Package: micromapST (via r-universe)

September 26, 2026

Type Package

Version 3.1.1

Date 2025-04-07

Title Linked Micromap Plots for U. S. and Other Geographic Areas

Author Jim Pearson [aut, cre, cph], Dan Carr [aut, cph], Linda Pickle [ctb, cph]

Maintainer Jim Pearson <jbpearson353@gmail.com>

Imports methods, graphics, grDevices, utils, stringr, RColorBrewer, labeling, sf, spdep, rmapshaper, tools, readxl, writexl, stats

Depends R (>= 4.0.0)

Suggests R.rsp

VignetteBuilder R.rsp

Description Provides the users with the ability to quickly create linked micromap plots for a collection of geographic areas. Linked micromap plots are visualizations of geo-referenced data that link statistical graphics to an organized series of small maps or graphic images. The Help description contains examples of how to use the 'micromapST' function. Contained in this package are border group datasets to support creating linked micromap plots for the 50 U.S. states and District of Columbia (51 areas), the U. S. 20 Seer Registries, the 105 counties in the state of Kansas, the 62 counties of New York, the 24 counties of Maryland, the 29 counties of Utah, the 32 administrative areas in China, the 218 administrative areas in the UK and Ireland (for testing only), the 25 districts in the city of Seoul South Korea, and the 52 counties on the Africa continent. A border group dataset contains the boundaries related to the data level areas, a second layer boundaries, a top or third layer boundary, a parameter list of run options, and a cross indexing table between area names, abbreviations, numeric identification and alias matching strings for the specific geographic area. By specifying a border group, the package create linked micromap plots for any geographic region.

The user can create and provide their own border group dataset for any area beyond the areas contained within the package with the 'BuildBorderGroup' function. In April of 2022, it was announced that 'maptools', 'rgdal', and 'rgeos' R packages would be retired in middle to end of 2023 and removed from the CRAN libraries. The 'BuildBorderGroup' function was dependent on these packages. 'micromapST' functions were not impacted by the retired R packages. Upgrading of 'BuildBorderGroup' function was completed and released with version 3.0.0 on August 10, 2023 using the 'sf' R package. References: Carr and Pickle, Chapman and Hall/CRC, Visualizing Data Patterns with Micromaps, CRC Press, 2010. Pickle, Pearson, and Carr (2015), micromapST: Exploring and Communicating Geospatial Patterns in US State Data., Journal of Statistical Software, 63(3), 1-25., <https://www.jstatsoft.org/v63/i03/>. Copyrighted 2013, 2014, 2015, 2016, 2022, 2023, 2024, and 2025 by Carr, Pearson and Pickle.

License GPL (>= 2)

LazyData no

LazyLoad no

BuildResaveData best

ByteCompile yes

Config/build/clean-inst-doc FALSE

NeedsCompilation no

RoxygenNote 7.2.3

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libnode-dev libx11-dev zlib1g-dev

Repository https://jpearson0525.r-universe.dev

Date/Publication 2025-04-08 08:41:44 UTC

RemoteUrl https://github.com/cran/micromapST

RemoteRef HEAD

RemoteSha faee0b0a98216583807f2936ffdcc6e07fdaa5c7

Contents

micromapST-package 4
AfricaBG 9
AfricaPopData 11
bordGrp 12
BuildBorderGroup 18
ChinaBG 32
ClnStrName_MST 34
cnPopData 35

detailsVariables . . . 36
Educ8thData . . . 37
glyph-arrow . . . 38
glyph-bar . . . 39
glyph-boxplot . . . 40
glyph-ctrbar . . . 41
glyph-dot . . . 43
glyph-dotconf . . . 44
glyph-dotse . . . 46
glyph-dotsignif . . . 47
glyph-id . . . 49
glyph-mapxxxx . . . 50
glyph-normbar . . . 52
glyph-rank . . . 54
glyph-scatdot . . . 55
glyph-segbar . . . 58
glyph-ts . . . 60
KansasBG . . . 62
KansPopInc . . . 65
LOWESSData . . . 66
MarylandBG . . . 67
mdPopData . . . 68
messages-BG . . . 69
messages-MM . . . 88
micromapGDefaults . . . 111
micromapGSetDefaults . . . 121
micromapGSetPanelDef . . . 122
micromapSEER . . . 123
micromapST . . . 124
NewYorkBG . . . 147
nyPopData . . . 149
panelDesc . . . 150
Plot_Vis . . . 158
Seer18Area . . . 160
SeoulPopData . . . 161
SeoulSKoreaBG . . . 161
statePop2010 . . . 163
SynTable . . . 164
TSdata . . . 165
UKIrelandBG . . . 166
UKIrelandPopData . . . 172
USSeerBG . . . 173
USStatesBG . . . 178
UtahBG . . . 180
UtahPopData . . . 182
wflung00and95 . . . 185
wflung00and95US . . . 186
wflung00cnty . . . 187

wmlung5070 188
wmlung5070US 189
X-Axis 190

Index **192**

`micromapST-package` *A graphics package to easily and quickly create linked micromaps for a specified geographic collection of areas.*

Description

The micromapST package provides a means of creating multiple column graphics representing data for a collection of geographic areas. The originally micromapST was limited to creating linked micromaps for each US 50 state and the District of Columbia. With this release, the package has been updated to be able to create linked micromaps for any collection of areas that is defined in a border group dataset (see details). Each area's graphical element is linked to a small map by means of color.

Details

Package:	micromapST
Type:	Package
Version:	3.1.1
Date:	2025-04-03
License:	GPL-2
LazyLoad:	no

The package uses the following R released packages: stringr, RColorBrewer, graphics, stats, grDevices, and labeling. When the package is download, these support packages should also be loaded. If not, please install these packages before loading micromapST.

This release is a test release prior to an official CRAN release.

Linked micromap plots link statistical graphs to an organized set of small maps, thus adding geographical context to the graphs and data. The *micromapST* package has been expanded to be able to use boundary data from any collection of geographic areas through the use of border group datasets. Each border group dataset contains five R objects that define the boundaries, run parameters and name, abbreviation, and ID relationships for a geographic region. When a border group is specified in the function call, the associated dataset is loaded and the five R objects become the key data structures used by micromapST to create the linked micromaps. The five R objects are:

- *areaParms* - run parameters for this border group
- *areaNamesAbbrsIDs* - the area name, abbreviation and numeric ID associates for each area.
- *areaVisBorders* - the boundary data for each area, indexed by the area abbreviation
- *L2VisBorders* - the boundary data for overlaying boundaries within the areas.

- *RegVisBorders* - the boundary data for a region of areas.
- *L3VisBorders* - the boundary data for the entire collection of areas.

The currently the supported sets of border groups contain in this package are:

- *USStatesBG* - The 50 U.S. states and the District of Columbia
- *USSeerBG* - The 20 U.S. Seer Areas within the U.S.
- *KansasBG* - The 105 counties in the state of Kansas
- *MarylandBG* - The 24 counties of Maryland
- *NewYorkBG* - The 62 counties in the state of New York
- *UtahBG* - The 29 counties of Utah
- *SeoulSKoreaBG* - The 25 districts in the city of Seoul.
- *UKIrelandBG* - The 218 counties in UK and Ireland, for testing only.
- *ChinaBG* - The 32 provinces, municipalities, autonomous regions, and Special Administrative divisions of China.
- *AfricaBG* - The 52 countries of the African continent.

The default border group is *USStatesBG* to allow users of previous releases of *micromapST* to run without changes to their code. To support users of the test release of *micromapSEER*, a function *micromapSEER* is included. The function *micromapSEER* sets up required parameters and then calls *micromapST* with the border group set to *USSeerBG* to create the same linked micromaps as the *micromapSEER* package created. All of the call parameters are the same. The two packages were merged to allow *micromapSEER* users to benefit from new features and fixes are released under *micromapST*.

Additional border groups will be added over time. The user may also create their own border group for use with *micromapST* (see paper on creating *micromapST* border groups.)

The entire micromap is created to fits on a single page. The page may be portrait or landscape and can range from an 8.5 x 11 up to a 11 to 17 page. Areas are grouped into panels from 3 to 5 areas each based on the sort variable, with the median-valued area set off in a separate panel in the middle of the page. If the median panel contains more than 1 area, a full link micromap panel is generated. Otherwise a single line representing the area is drawn and the median area is highlighted in the panels above and below the median. In the case of the U.S. Map and 51 states, there are 5 panels of 5 areas (states) above the median and 5 panels of 5 areas (states) below the median row.

The U.S. Seer Registry data may be groups of 9, 11, 13, 17 or 18 registries. Number of registries per panel and number of panels are dynamically setup based on the number of registries involved in the micromap. There are a variety of glyphs the caller can specify for each column of the micromap: the US map with areas colored, a list of registry names or abbreviations (the default) and one or more statistical graphics. The order of these panels is specified by the caller.

The statistical glyphs implemented in this version are plots of dots, dots with significant, dots with confidence intervals, dots and intervals based on Standard Error, horizontal bars, arrows, time series with or without confidence bands, horizontal stacked (segmented (SEGBAR), normalized (NORMBAR) or centered (CTRBAR)) bar charts, scatter plots and boxplots. The layout of the linked micromap plot is defined by the *panelDesc* data.frame that is passed to the *micromapST* function.

If the micromap cannot fit on one page, warnings are generated and the function is stopped. It is suggested the caller increase the size of the page (graphic space) being used to compensate.

The U.S. map of states and areas used by the *USStatesBG* and *USSeerBG* border groups are generalized boundary map, based on Mark Monmonier's visibility map. These maps are simplified to maximize the color areas shown for each state and to minimize the length of the boundary lines while still allowing identification of each area. At some future time, all border groups should have their boundaries characterized to enhance the linked micromap's readability.

One of the biggest enhancements in this version of *micromapST*, is support for other geographic areas. This has been added using border groups. Each border group data set contains the unique collection of information, run parameters, names, abbreviations, and boundary information for the geographic region. The package contains some pre-made border groups and the package user is encourage gather the required information and create their own border group. This is not a small task and required boundary file manipulation to reduce the complexity of the boundaries and researching to identify a suitable list of names, abbreviations and ID for each sub-area within the desired geographical area. The author is working on a guideline and a step by step procedure to help the user create their own border groups. This release includes border group to re-create the original link micromaps in earlier versions of *micromapST* for the 50 U. S. States and DC and micromapSEER for the 20 NCI Seer Registries. Several other test versions of border groups have been included in this release. The complete list of border groups included are:

- *USStatesBG* - the default border group to continue generating the original linked micromaps in earlier versions of micromapST.
- *USSeerBG* - a border group to support creating linked micromaps for NCI. A function micromapSEER is included to support earlier coding. This function automatically pulls in the USSeerBG border group.
- *KansasBG* - a border group to support the generation of linked micromaps for the 105 counties in the state of Kansas.
- *MarylandBG* - a border group to support the generation of linked micromaps for the 23 counties and the city of Baltimore in the state of Maryland.
- *NewYorkBG* - a border group to support the generation of linked micromaps for the 62 counties in the state of New York.
- *UtahBG* - a border group to support the generation of linked micromaps for the 29 counties in the state of Utah.
- *ChinaBG* - a border group to support the generation of linked micromaps for the 35 provinces, special administrative regions, and autonomous regions of China.
- *UKIrelandBG* - a border group for testing a large number of sub-areas (counties) in multiple regions. The package currently supports up to 110. This border group is designed to help in testing and developing support for up to 218 sub areas (counties, etc.)
- *SeoulSKoreaBG* - a border group to support the generation of linked micromaps for the 25 districts within the city of Seoul South Korea.
- *AfricaBG* - a border group to support creation of linked micromaps for 52 countries of Africa. This border group is for a continent with the sub-areas being countries.

The following datasets have also been included for use in many examples:

- *Educ8thData* - U.S. Education Data from an 8th grade survey.
- *TSdata* - Sample time series data.
- *statePop2010* - U. S. State Population data for 2010.

- *wflung00and95* - U. S. White Female Lung cancer data for 2000 and 1995 by state/terr.
- *wflung00and95US* - U. S. White Female Lung cancer data U. S. totals for 2000 and 1995.
- *wflung20cnty* - U. S. White Female Lung cancer sample data by county for 2000.
- *wmlung5070* - White Male Lung cancer sample data for 1950 and 1970.
- *wmlung5070US* - White Male Lung cancer sample data for 1950 and 1970 totals for the U. S.

Refer to the chapter on each border group for definitions on the Names, Abbreviations, and IDs used in the border group to link the user data to the boundary information to draw the micromap maps.

The sort order of the rows (areas) is based one of the statistical data columns as specified by the user. Correlation between multiple statistical columns can be judged visually by comparing the pattern of one column's values from top to bottom of the page with that of the sorted column. Spatial clusters of states with similar values of the sorting variable can be identified on the small maps that are linked to the graphics by color.

A area linked micromap plot is generated by 4 steps:

```
# load the micromapST package

library(micromapST)

# Read, create or collect your data into a data.frame.

statsDFrame <- data.frame(a row per area,
                          column per variable to be plotted,
                          row.names set to the area names
                          or abbreviations)

# now set up a data frame that defines the labels,
#       panel and page layout

panelDesc<-data.frame(...)  # (see section on the panelDesc data.frame)

# specify the data source, panelDesc, sorting variable and
#       order, and call the stateMicromap function

micromapST(statsDFrame, panelDesc, title=c("title1","title2"),
           details=list(options=values)
          )
```

The package contains a set of examples of how to produce linked area micromaps. The datasets used in each example are provided to help you learn how to use micromapSEER.

As of release 2.0.1 of the package, a new function has been added. The *BuildBorderGroup* function provide assistance to the user in building their own border groups from shape file and a name table as needed for geographic areas not covered by areas included in the package. Review the documentation on the *BuildBorderGroup* function for more details.

There are four primary sources for boundary data:

- U. S. Census Bureau for U. S. states and territories (https://www.census.gov/) (2021 - Cartographic Boundary Files) ;
- GADM (Global Administrative area Database) for all of the countries and sub-divisions in the world (https://gadm.org) (V4.1, July 16, 2022 release, older versions are available.);
- DIVA-GIS (https://diva-gis.org) (January 2012); and
- local governments.

All of the boundary data (shape files or json format) are free for public use. Care must be taken to ensure your data matches the boundary areas available in each collection. Data location IDs and boundaries change over time as areas are split, merged, added, or eliminated. The BuildBorderGroup function tries to clean up and repair the boundary data as best as it can. Always inspect the boundary data and make sure its usable for your application and is valid.

With the announcement of the support retirement of maptools, rgeos, rgdal, and effectively sp and spdep, this package had to be upgraded to continue to be useful by it's users. V3.0.0 completes that enhancement.

The newest features is the option to create a LOWESS line based on the data in the '`scatdot`' plot glyphic.

Note

The packaged is tuned to work with an area 7.5" wide and 10.5" high. Testing has shown it works well with portrait or landscape orientation and areas up to 11" x 17".

The examples in this package the output is directed to a PDF file for best clarity and resolution. File types of SVG, PNG, JPEG or TIFF can also be used. If the output is directed to a window, it is suggested a `windows( 7.5, 10.5, xpinch=72, ypinch=72, pointsize=9 )` command is used to set up the window to best display the linked micromap. The results will vary based on the resolution of the monitor being used.

Author(s)

James B Pearson, Jr <jbpearson353@gmail.com> and Daniel B. Carr <dcarr@gmu.edu>, with contributions from Linda Pickle
Maintainer: "James B. Pearson Jr." <jbpearson353@gmail.com>
Package compiled by "James B, Pearson, Jr." <jbpearson353@gmail.com>

References

Daniel B. Carr and Linda Williams Pickle, Visualizing Data Patterns with Micromaps, CRC Press, 2010
Linda Williams Pickle, James B. Pearson Jr., Daniel B. Carr (2015), micromapST: Exploring and Communicating Geospatial Patterns in US State Data., Journal of Statistical Software, 63(3), 1-25., https://www.jstatsoft.org/v63/i03/

`AfricaBG` *AfricaBG border group datasets to support creating micromaps for the 52 countries in Africa*

Description

The *micromapST* function has the ability to generate linked micromaps for any geographical area. To specify the geographical area, the '`bordGrp`' call argument is used to specify the border group dataset for the geographical area. The *AfricaBG* border group dataset supports creating linked micromaps for the 52 countries (sub-areas) on the African continent. When the '`bordGrp`' call argument is set to *AfricaBG*, the appropriate name table (country names and abbreviations) and the 52 sub-areas (countries) boundary data is loaded in *micromapST*. The user's data is then linked to the boundary data via the country's name, abbreviation, alternate abbreviation, or ID based on the table below.

