SeerMapper-package` *A graphics package to easily and quickly create U. S. maps at the state, Seer registry areas, county or census tract levels*

Description

The *SeerMapper* package provide a means of creating U.S. maps of categorized data on at the state, state/county, state/county/census tract or U.S. Seer registry levels. The *SeerMapper* package contain the R script to create the maps and the U. S. Census boundaries files for States, District of Columbia, and Puerto Rico, the counties for each state/territory, the census tracts for each state and NCI Seer Registries. The main package contains the code and the census year 2000 and 2010 state, registry and county boundary data and information. The census tract boundary data could not fit into the package and stay within the CRAN 5 megabyte package size limitation. Six(6) data only packages released with this package contain *SeerMapper*'s simplified U. S. 2000 and 2010 census tract boundary data: *SeerMapperRegs* and *SeerMapper2010Regs* packages containing the census tracts for any state with a Seer Registry for years 2000 and 2010, *SeerMapperEast* and *SeerMapper2010East* containing the census tracts for states east of the Mississippi river that don't have registries for years 2000 and 2010, and *SeerMapperWest* and *SeerMapperWest* containing the census tract boundary data for states without registries weat of the Mississippi for years 2000 and 2010. When *SeerMapper* is installed, the other six(6) packages are also installed. When *SeerMapper* is executing, it manages the loading of the six packages based on the level of map being requested by the caller.

To map at the state level, data must use the 2 digit U. S. state fips codes as the location identifiers.

To map at the HSA level, data must use the 3 digit HSA numbers as the location identifiers.

To map at the county level, data must use the 5 digit U. S. state/county fips codes as the location identifiers.

To map at the census tract level, data must use the 11 digit U. S. state/county/tract fips codes as the location identifiers.

To map at the U. S. NCI Seer registry area level, the location identifier must be the Seer Registry abbreviation (see documentation on the Seer registry boundary files for more details.) The basic packages only contain the census tract boundaries for the states which contain Seer Registries.

The goal was to allow a user the simpliest way to create maps of data and rates in the states, counties, Seer Registries, and census tracts in states with Seer Registries using one package for one census year. If additional states required census tract mapping, this is not precluded, just requires a little more setup work. Because of the numerous changes in the fips codes, census tracts boundaries and even county boundaries between 2000 and 2010, it was necessary to keep the boundary datasets separate. The result is two sets of packages. Make sure the location identifiers in your data match census year of the the base package your are using. If mis-matches are detected, warnings are generated document the missing or additional areas to help the user handle the situation. As much of the map is still generated as possible.

Call options are available to help manage: the inclusion, sizing and placement of the legend, hatching of areas, limiting mapping to only the contiguous 48 states, and the categorizing and coloring of areas based on the data.

The package will color the areas with data based on a user provided set of break points or on a number of categories desired. See the '`categ`' call parameter documentation for more details.

A hatching option controls how the package will overlay hatching of areas based on a critria and value. The default setting handle P Values in the range of 0 to 1, with a test value of 0.05 and areas hatched if their value is greater than the test value (>). See the documentation on the '`hatch`' call parameter for more details.

All 50 states, District of Columbia and Puerto Rico can be mapped using FIPS codes.

The boundaries for Alaska and Hawaii have been adjusted and scaled be provide a reasonable presentation with the mainland and relocated to just south of California and Arizona.

Details

Package:	SeerMapper
Type:	Package
Version:	1.2.4
Date:	2020-06-22
License:	GPL (>=2)
LazyLoad:	no

The function accepts a data.frame containing an area/location identifier, data (rates, data, or categories) for the areas and an hatching data. The hatching data is generaly a p_value for the data (rates), but can be any value. If the data is not a p_value, then the user must provide the test value and the test operation to determine if hatch is required. Each row represents a State, State/County, State/County/Census Tract, or U.S. Seer registry area. The data(rates) are categorized by default into 5 categories. The breakpoints for the categories are calculated based on the quartiles for the data. The caller can specify the categorization be done on from 3 to 11 categories. The caller can also specify the specific breakpoints to use in the categorization of the data via a numeric vector like c(0.6, 0.8, 1.0, 1.2, 1.4) for up to 10 breakpoints. The maximum number of categories is set by the color palette selected from the RColorBrewer palettes. The default palColors setting is the "RdYlBu" palette which can support up to 11 categories. The default '`categ`' value is *5*.

If hatching data is provided, the default is to treat the data as p_values. When hatching is enabled ('`hatch`'=TRUE), the function will overlay a hatch pattern on any area with a p_value >= 0.05. This can be changed using the *ops* and *value* settings in '`hatch`' list call parameter.

If the data is for states, state/counties, state/county/census traces, the area's (location) identify must be the U.S. FIPS codes. If the data is for U.S. Seer registry areas, the area identify is the Seer registry area abbreviation (See the *st99_d00* dataset documenation for more details.)

The boundary dataset for the Seer registry area is contained in this package as the 2000 Census.

The typical call sequence is:

```
library(SeerMapper)
library(maps)

# create data data.frame

rateData <- data.frame(Fips=state.fips$fips,
                       RateRatio=stateRateVector,
                       pValue=stateRatePValue)

SeerMapper(rateData, Title=c("Test Map"))
```

To map census tracts in state without Seer Registries, the call sequence is:

```
library(SeerMapper)
library(maps)

# create data data.frame

rateData <- data.frame(Fips=cenTract.fips,
                       RateRatio=ctRateVector,
                       pValue=ctRatePValue)

SeerMapper(rateData, Title=c("Test Non-Seer State Census Tract Map"))
```

The data passed to the function will dictate which areas of the U.S. are mapped. In general only states with data are mapped.

INSTALLATION:

Due to CRAN R package size limitations, the base *SeerMapper* package could only contain the boundary information datasets for: the states, DC and PR boundaries, the Seer Registry boundaries, and the boundaries for the counties in all states. The census tract boundaries are supplied via six(6) additional R packages: *SeerMapperRegs*, *SeerMapperEast*, *SeerMapperWest*. *SeerMapper2010Regs*, *SeerMapper2010East*, and *SeerMapper2010West*. The packages containing "2010" in their name support the census year of 2010. The packages without "2010" support the census year of "2000". When *SeerMapper* is installed, the package dependencies have been set up to force the installation of the six(6) census tract package automatically. The user should not have install these packages manually. The installation log will notify the user of the additional downloads and installations.

If the location IDs in the data cannot be associated with loaded boundary datasets, messages are generated to identify the packages that may need to be installed or loaded the library function.

The base package contains boundary information for:

- all states and DC,
- all Seer Registries,
- all Health Service Areas (HSA) for all states and DC,
- counties in all states,

The "Regs" boundary package contains the 2000 or 2010 census tract boundary data for 19 states:

Alaska, Arizona, California, Connecticut, Georgia, Hawaii, Idaho, Iowa, Kentucky, Louisiana, Massachusetts, Michigan, New Jersey, New Mexico, New York, Oklahoma, Utah, Washington, and Wisconsin.

The "East" boundary package contains the 2000 or 2010 census tract boundary data for 20 states and DC:

Alabama, Delaware, District of Columbia, Florida, Illinois, Indiana, Maine, Maryland, Mississippi, New Hampshire, North Carolina, Ohio, Pennsylvania, Rhode Island, South Carolina, Tennessee, Vermont, Virginia, West Virginia, and Puerto Rico.

The "West" boundary package contains the 2000 or 2010 census tract boundary data for 13 states:

Arkansas, Colorado, Kansas, Minnisota, Missouri, Montana, Nebraska, Nevada, North Dakota, Oregon, South Dakota, Texas, and Wyoming.

To do census tract mapping, the package automatically loaded needed census tract boundary packages according to the census year and location IDs in data.frame provided during the call. If a package is missing or cannot be loaded, the caller will be given a message identifying the problem.

```
library(SeerMapper)
```

and all of the boundary information is will be available.

Author(s)

James B Pearson, Jr <jbpearson353@gmail.com>
Maintainer: "Joe Zou" <zouj@imsweb.com>

`area_Metro_XX_Data` *Metro Combined Statistics Area datasets for County or census tract for Test data*

Description

This document described the contain of eight datasets included in the package. They represent demographic data at the county level and census tract level for the Washington-Baltimore and Kansas City Metro Combined Statistical areas for the 2000 and 2010 census years. Each metro area cosists of counties or tracts in multiple states. The Washington-Baltimore Metro CSA consists of an area that covers 5 states/districts (Maryland, Virginia, Pennsylvania, West Virinia, and the District of Columbia) and includes 40 counties and 2178 census tracts. The Kansas City Metro CSA consists of an area that covers 2 states (Kansas and Missouri) and 22 counties and 604 census tracts.

Usage

```
data(WashBaltMetro_Co_Data)
```

Format

The format of each datasets is the same. The only difference is the location ID used in each data set and the number of records and census year of the data. The file naming convention is: nnnnn-MetroYY_AA_Data, where nnnnn is the name of the metro area (KC or WashBalt), YY is "" for census year 2000 and "10" for census year 2010, and AA is the type of sub-areas ("Co" = country, and "Tr" = census tract.) In the county datasets, the location ID is the 5 digit state/county FIPS code. In the census tract datasets, the location ID is the 11 digit state/county/tract FIPS code.

These datasets are used to demonstrate how *SeerMapper* can work with county or census tracts data to create maps across multiple states.

The Kansas City and Washington-Baltimore Combined Statistics Area 2000 data at the county and census tract level contains the following demographic data for each sub-area. The data contains 9 columns:

FIPS is a character vector of the U.S. State/County/Tract FIPS code (11 digits)

pop2000 is an integer containing the 2000 tract population.

age.65.up is an integer representing the population of the people with the age of 65 or higher.

hh.units is an integer representing the number of physical household units in the census tract.

hh.occupied is an integer representing the number of hh.units that are occupied. (hh.units-hh.occupied) equals the number of households vacant.

hh.owner is an integer representing the number of occupied household units owned by the occupants. (hh.occupied-hh.owner) equals the number of units that are occupied by renters.

areasm is a numeric area of the tract in square miles.

popdens is a numeric containing the 2000 population density of the tract.

Source

The U. S. Census Bureau 2000 and 2010 census tract demographic dataset from the CENSUS.GOV website demographic data files for the 2000 census as pulled on February, 2017. The R package UScensus2000tract was used to get the census tract data based on the 2001 release. The dataset was then aggregated to create the county dataset.

---

co99_d00 *Information table for the counties in the US. Links the county FIPS code to Registries.*

Description

This dataset contains a data.frame table used to provide the county level information for all of the counties in the U. S. to the *SeerMapper* package. It provides the mapping of the county 5 digit FIPS code to the state 2 digit FIPS code, the associated Seer Registry abbreviation, the county name, and statistics on the number of census tracts in each county for the 2000 and 2010 census years. The information was originally stored in the Seer_stcoIDtosaID dataset and the coxx_dxx datasets for each county and was moved to this data.frame when the 2000 and 2010 census year county information was consolidated.

Usage

```
data(co99_d00)
```

Format

The dataset contains a single data.frame named *co99_d00* that supports both the 2000 and 2010 census years.

Since a registry area may contain multiple counties, this data.frame is used to mark each county with the Seer Registry abbrevation it is a member. The 20 U. S. Seer registry areas are identify by the following character abbreviations (*saID*):

saID	stID	Name
AK-NAT	02	Alaska Natives
AZ-NAT	04	Arizona Indians
CA-OTH	06	California Other (not SF-Oakland, SJ-Monterey, LA areas)

CA-SF	06	California San Fran/Oakland
CA-SJ	06	California San Jose/Monterey
CA-LA	06	California Los Angeles
CT	09	Connecticut
GA-OTH	13	Georgia Other (not Atlanta or Rural areas)
GA-ATL	13	Georgia Atlanta
GA-RUR	13	Georgia Rural
HI	15	Hawaii
IA	19	Iowa
KY	21	Kentucky
LA	22	Louisiana
MI-DET	26	Michigan Detroit
NJ	34	New Jersery
NM	35	New Mexico
OK-CHE	40	Oklahoma Cherokee Nation
UT	49	Utah
WA-SEA	53	Washington Seattle-Puget Sound

The above table also contains the Registry's full name and the state's 2 digit FIPS code where the registry resides. For more information on the NCI Seer Registry boundary data and information, refer to the document section on the *sa99_d00* dataset.

Details

The *co99_d00* dataset contains the following fields (columns):

row.names = a character vector equal to the 5 digit U. S. state and county FIPS code.

coName = a character vector of the county's name.

saID = a character vector identifying the Seer Registry Abbreviation containing the county. If the county is not a member of a registry, then the field is set to NA.

c_X_00 = A numeric value of the X coordinate of the centroid of the county in the 2000 census.

c_Y_00 = A numeric value of the Y coordinate of the centroid of the county in the 2000 census.

c_X_10 = A numeric value of the X coordinate of the centroid of the county in the 2010 census.

c_Y_10 = A numeric value of the Y coordinate of the centroid of the county in the 2010 census.

tracts00 = An integer value of the number of 2000 census tracts contained in the county.

tracts10 = An integer value of the number of 2010 census tracts contained in the county.

y = An integer indicating which census years the county was present. 1 = 2000, 2 = 2010, and 3 = both.

HSAID = A character vector containing the 3 digit HSA number that contains the county.

coXX_dXX *U. S. State County Boundary Spatial Polygons datasets for U. S. States, the District of Columbia and Puerto Rico for the 2000 and 2010 Census*

Description

This document describes the content and format of the 52 county boundary .rda datasets integrated in this package for 2000 and 2010 census years. The *SeerMapper* package contains a set of boundary ".rda" files for the 52 states', D. C.'s and Puerto Rico county or parish sub-areas for both the 2000 and 2010 census year. Each dataset contains the boundary information for all of the county/parish sub-area in that state, district or territory. The dataset name and object name convention used is coXX.dYY, where "co" indicates county boundaries, XX is the state FIPS code and YY is the last 2 digits of the census year (*00* or *10*). Only three states have a different set of counties and boundaries: Colorado, Alaska, and Washington. Only these states have two county ".rda" files in the package - one for each census year. All other states use the "00" version for both census years.

Usage

```
data(co01_d00)
```

Format

Each dataset contains a SpatialPolygons structure representing the county boundaries for all counties in a single state. Each "polygons" item in the structure represents one county and all of the polygons required to define the county, including lakes, holes, etc. The boundary files are based on the boundaries used in the 2000 and 2010 Census. The coordinates system used in these rda files have been transformed from the census Long/Lat values to an equal area projections for mapping.

Details

Each dataset (.rda file) represents the county boundaries for one state for the 2000 and 2010 U. S. census. In the 2010 census the counties in three states, Alaska, Colorado, and Virginia changed. In all other cases, the 2000 census county boundaries can be reused when mapping 2010 location IDs. To handle the three exceptions, three additional county boundary datasets are included and used by the package when the *censusYear* is set to *2010* and counties in these states are referenced in the data (at the county or tract level.)

Based on the locations on the rate or category data provided by the user, only needed county boundary information is loaded by the package for the referenced states. The multiple sate boundary data maybe accumulated by the package and used for mapping.

The data formats are the same for each county boundry .rda in this package, a single SpatialPolygons variable. The format of the dataset names is "co", the two digit state FIPS code, "_d", the two digit census year (00 or 10), and ".rda". The *SeerMapper* package contains the set of boundary datasets for all of the U. S. 2000 Census counties as individual datasets named coXX_d00 and the three additional boundary datasets for the three states where the counties changed in the 2010 census.

Each row.name of the SpatialPolygons is a polygons slot in the data structure and represents a set of Polygons that form the county. The row.names used are the the 5 digit FIPS state/county code associated with the county.

The county information that may normally be carried as a data.frame with a SpatialPolygonsDataFrame is stored in the *co99_d00* dataset in the package. This allows the package to all of the county information without having to load the boundary data.

The county boundary information has been transformed from the U. S. Census boundary data's lat/long coordinates under NAD83 to the Alber equal area cartisian coordinates.

The PROJ.4 projection used for transformation of these boundaries is:

"+proj=aea +lat_1=33 +lat_2=49 +lat_0=39 +lon_0=96W +ellps=WGS84"

All boundary data used by *SeerMapper* were transformed to this projection when the package is constructed.

All the ".rda" files are compressed using the "xy" method to reduce the disk space requirements.

All of the boundary shape file data was imported from the U. S. Census website on July 3, 2016, processed by the urlhttp://www.MapShaper.org website version 01.15 using modified Visvalingam method with intersection repair enabled, prevent shape removal enabled, and a coordinate precision value of 0.0 to simplify the boundaries from 100 space requirements by 90 and edge alignments. The simplified boundaries were then downloaded and converted to SpatialPolygons variables for use by *SeerMapper*.

Source

The county boundary files were obtained from the Census.gov website for the 2000 and 2010 year census under the Map & Data category, cartagraphic boundaries. The shapefiles used to create these boundaries were downlaoded for the U. S. Census website URLhttp://www.census.gov/geo/maps-data/data/cbf/cbf_counties.html on July 3, 2016.

`hs99_d00` *Information table for the U.S. Health Service Areas Links the HSA 3 digit numbers to Registries and states.*

Description

This dataset contains a data.frame table used to provide the health service area (HSA) level information for all of the HSAs in the U. S. to the *SeerMapper* package. It provides the mapping of the HSA 3 digit number to the state 2 digit FIPS code, the associated Seer Registry abbreviation, the HSA name, and statistics on the number of census tracts and counties in each HSA for the 2000 and 2010 census years.

Usage

```
data(hs99_d00)
```

Format

The dataset contains a single data.frame named *co99_d00* that supports both the 2000 and 2010 census years.

Since a registry area may contain multiple HSAs, this data.frame is used to mark each HSA with the Seer Registry abbrevation it is a member. The 20 U. S. Seer registry areas are identify by the following character abbreviations (*saID*):

saID	stID	Name
AK-NAT	02	Alaska Natives
AZ-NAT	04	Arizona Indians
CA-OTH	06	California Other (not SF-Oakland, SJ-Monterey, LA areas)
CA-SF	06	California San Fran/Oakland
CA-SJ	06	California San Jose/Monterey
CA-LA	06	California Los Angeles
CT	09	Connecticut
GA-OTH	13	Georgia Other (not Atlanta or Rural areas)
GA-ATL	13	Georgia Atlanta
GA-RUR	13	Georgia Rural
HI	15	Hawaii
IA	19	Iowa
KY	21	Kentucky
LA	22	Louisiana
MI-DET	26	Michigan Detroit
NJ	34	New Jersery
NM	35	New Mexico
OK-CHE	40	Oklahoma Cherokee Nation
UT	49	Utah
WA-SEA	53	Washington Seattle-Puget Sound

The above table also contains the Registry's full name and the state's 2 digit FIPS code where the registry resides. For more information on the NCI Seer Registry boundary data and information, refer to the document section on the *sa99_d00* dataset.

Details

The *hs99_d00* dataset contains the following fields (columns):

row.names = a character vector equal to the 3 digit U. S. HSA number.

HSAID = a charactor vector of the 3 digit HSA number. Same as the row.names.

HSA = a numeric value of the HSA number. Same as the row.names.

HSA_Name - a character vector containing the assigned name of the health service area.

stID the state 2 digit FIPS code containing the HSA.

y = integer indicating which census years the HSA was active during. 1=2000, 2=2010, 3=both.

Chg10 = logical flag indicating the HSA changed in the 2010 census and to use the _d10.rda files for to get boundaries for this HSA.

saID = a character vector identifying the Seer Registry Abbreviation containing the county. If the county is not a member of a registry, then the field is set to NA.

c_X_00 = A numeric value of the X coordinate of the centroid of the county in the 2000 census.

c_Y_00 = A numeric value of the Y coordinate of the centroid of the county in the 2000 census.

c_X_10 = A numeric value of the X coordinate of the centroid of the county in the 2010 census.

c_Y_10 = A numeric value of the Y coordinate of the centroid of the county in the 2010 census.

county00 = An integer value of the number of 2000 counties are contained in the HSA.

county10 = An integer value of the number of 2010 counties are contained in the HSA.

tracts00 = An integer value of the number of 2000 census tracts are contained in the HSA.

tracts10 = An integer value of the number of 2010 census tracts are contained in the HSA.

`hsXX_dXX` *U. S. Health Service Area Boundary Spatial Polygons datasets for U. S. States and the District of Columbia for the 2000 and 2010 Census*

Description

This document describes the content and format of the 942 health service area boundary .rda datasets integrated in this package for 2000 and 2010 census years. The *SeerMapper* package contains a set of boundary ".rda" files for the health service areas for all 51 states and the D. C. sub-areas for both the 2000 and 2010 census year. The dataset name and object name convention used is hsXX.dYY, where "hs" indicates health service area (HSA) boundaries, XX is the state FIPS code and YY is the last 2 digits of the census year (*00* or *10*). Only three states have a different set of counties and boundaries: Colorado, Alaska, and Virginia. These states have two HSA ".rda" files in the package - one for each census year. All other states use the "00" version for both census years.

Usage

```
data(hs01_d00)
```

Format

Each dataset contains a SpatialPolygons structure representing the health service area boundaries in a single state. Each "polygons" item in the structure represents one HSA and all of the polygons required to define the HSA, including lakes, holes, etc. The HSA boundary files are based on the census tract boundaries used in the 2000 and 2010 Census. The coordinates system used in these rda files have been transformed from the census Long/Lat coordinates values using NAD83 to an Alber equal area projections for mapping.

Details

Each dataset (.rda file) represents the boundaries for the HSAs in one state for the 2000 and 2010 U. S. census. In the 2010 census the counties in three states, Alaska, Colorado, and Virginia changed in size and shape. In all other cases, the 2000 census county boundaries can be reused when mapping 2010 location IDs. The county boundaries were used to build the HSA boundaries. To handle the three exceptions, three additional HSA boundary datasets are included and used by the package when the *censusYear* is set to *2010* and HSAs in these states are referenced in the data (at the HSA, county or tract level.)

Based on the locations on the rate or category data provided by the user, only needed HSA boundary information is loaded by the package for the referenced states. The multiple state boundary data maybe accumulated by the package and used for mapping.

The data formats are the same for each HSA boundry .rda in this package, a single SpatialPolygons variable. The format of the dataset names is "hs", the two digit state FIPS code, "_d", the two digit census year (00 or 10), and ".rda". The *SeerMapper* package contains the set of boundary datasets for all of the U. S. 2000 Census HSAs as individual datasets named hsXX_d00 and the three additional boundary datasets for the three states where the counties changed in the 2010 census.

Each row.name of the SpatialPolygons is a polygons slot in the data structure and represents a set of Polygons that form the HSA. The row.names used are the the 3 digit HSA numbers assigned by NIH.

The HSA information that may normally be carried as a data.frame with a SpatialPolygonsDataFrame is stored in the *hs99_d00* dataset in the package. This allows the package to all of the county information without having to load the boundary data.

The HSA boundary information has been transformed from the U. S. Census boundary data's lat/long coordinates under NAD83 to the Alber equal area cartisian coordinates.

The PROJ.4 projection used for transformation of these boundaries is:

"+proj=aea +lat_1=33 +lat_2=49 +lat_0=39 +lon_0=96W +ellps=WGS84"

All boundary data used by *SeerMapper* were transformed to this projection when the package is constructed.

All the ".rda" files are compressed using the "xy" method to reduce the disk space requirements.

All of the boundary shape file data was imported from the U. S. Census website on July 3, 2016 for the 2000 and 2010 census, processed by the urlhttp://www.MapShaper.org website version 01.15 using modified Visvalingam/weighted method with intersection repair enabled,prevent shape removal enabled, and a coordinate precision value of 0.0 to simplify the boundaries from 100 and disk space requirements by 90 a visually usable boundary image and edge alignments. The simplified boundaries were then downloaded and converted to SpatialPolygons variables for use by *SeerMapper* and transformed to the Alber equal area projection. The initial files were census tract boundaries. These files were aggregated into county boundaries. The county boundaries were aggregated into the HSA boundaries, Seer Registry, and State boundaries. The State boundaries were aggreated into the regional boundaries.