Usage

```
data(AfricaBG)
```

Details

The *AfricaBG* border group dataset contains the following data.frames:

areaParms - contains specific parameters for the border group

areaNamesAbbrsIDs - containing the names, abbreviations, and numerical identifier for the 59 countries in Africa.

areaVisBorders - the boundary point lists for each country in Africa.

L2VisBorders - the boundaries for an intermediate level and is not used in this border group and is set to L3VisBorders as a place holder.

RegVisBorders - the boundaries for regions in Africa. In this implementation of the border group, no regions are specified. This data frame is not used and is set to L3VisBorders as a place holder.

L3VisBorders - the boundary of the Africa continent.

The Africa continent border group contains 52 country sub-areas. Each country has a row in the areaNamesAbbrsIDs data.frame and a set of polygons in the areaVisBorders data.frame datasets. No regions are defined in the Africa border group, so the L2VisBorders dataset is not used and the regions option is disabled. The L3VisBorders dataset contains the outline of the Africa continent.

The details on each of these data.frame structures can be found in the "bordGrp" section of this document. The *areaNamesAbbrsIDs* data.frame provides the linkages to the boundary data for each sub-area (country) using the fullname, abbreviation, alternate abbreviation, and numerical identifier for each country to the *<statsDFrame>* data based on the setting of the '`rowNames`' call argument.

A column or the data.frame row.names must match one of the types of names in the *areaNamesAbbrsIDs* data.frame name table. If the data row does not match a value in the name table, an warning is issued and the data is ignored. If no data is present for a sub-area (country) in the name table, the sub-area is mapped but not colored.

The following are a list of the names, abbreviations, alternate abbreviations and IDs for each country in the *AfricaBG* border group.

name	ab	alt_ab	id
Algeria	ALG	DZ	01
Angola	ANG	AO	02
Benin	BEN	BJ	03
Botswana	BOT	BW	04
Burkina Faso	BUF	BF	05
Burundi	BUR	BI	06
Cameroon	CAM	CM	07
Cape Verde	CAP	CV	08
Central African Republic	CAR	CF	09
Chad	CHA	TD	10
Comoros	COM	KM	11
Congo-Brazzaville	CNG	CG	12
Cote d'Ivoire	CDI	CI	13
Democratic Republic of Congo	ZAI	ZR	14
Djibouti	DJI	DJ	15
Egypt	EGY	EG	16
Equatorial Guinea	EQG	GQ	17
Eritrea	ERI	ER	18
Ethiopia	ETH	ET	19
Gabon	GAB	GA	20
Gambia	GAM	GM	21
Ghana	GHA	GH	22
Guinea	GIN	GN	23
Guinea-Bissau	GUB	GW	24
Kenya	KEN	KE	25
Lesotho	LES	LS	26
Liberia	LIB	LR	27
Libya	LAJ	LY	28
Madagascar	MAD	MG	29
Malawi	MAA	MW	30
Mali	MAL	ML	31
Mauritania	MAU	MR	32
Morocco	MOR	MA	33
Mozambique	MOZ	MZ	34
Namibia	NAM	NA	35
Niger	NIG	NE	36
Nigeria	NIR	NG	37
Rwanda	RWA	RW	38
Sao Tome and Principe	STP	ST	39
Senegal	SEN	SN	40
Sierra Leone	SIL	SL	41
Somalia	SOM	SO	42
South Africa	SOU	SA	43
Sudan	SUD	SD	44

Swaziland	SWA	SZ	45
Tanzania	TAN	TZ	46
Togo	TOG	TG	47
Tunisia	TUN	TN	48
Uganda	UGA	UG	49
Western Sahara	WES	WS	50
Zambia	ZAM	ZM	51
Zimbabwe	ZIM	ZW	52

When this border group was created, there appeared to be no consistant set of abbreviations for the African countries. Therefore, the two most commonly found sets of abbreviations are included as the *abbr* and *alt_abbr* abbreviation sets. Set ‘`rownames`’ to "ab" to reference the primary set and "alt_ab" to reference the second set of abbreviates in the name table.

The ‘`rowNames`’ = "alias" and the ‘`regionB`’ and ‘`dataRegionsOnly`’ features are not supported in the *AfricaBG* border group.

`AfricaPopData` *Test data for the Africa border Group*

Description

This dataset contains the population and country data for the 52 countries in the African border group.

Usage

```
data(AfricaPopData)
```

Format

A data frame with 52 observations, 1 for each African country, on the following "x" variables.

Rank an integer rank of the country in Africa.

Name a character vector containing the Africa Country Name.

Abbr a character vector containing the African Country Abbreviation.

Projection a numeric vector of the number of the county’s population

AvrRelGw a numeric vector of the average relative population growth.

AvrAbsGw a numeric vector of the average absolute population growth.

Est2Time a numeric vector of the estimated time to double the population - years.

OfficialPop a numeric vector of the official population.

MMDDYY the date the information was last updated.

PercOf a numeric vector representing the percentage the country’s population is to the total population of Africa.

Details

This dataset was pulled from wikipedica on the population numbers for African countries.

Author(s)

Linda W. Pickle and Jim Pearson of StatNet Consulting, LLC, Gaithersburg, MD

`bordGrp` *Defining different spatial areas for Linked Micromap using the micromapST package*

Description

The *micromapST* function can be used to create linked micromaps for may different spatial areas by using different border groups. Several border group (bordGrp) examples are contained in the package and include the 51 states and DC of the United States, the counties of Kansas, Maryland, New York, Utah, the countries and provinces of the U.K. and China, and the U. S. Seer Registries used by the National Cancer Institute. Each border group is a different dataset containing the unique boundaries and operational information to allow micromapST to work in a different spatial area. The structure of each border group dataset is identical with the same variable names and types of structures. A user can build their own border group dataset to meet their specific spatial area needs. Because the package contains several border group datasets, the use of lazydata or lazyloading has been disabled.

The name of the border group is specified in the '`bordGrp`' call parameter. To permit a user to reference a border group dataset not contained in the package, and reside in a user's folder, the '`bordDir`' must be used to direct the package to the border group. The border group must be saved under R using the *save* function with the file extension of ".rda". For example: bordGrp="private", bordDir="c:/SavedBorderGroups"

Each border group contain six (6) datasets by the same data.frame names. This allows the micromapST package the ability to quickly load a particular border group and create the requested micromaps. The five data.frames are: areaParms, areaNamesAbbrsIDs, areaVisBorders, L2VisBorders, RegVisBorders, and L3VisBorders. Since the same data.frame names are reused in each border group, the R lazyload feature is disabled in the package.

The following describes the purpose and structure of each data.frame in the border group dataset:

'`areaParms`' - contains specific parameters and operational information for the border group

'`areaNamesAbbrsIDs`' - containing the names (full text), name abbreviations, wildcard string for name matching, alternate name abbrevations, regional (Level 2) association of the sub-area, and a numerical identifier for the areas in the border group. If the border group is for a state, the areas would be the counties within the state. If the border group is for a continent, then the areas are the countries on the continent. When the border group is for a country like the United States, then the areas are the administrative areas (like states, provinces, special administrative areas, or cities) within that country.

'`areaVisBorders`' - the boundary point lists for each area in the border group.

'`L2VisBorders`' - when a border group needs to have an intermediate set of boundaries drawn for clearity, the border group can provide the package a layer 2 set of boundaries via the *L2VisBorders* data.frame. The structure consists of the boundary information (point lists) of these areas and associated keys. Each area is linked to it's L2 boundary via the *L2_ID* variable (column) in the name table (*areaNamesAbbrsIDs* data.frame. At this time only the U. S. States and U. S. Seer Registry border group uses the L2 boundary overlays. It uses L2VisBorders to draw the boundaries of U. S. States around Seer Registries within a state. The *areaParms$Map.L2Borders* variable in the border group must be set to *TRUE* for the package to draw the layer 2 boundaries. If a border group does not use an intermediate level outline, L2VisBorders should be set to L3VisBorders or NA and the *areaParms$Map.L2Borders* variable set to FALSE. In this case, no L2 boundaries are drawn.

'`RegVisBorders`' - when the border group wants to work with the areas on a regional basis, the *regID* variable in the name table ('`areaNamesAbbrsIDs`'), the '`RegVisBorders`' boundary information data.frame, and the *areaParms$aP_Regions* are used to enable the feature and provide the information to map only regions of areas and overlay areas with regional boundaries highlights. When the '`regions`' call parameter is set to *TRUE* and the selected border group has the *areaParms$aP_Regions* set to *TRUE*, the package will scan the data provided by the caller and determine which regions are being referenced and which are not. The package uses the *regID* variable in the name table('`areaNamesAbbrsIDs`') to associate a area with a region and as a key into the *RegVisBorders* data.frame to draw the region boundaries. Any region not containing data and any L2 area and areas within those regions will not be drawn. If a border group does not use the regions feature the '`RegVisBorders`' data.frame in the border group should be set to the L3VisBorders data.frame or NA and the *areaParms$aP_Regions* variablel set to *FALSE*. This will disable the regions feature for the border group.

'`L3VisBorders`' - the boundary point list for an outline of the entire geographica area referenced by the border group. For the U.S. or a country border group, this is the outline of the country. For a state border group like Kansas, this is an outline of the state. For smaller areas like Seoul, it is an outline of the city. The L3VisBorders data frame must be present in the border group.

Details

The default border group is *USStatesBG* to be compatible with older R scripts using previous versions of the *micromapST* package.

This section provide the data frame structural details of each of the 6 data.frames and thier variables that make up a border group dataset.

areaParms The '`areaParms`' data.frame contains specific information about and in support of its border group. It provides column headers for the map and id glyphics and controls several features that related to handling a border group by the *micromapST* package. These controls are specific to a border group and do not changed from one *micromapST* call to the next and don't have to be part of the calling parameters.

For example, there are several built in titles and labels for the map and id glyphics. These will change for different border groups. For the *USStatesBG* border group, the map title is always "U.S. States", while in the *USSeerBG* border group the map glyphic title is "U.S. Seer Areas".

This data.frame contains the following variables that are used by micromapST when handling a border group.

The areaParms data.frame variables are:

areaUSData - logical variable. If *TRUE* then the border group represents the entire US area and labels will be placed on the first map for Alaska, Hawaii, and DC. This is option is only used with the *USStatesDF* and *USSeerBG* border groups. For the state/county border groups and foreign country border groups, *areaUSData* must be set to *FALSE*. This variable is being retired.

enableAlias - logical variable. If *TRUE*, enables the use of the "alias" name matching feature and the '`rowNames`'=*alias* call parameter. The "alias" name matching feature permits a partial ("contains") match of the loc id data in the '`rowNamesCol`' link column in the *statsDFrame* or the row.names of the *statsDFrame* data.frame. At the present time, the "alias" feature is only used in the *USSeerBG* bordGrp. It was implemented to be able to use directly use data generated by the SEER*Stat website and match the loc ids to the internal SEER*Stat registry names in the border group. This feature can be expanded when needed to other border groups if the "Alias" column in the name table is filled in with unique strings.

aP_Regions - logical variable. If *TRUE*, the package contains a valid *RegVisBorders* data.frame and the name table (*areaNamesAbbrsIDs*) contains the *regID* (or name) of a region boundary associated with the area. If the caller set the '`regions`' call parameter to *TRUE*, the package will only draw areas in regions and the region boundaries if the region contains areas with data. For examples: this allows you to provide data for the west coast U. S. states and not have the midwest, south, or northeastern states drawn. This feature is available to all border groups, but is currently only used by the USStatesBG, USSeerBG, UKIrelandBG and ChinaBG border groups. If set to *FALSE*, the regions feature is disabled and all regional information ignored.
The *RegVisBorders* should still exist, but should be set to equal the *L3VisBorders* data.frame. The '`regions`' call parameter will be ignored. As an example: In the 'USStatesBG' and 'USSeerBG' border groups, the regions are set up using the four U. S. census regions of: West, South, Midwest, and NorthEast. If only data for states in the NorthEast are passed to micromapST, only the NorthEast region and its states will be mapped when '`regions`' is set to *TRUE*. If '`regions`' is set to *FALSE* then all of the boundaries for all of the U. S. States and DC are drawn eventhough data was only provided for the states in the northeast region. This feature also allows a border group with a large number of sub-area, like the UK and Ireland border group to be assembled and used on a regional basis with fewer sub-area were the full border group is not really usable with linked micromap presentations.

ID.Hdr.1 - character variable. 1st title for the id type glyphics column.

ID.Hdr.2 - character variable. 2nd title for the id type glyphics column.

Map.Hdr.1 - character variable. 1st title element for the map type glyphics column. This variable is not implemented in this release.

Map.Hdr.2 - character variable. 2nd title element for the map type glyphics column. This variable provides the type of areas in the map. This string should be kept to less than 12-16 characters.

Map.Aspect - a numeric variable. The X/Y aspect ratio for the map borders in this border group. This is used to adjust the map glyphic to keep the area's aspect looking correct.

Map.MinH - a numeric variable. The minimum height of a group/row in inches

Map.MaxH - the maximum height of a group/row in inches.

LabelCex - a number representing the cex multiplier used on the *text* function when the map labels are drawn.

bordGrp - a character vector of the name of the border group.

Map.L2VisBorders - logical variable. If *TRUE*, the L2VisBorder border overlay is drawn on the map glyphics. If *FALSE*, the *L2VisBorders* boundaries are not drawn. This option is currently only used by USSeerBG to draw state boundaries for states containing multiple Seer Registries.

aP_Regions - logical variable. If *TRUE*, the regions feature is enabled. When the feature is enabled, the *RegVisBorders* data.frame should be included in the border group, but it is not required. The key to the regional feature is the *regID* column in the name table (*areaNamesAbbrsIDs*) that identifies the region associated with the area and the regional boundaries in the *RegVisBorders* data.frame. If *FALSE*, the regions feature is disabled.

aP_Proj - a character vector containing the projection used to create the areaVisBorders, L2VisBorders, RegVisBorders, and L3VisBorders boundary point lists.

aP_Units - a character vector containing the measurement units of the coordinates in the VisBorders boundary point lists.

Map.L3VisBorders - logical variable. If *TRUE*, the L2VisBorder boundary data is available to drawn on the map glyphics. If *FALSE*, the *L2VisBorders* boundaries are not available and are not drawn. This option is currently only used by the following border groups: USSeerBG.

Map.RegVisBorders - logical variable. If *TRUE*, the *RegVisBorder* boundary data is available in the border group and can be used to drawn a regional boundaryy overlay on the map glyphics. If *FALSE*, the *RegVisBorders* boundary data is not available and regional boundaries are not drawn. This set of boundaries are only only available in the following border groups: USStatesBG, USSeerBG, ChinaBG, and UKIrelandBG. The drawing of the regional boundaries is controlled by the '`regionB`' call parameter.

areaNamesAbbrsIDs The *areaNamesAbbrsIDs* data.frame provides the linkages between the full-name, abbreviation, and numerical identifier to link the *statsDFrame* data to the boundaries for the county areas. It is also used to validate the incoming data to ensure the linkage can be established. Within the boundary files, the area abbreviation is the key linkage.

The areaNamesAbbrsIDs dataset provides a table to permit the translation of a numerical ID (e.g., FIPS codes), area abbreviatation (should be less than 4-6 characters), optional alternate abbreviation, and full area names in the *micromapST* function.

Name - character string of each area name. Used to link the areas to boundaries when '`rowNames`' = "full" is specified. Used as the represented name of the sub-area in "ID" glyphics columns when '`plotNames`' = "full" is spedified (default).

Abbr - the name abbreviation for each area. Should be 2 to 3 character, but no more than 6 characters. Used to link the sub-areas to boundaries when '`rowNames`' = "ab" is specified. Used as the represented name of the area in "ID" glyphics columns when '`plotNames`' = "ab" is specified (default).

Alt_Abr - an alternate name abbreviation for each sub-area. Should be 2 to 3 characters, but no more than 6 characters. Most of the time this field is identical to the *Abbr* field. In some cases, multiple sets of authenticated abbreviations were found for the sub-areas in a continent or country. When this happened, the most common abbreviation should be placed in the *Abbr* field and the second abbreviation placed in the *Alt_Abr* field. This features allows the border group to be used by a wider audience. To access the *Alt_Abr* abbreviates, set the '`rowNames`' call argument/parameter to "alt_ab". The labels in the *statsDFrame* statistics data.frame will be matched against the alternate abbreviations.

ID - numerical identifier for each area. Used to link the data to boundary information when '`rowNames`' = *id* is specified. The row.names in the user provided data.frame or specificed '`rowNamesCol`' column must match the IDs in the name table. If no match a warning is generated and the data row ignored.

Alias - a character string for each area used to do a wildcard match ("contains") with the '`rowName`' or '`rowNamesCol`' specified in the *micromapST* call when the *USSeerBG* border group is used. When a match is completed, the abbreviation is used to link the user's data to the boundary data. The alias match is done when '`rowNames`' is set to "alias". There should be only one match in the data for each sub-area alias. If more are found, an error is raised. This feature is only supported in the *USSeerBG* border group.

L2_ID - is the level 2 identifier. Used to link the area to the L2VisBorders boundary data point data.frame. In the case of the USSeerBG border group, the L2 boundaries are state boundaries and the L2_ID value is the state 2 character abbreviation.

L2_ID_Name - is the full name L2 area.

regID - is the region identifier. Used to link the area to the RegVisBorders boundary data point data.frame. The USStatesBG and USSeerBG border groups use this field to like the sub-areas to the four (4) U. S. Census regions (Northeast, South, MidWest, and West) This association is used with the '`regions`' call parameter to determine which regions and their areas, etc. will be drawn when the caller provide data is mapped.

regName - is the full name of a region.

Key - is a character string used to link the name table to the boundary data in the VisBorders data.frames.

Link - when the initial border group is created, the *Key*, *Name* or the *Abbr* variables may not be able to provide a link to the original boundary data. When this happens, the border group creater must use an alternate "link" to tie the name table to the boundary data. The *link* field is used to accomplish this in the package provided border group building steps and functions. Once the border group is fully constructed, the *Link* field is not use.

MapL, MapX, and MapY These three columns have replaced the MapLabel column outlined below. The MapL column provides the label to be drawn at the coordinates provided for the area. Only a few areas should require labels. To many labels will make the map unreadable in a linked micromap. The MapX and MapY columns in the name table provide the x and y coordinates for the label on the map. The coordinates are in the same units as the boundary coordinates used.

areaVisBorders, L2VisBorders, RegVisBorders, and L3VisBorders There are four boundary dataset - The boundaries of the areas being micromapped (counties) are in areaVisBorders. The boundaries of an intermediate level (2) for general orientation are in L2VisBorders. The boundaries of regional areas for highlight overlays and regional only mapping and are in RegVisBorders. The boundaries of the an outline of the entire mapping area is in L3VisBorders.

These four data.frames contain the boundary point lists for the areas, regions and total map space. The L2VisBorders, RegVisBorders, and L3VisBorders data.frames are used to outline groups of areas, regions of the area and the entire space. The areaVisBorders data.frame contains the point lists for each sub-areas and are keyed to the name table (*areaNamesAbbrsIDs*).

The data structure for each of the following four boundary data.frames is:

seq Key x y hole

seq - a numerical sequence number of the boundary points in the data.frame.

Key - the Key field had different uses in each of the four VisBorders structured data.frames. In the areaVisBorders data.frame, the *Key* is the unique key for the sub-area as defined in

the name table (*areaNamesAbbrsIDs*). All of the points with the same *Key* form one or more polygons and represent boundaries of a sub-area. This allows the package to locate the boundary data for a specific sub-area when needed.
In the RegVisBorders boundary data.frame, the *Key* is the region ID associated with the boundary points (polygons). One or more sub-areas can be linked to an region boundary via the *regID* variable in the name table. The USStatesBG, USSeerBG, UKIrelandBG,and ChinaBG border groups may use of the regions feature.
In the L3VisBorders, all of the *Key* field values area set to a name that represents the entire border group area. The *Key* field is not used in drawning the area's outline. The level 3 boundary outline data.frame is always drawn when the entire geographic area is mapped. If not all of the regions are being mapped, then the L3 boundary is not drawn.

x - a numerical value for the *x* coordinates of a polygon. The end of the polygon is signaled by an *x* value of *NA*. The first point in the polygon does not have to be repeated. The polygon draw function used by micromapST will close the figure. An area may be composed of several polygons. Holes in areas are also represented by polygons and are associated with the sub-area.

y - a numerical value for the *y* coordinates of a polygon. The end of the polygon is signaled by an *y* value of *NA*.

hole - a logical value. If *TRUE*, the associated polygon is a hole within the sub-area identified by the *Key* field. A hole polygon is always drawn using the maps background color. For this reason, sub-areas containing holes (lakes, rivers, or other sub-areas), must proceed any sub-area in the data.frame that it may overlay with the hole. For example, if one sub-area "A" is contained within sub- area "B", sub-area "B" must have a hole where sub-area "A" is located and must preceed sub-area "A" in the VisBorders data.frame. In this way, sub-area "B" and its hole are drawn before sub-area "A" preventing sub- area "B" hole from overlaying sub-area "A".

Hole are processed by re-drawing the hole area with the current background color. Therefore, any area with holes must be in the data.frame prior to any areas that may occupy the hole's space in the map.

The order of the area's boundaries in these files is very important to allow correct processing of the holes and any areas that overlay holes. Holes are not unpainted polygons, but polygons repainted back to the background color. The order should be areas with holes must preceed areas that overlay hole space. This is required to ensure an area is not over-painted by an area's hole

Each data.frame should be validated before using to make sure they are clean and will not generate errors when used by *micromapST*.

Each border group contains the same six data.frames using the same six names. This allows the micromapST package the ability to quickly load a particular border group and create the requested micromaps.

See the write up on each included border group for details on the specific content of their border group dataset and the list of sub-area names, abbreviations, and id that should be used to link the data to the boundary information.

The '`regions`' feature and RegVisBorder overlays are only supported in the following border groups:

USStatesBG USSeerBG UKIrelendBG ChinaBG

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

`BuildBorderGroup` *Building new border groups for Linked Micromap created by the micromapST package*

Description

The package's *micromapST* function created linked micromaps for any geographic region. The information related to the selected regions is provided to the *micromapST* function via a *bordergrp* dataset. This dataset contains all of the operational information, the needed boundary dataset and a name table required for *micromapST* to draw the linked micromaps for the desired region of the world or elsewhere. The dataset must contain the general information data.frame (areaParms), the area boundary data.frame (list of points for each area in the boundery) and a name table that provide the the "Name", "Abbr", and "ID" location names for each area in the boundary group. The Name Table also assists in providing the L2 and Regional features. The border groups were originally created manually, one by one.

The *BuildBorderGroup* function is a compilation of the learning from building past border groups. The function tries to provide a common foundation to address many of the unique situations discovered building the border groups manually. However, there are still some situations that required the builder's intervention in order to produce a usable map for linked micromaps.

The *BuildBorderGroup* function accepts a shape file (ESRI format) and a user built name table. The name table provides the location ids (name, abbreviation or id) to allow the micromapST user ways to specify the identity of individual area using one of several forms of identifier. The forms used by the *BuildBorderGroup* and *micromapST* functions are "Name", "Abbr" (abbreviation), and the numerical "ID" identifiers that have been assigned and accepted over time for the area as the primary location identifiers.

In two special cases *micromapST* has been extended to support an "Alt_Abbr" and an "Alias" form of the location identifiers. The "Alt_abbr" form was added to handle geographic area that have two sets of commonly used abbreviations. The name table was expanded to contain both sets of abbreviations and allows the user of the border group to select the abbreviation that matches data frame location ids. The "Alias" identifier was implement to handle a special case where the source of the statistical data did not use the accepted "Name" or "Abbr" location ids, but used a more generallized string. To keep the matching as flexiable as possible, a wildcard matching alias was introduced. As each identifier in the data is examined, it is compared against the alias string with leading and traiing "*" matching character. As long as the Alias strings are unique and do not appear in two data rows, the matching provide the link between the data and the geographical area. Over the past 10 years, the source program has been changed many times, but the location id label matches has continued to work.
The user created name table also provides additional information regional identification and subsetting of the map, and area labeling. During the border group building process, the name table The Name Table also provides parameters to the *BuildBorderGroup* function to make specific modifications to areas. The modifications to areas include shifting it location, scaling it size, and rotating the area. These modifiers were used to scale Alaska to 35 it's normal size and move it to below

California to reduce the size of the total map. Hawaii was also moved below the U. S. to reduce the total size of the map.

The name table also allows the builder to assocate areas with regions in their map and Level 2 border outlining. Check the *micromapST* documentation on how the Level 2 and Region boundaries are to be used. The most important task the name table does is help provide the linkage between the rows in the user's data frames to the an area's collection of polygons from the shapefile. The Name Table also provide multiple location ids to allow the user to provide the full formal name ("Name"), one of two abbreviates (if available), or the numerical ID assigned to each area.

When the processing is completed, the user has a border group .RDA file ready for use with *micromapST* with boundaries usable in the small maps in linked micromaps and the name table documentation for the border group.

The *BuildBorderGroup* function validates all of the call parameters, inspects the information provided by the builder in the name table, inspects the shape file provided, inspects the projections used or requested, ensures the +unit= parameter in the projection is set to meters, The function performs the following steps to construct a border group: may different spatial areas by using different border groups.

In Version 3.0.0 of the *BuildBorderGroup* function, the function was upgraded to calls to functions in the retired R packages (rgdal, rgeos and maptools) as of October 16th, 2023. The spatial functions are now done by the "sf" package.

Usage

```
BuildBorderGroup(
   ShapeFile         = NULL,   # required
                ShapeFileDir      = NULL,   # defaults to NameTableDir
                # required if not the default value of "link"
                ShapeLinkName     = NULL,
                NameTableFile     = NULL,   # required
                NameTableDir      = NULL,   # required
                # required if not the default value of "link"
                NameTableLink     = NULL,
                BorderGroupName   = NULL,   # required
                # defaults to NameTableDir
                BorderGroupDir    = NULL,
                MapHdr            = NULL,   # optional
                MapMinH           = NULL,   # optional
                MapMaxH           = NULL,   # optional
                # required, default is the BorderGroupName and "Areas"
                IDHdr             = NULL,
                LabelCex          = NULL,   # optional
                # optional, but highly recommended
                ReducePC          = 1.25,   # percent value
                proj4             = NULL,   # optional
                checkPointReStart = NULL,   # optional
                debug             = 0      # debug only
             )
```

Arguments

`ShapeFile` a character string of the name of the ERSI formated shape file. Only the main part of the shape file name should be provided. The .shp, .shx, .dbf extensions should be omitted.

`ShapeFileDir` a character string defining the path to the folder containing the ERSI shape file. If the ShapeFileDir parameter is not provided or empty, the Name Table Directory will be used.

`ShapeLinkName` a character string defining the name of the variable within the shape file @data slot to use to match the polygons in the shape file with the area's row in the Name Table. The default value is "__Link".

`NameTableDir` a character string defining the path to the folder containing the Name Table File. There is no default value for this parameter.

`NameTableFile` a character string defining the name of the excel spreadsheet file or a .csv file containing the user built Name Table columns and information.

`NameTableLink` a character string defining the Name Table column name that should be used to match the *ShapeFileLink* variable to link the Name Table to the area's collection of polygons. The default value is "link".

`BorderGroupName`
a character string to use as the border group's name and dataset name. If the string ends with a "BG", it will be striped and re-added when the border group is built. If the string does not end with "BG", "BG" will be added to designate the file is a border group.

`BorderGroupDir` a character string defining the path to the folder where the border group dataset will be written at the end of the processing. If this parameter is not provided or "NA", the *NameTableDir* parameter will be used as the *BorderGroupDir* parameter. If the *BorderGroupDir* path does not exist, the *BuildBorderGroup* function will create the directory.

`MapHdr` is a two element character vector used to modify the pre-defined map header labels in *micromapST* for map type glyphs. The value is entered as `MapHdr = c("1stHdr", "2ndHdr")`. Check the *micromapST* documentation for more details. The first element (MapHdr[1]) is not implemented and is reserved for a future release. The MapHdr[2] element is used to generated the map headers for all of the map glyph tyeps. It should specify the type of area being mapped. For example for the US States map, it was set to "States". The default value is MapHdr=c(<border group name>,"Areas"). This call parameter is not required.

`MapMinH` is a numerical variable specifying the minimum height the maps should be in the group/rows in a linked micromap graphic. The default value is 0.5.

`MapMaxH` is a numerical variable specifying the maximum height the maps should be in the group/rows in a linked micromap graphic. The default value is 1.75.

`IDHdr` is a two element character vector used to modify the id glyph headers in *micromapST*. The two values are entered as `IDHdr=c("1stHdr","2ndHdr")`. The defaults for this parameter are *""* and *"States*. See *micromapST* documentation for more details.

`LabelCex` a numerical value indicating the cex multiplier to use when drawing the Map Labels (MapL) on the first map in a linked micromap graphic. The default value is 0.4 to match the micromapST maps.

`ReducePC` a numerical value between .01 and 100 tell the package to not reduce the number vertex in the shapefile. This is change from earlier releases where 0 to 1 could be used to represent 0 to 100 With the use of smaller keep values, the scale of the parameter can't be determined. Reduced percentage below 1 are common. Therefore, 0.65 is not 65 that be remaining in the shape file after simplification by rmapshaper. The default value is 1.25 to even 0.65 BuildBorderGroup to determine how this factor attects your boundary data.
The default setting for this variable is meant to handle large complex shapefiles. If the shapefile is already reduced and simplified, then it may start to look fractured and may have lines across the map. In this case, it is suggested the value for *ReducePC* be raised to keep more and more of the original shapefile data. A value as high as 50 a better map for the link micromap.

`proj4` is a character string representing a projection using the Proj4 notation. The transformation to this projection is done as the last step in the processing of the shape file before converting the boundary data into the micromapST boundary data.frame. '`proj4`' is provided, the projection in the original shape file is used. If the projection in the original shape file is missing or Long/Lat, then the function will create an Albers Equal Area protection centered on the centroid of the map's area.

`checkPointReStart`
The *BuildBorderGroup* function allows for the builder to manually adjust the shape file, just before building the *micromapST* boundary data.frame. During the normal process, the function writes check point images of the shape file, areaParms data.frame, and the Name Table data.frame to the "CheckPoint" folder in the '`NameTableDir`' folder. The builder can inspect and modify any of the tables and the shape file, but must be very careful with what and how they are modified. When done, the builder can re-issue the *BuildBorderGroup* function call with the '`checkPointReStart`' parameter set to *TRUE*. The function will bypass all of the processing up to the check point, then read in the check point files and continue building the border group. This will frequently be required when the map region contains many small area that will not be seen when shaped and simple scaling or shifting does not produce the desired arrangement of the areas.

`debug` is a numerical value from 0 to 65536. It is used by the developers to turn on specific actions or information displays to aid in the debugging and troubleshooting this function. The developers assigned a single actions to each bit in a 16 bit integer. In this way, multiple actions can be requested without any posibility of them interferring with each other. For example: the value of 512 (b'00000010 00000000') is assigned to plotting the border group map at each of the 4 stages of processing the shapefile. The four stages are: 1) RAW, just read; 2) after being process by rmapshaper; 3) after the polygons are manipulated as specified in the Name Table; and finally 4) the boundaries after they are converted into the point table format (VisBorders) for use by micromapST. The output plots are saved as PDF files. All that is needed is to set debug=512. Another action the developers created is to save the plots created by the 512 action in PNG type

files, not PDF. This can be done by setting debug = 512+128 = 640, in binary: b'00000010 00000000' OR b'00000000 10000000' = b'00000010 10000000'. The integer values are much easier to work with then the long 0s and 1s binary represention of the bits. Default is 0.

The full table of assigned values used by *debug* call parameter are:

bit #	value	definition
1	1	line by line debugging is being used and some code accomodates are required.
2	2	Outputs variable data to trace the function processes
3	4	Display Information related to projection processing
4	8	Plot intermediate Shape file, SF and SPDF (not the same plots as generated by 256 or 512)
5	16	Display processing and variables related to the SF
6	32	Display processing and variables related to the Name Table
7	64	Display other internal variables during processing of the data
8	128	= 0 sets output file type for the 512 option to PDF (default) = 1 sets output file type for the 512 option to PNG
9	254	Generate multiple plot graphics of the map using a small format similar to the image in the linked micromap with each plot with each plot having only 5 areas shaped. Number of images = Areas/5 + 1.
10	512	Generate a 4" x 4" plot of the area at key processing stage: RAW, After rmapshaping, After Name Table modifications, and after transformation and the conversion to the micromapST VisBorder format (Final).
11	1024	Same as 512, but only generates 4" x 4" plots for the RAW and Final images.
12	2048	Display the final BorderGroup map for each layer on the screen: Areas, Level 2, Regions, Map Outline (level 3). Each is display in a separate window and must be manually closed.
13	4096	Future Use - not assigned.
14	8192	Write to disk a PDF file of multiple area boundary maps with 5 areas colored in each map as done for the 256 option.
15	16384	Future Use - not assigned.
16	32768	Future Use - not assigned.