Source

The HSA boundary files were obtained from the Census.gov website for the 2000 and 2010 year census under the Map & Data category, cartagraphic boundaries. The census tract shapefiles used to

create these boundaries were downlaoded for the U. S. Census website URLhttp://www.census.gov/geo/maps-data/data/cbf/cbf_counties.html on July 3, 2016. The census tract boundaries were aggregated into county boundaries. The county boundaries were aggregated into the HSA, Registry, and State boundaries.

`messages` *SeerMapper Generated Warning and Error Messages*

Description

SeerMapper package tries to anticipates error and problems in the R environment and generate more specific warnings and error messages to help the user understand the problems and effect a solution and avoid as many cryptic R type messages.

Each message, warning and error message is documented in this section with the general form of the message and a friendly explanation and advice on what may be wrong and how to fix the issue identified.

All *SeerMapper* messages all start with "***" to help quickly find them in the warnings() logs and general output.

The general format of the messages is:

```
***XXX text of message
```

where the XXX is a unique numeric identifier for the message.

When requesting support, it is recommended to include a copy of the data and r script used to call the package and a copy of the output log of the preparation and execution of *SeerMapper*.

Details

Conventions: The message may contain dynamic values based on information and variables related to the problem and provided by the caller. These values appear in message below as <nnnnn> strings. The following is a list of the possible <nnnnn> strings the message may contain and their definition.

The <nnnnn> string will be replaced with a value relevent to the *SeerMapper* function call being serviced.

<badColors> a list of the bad colors found in the *dataCol* when *categ* was set to "COLORS".

<badIDs> A list of invalid location IDs found in the *idCol* column of the *<ndfName>* data.frame.

<badOptions> a list of invalid options in either the *hatch* or the *mLegend* call parameter lists.

<badValues> a list of bad data values found in the *<dataCol>* in the *<ndfName>* data.frame.

<catMaxNum> the maximum number of categories the package will support.

<censusYear> A character vector used to contain the 4 digit census year of the boundary data files to use with the data location IDs. The values of "2000" and "2010" are supported.

<color> A character vector representating the name of a colors from the *colors()* function or a 6 or 8 digit hexidecimal value (e.g., "#rrggbb" or "#rrggbbaa"). In the warning/error message this string is a color value that is not valid.

<dataBCol> A character vector defining the color to be used as the borders on the sub-areas containing data. This may be the state, county or tract boundaries depending on what is being mapped.

<dataCol> a column name or number provided in the *dataCol* call parameter. It identifies the column in the *<ndfName>* data.frame, that contains the data to be categorized, used a the integer category value, or the fill color for the sub-area.

<default> the default value for the specific parameter.

<hatch:dataCol> a column name or number provided in the hatching *dataCol* option. It identifies the column in the *<ndfName>* data.frame, that contains the data to be used to determine if hatchings should be done over a sub-area.

<high value> the numerical value. Represents the hign end of a range.

<idCol> The value of the '`idCol`' call parameter. It is the location id column name or number in the *<ndfName>* data.frame provided as the first call parameter.

<idList> A list of location IDs referred to my the message.

<low value> the numerical value. Represents the low end of a range.

<maxColumnN> The number of columns in the *<ndfName>* data.frame supplied by the caller.

<missingDataSets> A list of boundary datasets required for the Location IDs found in the *<ndfName>* data.frame that cannot be found in the list of loaded data() collection under R.

<missingPackages> A list of *SeerMapper* family packages that must be installed to be able to support the Location IDs in the *<ndrName>* data.frame and the census year call parameter. The packages include: *SeerMapperEast*, *SeerMapperWest*, *SeerMapper2010*, *SeerMapper2010East*, and *SeerMapper2010West*.

<ndfName> The name of the statistics data.frame provided as the first call parameter in the *SeerMapper* function call. The data.frame contains the location id of the data and the data to be categorized and mapped.

<number> The number that may be the value of a parameter or a range limit of a call parameter.

<packageName A character vector of a R package name. This is generally one of the supplement census tract packages: *SeerMapperEast*, *SeerMapperWest*, *SeerMapper2010*, *SeerMapper2010East*, and *SeerMapper2010West*.

<palColors> A character vector containing a character vector assigned to the palColors call parameter.

<palColorMaxNum> Based on the value of the *palColors* call parameter specified, the package obtains the maximum number of colors it can obtains from the RColorBrewer for palette selected. *palColorsMaxNum* is that value.

<row numbers> One or more logical row numbers in the <ndfName> data.frame. The package may use the sequence row number of a row to help the user identify the data.frame row being referred to in the message. The first row is 1.

<string> The value assigned to the call parameter mentioned in the message.

<typeof> The R variable type of the value provided in a call parameters.

<value> the numerical value of the call parameter in error.

<year> The last two digits of the census year: "00" or "10".

Messages:

The following is a listing of all *SeerMapper* generated messages and a description of possible causes and solutions. Each message contain specific variable information to help quickly identify the cause of the problems and allow the user to fix it and re-run the package.

The error and warning message generated by the *SeerMapper* and *SeerMapper2010* packages are listed below in numerical order:

010 The censusYear parameter is set to <censusYear> and is invalid. It must be '2000' or '2010'.

The call parameter '`censusYear`' is an internal parameter and is not generally used by the user. It's only allowed census years are 2000 and 2010. If the value of <censusYear> call parameter is any other value text, the package terminate execution. It is recommended the user use the *SeerMapper* and *SeerMapper2010* function calls without this call parameter

011 The Location IDs in column *idCol* in the *<ndfName>* data.frame are a mix of numbers and characters. They must be all digits for FIPS codes or characters for Seer Area Abbreviations.

The location ids supplied in the idCol column are not a valid type of variable. They must be either digits or character strings. Correct the location ids in the data.frame and re-run. An alternate cause may be the specification of the wrong column in the data.frame.

015 The palColors parameter value of <palColors> is not valid in the RColorBrewer package. The default of 'RdYlBu' will be used.

The value of the '`palColors`' call parameter must be acceptable to the RColorBrewer functions.

The acceptable values are: 'Blues', 'BuGn', 'BuPu', 'GnBu', 'Greens', 'Greys', 'Oranges', 'OrRd', 'PuBu', 'PuBuGn', 'PuRd', 'Purples', 'RdPu', 'Reds', 'YlGn', 'YlGnBu', 'YlOrBr', 'YlOrRd', 'BrBG', 'PiYG', 'PRGn', 'PuOr', 'RdBu', 'RdGy', 'RdYlBu', 'RdYlGn', 'Spectral', 'Accent', 'Dark2', 'Paired', 'Pastel1', 'Pastel2', 'Set1', 'Set2', and 'Set3'

Correct the call parameter and re-run.

020 us48Only parameter is not a logical value of TRUE or FALSE. The default of TRUE will be used.

The '`us48Only`' call parameter must be a logical value of TRUE or FALSE.

022 includePR parameter is not a logical value of TRUE or FALSE. The default of FALSE will be used.

The '`includePR`' must be a logicial TRUE or FALSE.

030 The first parameter should be the statistics data.frame, but is missing or NULL.

The first call parameter is not a data.frame or is missing. Verify the *<ndfName>* variable exists and is a data.frame with the correct structure. The 'str(<ndrName>)' R command can be used to help.

031 *<ndfName>* parameter is not a corrrectly formed data.frame.

The first call parameter is not a properly formed data.frame. It must have at least one row and one column. Inspect the variable using 'str(*<ndfName>*)' and correct the problem.

032 <ndfName> data.frame muat have at least one column for Data.

The *<ndfName>* data.frame is using the row.names of the data.frame for the location id. One column must be present to provide the data for the categorization or color for the sub-area. Inspect the data.frame using 'str(*<ndfName>*)' and correct the problem.

033 <ndfName> data.frame muat have at least two columns for ID and Data.

The *<ndfName>* data.frame must have at least two columns. One for the location ID and one for the data. Inspect the data.frame using 'str(*<ndfName>*)' and correct the problem.

034 <ndfName> data.frame does not have any data rows. Must have at least one row of data.

Check the *<ndfName>* data.frame, correct to provide Location ID column and data column and re-run.

040 idCol parameter is out of range. It must be a column number between 1 and <maxColumnN>.

The '`idCol`' call parameter specifies a column number in the <ndfName> data.frame. However, the number is less that 1 or greater than the number of columns in the data.frame (<maxColumnN>). Correct the value assigned to the '`idCol`' parameter.

041 idCol parameter is not a valid column name (<idCol>) in the <ndfName> data.frame.

The '`idCol`' call parameter specifies a column name in the *<ndfName>* data.frame. However, there is no column with the specified name (<idCol>). Correct the value assigned to the '`idCol`' parameter.

042 idCol parameter is not the correct data type (<typeof>). Must be numeric or character.

The '`idCol`' call parameter must be a numeric(integer) or character vector with length of 1. Verify and correct the value assigned to the '`idCol`' parameter.

043 idCol parameter is a character string, but is empty.

The '`idCol`' call parameter is a character vector, but is empty (""). It must point to a column in the *<ndfName>* data.frame. Correct the value assigned to the '`idCol`' parameter.

046 The following ID values are not valid FIPS codes. Check the first two digits for proper state codes: <badIDs>

Before loading the county and tract boundary data, the ID is check to see if all of the state fips codes are correct. This message identifies the row entry(ies) where the first to digits of the FIPS code is not a valid state Fips code. Review the IDs on the supplied data and correct any invalid ID codees.

050 *dataCol* **parameter is out of range. It must be a column number between 1 and <maxColumnN>.**

The *dataCol* call parameter is specified as the column number in the *<ndfName>* data.frame. However, the value is out of range (< 1 or > the number of columns in the data.frame <maxColumnN>. Correct the *dataCol* call parameter and rerun.

051 *dataCol* **parameter is not a valid column name (<dataCol>) in the <ndfName> data.frame.**

The *dataCol* call parameter is specified as the column name in the *<ndfName>* data.frame. However, the column name does not exist. Correct the *dataCol* call parameter and rerun.

052 *dataCol* **parameter is not the correct data type (<typeof>). Must be numeric or character.**

The *dataCol* call parameter is not the correct data type <typeof>. It must be a numeric or character vector. Correct the *dataCol* call parameter and rerun.

053 *dataCol* **parameter is a character string, but is empty.**

The *dataCol* call parameter is a character vector (string), but is empty (""). Correct the *dataCol* call parameter and rerun.

055 The *dataCol* parameter is set to NULL or NA, The default column name of '<default>' will be used.

The *dataCol* call parameter is missing, NULL or set to NA. The default of <default> witl be used. The default is specified by <default> and is the column name of 'Rate'. If *dataCol* is not specified, make sure the data is placed in the *<ndfName>* data.frame column named 'RATE'.

060 The following data rows in column <dataCol> of the <ndfName> data.frame contains missing (NA) values and will not be mapped. Location IDs: <idList>

The data in the column identified by <dataCol> or "Rate" contains one or more missing values (NA). The list of location IDs with missing values is provided in the <idList> section of the message. Any location with missing data will not be categorized or sub-area colored. It's boundary will be drawn, but colored "white". Correct any locations with missing values and re-run.

061 Some of the data values in the <dataCol> column in the <ndfName> data.frame are not numeric values. Sub-areas will not be mapped. Location IDs: <idList>

The data in column <dataCol> of the data.frame was a factor or character type. The values provided could not be converted to numeric. The <idList> provide the list of location ids with the bad numeric data values. Review and correct these values and re-run.

062 Some of values in data column <dataCol> are not numeric values. Value set to NA.

After initial validation of the data in column *dataCol*, when categorization is requested, one or more of the values were found to be NA (missing) The data row in the *<ndfName>* data.frame are ignored and the sub-area is not categorized mapping area filled with any color.

063 Bad value(s): <badValues>

In association with message 062, <badValues> is a the rows in the *<ndfName>* data.frame that contain bad data. The rows are dropped and not mapped.

064 The data column *<dataCol>* in *<ndfName>* data.frame is not numeric or character type data. Processing terminated.

The value provided at the *dataCol* call parameter is not a numeric or character value. It may be a 'logical' or other type of value that cannot be used to to identify the data column in the *<ndrName>* data.frame Execution is stopped.

065 Data in column (<dataCol>) in *<ndfName>* data.frame contains one or more values that are not a color and will be ignored.

The *categ* is set to "COLORS". The package will use the colors in the column as the fill colors for the sub-areas. However, one or more of the values in the *<dataCol>* column are not valid colors. The data must be character vectors and either the name of a color that matches one returned by the *colors()* function or a 6 or 8 digit hexidecimal character vector (e.g., "#rrggbb" or "#rrggbbaa")

The sub-areas with invalided colors will be drawn filled with "white".

066 Bad Color values: <badColors>

<badColors> is a list of bad colors found in the <dataCol> column related to message 095 when categ="COLORS". Sub-areas with these values will be drawn but filled with "white".

067 The dataCol (<dataCol>) contains <number> colors. It is recommended to limit the number of colors to 11.

Any number of colors can be used to create a map. However, is a legend is to be created, it's recommended 11 colors or less be sued. The package found <number> of colors present in the data. Complications may occur when the legend is drawn for the map.

080 The regionB call parameter is <string> and must be 'NONE', 'DATA' or 'ALL'. The default of '<default>' will be used.

The boundary control '`regionB`' must be a character vector and have a value of "NONE", "DATA", or "ALL". It is set to the default based on the data being mapped. <default> indicates the default being used, either "NONE" or "DATA".

082 The stateB call parameter is <string> and must be 'NONE', 'DATA' or 'ALL'. The default of '<default>' will be used.

The boundary control '`stateB`' must be a character vector and have a value of "NONE", "DATA", or "ALL". It is set to the default based on the data being mapped. <default> indicates the default being used, either "NONE" or "DATA".

084 The seerB call parameter is <string> and must be 'NONE', 'DATA', 'STATE', or 'ALL'. The default of '<default>' will be used.

The boundary control '`seerB`' must be a character vector and have a value of "NONE", "DATA", "STATE" or "ALL". It is set to the default based on the data being mapped. <default> indicates the default being used, either "NONE" or "DATA".

085 The hsaB call parameter is <string> and must be 'NONE', 'DATA', or 'STATE'. The default of '<default>' will be used.

The boundary control '`seerB`' must be a character vector and have a value of "NONE", "DATA", "STATE" or "ALL". It is set to the default based on the data being mapped. <default> indicates the default being used, either "NONE" or "DATA".

087 The countyB call parameter is <string> and must be 'NONE', 'DATA', 'SEER', or 'STATE'. The default of '<default>' will be used.

The boundary control '`countyB`' must be a character vector and have a value of 'NONE', 'DATA', 'HSA', 'SEER', or 'STATE'. It is set to the default based on the data being mapped. <default> indicates the default being used, either "NONE" or "DATA".

089 tractB call parameter is <string> and must be 'NONE', 'DATA', 'COUNTY', 'HSA','SEER' or 'STATE'. The default of '<default>' will be used.

The boundary control '`tractB`' must be a character vector and have a value of "NONE", "DATA", "COUNTY", "HSA", "SEER", or "STATE". It is set to the default based on the data being mapped. <default> indicates the default being used, either "NONE" or "DATA".

092 The fillTo call parameter is <string> and must be 'NONE', 'COUNTY', 'SEER', or 'STATE'. The default of 'NONE' will be used."

The boundary control '`fillTo`' must be a character vector and have a value of "NONE", "COUNTY", "SEER", or "STATE". It is set to the default of "NONE", This parameter is being retired.

094 The clipTo call parameter is <string> and must be 'NONE', 'DATA', 'HSA', 'SEER', 'STATE', 'REGION' or 'TRUE'/'FALSE'. The default of 'NONE' will be used."

The boundary control '`clipTo`' must be a character vector and have a value of "NONE", "DATA", "HSA", "SEER", "STATE" or "REGION" or a logical value of *TRUE* or *FALSE*. It is set to the default of "NONE",

096 The clipTo value specifies a geographic level lower than the data level. The clipTo value set to 'DATA'.

The clipTo value must be set to value higher than the geographic level of the data being mapped. If the data being mapped is at the HSA level, clipTo can only be set to "NONE", "DATA", "SEER", "STATE", or "REGION".

100 The dataBCol call parameter is not a valid color: <dataBCol>. The default of 'black' will be used.

The *dataBCol* call parameter value is not a valid color. The parameter was set to "<dataBCol>". The value must be the name of a color from the set of names returned by the *colors()* function or a 6 or 8 digit hexidecimal character string (e.g., "#rrggbb" or "#rrggbbaa"). The default color of 'black' is will be used.

103 The brkPtDigits call parameter must be greater than 0, no greater than 5 and not NA. Set to a value of 2.

The *brkPtDigits* call parameter is out of range. It must be > 0 and <= 5 digits. The calculated values for the break point when *categ* is set to a number of categories, will be rounded to the default value of 2 decimal digits. Correct or remove the call parameter for future maps.

105 The 'mTitle' option can only contain one or two strings. Only the first two will be used.

The *mTitle* option can consist of one or two charactor vectors (e.g. mTitle=c("Title Line 1","Sub-title line 2"). However, when checking the *mTitle* option it's length is > 2. Only the first two title lines will be used. For future maps, reduce the number of title lines to 2 or less.

106 The 'mTitle' option is not a character vector. 'mTitle' parameter is ignored.

The *mTitle* option is not a character vector. Only character vectors are supported for titles of maps. The *mTitle* option is ignored. Correct to get titles on future maps.

108 The 'mTitle.cex' call parameter is out of range (<=0 or >4). 'mTitle.cex' is set to the default of 1.

The *mTitle.cex* call parameter specifies the size multiplier (cex) to be used when creating the title lines on the map. The value of *mTitle.cex* was found out of the range for the parameter (<= 0 or >4). *mTitle.cex* was set to the default value of 1. Correct or remove the call parameter for future maps.

109 The 'mTitle.cex' option is not a numeric value. 'mTitle,cex' parameter is set to the default of 1.

The *mTitle.cex* call parameter is not a numeric value. *mTitle.cex* was set to the default value of 1. Correct or remove the call parameter for future maps.

110 hatch:dataCol parameter is out of range. It must be a column number between 1 and <maxColumnN>.

The '`hatch:dataCol`' call parameter is specified as the column number in the *<ndfName>* data.frame. However, the row number value is out of range (< 1 or > the number of columns in the data.frame <maxColumnN>.) The hatching data for the comparison cannot be found. Hatching is disabled. Correct the '`hatch:dataCol`' call parameter and rerun.

111 hatch:dataCol parameter is not a valid column name (<dataCol>) in the <ndfName> data.frame.

The '`hatch:dataCol`' hatch option is specified as the column name in the *<ndfName>* data.frame. However, the column name provided does not exist in the *<ndfName>* data.frame. The hatching data for the comparison cannot be found. Hatching is disabled. Correct the '`hatch:dataCol`' call parameter and rerun.

112 hatch:dataCol parameter is not the correct data type (<typeof>). Must be numeric or character.

The data type of the '`hatch:dataCol`' hatch option was found to be <typeof>. It must be a numeric or character value to be able to reference the hatch data column in the *<ndfName>* data.frame. Since the hatch data column cannot be found, hatching is disabled. Correct the '`hatch:dataCol`' call parameter and rerun.

113 hatch:dataCol parameter is a character string, but is empty.

The 'hatch:dataCol' call parameter is a character vector (string), but is empty (""). The hatch data column in the *<ndfName>* data.frame cannot be found. Hatching is disabled. Correct the 'hatch:dataCol' call parameter and rerun.

114 The data column for the hatch comparison could not be found. The hatch parameter has been disabled.

The 'hatch' dataCol option could not be found. Cannot preform the comparison to determine if sub-area should be hatched. The 'hatch' hatching has been disabled. Correct the 'hatch:dataCol' call parameter and rerun.

115 The following hatch options are not valid and will be ignored: <badOptions>

When validating the list of options in the *hatch* call parameter list, several invalid options were found. The '<badOptions>' contains a list of the invalid variables. These options will be ignored. Remove these items from the *hatch* call parameter named list or correct the entries. The list of valid hatching options are: dataCol, ops, value, range, col, lwd, den, angle, and incAngle.

116 The hatch option dataCol is not a character vector or numeric. The default value of 'pValue' will be used.

The name or number of the *hatch* dataCol, is not a valid column name or number. The default value of "pValue" as the column name will be used.

117 The hatch option range values out of order. First value must be less than second value. Reversed.

The low and high values in the hatch range option are out of order. There have been sorted.

118 The hatch option range is not valid. It must be NA or a vector containing 2 numeric values (low and high limits) for the range. Range checking is disabled.

The hatching 'hatch' range checking parameter must be "NA" to disable range changing or a vector of two numeric values representing the low and high limits of the range. If the values are out of order, they will be reversed. If either value is an NA, range checking will be disabled. An example of the use of the hatching option is: range=c(0,20) for a range change of 0 to 20.

119 The comparison operator provided in the hatch 'ops' options - <string> - is not valid. Hatching disabled.

The *hatch* call parameter contains a 'ops' options of <string> that is not valid.

The allowed *hatch* 'ops' is not a valid operation. The allowed comparision operation values are: "eq","==", "=", "ne", "<>", "!=", "lt", "<", "le", "<=", "=<", "gt", ">", "ge", ">=", and "=>". The comparison is <hatch:dataCol-value> <ops> <hatch:value>.

120 The following hatch2 options are not valid and will be ignored: <badOptions>

When validating the list of options in the *hatch2* call parameter list, several invalid options were found. The '<badOptions>' contains a list of the invalid variables. These options will be ignored. Remove these items from the *hatch2* call parameter named list or correct the entries. The list of valid hatching options are: dataCol, ops, value, and range.

121 The hatch2 option dataCol is not a character vector or numeric. The default value of 'pValue' will be used.

The name or number of the *hatch2* dataCol, is not a valid column name or number. The default value of "pValue" as the column name will be used.

122 The hatch2 range option values out of order. First value must be less than second value. Reversed.

The low and high values in the hatch2 range option are out of order. There have been sorted.

123 The hatch2 range option is not valid. It must be NA or a vector containing 2 numeric values (low and high limits) for the range. Range checking is disabled.

The 'hatch2' range checking parameter must be "NA" to disable range changing or a vector of two numeric values representing the low and high limits of the range. If the values are out of order, they will be reversed. If either value is an NA, range checking will be disabled. An example of the use of the hatching option is: range=c(0,20) for a range change of 0 to 20.

124 The comparison operator provided in the hatch2 *ops* option - <string> - is not valid. Hatching disabled.

The *hatch2* call parameter contains a 'ops' options of <string> that is not valid.

The allowed *hatch2* 'ops' is not a valid operation. The allowed comparision operation values are: "eq","==", "=", "ne", "<>", "!=", "lt", "<", "le", "<=", "=<", "gt", ">", "ge", ">=", and "=>". The comparison is <hatch:dataCol-value> <ops> <hatch2:value>.

125 The hatch col option is not a valid color : <color>. The default will be used.

The *hatch* 'col' does not specify a valid color. It must be a name that matches the color names in *colors()* or a 6 or 8 digit hexidecimal color value (e.g., "#rrggbb" or "#rrggbbaa".) The default hatch color value of "grey(0.66)" will be used.

126 The hatch lwd option is not numeric - <value> The default will be used.

The *hatch* call parameter 'lwd' option value is not a numeric value. The line weight values for the hatch lines must a numeric value > 0. The default value of 0.85 is used.

127 The hatch lwd option is out of the range of >0 to <= 5. The default will be used.

The *hatch* call parameter 'lwd' option value is out of range. It must be in a range > 0 and <= 5. Correct the call and re-run. The default value of 0.85 will be is used.

128 The hatch density option is not numeric - <value> The default value of 25 will be used.

The *hatch* call parameter 'den' option is not a numeric value. It was set to <value>. The option specifies the density of the hatching in number of lines per inch. The default is 25 lines per inch. The density may need to be adjusted if the size of the graphic space is not 7.5 x 10 inches.

129 The hatch density option is <value> and is out of the range of > 4 to <= 64 lines per inch. The default value of 25 will be used.

The *hatch* call parameter 'den' option is out of range. It must be a numeric value > 4 and <= 64 lines per inch. The default value of 25 lines per inch will be used.

130 The hatch angle option is not numeric. The default will be used.

The *hatch* call parameter 'angle' option is not a numeric value. The value is the degrees the hatching lines will be draw over the sub-area. The default value of 45 degress will be used.