Using any of these debug options will greatly increase the size of the output generated by BuildBorderGroup and should not be used unless requested by the package developer.

Details

The following describes the process of constructing a border group dataset:

- 1. Validate all of the calling parameters provided on the function call and provide targeted error and warning messages.
- 2. Read the shape file to gather initial shape file variables. The shape file can be read prior to the call to the BuildBorderGroup function, modified, and passed as a structure using the ShapeFile call parameter.
- 3. Read the name table file and verifies all required columns are present. The Name Table data can also be pre-read and passed to BuildBorderGroup function as a data.frame .

- 4. Validate the data in the name table columns: characters vs. numeric vs. NA; duplicate valids, and range of values.
- 5. Validate the geometry in the shape file data.
- 6. Simplify the shape file geometry to provide a caricatured map with minimal vectex, but maintaining the ability to recognize the areas. This generally reduces the size of the boundary information to between 1 using the *rmapshaper* package.
- 7. Preform a union the polygons in the shape file to organize polygons by their associated area. In SP, it would be collecting all polygon data for an area as a list of polygons as one group. In sf, it is collecting all of the polygons for an area and forming a multipolygon of the data. Since we have moved entirely to sf operation, this union is preformed by the aggregate.sf features in sf.
- 8. Match the name table rows (entries) with the area polygons in the spatial structure. Once the match is established, a key is assigned for the area and set in both the name table and spatial structure. The abbreviation id is normally used for the key, but if the abbreviation is not provided, a short key is created. Abbreviation for areas are not always available.
- 9. The projection of the map makes a difference in how usable the map in making linked micromap. It was desided to no use longitute/latitude in the final resulting map. Therefore, the user has three choices: specify a non-long/lat projection on the original shape file; specify final projection parameters in the "proj4" call parameter, or let the function construct a Albers Equal Area projection about the centroid of the map with the north and south latitudes half the distance from the middle to the north and south limits of the map. If there are any map labels requested, their location points are transformed. Any needed transformation is done at this time.
- 10. The areaParms data.frame table is constructed to permit restarting the build process and to save all of the other call parameters and operation parameters needed by *micromapST*. The '`MapHdrs`', '`IDHdrs`', '`MapMinH`', '`MapMaxH`', and '`LabelCex`' variables are saved in the areaParms data.frame.
- 11. The name table, areaParms, and the 4 boundary data.frame structures are check pointed to disk for possible manual editing. If manual editing of the shape file or name table is done, they result of the edits must be saved using the original file name and the file placed in the original check point directory. The check point files will be read when the BuildBorderGroup is call with the checkPointRestart parameter set to *TRUE* the name table directory provided is the same, and the check pointed files are located. Manual editing should be done very carefully.
- 12. On a "checkPointReStart=TRUE" the *BuildBorderGroup* function reloads all of the check pointed data.frame to pick up the Border Group building process from where it left off.
- 13. On either a restart or a normal run, the function gathers the boundary information for the area boundaries, layer 2 boundaries, regional boundaries, and a map outline boundary (Layer 3).
- 14. The name table becomes the areaNamesAbbrsIDs data.frame and any working columns not needed in the final name table are deleted.
- 15. The name table to aggregate the area boundaries to form the regional boundary spatial structure, the Level 2 spatial structure, and the Lever 3 spatial structure (outline of the entire set of areas.) This is done to make sure all of the layers of boundaries correctly overlay each other.
- 16. Each set of spatial images are converted into the *micomapST* boundary data.frame structures (not sp or sf) that are compatiable with the R polygon drawing function.

- 17. The final collection data.frames for the border group are: areaParms, areaNamesAbbrsID, areaVisBorders, L2VisBorders, RegVisBorders, and L3VisBorders data.frames are written out to a single .RDA dataset file as the completed border group dataset.

Each border group is a different dataset containing the unique boundaries and operational information to allow *micromapST* to work in a different spatial area. The structure of each border group dataset is identical with the same variable names and types of structures. A user can build their own border group dataset to meet their specific spatial area needs. Because the package contains several border group datasets each one using the same data.frame names and structure, the use of lazydata or lazyloading had to be disabled. This means the R system cannot preload the datasets and have them waiting for use, they would effectively overlay each other.

The name of the border group is specified in the bordGrp call parameter. To permit a user to reference a border group dataset not contained in the package, and reside in a user's folder, the bordDir must be used to direct the package to the border group. The border group must be saved under R using the *save* function with the file extension of ".rda".
For example: `bordGrp="private", bordDir="c:/SavedBorderGroups"`

Each border group contain six (6) datasets by the same data.frame names. This allows the micromapST package the ability to quickly load a particular border group and create the requested micromaps. The six data.frames are: areaParms, areaNamesAbbrsIDs, areaVisBorders, L2VisBorders, RegVisBorders, and L3VisBorders. Since the same data.frame names are reused in each border group, the R lazyload feature is disabled in the package.

Several border group '`bordGrp`' examples are contained in the package and include the 51 states and DC of the United States, the counties of Kansas, Maryland, New York, Utah, the countries and provinces of the U.K. and China, and the U. S. Seer Registries used by the National Cancer Institute.

The output of this function is a single R dataset containing 6 data.frames and a text report for inclusion in documentation for the border group. The details on each of the 6 data.frames and their contents can be found in the bordGrp section.

The default border group is *USStatesBG* to be compatible with older R scripts using previous versions of the *micromapST* package.

Each of the border groups contained in this package have detail descriptions of the specific geographic area they represent. See the "xxxxx"BG section for each border group:

USStatesBG USSeerBG KansasBG MarylandBG NewYorkBG UtahBG UKIrelendBG ChinaBG SeoulSKoreaBG AfricaBG

Refer to the "bordGrp" section of the manual for more details on the datasets and structure of the final border group dataset.

To help document the name table in the border group produced by the BuildBorderGroup function, the function prints a summary of the location IDs for each area defined in the boundary data and the draft name table. The row.names are the keys or abbreviations of the areas in the border group. There are columns for the "Name", "Abbr", "ID", "Alt-Abbr", "Alias"

```
PUBLICATION INFORMATION FOR NAME TABLE IN BORDER GROUP :
     USstBG   2025-04-07 11:59:16

   Abbr                 Name ID Alt_Abbr                Alias
AL   AL              ALABAMA 01       AL              ALABAMA
```

AK	AK	ALASKA	02	AK	ALASKA
AZ	AZ	ARIZONA	04	AZ	ARIZONA
AR	AR	ARKANSAS	05	AR	ARKANSAS
CA	CA	CALIFORNIA	06	CA	CALIFORNIA
CO	CO	COLORADO	08	CO	COLORADO
CT	CT	CONNECTICUT	09	CT	CONNECTICUT
DE	DE	DELAWARE	10	DE	DELAWARE
DC	DC	DISTRICT OF COLUMBIA	11	DC	DISTRICT OF COLUMBIA
FL	FL	FLORIDA	12	FL	FLORIDA
GA	GA	GEORGIA	13	GA	GEORGIA
HI	HI	HAWAII	15	HI	HAWAII
ID	ID	IDAHO	16	ID	IDAHO
IL	IL	ILLINOIS	17	IL	ILLINOIS
IN	IN	INDIANA	18	IN	INDIANA
IA	IA	IOWA	19	IA	IOWA
KS	KS	KANSAS	20	KS	KANSAS
KY	KY	KENTUCKY	21	KY	KENTUCKY
LA	LA	LOUISIANA	22	LA	LOUISIANA
ME	ME	MAINE	23	ME	MAINE
MD	MD	MARYLAND	24	MD	MARYLAND
MA	MA	MASSACHUSETTS	25	MA	MASSACHUSETTS
MI	MI	MICHIGAN	26	MI	MICHIGAN
MN	MN	MINNESOTA	27	MN	MINNESOTA
MS	MS	MISSISSIPPI	28	MS	MISSISSIPPI
MO	MO	MISSOURI	29	MO	MISSOURI
MT	MT	MONTANA	30	MT	MONTANA
NE	NE	NEBRASKA	31	NE	NEBRASKA
NV	NV	NEVADA	32	NV	NEVADA
NH	NH	NEW HAMPSHIRE	33	NH	NEW HAMPSHIRE
NJ	NJ	NEW JERSEY	34	NJ	NEW JERSEY
NM	NM	NEW MEXICO	35	NM	NEW MEXICO
NY	NY	NEW YORK	36	NY	NEW YORK
NC	NC	NORTH CAROLINA	37	NC	NORTH CAROLINA
ND	ND	NORTH DAKOTA	38	ND	NORTH DAKOTA
OH	OH	OHIO	39	OH	OHIO
OK	OK	OKLAHOMA	40	OK	OKLAHOMA
OR	OR	OREGON	41	OR	OREGON
PA	PA	PENNSYLVANIA	42	PA	PENNSYLVANIA
RI	RI	RHODE ISLAND	44	RI	RHODE ISLAND
SC	SC	SOUTH CAROLINA	45	SC	SOUTH CAROLINA
SD	SD	SOUTH DAKOTA	46	SD	SOUTH DAKOTA
TN	TN	TENNESSEE	47	TN	TENNESSEE
TX	TX	TEXAS	48	TX	TEXAS
UT	UT	UTAH	49	UT	UTAH
VT	VT	VERMONT	50	VT	VERMONT
VA	VA	VIRGINIA	51	VA	VIRGINIA
WA	WA	WASHINGTON	53	WA	WASHINGTON
WV	WV	WEST VIRGINIA	54	WV	WEST VIRGINIA

```
WI   WI            WISCONSIN 55       WI             WISCONSIN
WY   WY              WYOMING 56       WY               WYOMING
```

If the border group contains Layer 2 boundary and/or regional boundary data the section of the name table contains the L2_ID, L2_ID_Name, regID, and regName values if present. Otherwise the following table is not printed.

```
Name Table Layer 2 and Regional Values

   L2_ID L2_ID_Name regID   regName
AL     2      South     2     South
AK     0   Offshore     0  Offshore
AZ     4       West     4      West
AR     2      South     2     South
CA     4       West     4      West
CO     4       West     4      West
CT     1  Northeast     1 Northeast
DE     2      South     2     South
DC     2      South     2     South
FL     2      South     2     South
GA     2      South     2     South
HI     0   Offshore     0  Offshore
ID     4       West     4      West
IL     3    Midwest     3   Midwest
IN     3    Midwest     3   Midwest
IA     3    Midwest     3   Midwest
KS     3    Midwest     3   Midwest
KY     2      South     2     South
LA     2      South     2     South
ME     1  Northeast     1 Northeast
MD     2      South     2     South
MA     1  Northeast     1 Northeast
MI     3    Midwest     3   Midwest
MN     3    Midwest     3   Midwest
MS     2      South     2     South
MO     3    Midwest     3   Midwest
MT     4       West     4      West
NE     3    Midwest     3   Midwest
NV     4       West     4      West
NH     1  Northeast     1 Northeast
NJ     1  Northeast     1 Northeast
NM     4       West     4      West
NY     1  Northeast     1 Northeast
NC     2      South     2     South
ND     3    Midwest     3   Midwest
OH     3    Midwest     3   Midwest
OK     2      South     2     South
OR     4       West     4      West
```

```
PA     1  Northeast     1 Northeast
RI     1  Northeast     1 Northeast
SC     2      South     2     South
SD     3    Midwest     3   Midwest
TN     2      South     2     South
TX     2      South     2     South
UT     4       West     4      West
VT     1  Northeast     1 Northeast
VA     2      South     2     South
WA     4       West     4      West
WV     2      South     2     South
WI     3    Midwest     3   Midwest
WY     4       West     4      West
```

There is one row per area in the border group.

If the name table contained any modifying information (MapL, MapX, MapY, Xoffset, Yoffset, Scale, or Rotate), then one additional section of the name table is printed.

```
Name Table Modifications and Map Label Values

   Xoffset Yoffset Scale Rotate MapL     MapX         MapY
AL   0.000    0.00    NA     NA <NA>       NA           NA
AK  36.500  -36.00   0.3      0   AK -1505744 -1306750.302
AZ   0.000    0.00    NA     NA <NA>       NA           NA
AR   0.000    0.00    NA     NA <NA>       NA           NA
CA   0.000    0.00    NA     NA <NA>       NA           NA
CO   0.000    0.00    NA     NA <NA>       NA           NA
CT  -0.075   -0.09   1.2      0 <NA>       NA           NA
DE   0.000    0.00    NA     NA <NA>       NA           NA
DC   3.750   -2.15   4.0      0   DC  2503328    -6425.072
FL   0.000    0.00    NA     NA <NA>       NA           NA
GA   0.000    0.00    NA     NA <NA>       NA           NA
HI  52.000    6.00   1.0      0   HI  -139296 -1481684.362
ID   0.000    0.00    NA     NA <NA>       NA           NA
IL   0.000    0.00    NA     NA <NA>       NA           NA
IN   0.000    0.00    NA     NA <NA>       NA           NA
IA   0.000    0.00    NA     NA <NA>       NA           NA
KS   0.000    0.00    NA     NA <NA>       NA           NA
KY   0.000    0.00    NA     NA <NA>       NA           NA
LA   0.000    0.00    NA     NA <NA>       NA           NA
ME   0.000    0.00    NA     NA <NA>       NA           NA
MD   0.000    0.00    NA     NA <NA>       NA           NA
MA   0.000    0.00    NA     NA <NA>       NA           NA
MI   0.000    0.00    NA     NA <NA>       NA           NA
MN   0.000    0.00    NA     NA <NA>       NA           NA
MS   0.000    0.00    NA     NA <NA>       NA           NA
MO   0.000    0.00    NA     NA <NA>       NA           NA
```

```
MT   0.000    0.00    NA     NA <NA>       NA            NA
NE   0.000    0.00    NA     NA <NA>       NA            NA
NV   0.000    0.00    NA     NA <NA>       NA            NA
NH   0.000    0.00    NA     NA <NA>       NA            NA
NJ   0.000    0.00    NA     NA <NA>       NA            NA
NM   0.000    0.00    NA     NA <NA>       NA            NA
NY   0.000    0.00    NA     NA <NA>       NA            NA
NC   0.000    0.00    NA     NA <NA>       NA            NA
ND   0.000    0.00    NA     NA <NA>       NA            NA
OH   0.000    0.00    NA     NA <NA>       NA            NA
OK   0.000    0.00    NA     NA <NA>       NA            NA
OR   0.000    0.00    NA     NA <NA>       NA            NA
PA   0.000    0.00    NA     NA <NA>       NA            NA
RI   0.000   -0.09   1.4      0 <NA>       NA            NA
SC   0.000    0.00    NA     NA <NA>       NA            NA
SD   0.000    0.00    NA     NA <NA>       NA            NA
TN   0.000    0.00    NA     NA <NA>       NA            NA
TX   0.000    0.00    NA     NA <NA>       NA            NA
UT   0.000    0.00    NA     NA <NA>       NA            NA
VT   0.000    0.00    NA     NA <NA>       NA            NA
VA   0.000    0.00    NA     NA <NA>       NA            NA
WA   0.000    0.00    NA     NA <NA>       NA            NA
WV   0.000    0.00    NA     NA <NA>       NA            NA
WI   0.000    0.00    NA     NA <NA>       NA            NA
WY   0.000    0.00    NA     NA <NA>       NA            NA
```

The text is written to a file named after the border group name with "_rpt.txt" appended. For Example for the MarylandBG border group, the file written by the BuildBorderGroup functions is "MarylandBG_rpt.txt".

These tables would be included in any documentation to pass along to the users of the new border group.

Value

Path to the saved Border Group file.