131 The hatch angle option is out of the range of => -360 to <= 360 degrees. The default will be used.

The *hatch* call parameter 'angle' option is out of range. It must be a numeric value between -360 and 360 degrees. The default value of 45 degress will be used.

132 The hatch incremental angle option (incAngle) is not numeric. The default will be used.

The *hatch* call parameter 'incAngle' option is not a numeric value. The value is the degrees the hatching lines are shifted for each additional hatching overlay. The default value of 60 degress will be used.

133 The hatch incremental angle option (incAngle) is out of the range of => -120 to <= 120 degrees. The default will be used.

The *hatch* call parameter '`incAngle`' option is out of range. It must be a numeric value between -120 and 120 degrees. The default value of 60 degress will be used.

134 A hatch call parameter error was detected. The hatch parameter is disabled. See previous messages for details.

Previous message detail the errors detected in processing the *hatch* call parameter options list. Since errors were detected, hatching will be disabled. Correct the identified problems and re-run.

135 The hatch call parameter must be a logical value (T/F) or a list of options. Parameter is ignored.

The *hatch* call parameter can be a logical TRUE/FALSE value of a named list of hatch options. The *hatch* call parameter was neither a logical variable or a list. Hatching is disabled.

136 A hatch2 call parameter error was detected. The hatch2 parameter is disabled. See previous messages for details.

Previous message detail the errors detected in processing the *hatch2* call parameter options list. Since errors were detected, hatching will be disabled. Correct the identified problems and re-run.

137 The hatch2 call parameter must a list of options.

The *hatch2* call parameter can be a logical TRUE/FALSE value of a named list of hatch options. The *hatch* call parameter was neither a logical variable or a list. Hatching is disabled.

140 hatch2:dataCol parameter is out of range. It must be a column number between 1 and <maxColumnN>.

The '`hatch2:dataCol`' call parameter is specified as the column number in the *<ndfName>* data.frame. However, the row number value is out of range (< 1 or > the number of columns in the data.frame <maxColumnN>.) The hatching data for the comparison cannot be found. Hatching is disabled. Correct the '`hatch2:dataCol`' call parameter and rerun.

141 hatch2:dataCol parameter is not a valid column name (<dataCol>) in the <ndfName> data.frame.

The '`hatch2:dataCol`' hatch option is specified as the column name in the *<ndfName>* data.frame. However, the column name provided does not exist in the *<ndfName>* data.frame. The hatching data for the comparison cannot be found. Hatching is disabled. Correct the '`hatch2:dataCol`' call parameter and rerun.

142 hatch2:dataCol parameter is not the correct data type (<typeof>). Must be numeric or character.

The data type of the '`hatch2:dataCol`' hatch option was found to be <typeof>. It must be a numeric or character value to be able to reference the hatch data column in the *<ndfName>* data.frame. Since the hatch data column cannot be found, hatching is disabled. Correct the '`hatch2:dataCol`' call parameter and rerun.

143 hatch2:dataCol parameter is a character string, but is empty.

The '`hatch2:dataCol`' call parameter is a character vector (string), but is empty (""). The hatch data column in the *<ndfName>* data.frame cannot be found. Hatching is disabled. Correct the '`hatch2:dataCol`' call parameter and rerun.

144 The hatch/hatch2 data column <dataCol> does not contain numbers. Parameter hatch/hatch2 disabled.

The column identified to be used for the hatch data comparison, <hatch:dataCol>, does not contain numbers. Only numerical comparisons are supported. The hatch or hatch2 parameter is disabled.

145 The hatch/hatch2 data is not numeric. Parameter hatch/hatch2 disabled.

The column identified to be used for the hatch/hatch2 data comparison does not contain numbers. Only numerical comparisons are supported. The hatch or hatch2 parameter is disabled.

146 The hatch/hatch2 data column <dataCol> does not contain valid numbers. The hatch/hatch2 parameter is disabled.

The the data found in the dataCol option was not good numerical data. Cannot preform the comparison to determine if sub-area should be hatched. The '`hatch`' or '`hatch2`' parameter is disabled. Correct the data and rerun.

147 boldHatching data provided is not within the allowed range : <low value> to <high value>. Hatching disabled.

Range checking of the hatch data is enabled. The hatch data in the <hatch:dataCol> column is not within the range of <low value> to <high value>. Without all of the data within the specified range, hatching will be disabled.

150 The 'mLegend' parameter is set to NA. Set to the default of TRUE.

The *mLegend* call parameter is set to NA. The value must be a logical (TRUE/FALSE) or a named list of options. The default of TRUE will be used.

152 The 'mLegend' call parameter must be a list or logical value. Parameter Ignored.

The *mLegend* call parameter is not a logical variable or named list. The call parameter is ignored and the default of TRUE will be used. Check the function all and correct the mLegend=XXXX for the cause of the error.

153 The mLegend options list does not names for each list entry. The format must be mLegend=list(pos='left',numCols=4). Defaults used.

The *mLegend* call parameter is set to NA. The value must be a logical (TRUE/FALSE) or a named list of options. The default of TRUE will be used.

154 The following options in the Legend parameter are not valid and will be ignored: <badOptions>.

The *mLegend* call parameter is a named list, but one or more of the variables in the list are not valid. They are listed in the <badOptions> in the message. Valid variables are: counts, size, numCols, pos, noValue, pch, or labels. The invalid options are ignored.

156 The 'mLegend' parameter 'counts' must a a logical variable. Set to FALSE.

The *mLegend* call parameter '`counts`' option was specified, the variable is not a logical TRUE or FALSE value. Correct the value to TRUE or FALSE. The default value of FALSE will be used.

158 The mLegend parameter 'size' option must be a numeric value. The default is used.

The *mLegend* call parameter '`size`' option was specified, the variable is not a numeric value setting the font size multiplier (cex). The default value of 0.85 will be used. Correct the value to a numeric and re-run.

160 The mLegend parameter 'size' option must be in the range from 0 to 5. Set to 0.85.

The *mLegend* call parameter 'size' option was specified, the variable is not in the range of 0 to 5. The default of 0.85 is used as the font size multiplier. Correct the value to a value between 0 and 5 and re-run.

162 The mLegend parameter option 'numColumns' must be numeric and between 1 and 8. Set to 3.

The *mLegend* call parameter 'numColumns' option was specified, the variable is not a numeric setting the number of columns in the legend. The default value of 3 columns will be used. Correct the value and re-run.

164 The legendPos parameter is not "left", "center", or "right". Set to "left".

The *mLegend* call parameter 'pos' option was specified, the variable sets the position of the legend on the bottom of the map. The value is not valid. It must be 'left", "center" or "right". The default of "left" will be used. Correct the option and re-run.

166 The pch parameter must be a numeric value. Set to 19.

The *mLegend* call parameter 'pch' option was specified, the variable sets symbol used in the legend. The value provided is not valid. The pch value must be a value from 19 to 25 (filled symbols). The default symbol value of 22 will be used (a square). Correct the option and re-run.

167 The pch parameter must be a value between 19 and 25. Set to 19.

The *mLegend* call parameter 'pch' option was specified, the variable sets symbol used in the legend. The value provided is out of range. The pch value must be a value from 19 to 25 (filled symbols). The default symbol value of 22 will be used (a square). Correct the option and re-run.

168 The noValue parameter must be a logical (TRUE or FALSE) value. Set to FALSE.

The *mLegend* call parameter 'noValue' option was specified, the variable requests legend entries with no sub-areas be listed in the legend. The value is not a logical variable. The value must be a logical variable set to TRUE or FALSE. The default of TRUE will be used.

170 The labels parameter must be a vector of character strings. Set to an empty string.

The *mLegend* call parameter 'labels' option was specified, the variable is not a vector of replacement labels for the legend. The labels option was found to be empty (set to ""). The labels option is disabled. Correct the value assigned to the labels option and re-run.

172 The mLegends parameter is an empty list. The legend will be drawn using default values.

The *mLegend* call parameter named list of options was found to be empty. No options were specified. The legend will be drawn using the defaults for each option.

194 The regions parameter is not a logical value of TRUE or FALSE. The default of FALSE will be used.

Retired message

The *regions* call parameter must be a logical type variable with a value of TRUE or FALSE. Since it is not, the default value of FALSE will be used.

195 Package: <packageName> failed to load into the system. Verify package is install and try manually executing a 'library(<packageName>)' command.

To be able to map census tract boundaries, supplemental data packages must be installed on the local machine and available to be loaded. The package tries to dynamically load the needed package(s) when it detects census tracts are being mapped and which states are involved. If this process fails, this message will appear. Usually a manual reinstall of the supplemental data package will fix the problem. Another work around is to execute the library function with the name of the package to ensure it is loaded prior to calling *SeerMapper*.

196 The following boundary datasets are missing. Make sure the appropriate SeerMapper supplement packages have been installed and loaded.

SeerMapper has determined it requires census tract boundary data for states with Seer Registries. These boundary datasets are contained in the *SeerMapperEast* and *SeerMapperWest* packages for the 2000 census and in the *SeerMapper2010East* and *SeerMapper2010West* packages for the 2010 census. Message 197 lists the missing datasets.

Install these packages and load them via 'library' commands when mapping census tracts for states east and west of the Mississippi that do not contain Seer Registries.

197 Missing: <missingDataSets>

The boundary datasets (xxx.rda files) listed in <missingDataSets> are needed to map the census tracts found in the call. This should only occur when doing census tract maps in states without Seer Registries. In this case the *SeerMapperEast* and/or *SeerMapperWest* (or *SeerMapper2010East* and/or varSeerMapper2010West) are be required. Install and load via the 'library' command the packages in this list and re-run the mapping request.

198 The following supplemental SeerMapper Census Tract boundary packages are missing and must be installed and loaded:

SeerMapper requires suppliemental census tract boundary packages to map data at the census tract level. When tract boundaries are required, the package determines where packages are required to load the boundary datasets. These supplement packages should have been loaded into their own namespace when *SeerMapper* was first called. The names of the required packages do not appear in the list of loaded packages (loadedNamespace()). A list of missing packages is presented in message # 199, below. Make sure the packages are in the current R library and can be loaded with library(x) R commands. The main package and six supplemental packages are: *SeerMapper*, *SeerMapperRegs*, *SeerMapperEast*, *SeerMapperWest*, *SeerMapper2010Regs*, *SeerMapper2010East*, and *SeerMapper2010West*.

Install these packages and load them via 'library' commands when mapping census tracts for states east and west of the Mississippi that do not contain Seer Registries.

199 Missing: <missingPackages>

The message provides a list of the SeerMapper packages with supplemental census tract boundary data that must be installed from CRAN to be able to map the census tracts in the data. The supplemental packages are *SeerMapperRegs*, *SeerMapperEast*, *SeerMapperWest*, *SeerMapper2010Regs*, *SeerMapper2010East*, and varSeerMapper2010West.

200 After cleaning up <ndfName> data.frame to remove detected errors, there are no rows to process.

After all of the rows with detected errors were removed from the *<ndfName>* data.frame, the data.frame was found to be empty - no rows remaining. Resolve the error documented above and re-run.

202 The <ndfName> data.frame has duplicate location IDs. The duplicate rows will be removed.

The *idCol* column in the *<ndfName>* data.frame has rows with duplicate location IDs. There should be only one row of data per sub-area location ID. The duplicate row is removed. Message 204 contains a list of the duplicate IDs removed and their relative row number in the data.frame.

204 The duplicate IDs are: <badIDs>

This message provides a two column list of the duplicate IDs and their relative row numbers that were removed.

```
row#    ID
   2    01
```

Review and correct the contains of the data.frame and re-run.

206 The location IDs contain invalid state IDs. Assuming the Location IDs are HSA numbers.

The location ID in the dataframe could be either state location IDs or HSA locations IDs. Because the location ID provided contains invalid state IDs, the package is assuming the location IDs are really HSA IDs. Invalid state IDs are: 03, 07, 14, 43, 52, and values greater then 56 except for 72. If the location ID is really state IDs, correct the location IDs in the data frame and rerun.

220 Some of the data rows in the <ndfName> data.frame have location IDs with missing values (NA). These rows will be removed. Correct and rerun.

The location IDs in the *idCol* column of the *ndfName* data.frame contain some (one or more) missing values (NA). These rows cannot be linked to a geographic sub-area and will not be mapped. Message number 222 contains a list of the row numbers found to have the invalid location IDs in the *<ndfName>* data.frame. Correct the location IDs in the data.frame and re-run.

222 The following rows will not be mapped: <row numbers>

The <row numbers> list contains rows in the *ndrName* data.frame that cannot be mapped. This could because the location ID is missing, invalid, or incorrect or the data to be categorized is missing (NA) or not a valid number. The <row numbers> list starts with 1 as the first row and increments by 1 for each row.

250 The categ call parameter is missing, empty or contains NAs. The default value of 5 will be used.

The *categ* call parameter is NULL or NA (missing.) This is an invalid seting and the parameter is ignored. The default of *categ* <- 5 will be used.

252 The value entered for categ parameter is not valid : <string>. The default of categ=5 is used.

The value entered for the *categ* call parameter was a character vector. It was not set to "DATA" or "COLORS" and could not be converted to a numeric or vector of numerics (break point list.) The default value of 5 will be used.

253 The categ parameter provided does not contain any value. The default of categ=5 is used.

The *categ* call parameter was set to "". The default of 5 will be used.

256 The categ call parameter is not 'DATA', 'COLORS', or a valid single value numeric value. The default of categ=5 will be used.

The *categ* call parameter is not set to a valid value. The assigned value has a length of 1, so it cannot be a break point vector list. As a single value, it must be set to a number, 'DATA', or 'COLORS'. The default value of 5 is used. Correct the call parameter and re-run.

258 The categ call parameter has a single value of <string> and must be => 3 as a minimum. The default of 5 will be used.

When the *categ* call parameter is the number of categories to be calculated, the number of categories must be => 3. If less then < 3 the default value of 5 will be used.

259 The categ call parameter value has a single value of <string> and it is > than <catMaxNum>. The categ will be set to <catMaxNum>.

When the *categ* call parameter is a single value of the number of categories to calculate the break points. The maximum value is limited by the *palColors* call parameter used. With the default *palColors* of "RdYlBu", the maximum value is 10. The <catMaxNum> value in the message indicates the maximum for the active *palColors* parameter. Only the first <catMaxNum> values in the vector will be used.

Correct the break point vector and rerun.

260 The categ call parameter break point list has a length < 3 items. The default of *categ* = 5 will be used.

When *categ* call parameter specifies a break point list (vector), it must have a length equal to or greater than 3. The break point list cannot be used. The default of *categ* = 5 will be used.

261 The categ call parameter break point list has a length > <catMaxNum> items. Only the first <catMaxNum> values will be used.

When *categ* call parameter specifies a break point list (vector) with a length > <catMaxNum>. The package does not support more then <catMaxNum> categories. Only the first <catMaxNum> break points will be used.

262 The categ call parameter contains non-numeric value in the break point vector: <string> . The default of categ=5 is used.

The *categ* call parameter is a character vector (<string>) with length > 1. It appears to be a break point vector, but the values could not be converted to a numeric vector. The default value of 5 will be used.

263 One or more values in the categ breakpoint list is not a number. The default of categ=5 will be used.

The *categ* call parameter appears to be a break point list. One or more of the values in the vector are not numeric. The break point vector cannot be used. The default value of 5 will be used.

264 The categ call parameter breakpoint vector is not in order (low to high). Breakpoint vector has been sorted.

The *categ* call parameter is a break point numeric vector. However, the break point values are not in numerical order from low to high. The vector was sorted to compensate.

265 The categ call parameter contains a break point vector with duplicate values. The duplicate values will be removed.

When a vector of break point values is provided via the *categ* call parameter, each value must be unique and in order from low to high. If duplicate values are found, the duplicate value is removed and the number of break points used reduce by one. If the number of break point falls below 3, an additional warning will be generated.

266 The number of points in the categ call parameter list is out of range. It must be between 3 and <catMaxNum>. The default of categ=5 will be used.

The minimum number of values in a *categ* break point vector is 3. The maximum number of values is set by the *palColors* value used. The default maximum is 10 when *palColors* is equal to "RdYlBu". See the section on the *palColors* call parameter.

267 The <dataCol> column data are integer category values. The range of the values is greater than the maximum of <catMaxNum>. Reduce number of categories or select different palColors value.

When *categ* is set to "DATA", the data column (<dataCol> in the *<ndfName>* data.frame is used directly as the category index. The index should be from 1 to "n", but can be any set of

integers. However, the total number of sequential categories from the lowest value to highest value must be <= <catMaxNum> to be able to assign colors in the selected RColorBrewer palette to the sub-areas. Recommend reducing the number of categories, close up any gaps in the category range, or select a RColorBrewer palette that can provide more colors.

Execution is terminated.

268 The categ call parameter contains non-numeric value in the break point vector: <string>. The default of categ=5 is used.

The *categ* call parameter is a break point vector (<string>), but contains non-numeric values. It cannot be used to classify the data. The default value of 5 will be used.

270 The categ call parameter is 'DATA'. The <dataCol> column does not appear to contain integer values.

When *categ* is set to "DATA", the values in the data column of the data.frame are categories values and must be integers. The data.frame column is not integer values.

Execution is terminated.

280 The labels parameter must have one entry for each categories. Set to an empty string.

The '`labels`' legend option allows you to replace the calculated labels in the legend for each category. There was not one vector entry for each category to be included in the legend. Correct the '`labels`' option to have one entry for each legend category. The '`labels`' option is disabled.

290 The following area(s) in the data do not match the list of boundaries:

When the package compared the location IDs supplied with the data in the *<ndfName>* data.frame with the boundary information contained in the package, it was not able to match some of the data rows with the boundary information. Any data row that does not match the boundary information will not be mapped. This set of messages document which location IDs have been dropped.

291 > <badIDs>

The message contains a list (<badIDs>) of location IDs in the data that are not in the boundary dataset.

292 Please check to make sure your data matches the 20<year> census area identifiers and boundaries.

One of the reasons the data location IDs may not match the boundary information is the wrong census year boundary data is being used. Check to determine if the data and location IDs are build from the 2000 or 2010 U. S. Census data. Make sure the correct SeerMapper version is being used.

Check to see to see which census year the location IDs should match and re-run under the appropriate *SeerMapper*: *SeerMapper* for 2000, and *SeerMapper2010* for 2010. Also check the location IDs in the *<ndfName>* data.frame to make sure they are correct.

295 Number of locations found in the Data Table with borders: <number>

This is an informative message when '`debug`' is *TRUE* to indicate the number of rows of data in the supplied data.frame.

300 The following Seer Registry idenifiers do not match the abbreviations or aliases:

Messages 300, 301, and 302 are components of one message on three lines. The package has determined Seer Registry level data is being mapped. That is it is not U. S. Fips codes. The location ID column does not contain Registry abbreviations or character strings that would

match the alias for each registry. Message 301 list the location IDs that could not be matched to the boundary information. To allow the package to continue running, any row without a valid location ID is ignored and not mapped.

301 <badIDs>

See message 300 for details. <badIDs> is the list of Seer Registry location IDs that do not match the Registry boundary information.

302 These data rows will be ignored in the mapping.

See message 300 for details.

380 Internal Error - dataPList contains an NA in list: <idList>

The dataPList vector of location IDs should not contain a missing value. <idList> is the contains of the dataPList vector. The execution was stopped.

381 Internal Error - dataPList does not exist.

the dataPList vector of Location IDs should not be missing. The execution was stopped.

382 Internal Error - regionPList contains a NA in list: <idList>.

The regionPList of regional IDs should not contain any missing values. <idList> is the contains of the regionPList vector. The execution was stopped.

384 Internal Error - statePList contains a NA in list: <idList>.

The statePList vector of Location IDs should not contain any missing values. <idList> is the contains of the statePList vector. The execution was stopped.

386 Internal Error - seerPList contains a NA in list: <idList>.

The seerPList vector of Location IDs should not contain any missing values. <idList> is the contains of the seerPList vector. The execution was stopped.

387 Internal Error - countyPList contains a NA in list: <idList>.

The countyPList vector of Location IDs should not contain any missing values. <idList> is the contains of the countyPList vector. The execution was stopped.

388 Internal Error - tractPList contains a NA in list: <idList>.

The tractPList vector of Location IDs should not contain any missing values. <idList> is the contains of the tractPList vector. The execution was stopped.

389 Internal Error - hsaPList contains a NA in list: <idList>.

The tractPList vector of Location IDs should not contain any missing values. <idList> is the contains of the tractPList vector. The execution was stopped.

900 The provided proj4 string encountered an error when converted to the internal CRS parameter. The following error was reported, plesae correct and rerun. <CRS Error Message>

The package attempted to convert the provided PROJ.4 projection string into a CRS object for the boundary transformation. The CRS function reported the <CRS Error Message>. The proj4 call parameter is ignored.

901 The provided proj4 string encountered an warning when converted to the internal CRS parameter. The following warning was reported, plesae correct and rerun. <CRS Warning Message>

The package attempted to convert the provided PROJ.4 projection string into a CRS object for the boundary transformation. The CRS function reported the <CRS Warning Message>. The proj4 call parameter is ignored.

902 Unpredicted results when proj4 was translated to CRS format. Unknown problem. <CRS Message>

The package attempted to convert the provided PROJ.4 projection string into a CRS object for the boundary transformation. The CRS function reported an unexpected error and <CRS Message>. The proj4 call parameter is ignored.

909 Error on processing the proj4 parameter. No user specified projection will be done.

The package attempted to convert the provided PROJ.4 projection string into a CRS object for the boundary transformation. The CRS function could not translate the PROJ.4 string without any issues. See above messages. The proj4 call parameter is ignored.

980 Package: <packageName> failed to load into the system. Verify package was installed and try manually executing a 'library(<packageName>)' command.

During the mapping of census tract data, *SeerMapper* attempted to have the system load the boundary datasets for the sub-areas identified in the data. The package named <packageName> failed to load. The can be caused by a corrupted package, package is not installed, the package was not installed in the standard R library, permission prohibit access to the package, or the package has been deleted. Re-install the package from CRAN. The execution was stopped.

990 ERRORS found in statistic data.frame checking - Run Terminated.

During the validation of the first parameter data.frame, errors have been identified and warning messages generated. The package cannot continue processing the mapping request. The execution was stopped.

991 The location ID column could not be found. Run Terminated.

Previous messages identify the reason the location ID column in the *<ndfName>* data.frame was not found, was in error, or could not be found. If the location IDs are held in the row names attached to the data.frame, use *idCol* = "row.names". The *idCol* call parameter can be used to specify the column name or number in the *<ndfName>* data.frame. The default name of the location ID column is "FIPS". The execution was stopped.

992 The data column could not be found. Run Terminated.

Previous messages identify the reason the data column in the *<ndfName>* data.frame was not found, was in error, or could not be found. The *dataCol* call parameter can be used to specify the column name or number in the *<ndfName>* data.frame. The default name of the data column is "Rate". The execution was stopped.

993 The data column for the hatching comparison could not be found. Hatching has been disabled.

Previous messages identify the reason the hatch data column in the *<ndfName>* data.frame was not found, was in error, or could not be found. The *hatch* call parameter and '`dataCol`' option can be used to specify the column name or number in the *<ndfName>* data.frame. The default name of the *hatch* '`dataCol`' column is "pValue". Since the data column cannot be found, hatching is disabled.

End of Messages.

rg99_d00 *U. S. 2000 or 2010 Census Regional Boundary Definitions*

Description

This dataset contains the boundary data for the 4 U. S. 2000 or 2010 Census Regions for use by the NCI *SeerMapper* package.

Usage

```
data(rg99_d00)
```

Format

This dataset contain a single SpatialPolygonsDataFrame type object that contains the boundary polygon definitions for the 5 U. S. Census Regions for the year 2000 or 2010 and an extra region covering Alaska, Hawaii and Puerto Rico. The U. S. Census website defines the continental regions with values of 1 through 4 as NorthEast, South, MidWest and West. The states of Alaska, Hawaii, and the territory of Puerto Rico were assign regional values of 0, effectively a 5th region - offshore. The regional boundary data is the same for both the 2000 and 2010 census. This dataset maintains the codes as published by the U. S. Census Buearu. The regions are keyed using the U. S. 2000 or 2010 Census region codes based on which package being used. The *SeerMapper* package supports the 2000 and 2010 boundaries. Only one dataset is required: rg99_d00.

Details

```
The U. S. 2000 and 2010 Census Region codes:
    0 - Offshore
    1 - NorthEast
    2 - South
    3 - MidWest
    4 - West

The U. S. States/Terr. in the Offshore region (code = 0) are"
    02 - Alaska
    15 - Hawaii
    72 - Puerto Rico

The U. S. States in the NorthEast region (code = 1) are:
    FIPS Name
    09 - Connecticut
    23 - Maine
    25 - Massachusetts
    33 - New Hampshire
    34 - New Jersey
    36 - New York
    42 - Pennsylvania
```

```
    44 - Rhode Island
    50 - Vermont

The U. S. States in the South region (code = 2) are:
    FIPS Name
    01 - Alabama
    04 - Arkansas
    10 - Delaware
    11 - District of Columbia
    12 - Florida
    13 - Georgia
    21 - Kentucky
    22 - Louisiana
    24 - Maryland
    28 - Mississippi
    37 - North Carolina
    40 - Oklahoma
    45 - South Carolina
    47 - Tennessee
    48 - Texas
    51 - Virginia
    54 - West Virginia

The U. S. States in the Midwest region (code = 3) are:
    17 - Illinois
    18 - Indiana
    19 - Iowa
    20 - Kansas
    26 - Michigan
    27 - Minnesota
    29 - Missouri
    31 - Nebraska
    38 - North Dakota
    39 - Ohio
    46 - South Dakota
    55 - Wisconsin

The U. S. States in the West region (code = 4) are:
    06 - California
    08 - Colorado
    16 - Idaho
    30 - Montana
    32 - Nevada
    35 - New Mexico
    41 - Oregon
    49 - Utah
    53 - Washington
    56 - Wyoming
```

As of Version 1.0 of the *SeerMapper* package, The ".rda" datasets are compressed using the "xz" method to reduce the disk space requirements.

The region boundary data was created from the state boundary data used in this package to ensure the borders lined up. The state boundary data was created from county and census tract boundary data processed by "www.MapShaper.org" website version 01.15 using modified Visvalingam method with intersection repair enabled and prevent shape removal enabled and simplified to 13 while maintaining a visually usable boundary image and edge alignments.

The PROJ.4 projection used for these boundaries is:

"+proj=aea +lat_1=33 +lat_2=49 +lat_0=39 +lon_0=96W +ellps=WGS84"

The dataset provide a good functional characterization of the 5 regions.

The @data structure of the saved SpatialPolygonsDataFrame contains the following information for use by the SeerMapper function:

row.names a character vector of the region number (0-4)

ID a character vector of the region number (0-4). Save as the row.names of the SPDF

rgID a character vector containing the 1 digit U. S. region number from the U. S. 2000 or 2010 census state boundary datasets. (0 to 4) This field is also used as the row.names for the SpatialPolygons.

rgName a character vector containing the region name.

tracts00 an integer value of the number of census tracts in the region for the census year 2000.

tracts10 an integer value of the number of census tracts in the region for the census year 2010.

county00 an integer value of the number of counties in the region for the census year 2000.

county10 an integer value of the number of counties in the region for the census year 2010.

states an integer value of the number of states in the region for both the census years.

Source

The US 2000 and 2010 Regional Census boundaries are based on the CENSUS.GOV website state, county and census tract boundary shapefiles for the 2000 and 2010 Census.

Examples

```
data(rg99_d00)
```

sa99_d00 *The boundary dataset for the 20 U.S. Seer Registries.*

Description

This dataset contains a Spatial Polygons data frame for the 20 U. S. Seer registry areas boundaries as of the U. S. 2000 or 2010 Census years. Since the Seer Registries boundaries are the same for the 2000 and 2010 census, a single *sa99_d00* dataset is used by the *SeerMapper* mapping packages for the NCI Seer Registry boundary data and information. The boundaries are used to map data by U. S. Seer registry area, to provide an overlay to state, county and census tract maps with the Seer registry areas boundaries, and provide a intermediate boundary for drawing census tract or county boundaries. When doing counties or census tracts level mapping, the *SeerMapper* package can be used to map the sub-areas within a registry that only have data or can expand the hsa, county or census tract boundary's drawn to include all sub-areas up to a Seer Registry or State boundary. This feature is controled by the *hsaB*, *countyB* and *tractB* call parameters. For more details see below. The Seer registry area boundaries mapping is controlled by the *seerB* call parameter. When set to "NONE", no Seer registry area boundaries are drawn on county or census tract maps. If set to "DATA", the Seer registry boundaries will be drawn for any Seer registry area containing sub-areas with data. If set to "STATE", all Seer registry boundaries will be drawn in a state that contains a sub-area with data. If set to "ALL", all Seer registry boundaries will be drawn.

Usage

```
data(sa99_d00)
```

Format

The dataset contains a SpatialPolygon data.frame of the 20 U. S. Seer registry areas based on the 2000 and 2010 U. S. Census county boundary data and index information for use by *SeerMapper*. The Seer Registry boundary and information are the same for both census years.

An registry area may contain multiple polygons. For more details, refer to the SpatialPolygon structure descripted in the "sp" package documentation. The SpatialPolygons is constructed to have all of the polygons for a single registry contained under a single SpatialPolygon instead of several. The 20 U. S. Seer registry areas are identify by character abbreviations:

saID	stID	Name
AK-NAT	02	Alaska Natives
AZ-NAT	04	Arizona Indians
CA-OTH	06	California Other (not SF-Oakland, SJ-Monterey, LA areas)
CA-SF	06	California San Fran/Oakland
CA-SJ	06	California San Jose/Monterey
CA-LA	06	California Los Angeles
CT	09	Connecticut
GA-OTH	13	Georgia Other (not Atlanta or Rural areas)
GA-ATL	13	Georgia Atlanta
GA-RUR	13	Georgia Rural
HI	15	Hawaii

IA	19	Iowa
KY	21	Kentucky
LA	22	Louisiana
MI-DET	26	Michigan Detroit
NJ	34	New Jersery
NM	35	New Mexico
OK-CHE	40	Oklahoma Cherokee Nation
UT	49	Utah
WA-SEA	53	Washington Seattle-Puget Sound

When mapping Seer registry area level data, the Seer registry area abbreviated name must be used.

The data slot of the SpatialPolygonsDataFrame (sa99_d00) contains information on the associated state, region identifiers, the centroid of the registry, and the number of counties and tracts in the registry.

The data slot can be used to identify the states associated with the active Seer registry areas.

Details

When Seer Registry boundary be used to draw all of the county and tract boundaries up to the registry boundary to be able to provide a complete county or tract mapping within the registry. This is cone by setting either the *countyB* or *tractB* call parameter to "SEER". This feature works the same as setting these call parameters to "STATE", but the boundary is the Registry boundaries.

The PROJ.4 projection used for these boundaries is:

"+proj=aea +lat_1=33 +lat_2=49 +lat_0=39 +lon_0=96W +ellps=WGS84"

The *sa99_d00* dataset is a SpatialPolygonsDataFrame (SPDF). The spatial polygons represent the Seer Registry boundary data. The data slot of the SPDF contains the following information about Seer Registries:

row.names The row.names attribute of the SPDF are character vectors and are set to the Seer Registry abbreviations as listed above.

ID - character string representing the Seer Registry ID, the Seer Registry area abbreviation. *saID* and *ID* values provides the linkage between Seer registry area level data provided by the user and the Seer registry area boundary data. These abbreviations must be used. This field is the same as the row.names of the SPDF.

saID - same as ID. The Seer Registry Identifier.

stID - the state ID containing the Seer Registry. The *stID* is the 2 digit fips code for the state containing the Seer registry area. One state may be a Seer registry area or contain multiple Seer registry areas. But no Seer registry area spans multiple states.

stName - character vector of the state name containing the Seer Registry.

rgID - the U. S. census region ID containing the Seer Registry.

c_X - a numeric value of the centroid's X coordinates.

c_Y - a numeric value of the centroid's Y coordinates.

county00 - a numeric value of the number of tracts in the Seer Registry in the 2000 census.

county10 - a numeric value of the number of counties in the Seer Registry in the 2010 census.

tracts00 - a numeric value of the number of counties in the Seer Registry in the 2000 census.

tracts10 - a numeric value of the number of tracts in the Seer Registry in the 2010 census.

`SeerMapper` *Quick Data Mapper at State, Health Service Areas, State/County, State/County/Census Tract, or Seer Registry Area Levels*

Description

Provides a easy and quick means of creating U.S. maps of rates or data at the U.S. State, Health Service Areas (HSA), State/County, State/County/Census Tract or U.S. Seer Registry area levels. Send data is provide at the state level, the outlines of all states are overlaid, but only the states with data are categorized and colored. If the data is provided at the HSA, state/county or state/county/census tract level, only the states involved are outlined and only the registry areas with data are categorized and mapped. If the data is provided at the Seer Registry area level, all of the states are outlined, but only the Seer Registry areas with data are categorized and mapped. Each row in the statistics data.frame represents one area (a state, state/county, state/county/census tract, or Seer registry area.) Either the U.S. FIPS code, the HSA number or the Seer Registry area abbreviation is used to tie the data with the associated geographic area boundary for mapping. The package supports mapping with either the 2000 or 2010 U. S. Census Bureau boundaries at the state (district, territory), HSA, and county levels using a modified version of the census boundary data. The package also supports mapping of the NCI Seer Registries and/or counties or census tracts within the Seer Registries. The *SeerMapper* function and all of the supporting boundary data and information, even when compressed, could not fit within CRAN's 5 megabyte package size limitation. Therefore, the package and the boundary information for the 2000 and 2010 census tracts are distributed as a set six (6) boundary dataset data only packages; the *SeerMapperRegs*, *SeerMapperEast* and *SeerMapperWest* packages provide the 2000 census tract boundaries and the *SeerMapper2010Regs*, *SeerMapper2010East* and *SeerMapper2010West* packages provide the 2010 census tract boundaries. When *SeerMapper* is installed, its dependencies on the other six (6) packages will automatically installs them. When *SeerMapper* is called and census tract boundaries are required for a specific census year and area, the package ensures the appropriate supplement packages are loaded and the census tract boundary data and information are available. The caller does not have to install or load the six(6) packages. *SeerMapper* handles management of these extra packages. All of the boundary data and information is intended for mapping purposes and should not be used for locating geographical points or area estimating. The boundary data has been simplified to speed up the mapping process and reduce the size of the boundary distributed with the package. The purpose of this package is to be fully self-contained and provide quick and easy method of generating maps from data. The boundary data and information used by the package is distributed amoung the packages as follows:

1. main package - Code and all state, regional, Seer Registry, HSA, and county boundaries for census years 2000 and 2010.
2. Census Year 2000 census tracts boundary packages
   (a) *SeerMapperRegs* -the census tract boundaries for 19 states containing Seer Registries (Regs);

(b) *SeerMapperEast* -the census tract boundaries for 20 states not containing Seer Registries east of the Mississippi river and DC and Puerto Rico;

(c) *SeerMapperWest* -the census tract boundaries for 13 states not containing Seer Registries west of the Mississippi river;

3. Census Year 2010 census tracts boundary packages

(a) *SeerMapper2010Regs* -the census tract boundaries for 19 states containing Seer Registries (Regs);

(b) *SeerMapper2010East* -the census tract boundaries for 20 states not containing Seer Registries east of the Mississippi river and DC and Puerto Rico;

(c) *SeerMapper2010West* -the census tract boundaries for 13 states not containing Seer Registries west of the Mississippi river;

These packages are automatically installed when *SeerMapper* is installed and are loaded as required by *SeerMapper*.

Usage

```
SeerMapper( ndf,
censusYear     = NULL,    # default: 2000 census
proj4          = NULL,    # default: NULL - no transformation, original boundary
                          #                projection is used to draw maps.
idCol          = NULL,    # default: "FIPS"
dataCol        = NULL,    # default: "Rate"
categ          = NULL,    # default, "5"
mTitle         = NULL,
mTitle.cex     = NULL,    # default: 1
us48Only       = NULL,    # default: TRUE
includePR      = NULL,    # default: FALSE
regionB        = NULL,    # default: "NONE" *
stateB         = NULL,    # default: "NONE" *
seerB          = NULL,    # default: "NONE" *
hsaB           = NULL,    # default: "NONE" *
countyB        = NULL,    # default: "NONE" *
tractB         = NULL,    # default: "NONE" *
dataBCol       = NULL,    # default: 'black'
fillTo         = NULL,    # default: "NONE"
clipTo         = NULL,    # default: "NONE"
hatch          = NULL,    # T/F or List of hatching options
hatch2         = NULL,    # default: NULL (empty)
mLegend        = NULL,    # T/F or list of legend options.
brkPtDigits    = NULL,    # default: 2
palColors      = NULL,    # default: "RdYlBu" w/11 cat. colors
debug          = NULL     # default: FALSE
)
```

Arguments

ndf	a data.frame containing identifier, data and pValue for each area to map. The *ndf* must contain the *id* and *data* column providing the geographical link to the boundary data and the data to be classified and colored by the package. See the 'idCol' and 'dataCol' parameters below for more details. Based on the type of area identifer used, the package can produces maps at the state, state/county, state/county/census tract, Seer Registry area, state/seer/county or state/seer/county/tract levels.
censusYear	is a numeric or character value of the the census year boundary data to use in the validation of the data and mapping of the areas. The value must be *2000* or *2010*. The default is *2000* in the *SeerMapper* call. Using the *SeerMapper2010* function call, the censusYear call parameter is set to "2010".
proj4	is a character vector specifying the proj4 parameters for the projection transformation of the map and hatching prior to drawning. By default the map projection is CRS("+proj=aea +lat_1=33 +lat_2=49 +lat_0=39 +lon_0=96W +ellps=WGS84") The 'proj4' parameter would be set as follows: proj4 = "+proj=utm +zone=15"
idCol	a character vector specifying the name of the column in the *ndf* data.frame to use for the rate data to categorize and mapped. The default value is *FIPS*.
dataCol	a character vector specifying the name of the column in the *ndf* data.frame to use for the rate data to categorize and mapped. The default value is *Rate*. If 'categ' is set to "data", then the *dataCol* column contains the integer category numbers instead of rates.
categ	a chacacter or numeric vector specifying a single numeric value (the number of calculated categories), a numeric vector of multiple values (to be used as the breakpoints in the data categorization), "data", or "colors".

```
categ=5
  - or -
categ=c(0.6, 0.8, 1.0, 1.2, 1.4)
  - or -
categ="data" or "colors"
```

The default value is *5*. This default allows mapping of any range of data quickly. If more categories or specific categories are then required, the 'categ' parameter can be used to adjust the categorization.

When the number of categories is specified as a single value, the number of categories must be an integer and within the range of 3 to 11 and less than or equal the maximum number of categories supported by the color palette selected with the 'palColors' parameter. For example: 'categ' = 5. (See the 'palColors' section for more detail.)

When the actual break points are provided, the vector must have between 3 and the maximum number of categories supported by the color palette MINUS one. For example, if 'palColors'="Spectral", the maximum number of categories is 11. Therefore, the maximum number of breakpoints that can be provided in the 'categ' list is 10. For example: 'categ' = c(0.6, 0.8, 1.0, 1.2, 1.4).

If any map has the same color for all areas with data and a breakpoint list is used, check the range of the data and the breakpoint list to determine why all of the area were categorized into the same color.

If multiple maps of data need to be compared, it is recommended, the same breakpoint list be used for all package call to ensure the categories and coloring are the same on each map.

When the 'categ' parameter is set to *data* or *colors*, the data in the first parameter data.frame is not categorized.
When set to *data*, the data is treated as the sub-area's category value. The value must be a positive integer value and should in range from 1 to "n". The lowest value does not have to be 1, but the range of values should be < 11 to produce effective maps. The first color is assigned to the lowest value. Additional colors are assigned to each additional integer (+1) up to 11 colors. In this way the range of categories is used to determine the number of colors required. The palette specified in 'palColors' limits the number of category values the caller can specify. For example: for 'palColors'="Spectral" the data value range is limited to 11 values, such as c(1:11) or c(5:16). It is recommended the first category value should always be 1.
When set to *colors*, the data is treated at the color the sub-area will be filled with. The data values are validated to make sure they represent valid color names or "#hhhhhh" values. If a color name is used, it must match a name created by the color() function. If a color hex value is used, it must in the form of "#rrggbb" or "#rrggbbaa" where each digit is a digits (0-F) for the "r"ed, "g"reen, "b"lue and "a"lpha values of the color. See the *rgb* and *col2rgb* functions for more details.

`mTitle` A character vector with one or two character strings to be used as the title of the map. For example: mTitle=c("Major Title Line","minor title line")

`mTitle.cex` A numeric specifying the size of the title lines.

`us48Only` a logical value. If *TRUE*, only the contiguous 48 states and DC are mapped. Alaska, Hawaii and Puerto Rice are not mapped and any data for sub-areas in these states are ignored. This also includes data for Seer Registries in Alaska and/or Hawaii. If data for the Hawaii registry is being mapped, make sure to use the default or set 'us48Only' to *FALSE*. If *FALSE*, All 51 states and DC are mapped. PR can optionally be included is 'includePR' is set to *TRUE*. The default value is *FALSE*.

`includePR` a logical value. If set to *TRUE* and 'US48Only' = *FALSE*, Puerto Rico is mapped with the rest of the states and DC. If set to *FALSE*, Puerto Rico is not mapped. The default value is *FALSE*.

`regionB` Regional Boundary Option. This option has three values: *NONE*, *DATA*, and *ALL*. The default value is *NONE*. When set to *NONE*, no regional boundaries are mapped.
When set to *DATA*, regional boundaries drawn when a region contains a state, county or tract with data.
When set to varALL, all regional boundaries are drawn.
The regions are the 4 U. S. census regions of NorthEast, South, West, and Mid-West.

`stateB` State Boundary Option. This option has four values: *NONE*, *DATA*, *REGION*, and *ALL*. The default value is *DATA* with state data and *NONE* for all other levels of data. When set to *ALL*, all state boundaries are mapped.
When set to *REGION*, all states within a region are drawn when the region contains a sub-area with data.
When set to *DATA*, only the state boundaries are drawn if the state or a sub-area has data values.
When set to *NONE*, None of the state boundaries are drawn.

`seerB` Seer Registry Area Boundaries Option. This option has five values: *NONE*, *DATA*, *STATE*, *REGION*, and *ALL*. The default value is *DATA* with Seer Registry data and *NONE* for all other levels of data.
When set to *ALL*, all registry boundaries are drawn.
When set to *REGION*, all registry boundaries are drawn within a region when the region contains a state, registry or sub-area with data.
When set to *STATE*, all registry boundaries within a state are drawn when the state contains sub-areas with data.
When set to *DATA*, only registry boundaries are drawn if the registry contains areas with data values.
When set to *NONE*, no registry boundaries are drawn.

`hsaB` Health Service Area Boundary Option. This option has four values: *NONE*, *DATA*, *SEER*, and *STATE*. The default value is *DATA* with HSA data and *NONE* for all other levels of data.
When set to *STATE*, all HSA boundaries within a state are drawn if the state contains sub-areas with data.
When set to *SEER*, all HSA boundaries within a Seer Registry are drawn if the state contains sub-areas with data.
When set to *DATA*, only the HSA boundaries are drawn if the HSA or a sub-area in it has data.
When set to *NONE*, None of the HSA boundaries are drawn.

`countyB` County Boundaries Option. This option has five values: *NONE*, *DATA*, *HSA*, *SEER*, and *STATE*. This option is only valid when county or census tract level data is used. The default value is *DATA* with county level data and *NONE* for all other levels of data.
When set to *NONE*, No county boundaries are drawn.
When set to *DATA*, Only county boundaries are drawn, if the county has data or contains a tract with data values.
When set to *HSA*, All county boundaries are drawn within a HSA, if the HSA contains any county or tract with data values.
When set to *SEER*, All county boundaries are drawn within the Seer Registry, if the registry contains any county or tract with data values.
When set to *STATE*, All county boundaries are drawn within a state, when the state contains any county or tract with data.

`tractB` Census Tract Boundaries Option. This option has six values: *NONE*, *DATA*, *COUNTY*, *HSA*, *SEER*, and *STATE*. This option is only valid with census tract level data. The default value is *DATA* for tract level data and *NONE* for all other levels of data.
When set to *NONE*, No tract borders are drawn.
When set to *DATA*, All tract boundaries are drawn for tracts with data values.

When set to *COUNTY*, All tract boundaries are drawn for tracts within a county, if the county contains any tract with data values.
When set to *HSA*, All tract boundaries are drawn within a HSA, if the HSA contains any tract with data values.
When set to *SEER*, All tract boundaries are drawn for tracts within a registry, if the registry contains any tract with data values.
When set to *STATE*, All tracts boundaries are drawn for trats within a state, if the state contains any tract with data values

`fillTo` This parameter has been replaced by the full implementation of the '`stateB`', '`seerB`', '`countyB`', and '`tractB`' call parameters. Parameter is obsolete.

`clipTo` a character or logical value. This parameter controls how the map will be drawn in the in the grapics area and how the spatial box is calculated. The values for *clipTo* are: TRUE, FALSE, "NONE", "DATA", "SEER", or "STATE". If *clipTo* is set to FALSE or "NONE", the graphics box is set to cover all of the requested boundaries. So, if all of the state boundaries were requested and only one state has data, the entire U.S. will be mapped. If *clipTo* is set to TRUE or "DATA", the data sub-areas as would occur is all boundary controls were set to "NONE" and the level of the data set to "DATA". (e.g., for county data, stateB="NONE", seerB="NONE", and countyB="DATA". If additional boundaries are requested the extend beyond the data sub-areas, they will be clipped at the graphic edges. If *clipTo* is TRUE or "DATA", the spatial box is set to cover then the graphics is clipped to the spatial box that contains the data areas being mapped. Any boundaries extending beyond the data area would be clipped at the edge of the graphic space. If *clipTo* is "SEER" and the data is in Seer Registries, then the boundary of the Registries is used as the graphics box. If the *clipTo* is "STATE", then the state or states containing the data sub-areas are used as the graphics box for the mapping.

`dataBCol` is a character vector of one element. It is used to specify the color of the data level boundary on the maps. The default value is 'black'. The value must be a color name from the `colors()` name list or a string starting with a "#" with 6 or 8 hexidecimal digits representing the RGB and transparancy values of a color. If the color specified is not valid, the function will generate a warning and stop.

`mLegend` is a list of legend options. It must a list containing named items. The mLegend can be disabled by setting '`mLegend`' = *NA* is the call. The legend is placed on the bottom of the map. The following options are available to fine tune the map's legend: '`counts`', '`size`', '`numberColumns`', '`pos`', and '`noValue`'. See below for more details on each option. The options are specified in the call by setting `mLegend = list(counts=TRUE, size=0.6)`, for example.

numCols is the number of columns to use when creating a legend. The default is *3*. The number of columns can range from *1* to *8*. (Old name: *legendColn* and *ncol*)

size is a numeric value used to size the legend symbols and texts as required. The default value is 0.85. (Old name: legendCex)

pos is a character vector specifying the location of the legend along the bottom of the map. The acceptable values are *left*, *center*, and *right*. The default is *left*. (Old name: legendPos)

counts is a logical variable - *TRUE* or *FALSE*. If *TRUE*, the observation counts for each category are displayed in the legend after the description text. The default is *FALSE*. (Old name: *legendCnt*)

noValue is a logical variable - *TRUE* or *FALSE*. If *TRUE*, any category containing no observations (data) is tagged in the legend with "NV" after the category label. The default is *FALSE*.