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST, micromapSEER,

Examples

```
# Load libraries needed.
stt1 <- Sys.time()
```

```
library(stringr)
library(readxl)
library(sf)

# Generate a Kentucky County Border Group
#
# Read the county boundary files.  (Set up system directories.
#   Replace with your directories to run.)
TempD<-"c:/projects/statnet/"  # my private test PDF directory exist,
                               #don't use temp.
# get a temp directory for the output PDF files for the example.
if (!dir.exists(TempD)) {
     TempD <- paste0(tempdir(),"/")
     DataD <- paste0(system.file("extdata",package="micromapST"),"/")
} else {
     DataD <- "c:/projects/statnet/r code/micromapST-3.1.1/inst/extdata/"
}

cat("Temporary Directory:",TempD,"\n")
# get working data directory
#cat("Working Data Directory:",DataD,"\n")

KYCoBG  <- "KYCountyBG"  # Border Group name
KYCoCen <- "KY_County"   # shape file name(s)

KYCoShp <- st_read(DataD,KYCoCen)
st_crs(KYCoShp)  <- st_crs("+proj=lonlat +datum=NAD83 +ellipse=WGS84 +no_defs")

# inspect name table
KYNTname <- paste0(DataD,"/",KYCoCen,"_NameTable.xlsx")
#cat("KYNTname:",KYNTname,"\n")

KYCoNT    <- as.data.frame(read_xlsx(KYNTname))
#head(KYCoNT)
spt1 <- Sys.time()
cat("Time to get data and boundaries for Counties:",spt1-stt1,"\n")
## Not run:
#
#  building border group for all counties in Kentucky
#
stt2 <- Sys.time()
# Build Border Group
BuildBorderGroup(ShapeFile     = KYCoShp,
                 ShapeLinkName = "NAME",
                 NameTableLink = "Name",
                 NameTableDir  = DataD,
                 NameTableFile = paste0(KYCoCen,"_NameTable.xlsx"),
                 BorderGroupName = KYCoBG,
                 BorderGroupDir  = TempD,
                 MapHdr        = c("","KY Counties"),
                 IDHdr         = c("KY Co."),
                 ReducePC      = 0.9
                )
```

```
# Setup MicromapST graphic
spt2 <- Sys.time()
cat("Time to build KY Co BG:",spt2-stt2,"\n")
stt3 <- spt2
KYCoData  <- as.data.frame(read_xlsx(paste0(DataD,"/",
                           "KY_County_Population_1900-2020.xlsx")))
#head(KYCoData)

KY_Co_PD <- data.frame(stringsAsFactors=FALSE,
                 type=c("map","id","dot","dot"),
                 lab1=c(NA,NA,"2010 Pop","2020 Pop"),
                 col1=c(NA,NA,"2010","2020")
              )

KYCoTitle  <- c("Ez23ax-Kentucky County","Pop 2010 and 2020")
OutCoPDF   <- paste0(TempD,"Ez23ax-KY Co 2010-2020 Pop.pdf")
grDevices::pdf(OutCoPDF,width=10,height=13)   # on 11 x 14 paper.

micromapST(KYCoData,KY_Co_PD,sortVar=c("2020"), ascend=FALSE,
           rowNames="full", rowNamesCol = c("Name"),
           bordDir = TempD, bordGrp = KYCoBG,
           title   = KYCoTitle
          )

x <- dev.off()
spt3 <- Sys.time()
cat("Time to micromapST KY Co graph:",spt3-stt3,"\n")

## End(Not run)   # end of dontrun.

stt4 <- Sys.time()

# Aggregate Kentucky Counties into ADD areas
#
# The regions in the Kentucky County Name Table (KYCoNT) are the ADD districts
# the county was assigned to.
# The KYCoShp has the county boundaries.
#
KYCoShp$NAME   <- str_to_upper(KYCoShp$NAME)
KYCoNT$NameCap <- str_to_upper(KYCoNT$Name)

aggInx <- match(KYCoShp$NAME,KYCoNT$NameCap)
#print(aggInx)

xm     <- is.na(aggInx)  # which polygons did not match the name table?
if (any(xm)) {
  cat("ERROR: One or more polygons/counties in the shape file did not match\n",
      "the entries in the KY County name table. They are:\n")
      LLMiss <- KYCoNT[xm,"Name"]
      print(LLMiss)
      stop()
}
```

```
#

#####
#  aggFUN - a function to inspect the data.frame columns and determine
#    an appropriate aggregation method - copy or sum.
#
aggFUN <- function(z) { ifelse (is.character(z[1]), z[1], sum(as.numeric(z))) }
#
#####

#
aggList  <- KYCoNT$regID[aggInx]
#print(aggList)

KYADDShp <- aggregate(KYCoShp, by=list(aggList), FUN = aggFUN)
names(KYADDShp)[1] <- "regID"  # change first column name to "regNames"
row.names(KYADDShp) <- KYADDShp$regID

KeepAttr <- c("regID","AREA","PERIMETER","STATE","geometry")
KYADDShp <- KYADDShp[,KeepAttr]
st_geometry(KYADDShp) <- st_cast(st_geometry(KYADDShp),"MULTIPOLYGON")

#plot(st_geometry(KYADDShp))
spt4 <- Sys.time()
cat("Time to aggregate KY ADDs from Cos:",spt4-stt4,"\n")
stt5 <- spt4
# Build Border Group

BuildBorderGroup(ShapeFile       = KYADDShp,
        # sf structure of shapefile of combined counties into AD Districts
                 ShapeLinkName   = "regID",
                 NameTableFile   = "KY_ADD_NameTable.xlsx",
                 NameTableDir    = DataD,
                 NameTableLink   = "Index",
                 BorderGroupName = "KYADDBG",
                 BorderGroupDir  = TempD,
                 MapHdr          = c("","KY ADDs"),
                 IDHdr           = c("KY ADDs"),
                 ReducePC        = 0.9
                )

spt5 <- Sys.time()
cat("Time to build ADD BG:",spt5-stt5,"\n")
stt6 <- spt5
# Test micromapST
KYADDData <- as.data.frame(readxl::read_xlsx(
                            paste0(DataD,"KY_ADD_Population-2020.xlsx")),
                           stringsAsFactors=FALSE)
#
KY_ADD_PD <- data.frame(stringsAsFactors=FALSE,
                 type=c("map","id","dot","dot"),
                 lab1=c(NA,NA,"Pop","Proj. Pop"),
                 lab2=c(NA,NA,"2020","2030"),
```

```
                  col1=c(NA,NA,"DecC2020","Proj2030")
               )
#
KyTitle <- c("Ez23cx-KY Area Development Dist.",
             "Pop 2020 and proj Pop 2023")
OutPDF2 <- paste0(TempD,"Ez23cx-KY ADD Pop.pdf")

grDevices::pdf(OutPDF2,width=10,height=7.5)

micromapST(KYADDData,KY_ADD_PD,sortVar="DecC2020",ascend=FALSE,
           rowNames= "full", rowNamesCol = "ADD_Name",
           bordDir = TempD,
           bordGrp = "KYADDBG",
           title   = KyTitle
          )
x <- grDevices::dev.off()
spt6 <- Sys.time()
cat("Time to do micromapST of KY ADDs:",spt6-stt6,"\n")
```

ChinaBG *ChinaBG border group datasets to support creating micromaps for the 32 providences and municipalities in the country of Republic of China*

Description

The *micromapST* function has the ability to generate linked micromaps for any geographical area. To specify the geographical area, the 'bordGrp' call argument is used to specify the border group dataset for the geographical area. The *ChinaBG* border group dataset supports creating linked micromaps for the 34 provinces, special administrative regions, metropolitan areas in the China. When the 'bordGrp' call argument is set to *ChinaBG*, the appropriate name table (sub area names and abbreviations) and the 34 sub-areas (provinces, SAR, cities, etc.) boundary data is loaded in *micromapST*. The user's data is then linked to the boundary data via the county's name, abbreviated name or ID based on the table below.

Usage

```
data(ChinaBG)
```

Details

The *ChinaBG* border group dataset contains the following data.frames:

areaParms - contains specific parameters for the border group

areaNamesAbbrsIDs - containing the names, abbreviations, and numerical identifier for the providences and municipalities of China.

areaVisBorders - the boundary point lists for each area in China.

L2VisBorders - the boundaries for an intermediate level. This level is not used in this border group.

RegVisBorders - the boundaries for an regional level for China. This set of boundaries are used in conjunction with the regions call parameter.

L3VisBorders - the boundary of the country of China.

For the China border group, there are 34 county sub-areas listed in the areaNamesAbbrsIDs and areaVisBorders datasets. The *L2VisBorders* dataset is not used and is set to the *L3VisBorders* dataset as a placeholder. The *RegVisBorders* dataset represents the 6 regions of China in the *ChinaBG* border group. The L3VisBorders dataset contains the outline of the country of China.

The details on each of these data.frame structures can be found in the "bordGrp" section of this document. The *areaNamesAbbrsIDs* data.frame provides the linkages to the boundary data for each sub-area using the fullname, abbreviation, alternate abbreviation, and numerical identifier for each country to the *<statsDFrame>* data based on the setting of the '`rowNames`' call argument. A column or the data.frame row.names must match one of the types of names in the *areaNamesAbbrsIDs* data.frame name table. If the data row does not match a value in the name table, an warning is issued and the data is ignored. If no data is present for a sub-area (county) in the name table, the sub-area is mapped but not colored.

The following are a list of the names, abbreviations, alternate abbreviations, ids, and the region for each county, province or metro area in the *ChinaBG* border group.

name	ab	alt_ab	id	region
Anhui	AH	CN.AH	34	Huadong
Beijing	BJ	CN.BJ	11	Huabei
Chongqing	CQ	CN.CQ	50	Xinan
Fujian	FJ	CN.FJ	35	Huadong
Gansu	GS	CN.GS	62	Xibei
Guangdong	GD	CN.GD	44	Zhongnan
Guangxi	GX	CN.GX	45	Zhongnan
Guizhou	GZ	CN.GZ	52	Xinan
Hainan	HI	CN.HA	46	Zhongnan
Hebei	HE	CN.HB	13	Huabei
Heilongjiang	HL	CN.HL	23	Dongbei
Henan	HA	CN.HE	41	Zhongnan
Hubei	HB	CN.HU	42	Zhongnan
Hunan	HN	CN.HN	43	Zhongnan
Jiangsu	JS	CN.JS	32	Huadong
Jiangxi	JX	CN.JX	36	Huadong
Jilin	JL	CN.JL	22	Dongbei
Liaoning	LN	CN.LN	21	Dongbei
Nei Mongol	NM	CN.NM	15	Huadong
Ningxia Hui	NX	CN.NX	64	Xibei
Qinghai	QH	CN.QH	63	Xibei
Shaanxi	SN	CN.SA	61	Xibei
Shandong	SD	CN.SD	37	Huadong
Shanghai	SH	CN.SH	31	Huadong
Shanxi	SX	CN.SX	14	Huabei
Sichuan	SC	CN.SC	51	Xinan
Tianjin	TJ	CN.TJ	12	Huabei
Xinjiang Uygur	XJ	CN.XJ	65	Xibei

Xizang	XZ	CN.XZ	54	Xinan
Yunnan	YN	CN.YN	53	Xinan
Zhejiang	ZJ	CN.ZJ	33	Buadong
Hong Kong	HK	CN.HK	81	Zhongnan
Macao	MC	CN.MC	82	Zhongnan
Taiwan	TW	CN.TW	71	Huadong

The *ChinaBG* supports two sets of abbreviations for each county (province or metro area). No consistant source was found when the border group was originally created. Therefore, the two most common sets are included. The first abbreviation set can be referenced by setting the ‘`rowNames`’ call argument to "ab". The second (alternate) abbreviation set can be used by setting the ‘`rowNames`’ to "alt_ab".

The ‘`rowNames`’ = "alias" features are not supported in the *ChinaBG* border group.

`ClnStrName_MST` *A function to consistantly clean the character strings used for Location IDs*

Description

The *ClnStrName_MST* function is used to clean the location ID character string used as the "Name", "Abbr", "ID", and "Alt-Abbr" location ID columns in the name table in a border group. The packages automatically clean these string to make sure match are successful as much as possible. The function converts any punctuation or control characters to blanks. It also converts any of the following characters "+=$^|<>~‘" and will not convert "-" or "_" characters. The converted string is then squished to remove leading and trailing blanks and multiple groups of internal blanks to a single blank.

If a user wants to use to border group name table to convert their own location IDs, it is recommended passing the original Loc ID list through the ClnStrName_MST function to match sure their list will match the "Name", "Abbr", "ID", or "Alt-Abbr" columns in the border group location ID columns. Otherwise, most of the list may not match.

Usage

```
ClnStrName_MST(x)
```

Arguments

`x`	- a vector of one or more character string variables

Details

See description

Value

The function returns a vector of character strings the same size as the original vector. Each entry has been edited to remove unwanted characters.

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST BuildBorderGroup

---

cnPopData *Test data for the China border Group*

Description

This dataset contains the 2014 population and average income per person in each of the China Areas

Usage

```
data(cnPopData)
```

Format

A data frame with 18 observations, 1 for each Seer area, on the following 12 variables.

area a character vector containing the China area full names.

pop2013 a numeric vector of the number of the county's population

Details

This dataset was pulled from the China ??? government website on Dec. 2014. It contains the population and average income per person for each of Kansas' 105 counties.

`detailsVariables` *Validation and Translation table for details variables*

Description

An internal table containing a list of the details variables and parameters to do validation of user provided variable overrides of the defaults. The table also contains the information needed to translate the existing (as of 9/17/2015) details structure into a new details structure segmented by each type of glyphic.

Usage

```
data(detailsVariables)
```

Format

A data frame with 6 variables:

varName a character string of the exact details variable name

method a character string describing the type of test required to validate the variable. The supported tests are: colors, integer, numeric, logical, and vector3

v1 first variable for the "test" for this variable.

v2 second variable (if needed) for the "test" for this variable.

usedBy a vector of character string listing the glyphics that use this variable.
For Example: c('ts','tsconf')

newVarName the new variable name to be used within a glyphic. The glyphic name and variable name must be unique. This eliminates having to include the glyphic name in the variable name.

d_range defines the range of the dependency for the dependent relationship.

dependent indicates this variable is dependent on another variable and it's name.

default the default values or the variable.

comments operational comments on the variable.

Details

This dataset provide a table to *micromapST* for verification of the details variables provided by a user to override the packages default values. The *varName* is the exact name of the variable. If the variable name provide by the user does not match this list, it is flagged as an error and ignored. The test supported are: colors, integer, numeric, logical and vector3. The colors test calls the `is.Color` function. The integer and numeric tests use the range provided in *v1* and *v2* columns to check the range of the value for the variable. The logical test verifies the value is TRUE or FALSE. The vector3 test check to make sure the value is a vector or length 3 and each value is within the range in *v1* to *v2*.

The micromapST package is being updated to use a new variable structure. The new structure will allow options/parameters to be specified on glyphic and column basis. This allows each column to

be uniquely tuned to the user requirements. The variable names have also been simplifed to use the same name across glyphics when the purpose is the same. This should reduce the users learning curve.

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

`Educ8thData` *Education Survey of 8th Grade Proficiency in Math*

Description

Math Proficiency Survey Results for 8th Graders in 2011 by State

Usage

```
data(Educ8thData)
```

Format

A data frame with 51 observations (one per state) on the following 7 variables.

`StAbbrev` a character string representing the 2 character state Id for this row.

`State` a character string factor of the state full name.

`avgscore` a numeric vector of average proficiency score for each state

`PctBelowBasic` a numeric vector of percentage of students with below basic scores

`PctAtBasic` a numeric vector of percentage of students at basic proficiency.

`PctProficient` a numeric vector of percentage of students at the proficient level

`PctAdvanced` a numeric vector of percentage of students scoring at the advanced level.

Details

The dataset contains 51 records, one for each state/area. The data represents the percentage of 8th grade students in 2011 in that state who tested at each proficiency level in math: less than basic, basic, proficient and advanced. The row name is the state abbreviation - 2 characters. This dataset is used by the *micromapSEER* examples using the "USStatesDF" border group.

Source

The National Center for Education Statistics, Department of Education
http://nces.ed.gov/nationsreportcard/states/

`glyph-arrow` *The arrow glyph creates a graphic of a two data points, from a beginning point to an end point.*

Description

The *arrow* glyph creates a simple arrow with pointing heads from a beginning point (col1 value) to the ending point (col2 value). If the beginning and ending point values are the same, a solid dot will be used to represent the two points. The point value is provided through the col1 and col2 in *panelDesc* data.frame providing named/numbers of columns in the *statsDFrame* user provided data structure. Only the points related to the area rows associated to the group/row are used to create the arrow for that area. If the median consist of only a single area, the single arrow is plotted in the group/row. The *panelDesc* variables col1 or col2 provide the name/number of the data columns in the *statsDFrame* data.frame supplied by the user.

Details

The col1 snd col2 *panelDesc* variable provide the name/number of the column in the *statsDFrame* containing data values used to plot the beginning and end points of each arrow. The beginning value may be higher than the ending point to identify arrow going down between the col1 and col2 time points.