`hatch` is a logical value of *TRUE* or *FALSE* or a list structure with hatching settings. The call parameter enabled and provided parameters for hatching areas (state, Seer Registry, county or census tracts) based on data and a criteria. The most common use of hatching is to indicate the reliability of the data via a P-Value. In this case, the range of the data is 0 to 1, the comparison value is 0.05 and the criteria is greater than to hatch an area. If the '`hatch`' parameter is set to *TRUE*, then hatching will be implemented using the standard settings for hatching. The standard settings are: '`range`' = 0 to 1, '`value`' = 0.5, '`ops`' = *gt*, and '`dataCol`' = *pValue*. If set to *FALSE*, no hatching is done. If the '`hatch`' call parmeter is set to a list, hatching is enabled and the list is inspected for overriding values for the default setting. If two hatching patterns are required, the hatch2 call parameter can be used to specify a second data column, operator and test valid. (see below).

The following hatch settings may be changed using the '`hatch`' call parameter to pass a list of items and new values. The acceptable set of items are:

dataCol a character vector specifying the name of the column in the *ndf* data.frame to use to test the P_Value for each area. The value in this column used to test if the P_Value is < *0.05*. If the value is not < *0.05*, then the associated area is hatched as an indicator. The default column name value is *pValue*. Please note the same variable name is used as the statistical data, but this variable is used for the hatching feature.

ops a character value. The values may be one of the following: "eq", "ne", "lt", "le", "gt", or "ge". These translate to "==", "!=", "<", "<=", ">", and ">=" operations. Other notations are allowed and translated appropriately: "<>", "=>", "=", and "=<". The comparison is the data in the *dataCol* column to the '`hatch`' *value* option using the *ops* operation.

The formula is *dataCol ops value*.

For example, for P-Value testing the test is *data* > *0.05*. A *TRUE* result will cause the area to be hatched.

value a numeric break point for comparison with the hatching data column (pValue). Default is 0.05.

range a vector of two values, min and max inclusively. Range checking of is normally disabled, but can be used to validate the data used to determine if hatching should be done for an area. *range* can be set to: *NA*, *FALSE* to disable range checking, *TRUE* to enable range check for varc(0,1), or to a range checking vector of a low and high value, c(1234,532). If the low and high value of the range is out of order, it will be sorted and a warning message generated. If the range of the hatching data is not know, it is recommended to disable the range checking. If the data is pValue data, then range=TRUE will ensure all of the data is valid.

lab a character vector containing a label for this hatching activity. The default is "hatch#1".

col a character vector containing the name or `#A8A8A8` value of the color to be used for the area hatching pattern. The default value is `grey(0.66)` or `#A8A8A8`.

lwd a numeric value. The line weight for the hatching lines. Default value is *0.5*.

density a numeric value. *density* or *den* can also be used as the option name. When drawing the hatching over the area, this option specifies the density of the hatching pattern in lines per inch. The valid range is from 5 to 64 lines per inch. The default value is *25*.

angle a numeric value. When drawing the hatching over the area, this option specifies the angle of the hatching pattern in degrees counter clockwise. The default value is 45 degree. The value must be between 360 and -360 degrees.

incAngle a numeric value. When drawing multiple hatching lines, this option specifies the increment the angle will be increased for subsequent hatching. The default value is 60 degree. The value must be between 360 and -360 degrees.

`hatch2` is used to specify the options for a second hatching overlay on the maps. If the '`hatch2`' parameter is set a list of options, then the hatching is enabled. This list only contains the location of the data ('`dataCol`'), the operator ('`ops`'), the test value ('`value`') and the range test ('`range`') is needed. All other hatching options are taken from the '`hatch`' defaults or option values.

The most common use of hatching is to indicate the reliability of the data via a P-Value. The '`hatch2`' parameter uses the same defaults as '`hatch`'. The standard options and settings are: '`range`' = 0 to 1, '`value`' = 0.5, and '`ops`' = *gt*. The '`dataCol`' must be defined to enable '`hatch2`'.

The acceptable set of options are:

dataCol a character vector specifying the name of the column in the *ndf* data.frame containing the values for each sub-area to use in the hatching test. This option must be specified to enable the second hatching feature. The default column name value is NJLL.

ops a character value. The value may be one of the following: "eq", "ne", "lt", "le", "gt", or "ge". These translate to "==", "!=", "<", "<=", ">", and ">=" operations. Other notations are allowed and translated appropriately: "<>", "=>", "=", and "=<". The comparison is the data pointed to by *dataCol* column name using the '`ops`' operater to the '`value`' test value.
The formula is '`dataCol`' '`ops`' '`value`'.
For example, for P-Value testing the test is *data* > *0.05*. A *TRUE* result will cause the area to be hatched.

value a numeric test value for comparison with the hatch2 data column (dataCol) provided by the caller. Default is 0.05.

range a vector of two values, min and max inclusively. Range checking of is normally disabled, but can be used to validate the hatch2 data for the sub-areas. *range* can be set to: *NA* or *FALSE* to disable range checking, *TRUE* to enable range check for varc(0,1), or to a range checking vector of a low and high value, c(1234,532). If the low and high value of the range is out of order, they will be sorted and a warning message generated. If the range

of the hatching data is not know, it is recommended to disable the range checking.

lab a character vector containing a label for this hatching activity. The default is "hatch#2".

`brkPtDigits` specifies the number of digits to the right of the decimal place when calculating or pre-processing the break point list and resulting categorization intervals. Default is the calculated number of digits based on the interval between break points. For example: if '`brkPtDigits`' = *1*, the breakpoint rounded and limited to only having one digit. The valid range is 1 to 4. The default value is *2*. This parameter is only active when the '`categ`' call parameter specifies the number of categories.

`palColors` a character string specifying the *RColorBrewer* color palette used to generate the colors to map the data categorization. The specified palette is check for validity and the maximum number of colors for the palette is used to limit the number of categories the user can specify in *categ* parameter. See the *RColorBrewer* help for a list of the acceptable color palettes. This call parameter is only active when the '`categ`' is NOT set to "COLORS".

The permitted color palettes and maximum number of colors (categories) supported are:

Type of Palette	Palette Name	Maximum
Sequential	Blues	9
Sequential	BuGn	9
Sequential	BuPu	9
Sequential	GnBu	9
Sequential	Greens	9
Sequential	Greys	9
Sequential	Oranges	9
Sequential	OrRd	9
Sequential	PuBu	9
Sequential	PuBuGn	9
Sequential	PuRd	9
Sequential	Purples	9
Sequential	RdPu	9
Sequential	Reds	9
Sequential	YlGn	9
Sequential	YlGnBu	9
Sequential	YlOrBr	9
Sequential	YlOrRd	9
Diverging	BrBG	11
Diverging	PiYG	11
Diverging	PRGn	11
Diverging	PuOr	11
Diverging	RdBu	11
Diverging	RdGy	11
Diverging	RdYlBu	11
Diverging	RdYlGn	11
Diverging	Spectral	11

Qualitative	Accent	8
Qualitative	Dark2	8
Qualitative	Paired	12
Qualitative	Pastel1	9
Qualitative	Pastel2	8
Qualitative	Set1	9
Qualitative	Set2	8
Qualitative	Set3	12

All palettes have minimum number of colors (categories) of 3. For more details on each palette and their characteristics visit the http://www.colorbrewer.org website. The actual parameter testing of the palette names and maximum number of colors for each palettes is based on the brewer.pal.info data.frame provided by RColorBrewer package. It is strongly recommended the users stay with color schemes in the diverging group.

debug is a logical variable. When set to *TRUE*, debug prints of variables and the flow of the package are enabled. Use of this option adds a large number of debug lines to the package standard device output.

Details

The *SeerMapper* package (and the *SeerMapper2000* and *SeerMapper2010* functions) provide a simple means of mapping data for the U. S. at several levels: State, State/County, State/County/Census Tract, and U. S. Seer Registry areas. The package automatically determines the mapping level based on the data and the area identifiers provide at the time of the call. The data is categorised and each area is drawn and colored appropriately based on its data value.

The *censusYear* call parameter controls which census year name tables and boundary data will be used for the mapping. The *SeerMapper2000* function overrides the *censusYear* parameter and sets it to "2000" for the 2000 U. S. census. The *SeerMapper2010* function sets the *censusYear* parameter to "2010" for mapping with the 2010 U. S. location FIPS codes. The default value for the *censusYear* parameter is "2000", so calling *SeerMapper* without the *censusYear* parameter set, is equivalently the same as calling *SeerMapper2000*.

The level of the map is determined by the identifier used in the data provided when the function is called.

If the data uses the U. S. State fips code (2 digits) as the spatial identifer, then the state boundaries are drawn and colored. In this mode, the caller has two options: draw all of the states in the US (or continental 48 state), drawn only the states with data. This is controled by the '`stateB`' call parameter. It's default value for state level data is *ALL*. To only draw boundaries for states with data, specify '`stateB`' = *DATA*. Seer Registry, county and census tract boundaries will not be drawn.

If the data uses the U. S. State and County fips code (5 digits) as the spatial identifier, then the state and county boundaries are drawn and colored for areas with data. In this mode, there are several options:

a) Draw only the boundaries for the counties with data ('`countyB`' = *DATA*), b) Draw the counties with data and their state's boundaries (states with data) ('`stateB`' = *DATA*), c) Draw the boundaries for all of the counties in the states where data was provided for any county in the state (state

with data and counties within the state) ('stateB' = *DATA* and 'countyB' = *STATE*), d) Draw the bountaries of all of the counties within a Seer Registry where data was provided for any county in the Seer Registry (Seer Registry with data) ('countyB' = *SEER*),

If the data uses the U. S. State, County and Census Tract fips code (11 digits) as the spatial identifier, then the census tracts boundaries are drawn and colored for areas with data. In this mode, there are several options:

a) Draw only the boundaries for the census tracts with data ('tractB' = *DATA* default setting), b) Draw the tracts with data and their state's boundaries (states with data) ('stateB' = *DATA*), c) Draw the boundaries for all of the tracts in the states where data was provided for any tract in the state (state with data and tracts within the state) ('stateB' = *DATA* and 'tractB' = *STATE*), d) Draw the bountaries of all of the tracts within a Seer Registry where data was provided for any county in the Seer Registry (Seer Registry with data) ('tractB' = *SEER*),

If the spatial identifier does not match one of the above patterns, the package attempts to match the identifier against the list of Seer Registry abbreviations and an partial match against keywords for the Seer Registries. If there is no match, an error message is generated to inform the caller. When Seer Registry data is used, the package maps each Seer Registry based on the data presented.

In this mode, there are several options:

a) Draw only the boundaries for the Seer Registries with data without any state boundaries ('seerB' = *DATA* and 'stateB' = *NONE*, the default setting), b) Draw only the boundaries for the Seer Registries with data and their associated state boundaries ('seerB' = *DATA* and 'stateB' = *DATA*), c) Draw only the boundaries for the Seer Registries with data and all of the U.S. state boundaries ('seerB' = *DATA* and 'stateB' = *ALL*), d) Draw all of the Seer Registries boundaries within a state with data and only the state boundaries with data ('seerB' = *STATE* and 'stateB' = *DATA*).

For each level of data, there are more options related to how and when the state, county, census tract and Seer Registry are drawn. See the descriptions for the 'stateB', 'seerB', 'countyB', and 'tractB' call parameters.

The fips codes used to idenfity an area, but all be the same in the data; for states data, the 2 digit code, for state/countiesdata, the 5 digit code consisting of the 2 digit state and 3 digit county codes, for state/county/tract data, the 11 digit fips code consisting of the 2 digit state code, 3 digit county code, and the 6 digit census tract code.

For Seer Registry mapping, the identifier must be the Seer Registry abbreviate or a character string the uniquely partically matches one and only one alias string for the Seer Registry. The acceptable abbreviations and partial matches are:

Seer Registry Abbr	Seer Registry Alias	Description
AK-NAT	ALASKA	Alaska Native Registry
AZ-NAT	ARIZONA	Arizona Native Registry
CA-LA	LOS ANGELES	Los Angeles CA Registry
CA-OTH	CALIFORNIA EXCL	California Registry all county not in other CA registries
CA-OTH	GREATER CALIF	See Above
CA-SF	SAN FRAN	California San Fran/Oakland
CA-SJ	SAN JOSE	California San Jose/Monteray
CT	CONNECTICUT	State of CT Registry
GA-ATL	ATLANTA	Atlanta GA metro
GA-OTH	GEORGIA OTHER	All counties no in the Altanta or Rural registries
GA-OTH	GREATER GEOR	See above

GA-RUR	RURAL GEORG	Rural GA counties Registry
HI	HAWAII	State of HI Registry
IA	IOWA	State of Iowa Registry
KY	KENTUCKY	State of Kentucky Registry
LA	LOUISIANA	State of Louisiana Registry
MI-DET	DETROIT	Detroit Metro Registry
NJ	NEW JERSEY	State of NJ Registry
NM	NEW MEXICO	State of NM Registry
OK-CHE	CHEROKEE	Oklahoma Cherokee Nationn Registry
UT	UTAH	State of Utah Registry
WA-SEA	PUGET	Seattle/Puget Sound Registry
WA-SEA	SEATTLE	See Above

To be able to handle data from different sources, there may be multiple alias lines in the above table.

The package depends on the following other third party R packages for functionality: graphics, stringr, RColorBrewer, stats, sp, maptools, and rgdal.

Value

None

Author(s)

James B Pearson, Jr <jbpearson353@gmail.com> and Linda W Pickle <lwpickle353@gmail.com>
Maintainer: "Joe Zou" <zouj@imsweb.com>