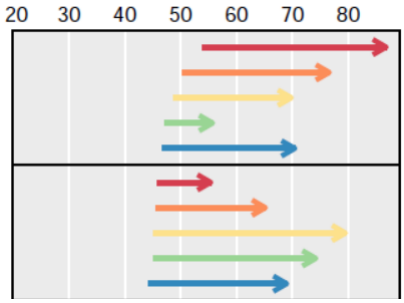
The user can specify the lab1, lab2, lab3, lab4, refval, and reftxt information/values in the appropriate column *j* related to the glyph in the *panelDesc* provide additional information on how to create the glyph graphic column. The information is used to create the column headers, trailers, and reference information.

An X-axis is drawn above and below the column based on the data provided in the col1 an col2 columns in the *statsDFrame* provided by the user.

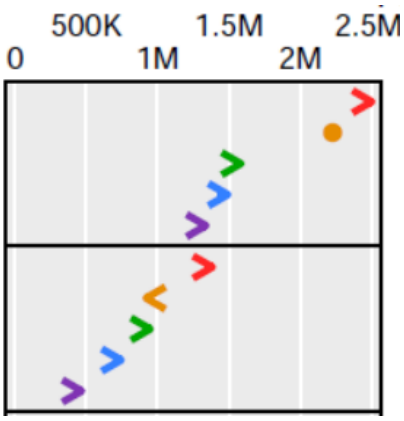
For the arrow glyph, the col1 and col2 column names/numbers must be supplied.

panelDesc variable rows not used are the col3, panelData, and parm variable rows. If these rows are present the value for the *arrow* glyph should be set to NA.

The following is an example of a section of a boxplot glyph column in a linked micromap:



If both the start and end points of an arrow are the same, a "dot" is draw. If the begin and end points of an arrow are very close to each other, the arrow may not be much more than an arrow head. See example below.



The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-bar` *The bar glyph creates a graphic of a single data point as the height of the bar.*

Description

The *bar* glyph creates a simple bar with the starting point for the bar being zero. The value is defined by the col1 column name/number in the *statsDFrame* data.frame provided by the user. The value may be a positive or negative number. The glyph determines the range of the values and sets the width of the graphic box accordingly. The zero point reference will be the same for all bars drawn for the glyphic. If the median consist of only a single area and bar, the single bar will be plotted in the group/row. The col1 data must be numeric values and can be used to sort the order of the areas using the sortVar call parameter.

Details

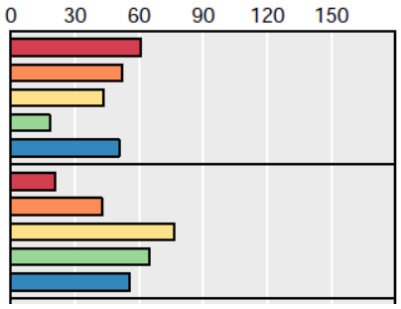
The col1 *panelDesc* parameter is indexed by *j* to obtain the data column's name/number of the data in the *statsDFrame* data.frame to use for the height of the bar. The bar is draw horizontally with negative values on the left and positive values on the right. *j* is used to index into *panelDesc* to get

the user specified lab1, lab2, lab3, lab4, refval, and reftxt information to create the column headers, trailers, and reference information. An X-axis is drawn above and below the column based on the data provided in the col1 column of the *statsDFrame* by the user.

For the bar glyph, the col1 value of the column name/numbers in the *statsDFrame* data.frame must be provided. The data values must be numeric.

panelDesc rows not used by a glyph should be set to NA if not used. The bar glyph does not use the col2, col3, panelData, and parm rows.

The following is an example of a section of a boxplot glyph column in a linked micromap:



The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-boxplot` *The boxplot glyph creates a graphic of a scatter dot plot of x and y coordinate points.*

Description

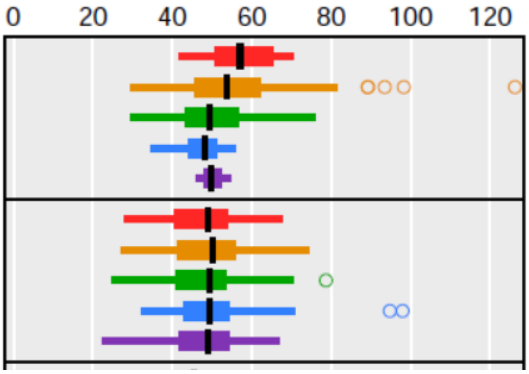
The *boxplot* glyph draws the boxplot for each of the areas being presented in the group/row. This can be up to 5 areas (boxplots) at a time. The color scheme of each boxplot matches the color assigned to the area being presented in this graphic row. Each boxplot is redrawn from the attribute data provided in the boxplot data.frame passed to the glyph code via the *panelDesc panelData* variable by name. The boxplot data.frame is the output of the standard R boxplot function when plot= parameter is set FALSE. This produces a data.frame of the information needed to draw the boxplots and outliers. The name of the data.frame is passed to the boxplot glyph via the *panelData* vector in the panelDesc data.frame used to describe the requested linked micromap.

Details

The boxplot data.frame created by calling R's boxplot function with plot=FALSE contain a row of boxplot attributes for each set of data passed to it originally. In this case it would be the data for each area to be analyzed by micromapST. The data.frame is passed by name using the '`panelData`' row in the *panelDesc* data.frame.

For the boxplot glyph, doesnot use the col1, col2, col3, refval or reftxt rows of the *panelDesc* data.frame.

The following is an example of a section of a boxplot glyph column in a linked micromap:



The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-ctrbar` *The ctrbar glyph creates a centered stacked bar diagram centered about the value of zero.*

Description

The *ctrbar* glyph creates a stacked bar diagram, where the centered segment stacked bar (ctrbar) glyph displays the bars in the order of the data in the statsDFrame data.frame. Each row is assumed to have a total value of 100 or at least equal values of all of the segments. Each row represents the same number of segment with the same total value or proprotion of participants in a study being examined. The *ctrbar* identifies the center of the set of bars, either between two bars or the middle of

an odd number of bars. The glyph lines up the middle of the segment or the boundary between two segments to allow comparison of the data between each row (or area). The values of each boundary is provided in a column in the statsDFrame data.frame. In panelDesc, the name or number of the FIRST data column is provided in the col1 column. The last column in the series is specified in the col2 column. The values may be specified as statsDRrame column names or numbers.

Details

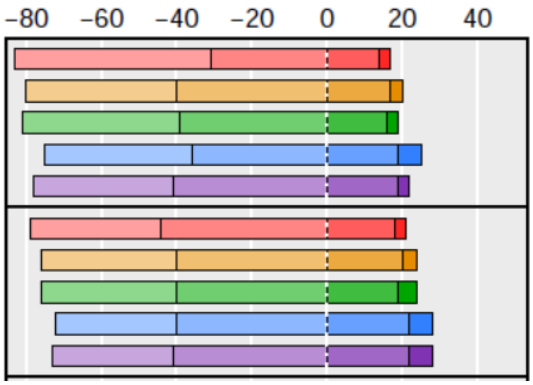
The glyph is called with a *j* identifying the graphic column number being drawn. *j* is used by all of the panel functions to scale, outline, and select the area within the total graphic page to draw a column of the ctrbar glyphs, one per group/row. It is also used to reference the panelDesc column to determine what type of glyph to draw, what variables (via col1, col2, and col3 rows), what labels to be used for the header and trailer titles when drawing the graphic. The graphic is a simple bar plot of the data points provide to the ctrbar routines.

The data must be sorted from lowest to highest values.

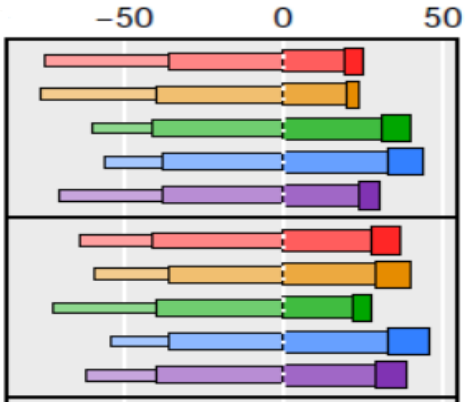
For the ctrbar glyph, the col1 and col2 column names/numbers must be supplied and represent the first and last data columns in the *statsDFrame* data.frame.

In the *details* data.frames there is a *CBar.varht* (a logical variable) that is used to indicate to the gryphic whether to use variable height bars in the gryphic or not. By default the value is set to *FALSE*. When set to *TRUE*, all Center Segmented Stacked Bar Charts will have their bars vary in height from the left to the right.

The following is an example of a section of a Centered Stacked Bar glyph column in a linked micromap:



If variable height stacked bars are used, then the glyph graphics will look like:



The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-dot` *The dot glyph create a graphic of a single data point as a dot on the graphic between the lowest and highest values.*

Description

The *dot* glyph creates a simple dot at the point on the graphic at the specified data point from the column name or number provided in the *col1* row in *panelDest* data.frame in the *statsDFrame* data.frame. The range of the data provided is used to set the range of the X-Axis and the dot is plotted in line with the associated area's color in the group/row. If the median consist of only a single area and dot, only one area row is plotted. The col1 data must be numeric values and can be used to sort the order of the areas using the *sortVar* call parameter.

Details

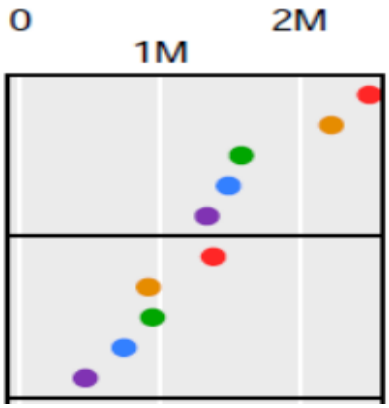
The col1 variable for the glyph column in *panelDesc* parameter links to the data column containing the dot value the *statsDFrame* data.frame. *j* is used to index into *panelDesc* to get the user specified *lab1*, *lab2*, *lab3*, *lab4*, *refval*, and *reftxt* information to create the column headers, trailers, and reference information. An X-axis is drawn above and below the column based on the data provided in the *col1* column of the *statsDFrame* by the user.

For the dot glyph, the *col1* value of the column name/numbers in the *statsDFrame* data.frame must be provided. The data values must be numeric.

This glyph routine creates both the "dot" and "dotsignif" graphs. If the *dsignif* call parameter is set to FALSE, the function will not check for *col2* and associated data in the *statsDFrame* for *pvalues* to check the significance of the dot value data. If TRUE, the function creates the dotsignif glyph creation.

panelDesc rows not used by the glyph should be set to NA. The "dot" glyph does not use the*col2*, *col3*, *panelData*, and *parm* rows in the *panelDesc* data.frame.

The following is an example of a section of a boxplot glyph column in a linked micromap:



The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-dotconf` *The dotconf glyph creates a graphic of a single data point as a dot with a line through the dot from the low to high confidence limits.*

Description

The *dotconf* glyph is an extension to the dot glyph graphic. The *dotconf* glyph uses two additional data columns in the *statsDFrame* user data data.frame to provide the low (col2) and high (col3) confidence interval values. col1 still represent the value of the dot's position in the graph. A line is drawn through the dot from the low confidence limit (provided by col2) to the highest confidence limit (provided by col3). The data values obtained from the the column names/numbers in col1, col2, and col3 variables in the *panelDesc* data.frame point to the columns containing the data in the *statsDFrame* data.frame provided by the user. All of the data must be numberic. The lowest confidence limit must be less than the data value for the dot position and the highest confidence limit must be greater than the data value for the dot. The range of the data for all three data columns is used to define the X-Axis range for the graphic column. The dot and the limits are drawn on the same row associated with the color used for the area it represents. The dot will be filled with the color of the related area in the geographic map. If the median consist of only a single area, only one area row and graphic will be plotted. The col1, col2, and col3 column in the *statsDFrame* data.frame must be numeric values and can be used to sort the order of the areas using the sortVar call parameter.

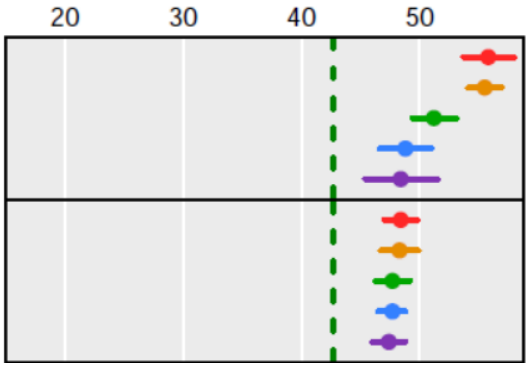
Details

The col1 *panelDesc* parameter is indexed by *j* to obtain the data column's names/numbers of the col1, col2, and col3 data in the *statsDFrame* data.frame to plot the dot and confidence interval. The col1 referenced data column is the value of the dot, the col2 referenced data is the low interval, and the col3 referenced data is the high interval. *j* is used to index into *panelDesc* to get the user specified lab1, lab2, lab3, lab4, refval, and reftxt information to create the column headers, trailers, and reference information. An X-axis is drawn above and below the column based on the data provided in the col1, col2, and col3 referenced column of data in the *statsDFrame* by the user.

For the dotconf glyph, the col1, col2, and col3 column name/numbers refereninc the *statsDFrame* data.frame must be provided and valid. All data in these columns must be numeric.

panelDesc rows not used by the glyph should be set to NA. The dotconf glyph does not use the panelData, and parm rows.

The following is an example of a section of a boxplot glyph column in a linked micromap:



The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-dotse` *The dotse glyph creates a graphic of a single data point and a line representing the standard error of the data.*

Description

The *dotse* glyph is also based on the *dat* gryph. In the *dotse* graphic, *col2* in the *panelDesc* data.frame provides the standard error value for the row. Using the standard error value, *dotse* creates a dot at the data point specified by col1 and a line through the dot representing the standard error associated with the dot's data. Value of the dot and it's standard error value is provided in the *statsDFrame* data columns referenced by the column name/number is *panelDesc* col1 and col2 rows. The upper and lower values are calculated based on

```
zval <- stats::qnorm( .5 + Dot.SE / 200 )  # where Dot.SE is defaulted to 95
inc  <- zval * col2 referenced data
upper <- col1 referenced data + inc
lower <- col1 referenced data - inc
```

The *dotse* is plotted at the col1 data value from *statsDFrame*. The line is drawn from the lower to upper calculated values based on the significant error values provided in the *statsDFrame* column referenced by the *panelDesc* col2 row. The range of the X-Axis of the graphic is based on the values of all of the data points to be graphed from all of the area. All of the data must be numberic. The dot will be filled with the color of the related area in the geographic map. If the median consist of only a single area, only one area row and graphic will be plotted. The data in the referred *statsDFrame* by col1 and col2 must be numeric values and can be used to sort the order of the areas using the sortVar call parameter.

Details

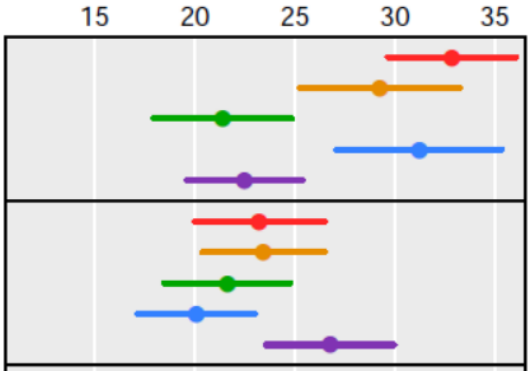
The col1 *panelDesc* parameter is indexed by *j* to obtain the data column's names/numbers of the col1 and col2 referred columns in the *statsDFrame* data.frame to plot the dot, calculate the standard error lower and upper values. The col1 referenced data column is the value of the dot, the col2 referenced data is the low interval, and the col3 referenced data is the high interval. *j* is used to index into *panelDesc* to get the user specified lab1, lab2, lab3, lab4, refval, and reftxt information to create the column headers, trailers, and reference information. An X-axis is drawn above and

below the column based on the data provided in the col1 and col2 referenced column of data in the *statsDFrame* by the user.

For the dotse glyph, the col1 and col2 column name/numbers refereninc the *statsDFrame* data.frame must be provided and valid. All data in these columns must be numeric.

panelDesc rows not used by the glyph should be set to NA. The dotse glyph does not use the *col3*, *panelData*, and *parm* rows.

The following is an example of a section of a boxplot glyph column in a linked micromap:



The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-dotsignif` *The dotsignif glyph creates a graphic of a single data point as a dot and overlay the dot with a "X" if the data value is significant.*

Description

The *dotsignif* glyph is the same as the *dot* glyphic, but adds the option of testing for significants and overlaying the dot with an "X" the dot's value is. The dot is placed using the data in column name/number in the *statsDFrame* user provided data. The data in *col2* is tested to determine if the dot's value is significant. The comparison is done against the standard pvalue of 0.05. This value can be changed in the *details* call parmater. The value of the dot is in the column named/numbers in the *panelDesc* data.frame's *col1* in the *statsDFrame* data.frame. The dot is considered significant

if the value provided in *statsDFrame* column referenced by the column name/number provided in the *col2* value is greater than the pvalue variable. The default pvalue is 0.05. If the *col2* referenced data is > then the reference pvalue (0.05), the an "X" is overploted on the dot. The range of the data provided is used to set the range of the X-Axis. The dot is plotted in line with the associated area's color in the group/row. If the median consist of only a single area and dot, only one area row is plotted. The *col1* and *col2* column names/numbers must be provided and the data in the *statsDFrame* must be numeric values. The statsDFrame columns can be used to sort the order of the areas using the *sortVar* call parameter.

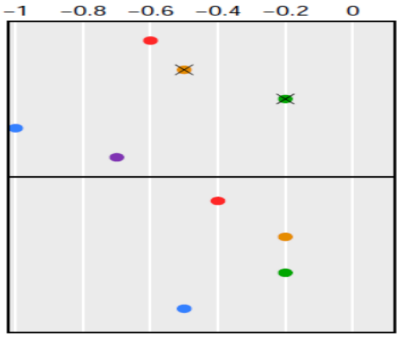
Details

The *col1* row in the *panelDesc* parameter is indexed by *j* to obtain the data column's name/number of the data in the *statsDFrame* data.frame to plot the dot. The data in the *statsDFrame* referred to by *col2* is compared with the set pvalue to determine if the value of the dot is significant. If it is, an "X" is ploted over the dot to indicate significance. *j* is used to index into *panelDesc* to get the user specified *lab1*, *lab2*, *lab3*, *lab4*, *refval*, and *reftxt* information to create the column headers, trailers, and reference information. An X-axis is drawn above and below the column based on the data provided in the col1 column of the *statsDFrame* by the user.

For the dotsignif glyph, the *col1* and *col2* values of column name/numbers must referrence columns in the *statsDFrame* data.frame and the data in these columns must be numeric.

panelDesc rows not used by the glyph should be set to NA. The dotsignif glyph does not use the *col3*, *panelData*, and *parm* rows.

The following is an example of a section of a boxplot glyph column in a linked micromap:



The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-id` *The id glyph creates a graphic of a names or abbreviations of each area in the map.*

Description

The *id* glyph creates a simple column containing the ID using the Abbreviation or Full name of each area in the Border Group dataset. The call parameter '`plotname`' tells micromapST which label to use in the ID glyph column. The label and the color representing the area are place in alignment with the statistical graphics in the group/row.

The only *panelDesc* rows (variables) used by the ID glyph are: *lab3* and *lab4* values from the *panelDesc* data.frame. All of the other rows are ignored and the values for this glyph should be set to "NA".

Details

For the id glyph, creates a location id label in the color associated with the area being presented in the statistical graphics in neigbhooring columns. The label can be the Full Name or the Abbreviation assigned to the area in the border group dataset.

The following is an example of a section of a id glyph column in a linked micromap:

The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-mapxxxx` *The collection of map glyphs creates a graphic of a micromap plots.*

Description

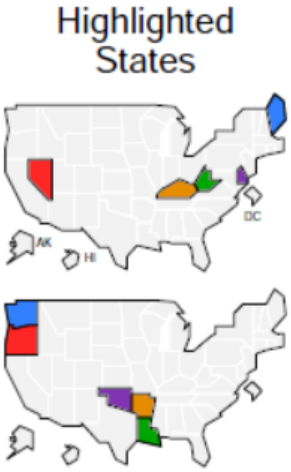
REWRITE The *map* collection of glyphs generate small maps from the boundary group for use in creating the linked micromaps and the statistical plots of the data. The collection of map glyphs do not use any values from the *col1*, *col2*, *col3*, *lab1*, *lab2*, *lab3*, *lab4*, *reftxt*, and *refval*, *panelData*, and *parm* vectors in the *panelDesc*. If any of there are present in the *panelDesc* data.frame, they value should be set to "NA". Each map is drawn in a single group/row section of the graphic representing up to 5 areas. A color is assigned to each "ACTIVE" area that is presented and carried over to the associated graphic in the columns on the left or right of the map.

Details

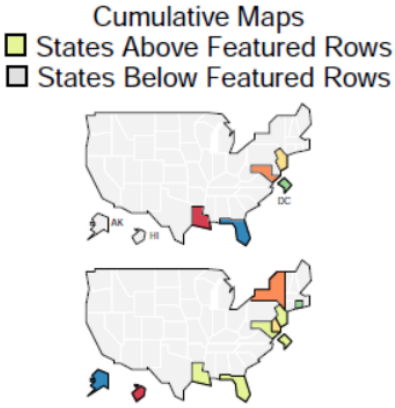
The only call parameter is *j* carrying the graphic column number being drawn. *j* is used by all of the panel functions to scale, outline, and select the area within the total graphic page to draw a column of the map glyphs, one per group/row.

The map collection consist of four ways the maps are used to present the area information moving down the page.

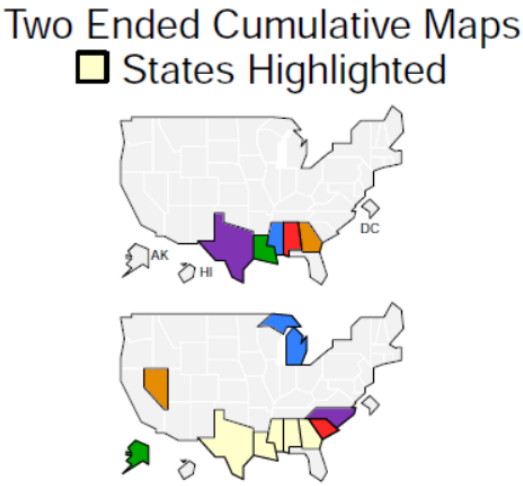
1) Map - is a simple map with out any extra areas colored in. Only the "active" areas being presented in the graphics on either side are colored. The same color is used in the statistical graphic to line the area to the data.

heightheight 

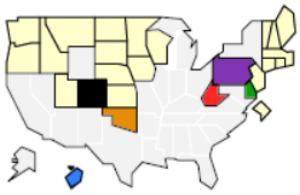
2) MapCum - is map, is the same as the "MAP" glypic, but after an area has been referenced, it is colored yellow to let the user know the area was presented in a group/row above as seen in the following figure:



3) MapTail - this map is the same as the MapCum in that it accumulate areas that have been presented. However, at the median point, the areas shaded are changed to the areas that have not been presented before to area to be repesented. As the graphics continue, the number of yellow area decreases. In the last row, only the presented states are colored and no area are colored yellow.

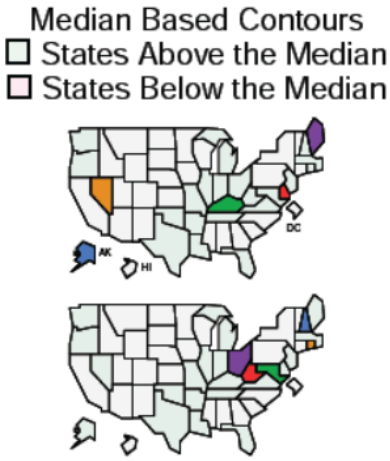


The other half of the tail leading away from the median can been seen in the lower maps.



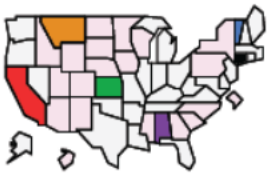
4) MapMedian - this map colors all of the areas with values above the median (as sorted) a very pale red. All of the areas below the median are colored a vary pale blue. The areas being presented are still colored with a unique color and linked to the statistical graphic using that color.

This figure shows maps above the median point:



heightheight

This figure shows what the map looks like below the median point:



heightheight

The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-normbar` *The Normbar glyph creates a graphic of a stacked bar chart dot plot of x values.*

Description

The *normbar* glyph creates a set of stacked bars with each bar in the stack with a length equal to the data value provided in a column of the *statsDFrame* data.frame. Since the stacked bar diagrams contain multiple bars, the data in the *statsDFrame* must provide the data in consecutive columns.

The first column in the series is specified in *panelDesc* data.frame in *col1*. The last column in the sequence is specified in *col2*. The stacked bars can contain from 2 to 9 bars.

The normbar graphic converts the length for each bar to values that will sum up to 100. Each normbar is drawn with bars from the left to right edge of the panel.

Each bar is filled with the same color as the area the data represents. The transparency of the color varies from left to right and a line is used to separate each bar for an area.

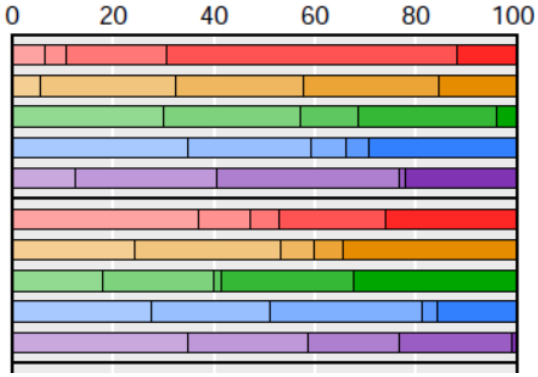
Details

The stacked normal bars only use the *panelDesc* col1, col2, lab1, lab2, lab3, lab4, refVal, and refTxt variable. The glyph uses the R draw rectangle function to draw and fill each part of the stack.

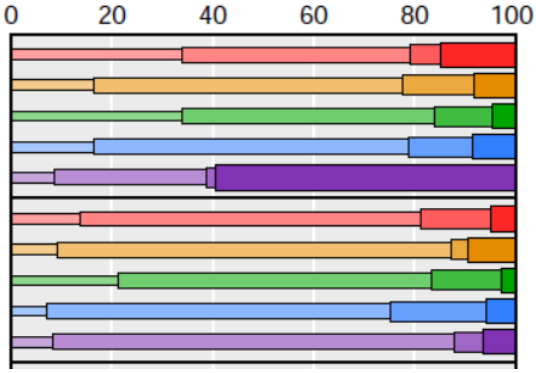
In the *details* data.frames there is a *SNBar.varht* (a logical variable) that is used to indicate to the gryphic whether to use variable height bars in the gryphic or not. By default the value is set to *FALSE*. When set to *TRUE*, all Center Segmented Stacked Bar Charts will have their bars vary in height from the left to the right.

The colors used in each segment are adjusted based on the number of segments in the glyph. If 5 bar are being displayed, the 1/5, 2/5, 3/5, and so on of the original color are used to fill in each bar as transparent colors. The last bar is always 100

The following is an example of a section of a normalized stacked bar glyph in a linked micromap:



The stacked bar glyphs have a unique property. The height of each segment can vary from small to full height from left to right. While the colors of the segments are varied in intensity from The following is an example of a section of a time series glyph in a linked micromap:



To have micromapST do variable heights in the Segmented Stacked Bar, and Normalized Stacked Bar, the detail call parameter is used to enable the feature.

```
details=list(SNBar.varht=TRUE,...)
```

The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-rank` *The rank glyph creates a graphic of the area rankings.*

Description

The *rank* glyph presents a ranking number for each area in the linked micromap. The rank number presented is the ranking value based on the sortVar results.

This glyph needs to be rewritten to properly handle order ties in the data consistantly.

Details

The ranking function does not use any data or label or reference column information from the *panelDesc* data.frame.

The ranking is based on the resulting sorted order specified in the sortVar call parameter.

The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-scatdot` *The SCATDOT glyph creates a graphic of a scatter dot plot of x and y coordinate points.*

Description

The *SCATDOT* glyph creates a scatter dot plot of all of the x and y coordinates for the data referenced by the scatdot through the col1 and col2 column numbers/names in *panelDesc* data.frame structure that point to columns in the *statsDFrame* data.frame provide by the USER with the data. The data from each area row is repersented by a dot in the *scatdot* graphic. The areas highlighted in the current group-row are shown using slightly large dots and the color of their associated area in the group-row set. There may be from 3 to 5 areas represented in a single group/row set and link by color to the geographic area in the micromap drawn from the boundary data provided in the boundary group data.

Details

The only internal argument is *j* identifying the graphic column number being drawn. *j* is used by all of the panel functions to scale, outline, and select the area within the total graphic page to draw a column of the *scatdot* glyphs, one per group-row. It is also used to reference the panelDesc column to retreive related information and links to be able to draw and label the *scatdot* graphic and column. The *scatdot* glyph uses the *col1* and *col2* variable to identify the *statsDFrame* columns that contain the x and y coordinates for each dot representing an area. The *lab1*, *lab2*, *lab3*, and *lab4* variables provide the header and trailing character strings for the column.

The graphic is a simple x-y plot of the data points provide to the *scatdot* routines.

The *refval*, *reftxt*, and *panelData* variable in the *panelDesc* structure are not used.

scatdot glyph has the ability to add a "DIAGONAL" or "LOWESS" line to the graphic as an option. The default is a white diagonal line. The parm variable list in the panelDesc is used by the user to request the type of line, the color, R type, and R width. In the list for the glyph column, the user starts by specifying "LINE=" to request "NONLINE", "DIAGONAL" or a "LOWESS" based line (upper or lower case). If a "LOWESS" type line is desired, the f value for the LOWESS function is set with "f=<value>" in the parm list. The value must be a decimal number ranging from .01 to 100. The range is not checked. All *f* values may not work if to small or large. Check with the R documentation on the "LOWESS" function in the *stats* package. The user cannot change the "iter" parameter. It defaults to 3. If the "LOWESS" function parameters are not acceptable to R, the system may terminate the micromapST function call and report an error. The user can also specify the line color (line.col, any valid hex value or "R" color name), line width (line.lwd, range of .5 to 2 is reasonable, default is 0.5) or the "R" line type (line.lty, the default is "solid" or any of the "R" types.) There extra parameters are passed to the glyph code via the panelDesc "parm=list( )" variable. This is the only panelDesc variable that must be a list of lists. Because of the R language there are two ways to successfully add the *parm* list to *micromapST* package:

```
panelDesc <- data.frame(type = c("map","scatdot"),
                        lab1 = c("US", "population"),
                        lab2 = c(NA, "1920"),
                        col1 = c(NA, "x"),
```

```
                         col2 = c(NA, "y")
                        )
wparm <- list(NA, list(line="LOWESS",f=.77, line.col="blue",
                        line.lwd=1, line.lty="solid")
             )
panelDesc$parm <- wparm

 or

panelDesc <- data.frame(type = c("map","scatdot"),
                        lab1 = c("US", "population"),
                        lab2 = c(NA, "1920"),
                        col1 = c(NA, "x"),
                        col2 = c(NA, "y"),
                        parm = I(list(NA,
                                      list(line="LOWESS", f=.77,
                                      line.col="blue", line.lwd=1,
                                      line.lty="solid")
                                     )
                                 )
                        )
```

In the second example, the "I()" function must be used around the list structure to make sure R does not parse the *parm* value when adding to the data.frame.

The resulting panelDesc structure should look like:

```
  str(panelDesc)
 'data.frame':   2 obs. of  6 variables:
$ type: chr  "map" "scatdot"
$ lab1: chr  "US" "population"
$ lab2: chr  NA "1920"
$ col1: chr  NA "x"
$ col2: chr  NA "y"
$ parm:List of 2
 ..$ : logi NA
 ..$ :List of 5
 .. ..$ line    : chr "LOWESS"
 .. ..$ f       : num 0.77
 .. ..$ line.col: chr "blue"
 .. ..$ line.lwd: num 1
 .. ..$ line.lty: chr "solid"
 ..- attr(*, "class")= chr "AsIs"
```

For details on the lowess R function refer to the stats package documention and the lowess function.

This allows the user to customize these optiona for each scatdot glyphic column, as needed.