Examples

```
#####
#
#  Generate data.frame of all data for the US States and DC
#

data(USStates_CM_St_Data,envir=environment())
str(USStates_CM_St_Data)

#
#####

#####
#
#  Example # s01 - States Mapping Basic defaults
#
#  In this example rate and pValue data is provided for each state.
#  The number of categories requested is set to 5.
#  The "pValue" column in the data is used to hatch states when
#  the pValue > 0.05.  The defaults on the hatch feature are tuned
#  support hatching of pValue data that is not-significate.
```

```
#
#  Border defaults for state data are = stateB=ALL, seerB=NONE.
#
#  This example uses example s1's data.  All state boundaries are drawn.
#

TT   <- c("Ex-s01 States Mapping Cancer Mortality Rate",
          "all defaults")
save.f<-tempfile(pattern = "", fileext = "SM-Ex-s01 States Map of Cancer Mortality-defaults.pdf")
pdf(save.f,width=8,height=10.5)
SeerMapper(USStates_CM_St_Data,mTitle=TT)
dev.off()

#
#  The pValue data in the dataset was assigned 0.02 or 0.2 based on a
#  comparison the state's confidence interval values and the US's rate,
#  for age adjusted rates for all cancers and cancer deaths for the years
#  2009 to 2013.
#
####

#####
#
#  Example # s02 - States Mapping Basic with Hatching
#
#  In this example rate and pValue data is provided for each state.
#  The number of categories requested is set to 5.
#  The "pValue" column in the data is used to hatch states when
#  the pValue > 0.05.  The defaults on the hatch feature are tuned
#  support hatching of pValue data that is not-significate.
#
#  Border defaults for state data are = stateB=ALL, seerB=NONE.
#
#  This example uses example s1's data.  All state boundaries are drawn.
#

TT   <- c("Ex-s02 States Mapping Cancer Mortality Rate",
          "w hatching and defaults")
save.f<-tempfile(pattern = "", fileext = "SM-Ex-s02 States Map of Cancer Mortality-w hatching.pdf")
pdf(save.f,width=8,height=10.5)

SeerMapper(USStates_CM_St_Data,
           hatch   = TRUE,
           mTitle  = TT
       )

dev.off()

#
#  The pValue data in the dataset was assigned 0.02 or 0.2 based on a
#  comparison the state's confidence interval values and the US's rate,
#  for age adjusted rates for all cancers and cancer deaths for the years
#  2009 to 2013.
```

```
#
####

####
#
#  Generate Partial States data.frame for Examples s04 through s15
#
#  This dataset is created one and re-used.  If the examples
#  are unindependly, the code to generate the dataset must
#  be run first.
#

data(USStates_CM_St_Data,envir=environment())

USStates_P   <- USStates_CM_St_Data         # get copy
numStates    <- dim(USStates_P)[1]          # get number of rows (states)
selectStates <- (runif(numStates) <= 0.75 ) # select random 75% of states
USStates_P   <- USStates_P[selectStates,]   # pull out data

#
#####

#####
#
#  Example # s03 - Partial State Mapping with pValue Hatching
#
#  The package does not have to have data for every state/DC.  Partial
#  data can also be mapped.  States without data are not colored (white).
#
#  This example uses a randomly selected set data for 75 % of the states/DC.
#
#  The number of categories is set to 5 (categ=5), and hatching
#  is enabled using the default hatching options on the data column "pValue".
#
#  By default for state level data, the boundaries for all U.S. states/DC are
#  drawn to provide a complete map (stateB="ALL").
#

TT   <- c("Ex-s03 Partial States Map",
          "all defaults with hatching")
save.f<-tempfile(pattern = "", fileext = "SM-Ex-s03 Partial States Map w hatching defaults.pdf")
pdf(save.f,width=8,height=10.5)

SeerMapper(USStates_P,
               hatch   = list(dataCol="pValue"),       # test pValue column for < 0.05
               mTitle  = TT
            )

dev.off()

#
####
```

```
#####
#
#  Example # s04 - Partial State Mapping with pValue Hatching
#         and boundaries for all states and seer registries.
#
#  If stateB="ALL" and seerB="ALL", then boundaries for all of the states and Seer
#  Registries are drawn.  This is one solution for the map generated in example # s06.
#

TT   <- c("Ex-s04 Partial State Map w hatching",
          "Outline all States and Regs")
save.f<-tempfile(pattern = "", fileext = "SM-Ex-s04 Partial States Map w hatching stB-A srB-A.pdf")
pdf(save.f,width=8,height=10.5)

SeerMapper(USStates_P,
                stateB  = "ALL",             # outline all states
                seerB   = "ALL",             # outline all registries
                hatch   = TRUE,              # test pValue column for < 0.05
                mTitle  = TT
              )
dev.off()

#####
#
#  Example # s05 - Partial States Mapping with pValue Hatching
#         No state boundaries, but boundaries for all Registries
#           stateB=NONE and seerB=ALL
#

TT   <- c("Ex-s05 Partial States Map w hatching, cat=7",
          "No State boundaries, All Regs, w/column names")

ex.f<-"SM-Ex-s05 Partial States Map w hatching cat-7 stB-N srB-A w column names.pdf"
save.f<-tempfile(pattern = "", fileext = ex.f)
pdf(save.f, width=8, height=10.5)

SeerMapper(USStates_P,
           idCol   = "FIPS",dataCol="AA_Rate",
           stateB  = "NONE",        # no state outlines
           seerB   = "ALL",         # all registries
           categ   = 7,             # number of categories to generate and use.
           hatch   = TRUE,          # test pValue column for < 0.05
           mTitle  = TT
)
dev.off()

#
####

######
#
#  Generate Seer Regs data.frame for 17 of the 20 registries and
#    a smaller Seer Regs data.frame for the original 12 registries
```

```
#    All of the registry data.frames serve as partial data sets.
#    The 12 registry data.frame shows the features the best.
#
#  The following script creates the dataset for use in examples sr30-sr41.
#  Since it is not re-created in the code for each examples, this code
#  must be run or copied to the example as needed.
#

data(SeerRegs_CM_Data,envir=environment())
str(SeerRegs_CM_Data)

# Get US rate for "All_Both" sexes and races.
USRate <- SeerRegs_CM_Data[2,"All_Both"]
cat("USRate:",USRate,"\n")

# strip off first to rows as required
SeerRegs_CM_Data <- SeerRegs_CM_Data[c(-1,-2),]
# this gets ride of Seer Reg and U.S data.

#  Select data for the original Seer 13 Registries without Alaska.
srList <- c("CT", "MI-DET", "GA-ATL", "GA-RUR",
            "CA-SF", "CA-SJ", "CA-LA", "HI", "IA",
            "NM", "WA-SEA", "UT")
SeerRegs_CM_Data_12 <- SeerRegs_CM_Data[srList,]

#

#####
#
#  Example # sr10 - Seer Registry 12 Mapping
#
#  Of the 21 NCI Seer Registries, most mapping occurs using the
#  12 primary registries. They include: Connecticut, Detroit,
#  Atlanta, Rural Georgia, Hawaii, Iowa, Utah, New Mexico,
#  Greater California, Greater Georgia, New Jersey,
#  Kentucky, Louisiana, San Francisco/Oakland, San Jose/Monterey,
#  Los Angeles, and Seattle-Puget Sound.
#  The default stateB and seerB call parameter values for
#  for Seer Registry data are:  stateB="NONE" and seerB="DATA".
#  The countyB and tractB parameters are ignored.
#
#  This example drawn boundaries for Seer Registries with data,
#  but does not include any state boundaries.  The registries
#  jsut float.  This is useful when you are mapping a few
#  Seer Registries, like Georgia Rural and AtLanta Metro.
#
TT   <- c("Ex-sr10 Seer Reg 12 Map-Cancer Mort. Rates All Both",
          "cat=6, def: stateB-NONE, seerB-DATA" )

ex.f<-"SM-Ex-sr10 Seer Reg 12 Map cat-6 stB-N srB-D.pdf"
save.f<-tempfile(pattern = "", fileext = ex.f)
pdf(save.f, width=8, height=10.5)
```

```
SeerMapper(SeerRegs_CM_Data_12,
           idCol   ="Registry",dataCol="All_Both",
           categ   =6,
           mTitle  =TT
)
dev.off()

#
####

#####
#
#  Example # sr11 - Seer Reg All States Map w Hatching
#                 stateB=DATA   seerB=DATA
#
#  If stateB = "DATA", the boundaries for the states/DC are drawn that
#  contain Seer Registries with data.
#  This provides the state outlines around Registries with data.
#  Since only a few of the Seer Registries have data, nott all of the
#  state boundaries are drawn.  A partial U.S. map appears.
#

TT   <- c("Ex-sr11 Seer Reg 12 Map Seer wD and States wD",
          "stateB=DATA, seerB=DATA")

ex.f<-"SM-Ex-sr11 Seer Reg 12 Map-stB-D, srB-D.pdf"
save.f<-tempfile(pattern = "", fileext = ex.f)
pdf(save.f, width=8, height=10.5)

SeerMapper(SeerRegs_CM_Data_12,
           idCol   = "saID",dataCol="All_Both",
           stateB  = "DATA",  # drawn boundaries for all states.
           mTitle  = TT
)
dev.off()


#####
#
#  Example # sr12 - Seer Reg All States Map w Hatching
#                   stateB=ALL   seerB=ALL
#
#  With stateB = "ALL", the boundaries for all of the states/DC are drawn.
#  This provides a full U.S. map.  To add the boundaries for all of the
#  Seer Registries, seerB = "ALL" is set.
#

TT   <- c("Ex-sr12 Seer Reg 12 Map Seer-A and States-A",
          "stateB=ALL, seerB=ALL")
ex.f<-"SM-Ex-sr12 Seer Reg 12 Map-stB-A, srB-A.pdf"
save.f<-tempfile(pattern = "", fileext = ex.f)
pdf(save.f, width=8, height=10.5)
```

```
SeerMapper(SeerRegs_CM_Data_12,
           idCol   = "saID",dataCol="All_Both",  # specify the column names.
           stateB  = "ALL",  # drawn boundaries for all states.
           seerB   = "ALL",
           mTitle  = TT
)
dev.off()

# #####
# The below examples are uncommented due to the example running time limit on CRAN
# #####
# #
# #  Example # sr20 - Seer Registry West - Level Mapping
# #
# #  With seerB="DATA", the package only maps Seer Registries with data.
# #  To include the state boundaries, it's best to use stateB="DATA".
# #  Otherwise, the entire US is mapped.
# #
# #  This example maps the western area Seer Registries in California,
# #  New Mexico, Utah and Washington-Seattle/Puget sound.
# #
#
# TT   <- c("Ex-sr20 Seer Registry Area West Males",
#           "stateB=DATA, seerB=DATA")
#
# pdf("SM-Ex-sr20 Seer Regs West Males stB-D srB-D.pdf",
#     width=8, height=10.5)
#
# SeerMapper(SeerRegs_CM_Data_West,
#            idCol   = "Registry",dataCol="All_Males",
#            stateB  ="DATA",
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# ####
#
# #####
# #
# #  Example # sr21 - Seer Registry West - Level Mapping
# #
# #  To drawn the state boundaries in the region, but not the
# #  entire U.S., stateB can be set to "REGION".  The package
# #  is aware of the 4 U. S. Census regions.  The "REGION" option
# #  is available with the stateB and seerB boundary controls.
# #
# #  This example maps the western area Seer Registries in California,
# #  New Mexico, Utah and Washington-Seattle/Puget sound with state boundaries
# #  drawn to the western region boundary.
# #
#
```

```
# TT   <- c("Ex-sr21 Seer Registry Area West Males",
#           "w hatching, stateB=REGION, seerB=DATA")
#
# pdf("SM-Ex-sr21 Seer Regs West Males stB-R srB-D.pdf",
#     width=8, height=10.5)
#
# SeerMapper(SeerRegs_CM_Data_West,
#            idCol   = "Registry",dataCol="All_Males",
#            stateB  ="REGION",
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# ####
#
# ####
# #
# #  The next set of examples show mapping of county data in many different situations.
# #  The key controls of the border drawing are the stateB, seerB, and countyB call
# #  parameters.  The default values are DATA.  This says, draw the area border only
# #  if there is data provided within the area.  This applies to Seer Registry data,
# #  county data, and census tract data.  If the call parameter is set to "ALL", the
# #  associated border is ALWAYS drawn.   If the call parameter is set to "NONE", the
# #  associated border is almost always not drawn.  See notes below.
# #
# #  The caller has one additional control over when area borders are drawn: the fillTo
# #  call parameter.  Additional borders may be drawn for higher level areas as needed.
# #  If the fillTo call parameter is set to "NONE", only sub-areas (county or tract level)
# #  with data are colored and their borders drawn. This is the default, in most cases.
# #  If fillTo is set to "SEER", when a Seer Registry area contain any sub-area with data,
# #  all of the borders at that level are drawn within the Seer Registry area. But not
# #  within the state or neighboring Seer Registry areas.
# #  If fillTo is set to "STATE", when a state contains any sub-area (Seer Registry, county,
# #  tract level) with data, all borders at that level are drawn within the state.
# #
# ####
#
# #####
# #
# #  Create a data.frame for All and Partial Kentucky Counties.
# #
#
# data(Kentucky_CM_Co_Data,envir=environment())
# str(Kentucky_CM_Co_Data)
#
# KY_Co_DF <- Kentucky_CM_Co_Data         # start with the fill set of counties.
#
# lKY      <- dim(KY_Co_DF)[1]            # get number of counties
# selKY    <- (runif(lKY) <= 0.75 )       # select random 75% of counties
# KY_Co_P  <- KY_Co_DF[selKY,]
#
```

```
# #
# #####
#
# #####
# #
# #  Example # c30 Kentucky All Co Map w hatching,
# #         default - countyB="DATA", seerB="NONE", stateB="NONE"
# #
# #  In this example, countyB is set to "STATE", to tell the package
# #  to draw all of the county bountaries within any state containing
# #  a county with data.
# #
#
# TT   <- c("Ex-c30 Kentucky All County Map w hatching",
#           "defaults")
#
# pdf("SM-Ex-c30 Kentucky All Co Map w hatching-defaults.pdf",
#     width=8,height=10.5)
#
# SeerMapper(KY_Co_DF,
#            idCol   ="FIPS",dataCol="AA_Rate",
#            hatch   = list(dataCol="pValue"),
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# ####
#
#
# #####
# #
# #  Example # c31 Kentucky Partial Co Map
# #         default - countyB="DATA", seerB="NONE", stateB="NONE"
# #
# #  In this example, the only 75 percent of the county have data.
# #  The default settings are used.
# #
#
# TT   <- c("Ex-c31 Kentucky Partial County Map",
#           "defaults")
#
# pdf("SM-Ex-c31 Kentucky Partial Co Map-coB-D srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(KY_Co_P,
#            idCol   ="FIPS",dataCol="AA_Rate",
#            mTitle  = TT
# )
#
# dev.off()
#
```

```
# #
# #  Not very pretty.
# #
# ####
#
#
# #####
# #
# #  Example # c32 Kentucky Partial Co Map
# #         default - countyB="DATA", seerB="NONE", stateB="NONE"
# #
# #  To improve the c31 map, there are several direction that could be
# #  taken:  Add the state boundaries (stateB="DATA") or draw all of the
# #  county boundaries (countyB="STATE").  countyB="ALL" is not supported.
# #  In this case, since Kentucky is a single registry, seerB="DATA" has the
# #  same effect as stateB="DATA".  The difference is stateB="DATA" will not
# #  draw the missing county boundaries, while countyB="STATE" will draw all
# #  of the county boundaries up to the state border.
# #
#
# TT   <- c("Ex-c32 Kentucky Partial County Map",
#           "stateB='DATA'")
#
# pdf("SM-Ex-c32 Kentucky Partial Co Map-coB-D srB-N stB-D.pdf",
#     width=8,height=10.5)
#
# SeerMapper(KY_Co_P,
#            idCol   ="FIPS",dataCol="AA_Rate",
#            stateB  = "DATA",
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# ####
#
# #####
# #
# #  Example # c33 Kentucky Partial Co Map
# #         default - countyB="DATA", seerB="NONE", stateB="NONE"
# #
# #  This example has the countyB="STATE" set instead of the stateB="DATA"
# #  that was used in c32.
# #
#
# TT   <- c("Ex-c33 Kentucky Partial County Map",
#           "countyB='STATE'")
#
# pdf("SM-Ex-c32 Kentucky Partial Co Map-coB-St srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(KY_Co_P,
```

```
#            idCol   ="FIPS",dataCol="AA_Rate",
#            countyB  = "STATE",
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# ####
#
# #####
# #
# #  Create a data.frame of the Kentucky and Georgia counties (all)
# #   and a combined county data.frame.
# #
# #   Create a partial list of Georgia Counties.
# #
#
# data(Georgia_CM_Co_Data,envir=environment())
# GA_Co_Data  <- Georgia_CM_Co_Data
#
# data(Kentucky_CM_Co_Data,envir=environment())
# KY_Co_Data  <- Kentucky_CM_Co_Data
#
# TwoStatesData <- rbind(GA_Co_Data,KY_Co_Data)
#
# lGA         <- dim(GA_Co_Data)[1]       # get number of counties
# selectedGA  <- (runif(lGA) <= 0.75 ) # select random 75% of counties
# GA_Co_P     <- GA_Co_Data[selectedGA,]
#
# #
# #####
#
# ####
# #
# #  Example # c35 Multiple States - KY and GA County Mapping
# #             with hatching and stateB = DATA.
# #
# #  This example expands example c36 by setting stateB="DATA".  This
# #  drawns the Seer Registries around any sets of counties with data.
# #  Counties without data will have boundaries missing.
# #
#
# TT   <- c("Ex-c35 KY-GA All County Map",
#           "defaults")
#
# pdf("SM-Ex-c39 KY-GA All Co Map def:coB-D srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(TwoStatesData,
#            idCol   = "FIPS",dataCol="AA_Rate",
#            mTitle  = TT
# )
```

```
#
# dev.off()
#
# #
# #  seerB="ALL" and stateB="ALL" could be used, but will drawn
# #  a map the size of the entire U.S. and the counties and their
# #  colors will be very hard to see.
# #
# ####
#
# ####
# #
# #   Create a data.frame for Georgia counties in the Atlanta Registry,
# #   all counties.
# #
#
# GA_Co_Data_Atl <- GA_Co_Data[GA_Co_Data$saID == "GA-ATL",]
# # pull out of the data the Atlanta Registry.
#
# #
# #   Create a data.frame for the Georgia counties in the
# #   Atlanta and Rural registries - All and Partial
# #
#
# GA_Co_Data_Atl_Rur <- GA_Co_Data[(GA_Co_Data$saID == "GA-ATL" |
#                                     GA_Co_Data$saID == "GA-RUR"),]
# # pull out of the data the Atlanta Registry.
# lGA         <- dim(GA_Co_Data_Atl_Rur)[1]          # get number of counties
# selectedGA  <- (runif(lGA) <= 0.75 )            # select random 75% of counties
# GA_Co_Data_Atl_Rur_P <- GA_Co_Data_Atl_Rur[selectedGA,]
#
# #
# #####
#
# #####
# #
# #  Example # c45  GA Single Seer Registry (Atlanta) All Counties
# #           def: countyB=DATA, seerB=NONE, stateB=NONE
# #
# #  This example the all of the counties in the Georgia Atlanta Metro Seer
# #  Registry are selected for mapping. The other counties in the state are not
# #  listed in the data.frame, so have no data associated.  This example
# #  shows the map using the default boundary
# #  settings:  countyB="DATA", seerB="NONE", and stateB="NONE".
# #
#
# TT   <- c("Ex-c45 GA Atlanta Reg Co ",
#           "def countyB=DATA, seerB=NONE, stateB=NONE")
#
# pdf("SM-Ex-c45 GA Atl Reg Co def coB-D srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Co_Data_Atl,
```

```
#            idCol   ="FIPS",dataCol="AA_Rate",
#            mTitle  = TT
# )
# dev.off()
#
# #
# ####
#
# #####
# #
# #  Example # c46  GA Single Seer Registry (Atlanta) All Counties
# #                countyB=DATA, seerB=NONE, stateB=ALL
# #
# #  In this extreme example the more boundaries are drawn then really needed.
# #  The data is lost in the size of the map.  Not a good practice.
# #  If only a small area contains data, don't enable any set of
# #  boundaries that cover more than is needed.
# #  In this case, stateB is set to "ALL".
# #
#
# TT   <- c("Ex-c46 GA Atlanta Reg Co ",
#           "def countyB=DATA, seerB=NONE, stateB=ALL")
#
# pdf("SM-Ex-c46 GA Atl Reg Co def coB-D srB-N stB-A.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Co_Data_Atl,
#            idCol   ="FIPS",dataCol="AA_Rate",
#            stateB  ="ALL",
#            mTitle  = TT
# )
# dev.off()
#
# #
# ####
#
# ####
# #
# #  Example # c47 GA Two Seer Registry Partial Co Map,
# #           countyB="DATA", seerB="NONE", stateB="NONE"
# #
# #  This example maps the counties in two Georgia Seer Registries: Atlanta Metro
# #  and Rural with partial data can be mapped and the affects of the boundary options.
# #  By default only the boundaries and counties with data are mapped.
# #
#
# TT   <- c("Ex-c47 GA Atl-Rur Reg Partial w def",
#           "def: countyB=DATA seerB=NONE stateB=NONE")
#
# pdf("SM-Ex-c47 GA Atl-Rur Reg Partial Co def coB-D srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Co_Data_Atl_Rur_P,
```

```
#            idCol   ="FIPS",dataCol="AA_Rate",
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# #  NOTE: The boundaries of the state and Seer Registries at not clearly defined.
# #   The package works the same at the tract, county, seer and state levels
# #   with partial data.  The default is draw and fill sub-areas with data.
# #   Each level can be expanded to drawn sub-areas without data to the next
# #   level boundary with xxxxxB = "ALL", "REGION", "STATE", "SEER",
# #   and "COUNTY" options.
# #
# ####
#
#
#
# #####
# #
# #   Data for the following examples.  2010 census boundaries
#
# data(GA_Dem10_Co_Data,envir=environment())
# GA_D_Co_Data  <- GA_Dem10_Co_Data
#
# #####
# #
# #  Examples c60 to c66 use 2010 census demographic data and use the
# #  SeerMapper2010 function call to activate the 2010 boundary data collection.
# #
# #####
#
# #####
# #
# #  Example c60 - Georgia County Data-Population Density with defaults
# #
# #  Uses 2010 demographic County data.frame (GA_Dem10_Co_Data) loaded above.
# #
#
# TT   <- c("Ex-c60 Georgia Counties Population Density10, c=7",
#           "def: countyB='DATA' seerB='NONE' stateB='NONE'")
#
# pdf("SM-Ex-c60 GA Counties-PopDens10-c=7 def coB-D srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper2010(GA_D_Co_Data,
#                idCol   = "FIPS",dataCol="popdens",
#                categ   = 7,
#                mTitle  = TT
# )
#
# dev.off()
#
```

```
# #####
# #
# #  Example c61 - Georgia County Dem. Data for age 65 and up
# #
# #  Uses 2010 demographic county data.frame (GA_Dem10_Co_Data) loaded above.
# #
#
# TT   <- c("Ex-c61 Georgia County Dem10 Counts for age 65 and up.",
#           "def-countyB=DATA, seerB=NONE, stateB=NONE")
#
# pdf("SM-Ex-c61 GA Counties10-age 65 and up-def coB-D srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper2010(GA_D_Co_Data,
#                idCol   = "FIPS",dataCol="age.65.up",
#                categ   = 7,
#                mTitle  = TT
# )
#
# dev.off()
#
# #
# ####
#
# #####
# #
# #  Example c64 - Georgia County Dem10 Data for Percentage age 65 and up
# #
# #  The Georgia 2010 demographic county Dem dataset (GA_Dem10_Co_Data)
# #  is used in this example.
# #  The percentage (0% to 100%) of individuals in each tract that is
# #  65 year old or older is calculated and mapped.
# #
#
# # calculate the percentage of age 65 up vs population
#
# GA_D_Co_Data$PC.age.65.up <- ( GA_D_Co_Data$age.65.up / GA_D_Co_Data$pop2010 ) * 100
#
# TT   <- c("Ex-c64 Georgia County10 for PC 65 and up",
#           "def: countyB='DATA' seerB='NONE' stateB='NONE'")
#
# pdf("SM-Ex-c64 GA Counties10-PC age 65 and up-def.pdf",
#     width=8,height=10.5)
#
# SeerMapper2010(GA_D_Co_Data,
#                idCol   = "FIPS",dataCol="PC.age.65.up",
#                categ   = 7,
#                mTitle  = TT
# )
#
# dev.off()
#
# #
```

```
# ####
#
# #####
# #
# #  Example c65 - Georgia County Dem10 Data for Percentage households occupied
# #
# #  Using the Georgia 2010 demographic county dataset (GA_Dem_Co_Data),
# #  the percentage (0% to 100%) of the households that are occupied
# #  in each county is calculated and mapped.
# #
#
# # Calculate percentage of HH occupied vs HH units.
#
# GA_D_Co_Data$PC.hh.occupied <- ( GA_D_Co_Data$hh.occupied / GA_D_Co_Data$hh.units ) * 100
#
# TT   <- c("Ex-c65 GA County10 for PC HH Occupied",
#           "def: countyB='DATA' seerB='NONE' stateB='NONE'")
#
# pdf("SM-Ex-c65 GA Counties10-PC HH Occupied-trB-D coB-N srB-N stB-N",
#     width=8,height=10.5)
#
# SeerMapper2010(GA_D_Co_Data,
#                idCol   = "FIPS",dataCol="PC.hh.occupied",
#                categ   = 7,
#                mTitle  = TT
# )
#
# dev.off()
#
# #
# ####
#
# #####
# #
# #  Example c66 - Georgia County Dem10. Data for Percentage Household Renters
# #
# #  Using the Georgia 2010 demographic county dataset (GA_Dem_Co_Data),
# #  the percentage (0% to 100%) of the households that have renters
# #  in each county is calculated and mapped.
# #
#
# # calculate percentage of renters (1-owners) vs units
#
# GA_D_Co_Data$PC.hh.renter <- (1-( GA_D_Co_Data$hh.owner / GA_D_Co_Data$hh.units )) * 100
#
# TT   <- c("Ex-c66 GA County10 for PC Renters",
#           "countyB='DATA' seerB='NONE' stateB='NONE'")
#
# pdf("SM-Ex-c66 GA Counties10-PC Renters-trB-D coB-N srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper2010(GA_D_Co_Data,
#                idCol   = "FIPS",dataCol="PC.hh.renter",
```

```
#                categ   = 7,
#                mTitle  = TT
# )
# dev.off()
#
# #
# #####
#
#
#
# ####
# #
# #   In the situation where all of the counties with data do not reside
# #   in a Seer Registry, the behavor of the boundary options is a little
# #   different.
# #
# #   Example # c70 - Washington Seattle-Puget Sound partial Counties.
# #
# #   In this example we have selected a random set of counties from the
# #   state of Washington.  Some are in the Seattle-Puget sound Seer Registry
# #   and some are not.
# #
# #   The default settings of the boundary options works the same way.
# #   The countyB options works the same, except countyB="SEER" only
# #   adds the boundaries for counties within a Seer Registry with counties
# #   with data.  The counties outside the Registry is not drawn.
# #
#
# data(Washington_CM_Co_Data,envir=environment())
#
# WA_Data        <- Washington_CM_Co_Data
#
# # have to compensate for NA in the saID list (no registry)
# isNAsa         <- is.na(WA_Data$saID)
# sL             <- !isNAsa & (WA_Data$saID    == "WA-SEA")
# # counties with saID set and == "WA-SEA"
#
# nSL            <- isNAsa | (WA_Data$saID    != "WA-SEA")
# # counties with saID not set (NA) or != "WA-SEA"
#
#
# WA_Data_Seat   <- WA_Data[sL,]
# WA_Data_NotSeat<- WA_Data[nSL,]
#
# # pull out the data for the Washingto-Puget Registry.
# lWA            <- dim(WA_Data_Seat)[1]        # get number of counties
# selectedWA     <- (runif(lWA) <= 0.7 )       # select random 80% of CO in Puget area.
# WA_Data_Seat_P <- WA_Data_Seat[selectedWA,]
#
# lWA            <- dim(WA_Data_NotSeat)[1]
# selectedNotWA  <- (runif(lWA) <= 0.3 )
# WA_Data_NotSeat_P<- WA_Data_NotSeat[selectedNotWA,]
#
```

```
# WA_P_Data      <- WA_Data_Seat_P
# WA_P_Data      <- rbind(WA_P_Data,WA_Data_NotSeat_P)
# str(WA_P_Data)
# #
# ####
#
# #####
# #
# #   Example c70 - WA Partial Counties - one Registry (WA-SEAT)
# #
#
# TT   <- c("Ex-c70 WA-Seat Partial Reg plus Partial Co",
#           "def basic:countyB=DATA, seerB=NONE, stateB=NONE")
#
#
# pdf("SM-Ex-c70 WA-Seat Reg Partial Co default basic.pdf", width=8,height=10.5)
# SeerMapper(WA_P_Data)
# dev.off()
#
#
# ####
# #
# #  Example - c71 - Washington State - Partial Co. - one Registry.
# #
# #  To make the mapping of partial counties in Washington State
# #  which has one Registry in part of the state, the following
# #  enhancement can be made for clarity: draw the outline of the
# #  states with data (Washington) via stateB="DATA",  draw all of
# #  county boundaries within the Registry with data (WA-SEA),
# #  increase the number of categories from the default of 5 to 7 (categ=7),
# #  add a two (2) line title (mTitle=TT), use column names to locate
# #  the location ID and data in the data data.frame (idCol="FIPS" and
# #  dataCol="AA_Rate", and add hatching of counties with a pValue > 0.05
# #  hatch=list(dataCol="pValue").
# #
#
# TT   <- c("Ex-c71 WA-Seat Partial Reg plus Partial Co",
#           "cat-7, hatching, countyB=SEER, seerB=NONE, stateB=DATA")
#
#
# pdf("SM-Ex-c71 WA-Seat Reg Partial Co enhd hatch, cat-7, coB=SR, stB-ST.pdf",
#     width=8,height=10.5)
#
# SeerMapper(WA_P_Data,
#            idCol = "FIPS", dataCol="AA_Rate", # use column names
#            categ = 7,                         # increase from 5 to 7 categories
#            stateB="DATA",                     # provided state outline
#          countyB="SEER",                   # draw all county boundaries up to Registry
#         hatch = list(dataCol="pValue"),     # use column name for pValue and do hatching.
#            mTitle=TT                          # add title (2 lines)
# )
#
# dev.off()
```

```
#
# #
# ####
#
# ####
# #
# #  Example - c73 - Washington State - Partial Co. - one Registry.
# #
# #  A different variation on inproving example c70 is to
# #  draw all of the counties within the state with data (countyB="STATE"),
# #  the outline of the state would only be needed is further accent is required
# #  (stateB="DATA", move the legend from the left to right side and include the
# #  counts in each category mLegend=list(pos="right",counts=TRUE), and
# #  since the categories calculated are:
# #    [152.69-157.16], (156.16-164.00], (164.00-168.76], (168.76-172.91],
# #    (172.91-174.93], (174.93-179.76], and (179.76-194.80]
# #  we can manually set a reasonable set of breakpoints with
# #  categ=c(157.5, 164.0, 168.75, 173, 175, 180), also 7 categories.
# #
# #
#
# TT   <- c("Ex-c73 WA-Seat Partial Reg plus Partial Co",
#           "countyB=STATE, seerB=NONE, stateB=NONE brkpointlist")
#
# pdf("SM-Ex-c73 WA-Seat Reg Partial Co coB-ST srB-N stB-N brkpt.pdf",
#     width=8,height=10.5)
#
# SeerMapper(WA_P_Data,
#            idCol   = "FIPS",dataCol="AA_Rate",
#            countyB = "STATE",
#            categ   = c(157.5, 164, 168.75, 173, 175, 180),
#            mLegend = list(pos="right",counts=TRUE),
#            brkPtDigits = 2,
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# #####
#
#
# #
# #   Example # c72 shows the default settings with countyB="DATA" for atlanta all tracts.
# #    Example # c73 partial data in Washington State - countyB="SEER"
# #    Example # c74 partial data in Washington State - countyB="STATE"
# #
#
# #####
# #
# #  Example # c72 Washington Partial Counties
# #         countyB=DATA, seerB=NONE, stateB=NONE (def)
# #
```

```
# TT <- c("Ex-c72-Washington Partial Counties","Rates - defaults")
#
# pdf("SM-Ex-c72-WA-P-Co-Rate-def.pdf", width=7.5, height=10)
#
# SeerMapper(WA_P_Data,
#            idCol="FIPS",dataCol="AA_Rate",
#            mTitle=TT
# )
#
# dev.off()
# #
# ####
#
# #####
# #
# #  Example # c73 Washington Partial Counties
# #         countyB=SEER, seerB=NONE, stateB=NONE (def)
# #
# TT <- c("Ex-c73-Washington Partial Counties","Rates - countyB=SEER")
#
# pdf("SM-Ex-c73-WA-P-Co=Rate-coB-sr.pdf", width=7.5, height=10)
#
# SeerMapper(WA_P_Data,
#            idCol="FIPS",dataCol="AA_Rate",
#            countyB="SEER",
#            mTitle=TT
# )
#
# dev.off()
# #
# ####
#
# #####
# #
# #  Example # c74 Washington Partial Counties
# #         countyB=STATE, seerB=NONE, stateB=NONE (def)
# #
# TT <- c("Ex-c74-Washington Partial Counties"," Rate Data - countyB=STATE")
#
# pdf("SM-Ex-c74-WA-P-Co-Rate-coB-st.pdf", width=7.5, height=10)
#
# SeerMapper(WA_P_Data,
#            idCol="FIPS",dataCol="AA_Rate",
#            countyB="STATE",
#            mTitle=TT
# )
#
# dev.off()
# #
# ####
#
#
# ####
```

```
# #
# #  2000 census County Demographic data for Washington State
# #
# data(WA_Dem_Co_Data,envir=environment())
#
# WA_D_Data        <- WA_Dem_Co_Data
#
# # have to compensate for NA in the saID list (no registry)
# isNAsa           <- is.na(WA_D_Data$saID)
# sL               <- !isNAsa & (WA_D_Data$saID == "WA-SEA")
# # counties with saID set and == "WA-SEA"
#
# nSL              <- isNAsa | (WA_D_Data$saID  != "WA-SEA")
# # counties with saID not set (NA) or != "WA-SEA"
#
# WA_D_Data_Seat   <- WA_D_Data[sL,]
# WA_D_Data_NotSeat<- WA_D_Data[nSL,]
#
# # pull out the data for the Washington-Puget Registry.
# lWA              <- dim(WA_D_Data_Seat)[1]     # get number of counties
# selectedWA       <- (runif(lWA) <= 0.6 )       # select random 80% of CO in Puget area.
# WA_D_Data_Seat_P <- WA_D_Data_Seat[selectedWA,]
#
# lWA              <- dim(WA_D_Data_NotSeat)[1]
# selectedNotWA    <- (runif(lWA) <= 0.2 )
# WA_D_Data_NotSeat_P<- WA_D_Data_NotSeat[selectedNotWA,]
#
# WA_D_P_Data      <- WA_D_Data_Seat_P
# WA_D_P_Data      <- rbind(WA_D_P_Data,WA_D_Data_NotSeat_P)
# str(WA_D_P_Data)
#
# #
# #   Example # c76 shows the default settings with countyB="DATA" for atlanta all tracts.
# #    Example # c77 partial data in Washington State - countyB="SEER"
# #    Example # c78 partial data in Washington State - countyB="STATE"
# #
#
# #####
# #
# #  Example # c76 Washington Partial Counties
# #         countyB=DATA, seerB=NONE, stateB=NONE (def)
# #
# TT <- c("Ex-c76-Washington Partial County-Dem","defaults")
#
# pdf("SM-Ex-c76-WA-Dem-P-Co-def.pdf", width=7.5, height=10)
#
# SeerMapper(WA_D_P_Data,
#            idCol="FIPS",dataCol="popdens",
#            mTitle=TT
# )
#
# dev.off()
# #
```

```
# ####
#
# #####
# #
# #  Example # c78 Washington Partial Counties
# #         countyB=SEER, seerB=NONE, stateB=NONE (def)
# #
# TT <- c("Ex-c78-Washington Partial Dem County","countyB=SEER")
#
# pdf("SM-Ex-c78-WA-Dem-P-Co-coB-sr.pdf", width=7.5, height=10)
#
# SeerMapper(WA_D_P_Data,
#            idCol="FIPS",dataCol="popdens",
#            countyB="SEER",
#            mTitle=TT
# )
#
# dev.off()
# #
# ####
#
# #####
# #
# #  Example # c79 Washington Partial Counties
# #         countyB=STATE, seerB=NONE, stateB=NONE (def)
# #
# TT <- c("Ex-c79-Washington Partial Dem Counties","countyB=STATE")
#
# pdf("SM-Ex-c79-WA-Dem-P-Co-trB-st.pdf", width=7.5, height=10)
#
# SeerMapper(WA_D_P_Data,
#            idCol="FIPS",dataCol="popdens",
#            countyB="STATE",
#            mTitle=TT
# )
#
# dev.off()
# #
# ####
#
#
#
# ####
# #
# #  Have data at the census tract level works exactly the same as data
# #  at the county level.  The only exception is supplemental boundary
# #  information datasets may be needed.
# #
#
# #
# #  Example - c80 - Wash-Balt CSA county level - defaults
# #
#
```

```
#
# data(WashBaltMetro_Co_Data)
#
# TT <- c("SM-Ex-c80-Washington-Baltimore Metro","County-Combined Statistics Area-def")
#
# pdf("SM-Ex-c80-WashBalt-County-CSA-def.pdf", width=7.5, height=10)
#
# SeerMapper(WashBaltMetro_Co_Data,
#            idCol="FIPS",dataCol="popdens",
#            categ=7,
#            mTitle=TT
# )
#
# dev.off()
#
# #
# #  Example - c81 - Wash-Balt CSA County level - stateB="DATA"
# #
#
#
# TT <- c("SM-Ex-c81-Washington-Baltimore Metro","County-Combined Statistics Area-stB=D")
#
# pdf("SM-Ex-c81-WashBalt-County-CSA-stB-D.pdf", width=7.5, height=10)
#
# SeerMapper(WashBaltMetro_Co_Data,
#            idCol="FIPS",dataCol="popdens",
#            categ=7,
#            stateB="DATA",
#            mTitle=TT
# )
#
# dev.off()
# #
# ####
#
#
# ####
# #
# #  Example - c83 - Kansas City CSA county level - defaults
# #
#
#
# data(KCMetro_Co_Data)
#
# TT <- c("SM-Ex-c83-Kansas City Metro","County-Combined Statistics Area-def")
#
# pdf("SM-Ex-c83-KCMetro-County-CSA-def.pdf", width=7.5, height=10)
#
# SeerMapper(KCMetro_Co_Data,
#            idCol="FIPS",dataCol="popdens",
#            categ=7,
#            mTitle = TT
# )
```

```
#
# dev.off()
#
# #
# #  Example - c84 - Kansas City CSA County level - stateB="DATA"
# #
#
#
# TT <- c("SM-Ex-c84-Kansas City Metro","County-Combined Statistics Area-stB=D")
#
# pdf("SM-Ex-c84-KCMetro-County-CSA-stB-D.pdf", width=7.5, height=10)
#
# SeerMapper(KCMetro_Co_Data,
#            idCol="FIPS",dataCol="popdens",
#            categ=7,
#            stateB="DATA",
#            mTitle = TT
# )
#
# dev.off()
# #
# ####
#
#
#
# #
# #  End of County Examples
# #
#
# ########################
#
# #####
# #
# #  Example tr60 - Georgia Census Tract Data-Population Density with defaults
# #
# #  Uses 2000 census demographic tract data from the GA_Dem_Tr_Data dataset.
# #
#
# data(GA_Dem_Tr_Data)
# GA_Tr_DF   <- GA_Dem_Tr_Data
#
# #
#
# TT   <- c("Ex-tr60 Georgia Census Tracts Population Density, c=7",
#           "def: tractB=DATA countyB=NONE seerB=NONE stateB=NONE")
#
# pdf("SM-Ex-tr60 GA Tracts-PopDens-c=7 def trB-D coB-N srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_DF,
#            idCol   = "FIPS",dataCol="popdens",
#            categ   = 7,
#            mTitle  = TT
```

```
# )
#
# dev.off()
#
# #
# #####
#
# #####
# #
# #  Example tr61 - Georgia Census Tract Data for age 65 and up
# #
# #  Uses demographic tract data.frame (GA_Dem_Tr_Data) loaded above.
# #
#
# TT   <- c("Ex-tr61 Georgia Census Tract for age 65 and up.",
#           "def-tractB=DATA, countyB=NONE, seerB=NONE, stateB=NONE")
#
# pdf("SM-Ex-tr61 GA Tracts-age 65 and up-def trB-D coB-N srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_DF,
#            idCol   = "FIPS",dataCol="age.65.up",
#            categ   = 7,
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# #####
#
# #####
# #
# #  Example tr63 - Georgia Census Tract Data for Percentage age 65 and up
# #
# #  The Georgia demographic census tract dataset (GA_Dem_Tr_Data)
# #  is used in this example.
# #  The percentage (0% to 100%) of individuals in each tract that is
# #  65 year old or older is calculated and mapped.
# #
#
# GA_Tr_DF$PC.age.65.up <- ( GA_Tr_DF$age.65.up / GA_Tr_DF$pop2000 ) * 100
#
# TT   <- c("Ex-tr63 Georgia Tract for PC 65 and up",
#           "def: tractB=DATA countyB=NONE seerB=NONE stateB=NONE")
#
# pdf("SM-Ex-tr63 GA Tracts-PC age 65 and up-def.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_DF,
#            idCol   = "FIPS",dataCol="PC.age.65.up",
#            categ   = 7,
#            mTitle  = TT
```

```
# )
#
# dev.off()
#
# #
# #####
#
# #####
# #
# #  Example tr64 - Georgia Census Tract Data for Percentage age 65 and up
# #
# #  The Georgia demographic census tract dataset (GA_Dem_Tr_Data)
# #  is used in this example. Same as tr63, but tractB=NONE, and countyB=DATA to
# #  turn off the drawing of the tract boundaries, but make sure
# #  county boundaries are drawn around areas with data.
# #
# #  The value categorized in the map is the percentage (0% to 100%) of
# #  individuals in each tract that are 65 year old or older.
# #
#
# GA_Tr_DF$PC.age.65.up <- ( GA_Tr_DF$age.65.up / GA_Tr_DF$pop2000 ) * 100
#
# TT   <- c("Ex-tr64 Georgia Tract for PC 65 and up",
#           "tractB=NONE countyB=DATA seerB=NONE stateB=NONE")
#
# pdf("SM-Ex-tr64 GA Tracts-PC age 65 and up-tB-N cB-D.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_DF,
#            idCol   = "FIPS",dataCol="PC.age.65.up",
#            categ   = 7,
#            tractB  = "NONE",
#            countyB = "DATA",
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# #####
#
# #####
# #
# #  Example tr65 - Georgia Census Tract Data for Percentage households occupied
# #
# #  Using the Georgia demographic census tract dataset (GA_Dem_Tr_Data),
# #  the percentage (0% to 100%) of the households that are occupied
# #  in each county is calculated and mapped.
# #
#
# GA_Tr_DF$PC.hh.occupied <- ( GA_Tr_DF$hh.occupied / GA_Tr_DF$hh.units ) * 100
#
# TT   <- c("Ex-tr65 GA Tract for PC HH Occupied",
```

```
#           "def: tractB=DATA countyB=NONE seerB=NONE stateB=NONE")
#
# pdf("SM-Ex-tr65 GA Tracts-PC HH Occupied-trB-D coB-N srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_DF,
#            idCol   = "FIPS",dataCol="PC.hh.occupied",
#            categ   = 7,
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# ####
#
# #####
# #
# #  Example tr66 - Georgia Census Tract Data for Percentage Household Renters
# #
# #  Using the Georgia demographic census tract dataset (GA_Dem_Tr_Data),
# #  the percentage (0% to 100%) of the households that have renters
# #  in each county is calculated and mapped.
# #
#
# GA_Tr_DF$PC.hh.renter <- (1-( GA_Tr_DF$hh.owner / GA_Tr_DF$hh.units )) * 100
#
# TT   <- c("Ex-tr66 GA Tract for PC Renters",
#           "tractB=DATA countyB=NONE seerB=NONE stateB=NONE")
#
# pdf("SM-Ex-tr66 GA Tracts-PC Renters-trB-D coB-N srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_DF,
#            idCol   = "FIPS",dataCol="PC.hh.renter",
#            categ   = 7,
#            mTitle  = TT
# )
# dev.off()
#
# #
# #####
#
#
# ####
# #
# #  To do the same Washington Baltimore Metro area a the census tract level,
# #  the data is collected at the census tract level and filtered
# #  to just the CSA tracts.
# #
# #  To cover all of the conditions in the next two examples:
# #
#
```

```
# #
# #  With the \var{SeerMapperEast} and \var{SeerMapperWest} supplemental
# #  packages loaded, maps can be created for census tracts in any state or
# #  district.
# #
#
# ####
# #
# #  County Cancer Mortality data for Washington State
# #
# data(WA_Dem_Tr_Data,envir=environment())
#
# WA_D_Data        <- WA_Dem_Tr_Data
#
#
# # have to compensate for NA in the saID list (no registry)
# isNAsa           <- is.na(WA_D_Data$saID)
# sL               <- !isNAsa & (WA_D_Data$saID    == "WA-SEA")
# # counties with saID set and == "WA-SEA"
#
# nSL              <- isNAsa | (WA_D_Data$saID    != "WA-SEA")
# # counties with saID not set (NA) or != "WA-SEA"
#
# WA_D_Data_Seat   <- WA_D_Data[sL,]
# WA_D_Data_NotSeat<- WA_D_Data[nSL,]
#
# # pull out the data for the Washingto-Puget Registry.
# lWA            <- dim(WA_D_Data_Seat)[1]        # get number of counties
# selectedWA     <- (runif(lWA) <= 0.6 )       # select random 80% of CO in Puget area.
# WA_D_Data_Seat_P <- WA_D_Data_Seat[selectedWA,]
#
# lWA            <- dim(WA_D_Data_NotSeat)[1]
# selectedNotWA  <- (runif(lWA) <= 0.2 )
# WA_D_Data_NotSeat_P<- WA_D_Data_NotSeat[selectedNotWA,]
#
# WA_D_P_Data      <- WA_D_Data_Seat_P
# WA_D_P_Data      <- rbind(WA_D_P_Data,WA_D_Data_NotSeat_P)
# str(WA_D_P_Data)
#
# #
# #   Example # tr76 shows the default settings with tractB="DATA" for atlanta all tracts.
# #    Example # tr77 partial data in Washington State - tractB="COUNTY"
# #    Example # tr78 partial data in Washington State - tractB="SEER"
# #    Example # tr79 partial data in Washington State - tractB="STATE"
# #
#
# #####
# #
# #  Example # tr76 Washington Partial Tracts
# #         tractB=DATA, countyB=NONE seerB=NONE, stateB=NONE (def)
# #
# TT <- c("Ex-tr76-Washington Partial Tracts-Demog","defaults")
#
```

```
# pdf("SM-Ex-tr76-WA-Dem-P-Tracts-def.pdf", width=7.5, height=10)
#
# SeerMapper(WA_D_P_Data,
#            idCol="FIPS",dataCol="popdens",
#            mTitle=TT
# )
#
# dev.off()
# #
# ####
#
# #####
# #
# #  Example # tr77 Washington Partial Tracts
# #         tractB=COUNTY, countyB=NONE, seerB=NONE, stateB=NONE (def)
# #
# TT <- c("Ex-tr77-Washington Partial Tracts","tractB=COUNTY")
#
# pdf("SM-Ex-tr77-WA-Dem-P-Tract-trB-co.pdf", width=7.5, height=10)
#
# SeerMapper(WA_D_P_Data,
#            idCol="FIPS",dataCol="popdens",
#            tractB="COUNTY",
#            mTitle=TT
# )
#
# dev.off()
# #
# ####
#
# #####
# #
# #  Example # tr78 Washington Partial Tracts
# #         tractB=SEER, seerB=NONE, stateB=NONE (def)
# #
# TT <- c("Ex-tr78 Washington Partial Tracts","tractB=SEER")
#
# pdf("SM-Ex-tr78-WA-Dem-P-Tract-trB-sr.pdf", width=7.5, height=10)
#
# SeerMapper(WA_D_P_Data,
#            idCol="FIPS",dataCol="popdens",
#            tractB = "SEER",
#            mTitle=TT
# )
#
# dev.off()
# #
# ####
#
#
# #####
# #
# #  Example # tr79 Washington Partial Tracts
```

```
# #         tractB=STATE, seerB=NONE, stateB=NONE (def)
# #
# TT <- c("Ex-tr79-Washington Partial Tracts","tractB=STATE")
#
# pdf("SM-Ex-tr79-WA-Dem-P-Tract-trB-st.pdf", width=7.5, height=10)
#
# SeerMapper(WA_D_P_Data,
#            idCol="FIPS",dataCol="popdens",
#            countyB="STATE",
#            mTitle=TT
# )
#
# dev.off()
# #
# ####
#
# ####
# #
# #  Example - tr83 - Kansas City CSA tract level - defaults
# #
#
# data(KCMetro_Tr_Data)
#
# TT <- c("SM-Ex-tr83-Kansas City Metro","Tract-Combined Statistics Area-def")
#
# pdf("SM-Ex-tr83-KCMetro-Tract-CSA-def.pdf", width=7.5, height=10)
#
# SeerMapper(KCMetro_Tr_Data,
#            idCol="FIPS",dataCol="popdens",
#            categ=7,
#            mTitle=TT
# )
#
# dev.off()
#
# #
# #  Example - tr84 -Kanssa City CSA tract level - stateB="DATA"
# #
#
#
# TT <- c("SM-Ex-tr84-Kansas City Metro","Tract-Combined Statistics Area-stB=D")
#
# pdf("SM-Ex-tr84-KCMetro-CSA-stB-D.pdf", width=7.5, height=10)
#
# SeerMapper(KCMetro_Tr_Data,
#            idCol="FIPS",dataCol="popdens",
#            categ=7,
#            stateB="DATA",
#            mTitle=TT
# )
#
# dev.off()
# #
```

```
# ####
#
#
# #####
# #
# #  Data.frame for Georgia Tracts ALL, Partial Atlanta Registry  2000 census
# #
#
# data(GA_Dem_Tr_Data,envir=environment())
#
# GA_Tr_DF         <- GA_Dem_Tr_Data
#
# lGA              <- dim(GA_Tr_DF)[1]
# selectedTr       <- (runif(lGA) <= 0.75)
# GA_Tr_P          <- GA_Tr_DF[selectedTr,]
# # select a random part (75%) of Georgia Tracts.
#
# GA_Tr_ATL_DF     <- GA_Tr_DF[GA_Tr_DF$saID == "GA-ATL",]
# # select Atlanta Reg. Tracts
#
# lATL             <- dim(GA_Tr_ATL_DF)[1]
# GA_Tr_ATL_P      <- GA_Tr_ATL_DF[(runif(lATL) <= 0.75),]
#
# GA_Tr_ATLRUR_DF  <- GA_Tr_DF[(GA_Tr_DF$saID == "GA-ATL" | GA_Tr_DF$saID == "GA-RUR"),]
# # select Atlanta & Rural Regs
#
# lATLRUR          <- dim(GA_Tr_ATLRUR_DF)[1]
# GA_Tr_ATLRUR_P   <- GA_Tr_ATLRUR_DF[(runif(lATLRUR) <= 0.75),]
# # select a random set of tracts
# #
# #####
#
# #####
# #
# #  The use of census tract location IDs and data works identically
# #  to the county mapping shown above.  The only difference is th e
# #  extra county boundary layer.  The tractB call parameter
# #  allows tract boundaries to be drawn at the data level or up to the
# #  county, Seer Registry, or State levels.
# #
#
# #
# #  Example # tr90 - Georgia ALL Tracts Map with defaults
# #    demonstrate the defaults setting - map tracts with data,
# #    tractB="DATA", countyB="NONE", seerB="NONE", stateB="NONE".
# #
# #    Demonstrates simple census tract mapping with 7 categories and
# #    the default boundary controls.
# #
#
# TT   <- c("Ex-tr90 Georgia All Tracts Map-2000 popDensity",
#           "Defaults")
#
```

```
# pdf("SM-Ex-tr90 GA All Tracts Map-Def trB-D coB-N srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_DF,
#            idCol   = "FIPS",dataCol="popdens",
#            mTitle  = TT
# )
#
# dev.off()
#
# #
# ####
#
# #####
# #
# #     This example maps the tracts assocated with the Georgia Atlanta Seer Registry.
# #     The defaults of tractB="DATA", countyB="NONE", sserB="NONE", and stateB="NONE"
# #     are used.
# #
# #     Alternate boundary controls would have the following affects:
# #      tractB="SEER" has little effect, since all of the tracts within the registry are
# #          present.
# #       countyB="DATA". countyB="SEER" and seerB="DATA" would only affect the map
# #          by drawing heavier bounaries at each level.  Remember registry boundaries
# #          are on county boundaries.
# #
# #     The drawing of the tract boundaries can be expanded by:
# #       tractB="COUNTY" will draw all of the tracts within the counties
# #          with tracts with data.  Any tract or county in the Seer Registry or State
# #          will not be drawn or colors.  You will not see any shared boundaries.
# #     tractB="STATE" will draw all of the tracts within the states with tracts with data.
# #         All tracts within a state with a tract with data will be drawn.  Nothing will
# #          be missed.
# #
#
# ####
# #
# #  Example tr91 - Georgia Tract - Altanta Registry
# #
# #     When partial tract data is present, use the settings that will present the
# #     data and it's relationship to the surroundings and nothing more.  Don't use
# #     stateB="ALL" or seerB="ALL.  For small areas of data, these are very useful.
# #
# TT   <- c("Ex-tr91 GA ATL Reg All Tracts",
#           "default:tractB=DATA, countyB=NONE, seerB=NONE, stateB=NONE")
#
# pdf("SM-Ex-tr91 GA ATL Reg Tracts def-trB-D coB-N Sr-N St-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_ATL_DF,
#            idCol   ="FIPS",dataCol="popdens",
#            mTitle  =TT
# )
```

```
#
# dev.off()
#
# #
# ####
#
#
# ####
# #
# #  Example tr92 - Georgia Tract - Altanta Registry
# #
# #     When partial tract data is present, use the settings that will present the
# #     data and it's relationship to the surroundings and nothing more.  Don't use
# #     stateB="ALL" or seerB="ALL.  For small areas of data, these are very useful.
# #
# TT   <- c("Ex-tr92 GA ATL Reg All Tracts",
#           "def:tractB=SEER, countyB=NONE, seerB=NONE, stateB=NONE")
#
# pdf("SM-Ex-tr92 GA ATL Reg Tracts def-trB-sr coB-N Sr-N St-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_ATL_DF,
#            idCol   ="FIPS",dataCol="popdens",
#            tractB  ="SEER",
#            mTitle  =TT
# )
#
# dev.off()
#
# #
# ####
#
#
# ####
# #
# #  Example tr93 - Georgia Tract - Altanta Registry
# #
# #     When partial tract data is present, use the settings that will present the
# #     data and it's relationship to the surroundings and nothing more.  Don't use
# #     stateB="ALL" or seerB="ALL.  For small areas of data, these are very useful.
# #
# TT   <- c("Ex-tr93 GA ATL Reg All Tracts",
#           "def:tractB=STATE, countyB=NONE, seerB=NONE, stateB=NONE")
#
# pdf("SM-Ex-tr93 GA ATL Reg Tracts def-trB-st coB-N Sr-N St-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_ATL_DF,
#            idCol   ="FIPS",dataCol="popdens",
#            tractB  ="STATE",
#            mTitle  =TT
# )
#
```

```
# dev.off()
#
# #
# ####
#
# ####
# #
# #  Example tr94 - Georgia Tract - Altanta Registry
# #
# #     When partial tract data is present, use the settings that will present the
# #     data and it's relationship to the surroundings and nothing more.  Don't use
# #     stateB="ALL" or seerB="ALL.  For small areas of data, these are very useful.
# #
#
# TT   <- c("Ex-tr94 GA ATL Reg Partial Tracts-Def",
#           "tractB=DATA, countyB=NONE, seerB=NONE, stateB=NONE")
#
# pdf("SM-Ex-tr94 GA ATL Partial Tracts trB-D coB-N srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_Tr_ATL_DF,
#            idCol   ="FIPS",dataCol="popdens",
#            categ   =7,
#            tractB  ="STATE",
#            mTitle  =TT
# )
#
# dev.off()
#
# #
# ####
#
# #####
# #
# #  Example H02 & H04 - Hatching of non-pValue data, line density,
# #       and color
# #
# #  Using the Georgia demographic county dataset (GA_Dem_Co_Data)
# #   population density to demonstrate:
# #      Hatching of non-pValue type data (popdens), operation set
# #      to "<", value set to mean of all population densities, color
# #      changed to blue, palette changed reds, request 6 categories.
# #
# #
#
# data(GA_Dem_Co_Data, envir=environment())
# GA_D_Co_Data  <- GA_Dem_Co_Data
#
# meanPopDens     <- mean(GA_D_Co_Data$popdens)
# cat("meanPopDens:",meanPopDens,"\n")
#
# TT   <- c("Ex-H02 Georgia Co w hatching of > mean popdens", "")
#
```

```
# pdf("SM-Ex-H02 GA Co hatch-ge mean popdens.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_D_Co_Data,
#            idCol     = "FIPS",dataCol="popdens",
#            categ     = 6,               # use 6 categories
#            hatch     = list(col="black",
#                             dataCol="popdens",ops=">", value=meanPopDens),
#            palColors = "RdYlGn",
#            mTitle    = TT
# )
#
# dev.off()
#
# #
# #####
#
# #####
# #
# #  Example H04 - Hatching of non-pValue data, line density,
# #       and color
# #
#
# meanPopDens     <- mean(GA_D_Co_Data$popdens)
# cat("meanPopDens:",meanPopDens,"\n")
#
# TT   <- c("Ex-H04 Georgia Co w hatching of > mean popdens",
#           "den=15, angle=60")
#
# pdf("SM-Ex-H04 GA Co hatch-ge mean popdens-d15 a60.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_D_Co_Data,
#            idCol     = "FIPS",dataCol="popdens",
#            categ     = 6,               # use 6 categories
#            hatch     = list(col="red", den=15, angle=60,
#                             dataCol="popdens",ops=">",value=meanPopDens),
#            palColors = "BuGn",
#            mTitle    = TT
# )
#
# dev.off()
#
# #
# #####
#
# #####
# #
# #  The next set of examples look at options to change how the maps will look.
# #
# #  Example P01 - Georgia County Data for Percentage Household Renters
# #
# #  In the extreme you can assign your own colors to each sub-area.
```

```
# #  In this example, I have randomly assigned a color from a
# #  RColorBrewer "Accent" palette to each sub-areas.  The color
# #  is placed in the "dataCol" for the sub-area and categ is set to "COLORS".
# #
# #  Options:  - Use colors instead of data
# #            - turning off the legend.
# #
# library(RColorBrewer)
#
# data(GA_Dem_Co_Data,envir=environment())
# GA_D_Co_Data <- GA_Dem_Co_Data
#
# lGA_D_Co    <- dim(GA_D_Co_Data)[1]
# numColrs  <- 8
# ColorSet  <- brewer.pal(numColrs,"Accent")
#
# numRep      <- (lGA_D_Co/numColrs)+1
# # place colors into the data.frame.
# GA_D_Co_Data$Col <- replicate(numRep, ColorSet)[1:lGA_D_Co]
#
# TT   <- c("Ex-p01 Georgia All County-Random Accent Colors Test",
#           "def: countyB=DATA seerB=NONE stateB=NONE")
#
# pdf("SM-Ex-p01 GA Co-Random-Colors-coB-D srB-N stB-N.pdf",
#     width=8,height=10.5)
#
# SeerMapper(GA_D_Co_Data,
#            idCol   ="FIPS",dataCol="Col",  # new data column name
#            categ   ="colors",            # specify how to do the categorization
#            mLegend = FALSE,              # turn off the legend.
#            mTitle  =TT
# )
#
# dev.off()
#
# #
# ####
#
#
#
# #
# #  end of examples
# #
# #####

  #
  # Reference: Mapping it out, Mark Monnier - benefit of characterization
  #    and simplification for a more visible map.
  #
  #    Still need simplification and maintain same area and
  #    relationships/neigborships.
  #
  #    Purpose of map is to illustrate spatial patterns of the mapped
```

```
#    variable.  It is important that each area be visible and that
#    the spatial relationship be maintain.  It is not important the
#    area's shape if maintained.
#
#    Centroids and Convex Hulls...  Also population centroid.
```

`SeerMapper.Version` *Display SeerMapper Package Version and Build Date*

Description

Provides a means of retreiving SeerMapper's or Seer2010Mapper's version number and build date and time.

Usage

```
SeerMapper.Version()
```

Details

The *SeerMapper* package provide a simple means of mapping data for the U. S. at several levels: State, State/County, State/County/Census Tract, and U. S. Seer Registry areas. The package automatically determines the mapping level based on the data and the area identifiers provide at the time of the call. The data is categorised and each area is drawn and colored appropriately based on its data value.

The SeerMapper.Version function return the version number and build date and time for package.

Value

The return value is "SeerMapper Version X.X.X YY/MM/DD HH:MM Xm"

Author(s)

James B Pearson and Linda Pickle, StatNet Consulting, LLC, Gaithersburg, MD

SeerMapper2010 *Quick Data Mapper at State, County, Census Tract or Seer Registry levels*

Description

Provides a call to the *SeerMapper* main function in the package to create maps of data using the U. S. 2010 census boundary identifiers and data. While the *SeerMapper* function can map against either the 2000 or 2010 census boundary data by using the *censusYear* call parameter, the *SeerMapper2010* and *SeerMapper2000* function are provided minimize the call parameters required and also simplify the overall process. All of the *SeerMapper* call parameters can be specified and are passed along to *SeerMapper*.

Usage

```
SeerMapper2010(...)
```

Arguments

...	any of the *SeerMapper* call parameters (see section on the *SeerMapper* function for details). All parameters are passed through to *SeerMapper*.

Details

For details on the supported call parameters, see the *SeerMapper* section of this doccument.

Author(s)

James B Pearson and Linda Pickle, StatNet Consulting, LLC, Gaithersburg, MD

SeerRegs_CM_Data *Seer Registry Cancer Mortality Rates for all sites for 2009 to 2013*

Description

These datasets containing the cancer mortality rates for 17 of the 20 U. S. Seer Registries. The cancer mortality data was extracted from the Seer*Stat website at the Seer Registry level over a 5 year period from 2009 to 2013 for all cancer sites on September 30, 2016.

Usage

```
data(SeerRegs_CM_Data)
```

Format

All rates are 5 year age adjusted rates for the period 2009 to 2013. The dataset contains 10 columns:

Registry is a character string name for the Seer Registry.

saID is a character string abbreviation for the Seer Registry. See below for a list of supported abbreviations.

All_Both is a numeric value for the age adjusted rate for all races for both sexes.

All_Males is a numeric value for the age adjusted rate for all races for males.

All_Females is a numeric value for the age adjusted rate for all races for females.

Whites_Both is a numeric value for the age adjusted rate for whites and both sezes.

Whites_Males is a numeric value for the age adjusted rate for white males.

Whites_Females is a numeric value for the age adjusted rate for white females.

Blacks_Both is a numeric value for the age adjusted rate for blacks and both sezes.

Blacks_Males is a numeric value for the age adjusted rate for black males.

Blacks_Females is a numeric value for the age adjusted rate for black females.

The first two rows of the data.frame represent the rate information for the 17 of the 20 U. S. Seer Registries and the USA. These rows MUST be deleted before using the data.frame with the package.

The Seer Registry abbreviations, state fips ID and names used in the package are:

saID	stID	Alias Match	Name
AK-NAT	02	Alaska	Alaska Natives
AZ-NAT	04	Arizona	Arizona Natives
CA-OTH	06	Greater Calif	CA Other (not SF-Oakland, SJ-Monterey, or LA)
CA-SF	06	San Fran	California San Francisco/Oakland
CA-SJ	06	San Jose	California San Jose/Monterey
CA-LA	06	Los Angeles	California Los Angeles Co
CT	09	Connecticut	Connecticut
GA-OTH	13	Greater Geor	Georgia Other (not Atlanta or Rural areas)
GA-ATL	13	Atlanta	Georgia Atlanta Metro
GA-RUR	13	Rural Geor	Rural Georgia
HI	15	Hawaii	Hawaii
IA	19	Iowa	Iowa
KY	21	Kentucky	Kentucky
LA	22	Louisiana	Louisiana
MI-DET	26	Detroit	Michigan Detroit
NJ	34	Jersey	New Jersery
NM	35	New Mexico	New Mexico
OK-CHE	40	Cherokee	Oklahoma Cherokee Nation
UT	49	Utah	Utah
WA-SEA	53	Puget Sound	Washington Seattle-Puget Sound

Data is not available for the three native american Seer Registries: Alaska (AK-NAT), Arizona (AZ-NAT), and a set of counties in Oklahoma (OK-CHE). These Seer Registries are included for

completeness. The alias match strings allows the package to attempt to match location identifiers generated by the Seer*Stat website instead of having the caller re-label all of the records and possibly made an error. The recommendation is to point package at the column and see what happens.

Source

The U. S. Seer Registry age adjusted Cancer Mortality rates for the years 2009 to 2013 for all races, whites, and blacks for both sexes, males and females for all sites was extracted from the NCI Seer*Stat website on September 30, 2016. The data selected is at the Seer Registry geographic level for a 5 year period from 2009 to 2013 for cancer mortality for all sites to allow it to be used as example data in the package.

SM_XXXXX *Internal Functions to utilize boundary data and create maps.*

Description

Provides support for *SeerMapper* and other packages that can use the *SeerMapper* core functions and boundary datasets to map state, registry, county or tract sub-areas in the U. S. Based on the location IDs provided loads into memory the U. S. state and registry boundary data, determines the sub-area level being mapped, then loads the county and (if needed) census tract boundary data for any state containing a sub-area with data and maps the data and requested boundaries.

Usage

```
SM_Mapper(rPM,MV)
```

Arguments

rPM	This a named list of *SeerMapper* variables and call parameter values used by the internal functions of *SeerMapper*.
MV	This a named list of *SeerMapper* boundary data and location IDs to support the requested map.

Details

Three functions are exported by *SeerMapper* for use by other packages offered by this author. These functions are internal functions and not designed for general use. The three exposed functions are: *SM_GlobInit*, *SM_Build*, and *SM_Mapper*. Each uses the *rPM* and *MV* named lists to receive and return information to the caller. *SM_GlobInit* builds the initial *rPM* named list and populates it with default values. *SM_Build* receives the *rPM* and *MV* named lists, validates the location ID supplied, sets the level of the boundaries needed, loads the boundaries from the datasets contained in the packages of *SeerMapper*, implements the expanded boundaries to be drawn beyond the data sub-areas, and returns the loaded boundaries in the *MV* named list. *SM_Mapper* receives the *rPM* and *MV* and based on the information provided, creates the map. The functions are very depended on the information in the *rPM* and *MV* begin correct, the use of these functions or modification to there data structures is not recommended.

Value

The functions returns named list containing the *rPM* named list (*SM_GlobInit*, *SM_Build*), the *MV* named list (all three functions) and a named list of the graphic x and y limits used in the mapping.

Author(s)

James B Pearson and Linda Pickle, StatNet Consulting, LLC, Gaithersburg, MD

`st99_d00` *US State Boundary Data and Information*

Description

US State and Territory Boundary datasets from the 2000 or 2010 census for use with the Seer Mapper packages (*SeerMapper* and the SatScan Mapping Program.

Usage

```
data(st99_d00)
```

Format

SpatialPolygons class of object. Keyed on the State FIPs code. Each list element is a "polygons" class object contains a list of "Polygons" class objects describing the state boundaries and several attributes: Boundary Box, Label Point, Area, and ID.

Details

```
The state FIPS code are:
    01 - Alabama
    02 - Alaska
    04 - Arkansas
    06 - California
    08 - Colorado
    09 - Connecticut
    10 - Delaware
    11 - District of Columbia
    12 - Florida
    13 - Georgia
    15 - Hawaii
    16 - Idaho
    17 - Illinois
    18 - Indiana
    19 - Iowa
    20 - Kansas
    21 - Kentucky
    22 - Louisiana
```

```
23 - Maine
24 - Maryland
25 - Massachusetts
26 - Michigan
27 - Minnesota
28 - Mississippi
29 - Missouri
30 - Montana
31 - Nebraska
32 - Nevada
33 - New Hampshire
34 - New Jersey
35 - New Mexico
36 - New York
37 - North Carolina
38 - North Dakota
39 - Ohio
40 - Oklahoma
41 - Oregon
42 - Pennsylvania
44 - Rhode Island
45 - South Carolina
46 - South Dakota
47 - Tennessee
48 - Texas
49 - Utah
50 - Vermont
51 - Virginia
53 - Washington
54 - West Virginia
55 - Wisconsin
56 - Wyoming
72 - Puerto Rico
```

As of Version 1.0 of *SeerMapper* package, the .rda files are compressed using the "xz" method to reduce the disk space requirements.

The *SeerMapper* package contains the state boundary files from the U. S. 2000 Census state boundary data. Since the U. S. 2000 state boundaries are very similar to the U. S. 2010 state boundaries, the U. S. 2000 state boundaries are used for mapping data for both census years.

The *censusYear* call parameter or using the *SeerMapper2010* function call informs the package which census year boundaries to use during the mapping. In the case of the state boundaries, they are the same. Three states changed counties and have additional 2010 counties boundary files. Most of the change was with the tract boundaries, so there are two set of three supplemental packages containing the tract boundaries for each state.

The state, county and tract boundary shape file data was downloaded from the U. S. census website on July 3, 2016. The shapefiles were uploaded to the "www.MapShaper.org" website and using the modified Visvalingam method with intersection repair enabled, prevent shape removal enabled, and

coordinate precision set to 0.0, the boundaries were simplified from 100 the original. This reduced the space requirements by 90 The shapefiles were downloaded and converted to SpatialPolygons structures and saved as one county dataset per state, one tract dataset per state and census year and the state boundary set (st99_d00). The coordinates started as lat/long values and have been transformed to a cartisian coordinates system based on proj4 string "+proj=eqdc +lat_1=33 +lat_2=49 +lat_0=39 +lon_0=96w +units=m" for mapping. All boundary data used by the *SeerMapper* package is pre-transformed to this project for better visualization and to reduce the time required to build each map.

The *st99_d00* is used to support both census years (2000 and 2010). It contains the information to identify the three states that had county boundary changes and triggers the program to use different county boundary files when mapping census year 2010.

The boundary spatial data

The @data structure of the saved *st99_d00* SpatialPolygonsDataFrame contains the following information for use by the *SeerMapper* function:

row.names the row.names of the SPDF are the 2 digit state FIPS codes.

ID the state 2 digit FIPS code.

stID the state 2 digit FIPS code.

abbr = A character vector of the 2 letter state abbreviation.

stName = A character vector of the state full name.

rgID = a value representing the U. S. census region containing the state.

rgName = A character vector of the name of the region containing the state.

dvID = An integer representing the U. S. census regional division containing the state.

dvName = A character vector of the name of the regional division containing the state.

loc = A character vector of the last part of the name of the supplemental package containing the census tract boundary data for this state.

DoAdj = a value indicating if adjustments are required to the original shapefile boundary data. (e.g. moveX, moveY, scale) (Not used by *SeerMapper*.)

moveX = A numeric value of the number of degrees latitude the centroid of the state was moved to create the boundary data for this state. (Not used by *SeerMapper*.)

moveY = A numeric value of the number of degrees longitude the centroid of the state was moved to create the boundary data for this state. (Not used by *SeerMapper*.)

scale = A numeric value represntating the scaling factor used on the original shapefile data to create the boundary data in this dataset. The value of 1 indicates no scaling change. (Not used by *SeerMapper*.)

proj = A character vector representing the proj4 values of the boundary data projection. All boundaries are transformed to an equal area projection for the mapping. (Not used by *SeerMapper*.)

county00 = An integer value of the number of counties in the state during the 2000 census year.

county10 = An integer value of the number of counties in the state during the 2010 census year.

tracts00 = An integer value of the number of 2000 census tracts contained in the state.

tracts10 = An integer value of the number of 2010 census tracts contained in the state.

change10 = A logical value (T/F) indicateing (if TRUE), the states had county boundary changes in the U. S. 2010 Census. If TRUE, the backages uses the coXX_d10 datasets for the states instead of the standard cXX_d00 datasets for the county boundaries in the state for mapping.

c_X = A numeric value of the X coordinate of the centroid of the state.

c_Y = A numeric value of the Y coordinate of the centroid of the state.

To ensure the boundary datas correctly align when census tracts are mapped within counties and counties within states, the state boundaries based on their census tract boundaries data. The state census tract data was imported from the U. S. census website and uploaded into URL:www.mapshaper.org and reduced as stated above to 10 The resulting shapefile for tracts in a state are saved as the census tract baseline shapefile for the state. The census tract boundaries are joined together into counties to create the state's county boundary dataset. The county boundaries are then joined together to create the state boundary and added to this state boundary dataset.

Source

The US State boundary SpatialPolygon structures are generated by collapsing the census tract bountaries from the U. S. Census website urlhttp://www.census.gov/geo/maps-data/data/cbf/cbf_tracts.html for the appropriate census year - 2000 or 2010. The shapefiles were downloaded on July 3, 2016.

Examples

```
data(st99_d00)
```

`state_CM_Co_Data` *County Level Cancer Mortality Rates for se;ected state for 2009 to 2013*

Description

These datasets containing the cancer mortality rates from the State Cancer Profile website at the county level for several states containing Seer Registries for the period of 2009 to 2013. The county level data for Georgia, Washington, California, and Kentucky have been included to support examples demonstrating difference features of the *SeerMapper* package. It was obtained through the State Cancer Profiles website at www.statecancerprofiles.cancer.gov. This data represents all races and sexes for all sites of cancer. The data was edited to remove the recent trend data and to add a pseudo-pValue based on the US age adjusted rate (AA_Rate = 168.5) and the abbreviation of the associated Seer Registry (if any). The dataset names for the state dataset are: Georgia_CM_Co_Data, Washington_CM_Co_Data, Kentucky_CM_Co_Data, and California_CM_Co_Data.

Usage

```
data(Georgia_CM_Co_Data)
```

Format

All of the state Cancer Mortality Rate datasets are at the county level and contains 10 columns:

stcoID is a numeric value for the state and county U. S. FIPS codes. This is used to link the data to the county and state boundary data. It must be a 4 or 5 character string.

County is a character vector county name provided by State Cancer Profiles.

stID is a character vector containing the 2 digit state FIPS code.

AA_Rate is a numeric value of the age-adjust rate for the county.

Lower_CI is a numeric value for the lower confidence interval value for the age-adjusted rate.

Upper_CI is a numeric value for the upper confidence interval value for the age-adjusted rate.

Avg_DpY is a numeric value for the average number of deaths per year. This variable is not used in this package examples.

saID is a character factor vector of the abbreviation of the Seer Registry the tract is a member. The Seer Registry abbreviations match the abbreviations used in the package boundaries datasets for each Seer Registry state. For Georgia the three Seer Registries are: GA-ATL, GA-RUR, and GA-OTH.

pValue is a numeric value of 0.02 or 0.2. The pValue is set to 0.02 if the U. S. rate is not within the county's confidence interval range. It is set to 0.2 if the U. S. rate is within the interval. 0.02 indicates the county's rate is significant, while 0.2 indicates the rate is not significant.

pV is a logical value indicating whether the county is significant or not.

The first two rows of the data.frame represent the rate information for the state and the USA. These rows MUST be deleted before using the data.frame with the package.

In the Georgia and Kentucky datasets, there are two additional columns: "FIPS" and "Rate". These columns are identical to the "stcoID" and "AA_Rate" columns and are used to demonstrate how the default values for the *idCol* and *dataCol* call parameters operate.

Source

The State Cancer Mortality data for the years 2009 to 2013 for all races and sexes for each county for all cancer sites was exported from the State Cancer Profile website www.statecancerprofiles.cancer.gov on September 2, 2016. The data was modified to include the pValue, county's Seer Registry Abbreviation (if associated), and a stripped down county name. Data was collected for the counties of Georgia, California, Kentucky, and Washington.

`state_DemYY_XX_Data` *state 2000 or 2010 County / Census Tract level Test demographic data*

Description

These datasets are based on the 2000 and 2010 census demographic data for all of the counties or census tracts in the identified state. This data supports the examples in the *SeerMapper* package and contains demographics for Georgia (GA) and Washington (WA) state. Each dataset contsins the location IDs (either a county FIPS codes (5 digits) or census tract FIPS code (11 digits)), the abbreviation of the associated Seer Registry (if any), 2000/2010 population, area in sq. miles of each county or tract, 2000/2010 population density (pop/area), population 65 years old or older, the number of housing units, the number of units occupied, the number of units occupied by owner, and population of hispanics.

In Georgia all counties and census tracts belong to registries. The three Seer registry areas in Georgia are identifed, using the abbreviations of: GA-ATL, GA-RUR, and GA-OTH. In the examples, this data is used for full state mapping, Seer registry areas mapping (subdivided by Seer registry area abbreviations), and randomly selected data to represent different situations for partial data collections. This data can be used by both the *SeerMapper* package,

Usage

```
data(GA_Dem_Tr_Data)
```

Format

The naming convention for the files are: SS_DemYY_AA_Data, where SS is the same 2 character postal abbreviation (GA = Georgia and WA = Washingon), YY is "" for 2000 and "10" for 2010 census demographic data, and AA is the area type (Co = County, and Tr = Census Tract). The files contained in this package are:

- GA_Dem_Co_Data - Georgia County Demographic data for 2000
- GA_Dem10_Co_Data - Georgia County Demographic data for 2010
- GA_Dem_Tr_Data - Georgia Census Tract Demographic data for 2000
- WA_Dem_Co_Data - Washington County Demographic data for 2000
- WA_Dem10_Co_Data - Washington County Demographic data for 2010
- WA_Dem_Tr_Data - Washington Census Tract Demographic data for 2000

The U. S. Census demographic data files contains hundreds for variables and counts. These files are used in the examples and only contain the following 11 columns of data:

FIPS is a character vector of the U.S. County (5 digits) or Census Tract (11 digits) FIPS codes.

pop2000 or pop2010 is a integer containing the 2000 or 2010 sub-area population.

popdens is a integer containing the 2000 or 2010 population density of the sub-areat.

areasm is a numeric area of the sub-area(county or tract) in square miles.

saID is a character factor vector of the abbreviation of the Seer Registry the sub-area is a member. The Seer Registry abbreviations match the abbreviations used in the package boundaries datasets for each Seer Registry state. For Georgia the three Seer Registries are: GA-ATL, GA-RUR, and GA-OTH.

age.65.up is an integer representing the population with the age of 65 or higher.

hh.units is an integer representing the number of physical housing units in the county or census tract.

hh.occupied is an integer representing the number of housing units that are occupied. (hh.units-hh.occupied) equals the number of housing units vacant.

hh.owner is an integer representing the number of housing units that are occupied by the owner. (hh.occupied-hh.owner) equals the number of units that are occupied by renters.

Source

The Georgia and Washington 2000 and 2010 Census demographis at the county and census tract levels are based on the American Fact Finder data from the CENSUS.GOV website demographic data and the the R package UScensus2000tract package.

`USStates_CM_St_Data` *State Level Cancer Mortality Rates for U. S. for 2009 to 2013*

Description

This dataset containing the cancer mortality rates from the State Cancer Profile website at the state level for United States for the period of 2009 to 2013. The dataset is used to support examples demonstrating difference features of the *SeerMapper* and *Seer2010Mapper* packages. It was obtained through the State Cancer Profiles website at www.statecancerprofiles.cancer.gov. This data represents all races and sexes for all sites of cancer. The data was edited to remove the recent trend data and to add a pseudo-pValue based on the US age adjusted rate (AA_Rate = 168.5) and the abbreviation of any associated Seer Registry (if any). When a state contains multiple Seer Registries, the Abbreviation of "xx-MIX" is used where xx is the 2 character state abbreviation.

Usage

```
data(USStates_CM_St_Data)
```

Format

All of the state Cancer Mortality Rate datasets are at the county level and contains 10 columns:

stcoID is a numeric value for the state and county U. S. FIPS codes. This is used to link the data to the county and state boundary data. It must be a 4 or 5 character string. For the US States data, the 5 digit FIPS code contains 2 characters for the state FIPS and three "0" for the county.

State is a character vector state name provided by State Cancer Profiles.

stID is a character vector containing the 2 digit state FIPS code.

AA_Rate is a numeric value of the age-adjust rate for the county.

Lower_CI is a numeric value for the lower confidence interval value for the age-adjusted rate.

Upper_CI is a numeric value for the upper confidence interval value for the age-adjusted rate.

Avg_DpY is a numeric value for the average number of deaths per year. This variable is not used in this package examples.

saID is a character factor vector of the abbreviation of the Seer Registry the tract is a member. The Seer Registry abbreviations match the abbreviations used in the package boundaries datasets for each Seer Registry state. For Georgia the three Seer Registries are: GA-ATL, GA-RUR, and GA-OTH.

pValue is a numeric value of 0.02 or 0.2. The pValue is set to 0.02 if the U. S. rate is not within the county's confidence interval range. It is set to 0.2 if the U. S. rate is within the interval. 0.02 indicates the county's rate is significant, while 0.2 indicates the rate is not significant.

pV is a logical value indicating whether the county is significant or not.

FIPS is a numeric value for the state and county U. S. FIPS codes. This is used to link the data to the county and state boundary data. It must be a 4 or 5 character string. For the US States data, the 5 digit FIPS code contains 2 characters for the state FIPS and three "0" for the county. This is a duplicate column of the stcoID column for use in demonstrating using the default location ID column name.

Rate is a numeric value of the age-adjust rate for the county. This is duplicate column to the AA_Rate column for use in demonstrating using the default dataCol value of "Rate".

The first two rows of the data.frame represent the rate information for the state and the USA. These rows MUST be deleted before using the data.frame with the package.

Source

The US Cancer Mortality data for the years 2009 to 2013 for all races and sexes for each state for all cancer sites was exported from the State Cancer Profile website www.statecancerprofiles.cancer.gov on September 2, 2016. The data was modified to include the pValue, state's Seer Registry abbreviation (if associated), and a validated state name.

Index