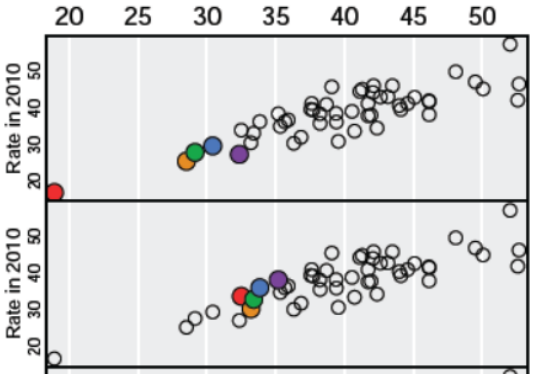
The defaults for the *scatdot* "parm list variable" are:

```
  if "line='NOLINE'" There are no defaults.

  if "line='DIAGONAL'" or "line='LOWESS'" then defaults are:

line.col = "white" or "gray20", respectfully
line.lwd = 0.5
line.lty = "solid"

  if "line='LOWESS'" then

   f = .666666

  if the parameter list is empty or there is no "line=" parameter,
     "line='DIAGONAL'" is assumed.
```

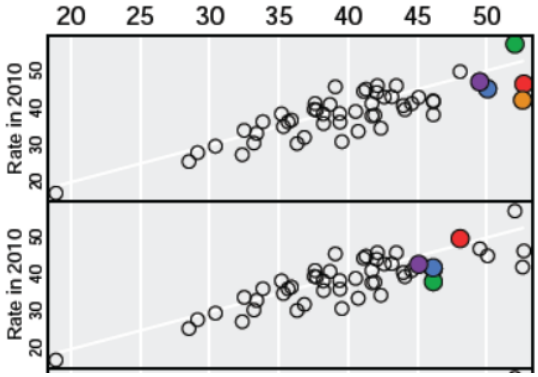
The user can change any of the *scatdot* options using in the parm row list entry.

The scatter dot plot can be produced with three types of lines:

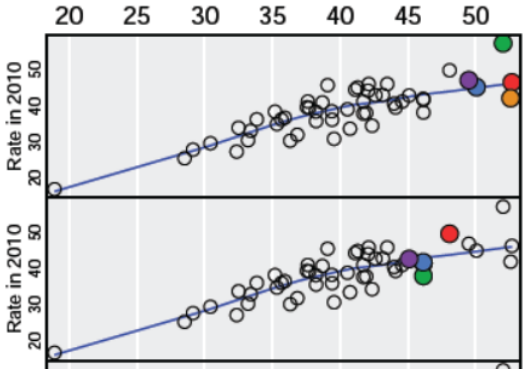
No Line:



A Diagonal Line (shown in white):



A Lowess Calculated Line (shown in black):



The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-segbar` *The Segbar glyph creates a graphic of a stacked bar chart dot plot of x values.*

Description

The *segbar* glyph creates a set of stacked bars with each bar in the stack with a length equal to the data value provided in the *statsDFrame* data.frame. Since the stacked bar diagrams can contain multiple bars, the data in the *statsDFrame* must provide the data in consecutive columns. The first column in the series is specified in the *panelDesc* data.frame in variable col1. The last column in the sequence is specified in variable col2. The stacked bars can contain from 2 to 9 bars.

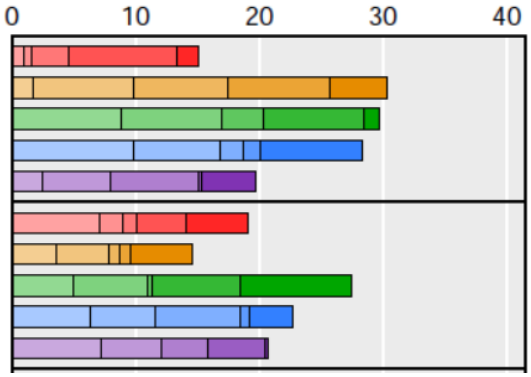
The segbar graphic starts the first bar at the left edge of the panel. Each bar is added to the right starting at the end of the previous bar. The width of the panel is determined based on the space require to fit all of the segment stacked bars for all of the areas. Each bar is filled with the same color as the area the data represents. The transparency of the color varies from left to right and a line is used to separate each bar for an area.

Details

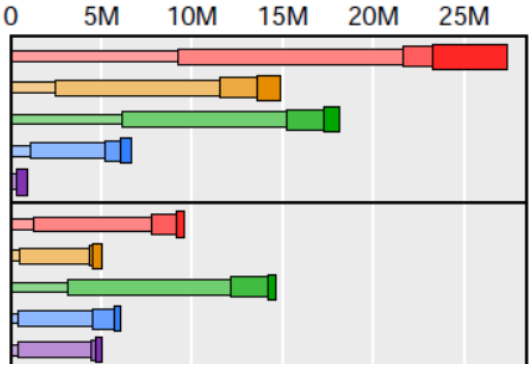
The segment stacked bars only use the *panelDesc* col1, col2, lab1, lab2, lab3, lab4, refVal, and refTxt variable. The glyph uses the R draw rectangle function to draw and fill each part of the stack.

The colors used in each segment are adjusted based on the number of segments in the glyph. If 5 bar are being displayed, the 1/5, 2/5, 3/5, and so on of the original color are used to fill in each bar as transparent colors. The last bar is always 100

The following is an example of a section of a segmented stacked bar glyph in a linked micromap:



The stacked bar glyphs have a unique property. The height of each segment can vary from small to full height from left to right. While the colors of the segments are varied in intensity from The following is an example of a section of a time series glyph in a linked micromap:



To have micromapST do variable heights in the Segmented Stacked Bar, and Normalized Stacked Bar, the detail call parameter is used to enable the feature.

```
details=list(SNBar.varht=TRUE,...)
```

The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

In the *details* data.frames there is a *SNBar.varht* (a logical variable) that is used to indicate to the gryphic whether to use variable height bars in the gryphic or not. By default the value is set to *FALSE*. When set to *TRUE*, all Center Segmented Stacked Bar Charts will have their bars vary in height from the left to the right.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`glyph-ts` *The TS and TSCONF glyph creates a time series plot for up to 64 time periods for the data based on the provided matrix.*

Description

The *ts* and *tsconf* glyphs create time series plot of data for each area over time with or without confidence internals. The plot is of a "y" value at a time ("x") where you can have as many samples as you want. When each area is presented for the analysis, the time series line will be highlighted in the panel. If you provided and requested confidence interval, a line is drawn for the upper and lower values over time and the space between the original value ("Y") and the upper and lower values are shaded in with a lighter variation of the color assigned to the area for presentation.

Since the data required for this glyph is more complicated, it is collected into a 3 dimensional matrix and passed to the glyph code via the *panelData* variable in the *panelDesc* data.frame.

Details

The data structure used to carry the time series data with confidence intervals is a 3 dimensional matrix. The first level index is by the area the data is associated. The second level under the area is the time series matrixes representing the data for each time period being graphed. The third level under each time period is A vector containg the "x", "y", "hy", and "ly" values a the time point for the area. The "x" is time point idenfier, "y" is the observed value, "hy" is the high confidence value, and "lY" is the low confidence value. It is best to assemble the matrix for the time series for each area, then gather the areas in to the final matrix.

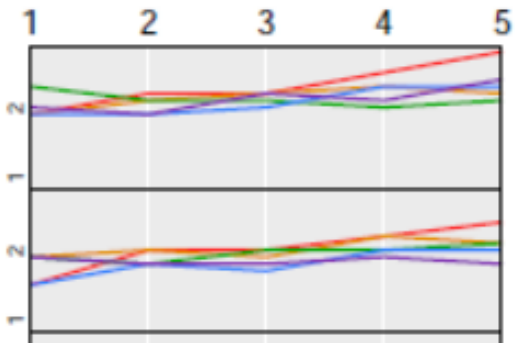
The easiest way to construct the time series 3 dimensional array is to use the following steps:

```
TSAreaNames <- AreaNames
TSArr       <- array(dim=c(51,10,2),dimnames=list(TSAreaNames))
                     # This build an empty 3d structure to be filled in.

for (index in TSAreaNames) {
 x   <- c(1,2,3,4,5,6,7,8,9,10)  # time points or "x" data values for this area.
 y    <- DATA[index,]         # pick up the 10 "y" data values for "x" points.
    TSArr[index,,1] <- x
    TSArr[index,,2] <- y
```

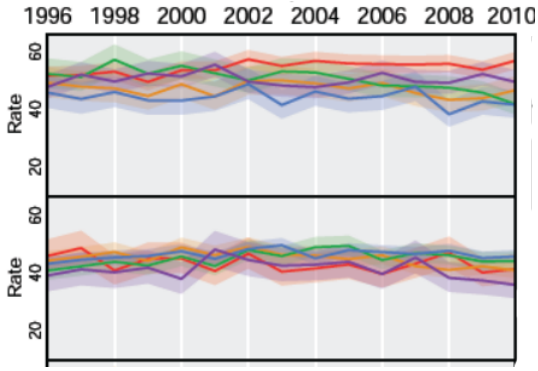
```
    # if the confidence values are available, then TSArr[index,,3]
  # and TSArr[index,,4] vectors would beadded for the high and low y values
}  # repeat until the TSArr is fully loaded.

# verify the 3d array was build correctly
row.names(TSArr) <- TSAreaNames   # add the row.names to the array.
```

If any data point is missing from the time series or the confidence data, that point will not be plotted, but the rest of the time series will be plotted.

The following is an example of a section of a time series glyph in a linked micromap:



An example of a time series glyph with confidence intervales is:



The shaded areas of the confidence interval around a time series line is based on the color of the area and line. It is a translucent version of the original color set to 20 time series lines to be seen and provides a reasonable presentation of the confidence spaces when the areas overlap on the graph.

The time series can also be produced in the Black and White color mode with good results.

The *statsDFrame* and *panelDesc* data.frames reside in the global environment and automatically accessible to the process along with several other major structures.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`KansasBG` *KansasBG border group datasets to support creating micromaps for the 105 counties in the state of Kansas*

Description

The *micromapST* function has the ability to generate linked micromaps for any geographical area. To specify the geographical area, the '`bordGrp`' call argument is used to specify the border group dataset for the geographical area. When *micromapST* function is used to micromap data for Kansas County area, the border group option ('`bordGrp`') is set to "KansasBG". This instructs micromapST to load the area Name, Abbreviation, ID and boundaries files for Kansas 105 counties. The datasets contained in the border group are areaNamesAbbrsIDs, areaVisBorders, L2VisBorders, and L3VisBorders for the counties of the state Kansas. The user's data is then linked to the boundary data via the county's name, abbreviated name or ID based on the table below.

Usage

```
data(KansasBG)
```

Details

The border group contains the following data.frames::

areaParms - contains specific parameters for the border group

areaNamesAbbrsIDs - containing the names, abbreviations, and numerical identifier for the counties in the state of Kansas.

areaVisBorders - the boundary point lists for each county area in Kansas.

L2VisBorders - the boundaries for an intermediate level. For state areas, this boundary data is not used and set to L3VisBorders as a place holder.

RegVisBorders - the boundaries for an intermediate level. For state areas, this boundary data is not used and set to L3VisBorders as a place holder.

L3VisBorders - the boundary of the state of Kansas

The Kansas county border group contains 105 county sub-areas. Each county has a row in the areaNamesAbbrsIDs data.frame and a set of polygons in the areaVisBorders data.frame datasets. No regions are defined in the Kansas county border group, so the *L2VisBorders* and *RegVisBorders* datasets are not used and the regions feations is disabled. The L3VisBorders dataset contains the outline of the state of Kansas.

The details on each of these data.frame structures can be found in the "bordGrp" section of this document. The *areaNamesAbbrsIDs* data.frame provides the linkages to the boundary data for each sub-area (county) using the fullname, abbreviation, and numerical identifier for each country to the *<statsDFrame>* data based on the setting of the '`rowNames`' call parameter.

A column or the data.frame row.names must match one of the types of names in the *areaNamesAbbrsIDs* data.frame name table. If the data row does not match a value in the name table, an warning is issued and the data is ignored. If no data is present for a sub-area (county) in the name table, the sub-area (county) is mapped but not colored.

The following are a list of the names, abbreviations, and IDs for each country in the *KansasBG* border group.

name	ab	id
Allen	AL	20001
Anderson	AN	20003
Atchison	AT	20005
Barber	BA	20007
Barton	BT	20009
Bourbon	BB	20011
Brown	BR	20013
Butler	BU	20015
Chase	CS	20017
Chautauqua	CQ	20019
Cherokee	CK	20021
Cheyenne	CN	20023
Clark	CA	20025
Clay	CY	20027
Cloud	CD	20029
Coffey	CF	20031
Comanche	CM	20033
Cowley	CL	20035
Crawford	CR	20037
Decatur	DC	20039
Dickinson	DK	20041
Doniphan	DP	20043
Douglas	DG	20045
Edwards	ED	20047
Elk	EK	20049
Ellis	EL	20051
Ellsworth	EW	20053
Finney	FI	20055
Ford	FO	20057
Franklin	FR	20059
Geary	GE	20061
Gove	GO	20063
Graham	GH	20065
Grant	GT	20067
Gray	GY	20069
Greeley	GL	20071

Greenwood	GW	20073
Hamilton	HM	20075
Harper	HP	20077
Harvey	HV	20079
Haskell	HS	20081
Hodgeman	HG	20083
Jackson	JA	20085
Jefferson	JF	20087
Jewell	JW	20089
Johnson	JO	20091
Kearny	KE	20093
Kingman	KM	20095
Kiowa	KW	20097
Labette	LB	20099
Lane	LE	20101
Leavenworth	LV	20103
Lincoln	LC	20105
Linn	LN	20107
Logan	LG	20109
Lyon	LY	20111
Marion	MN	20115
Marshall	MS	20117
McPherson	MP	20113
Meade	ME	20119
Miami	MI	20121
Mitchell	MC	20123
Montgomery	MG	20125
Morris	MR	20127
Morton	MT	20129
Nemaha	NM	20131
Neosho	NO	20133
Ness	NS	20135
Norton	NT	20137
Osage	OS	20139
Osborne	OB	20141
Ottawa	OT	20143
Pawnee	PN	20145
Phillips	PL	20147
Pottawatomie	PT	20149
Pratt	PR	20151
Rawlins	RA	20153
Reno	RN	20155
Republic	RP	20157
Rice	RC	20159
Riley	RL	20161
Rooks	RO	20163
Rush	RH	20165
Russell	RS	20167

Saline	SA	20169
Scott	SC	20171
Sedgwick	SG	20173
Seward	SW	20175
Shawnee	SN	20177
Sheridan	SD	20179
Sherman	SH	20181
Smith	SM	20183
Stafford	SF	20185
Stanton	ST	20187
Stevens	SV	20189
Sumner	SU	20191
Thomas	TH	20193
Trego	TR	20195
Wabaunsee	WB	20197
Wallace	WA	20199
Washington	WS	20201
Wichita	WH	20203
Wilson	WL	20205
Woodson	WO	20207
Wyandotte	WY	20209

There are no alternate abbreviations or regions assocated with counties in Kansas.

The *id* field value is the U. S. state and county FIPS code.

The '`rowNames`' = "alias" or "alt_ab" and the '`regionB`' and '`dataRegionsOnly`' features are not supported in the *KansasBG* border group.

`KansPopInc` *Test data for the Kansas border Group*

Description

This dataset contains the 2014 population and average income per person in each of the 105 Kansas counties.

Usage

```
data(KansPopInc)
```

Format

A data frame with 18 observations, 1 for each Seer area, on the following 12 variables.

County a character vector containing the Kansas County Name.

Pop a numeric vector of the number of the county's population

AvgInc a numeric vector of the average person's income for the county.

Details

This dataset was pulled from the Kansas government website on Dec. 2014. It contains the population and average income per person for each of Kansas' 105 counties.

Author(s)

Linda W. Pickle and Jim Pearson of StatNet Consulting, LLC, Gaithersburg, MD

LOWESSData *Test data for US LOWESS ScatDot glyphs.*

Description

This dataset contains the data for the USStatesBG for testing and demonstrations of the ScatDat LOWESS line feature in the glyph.

Usage

```
data(LOWESSData)
```

Format

A data frame with 51 observations, 1 for each US State and DC area, on the following 3 variables. The row.names are set to the strings in the "ABBR" column.

x the x coordinate value.

y the y coordinate value.

Abbr a character vector containing the abbreviated names for each US state and DC.

Details

This dataset was manually created to provide set of x, y and w data point to test and demostrate how a LOWESS line can be implement in the ScatDot glyph with the USStatesBG border group.

`MarylandBG` *MarylandBG border group datasets to support creating micromaps for the 24 counties in the state of Maryland*

Description

The *micromapST* function has the ability to generate linked micromaps for any geographical area. To specify the geographical area, the '`bordGrp`' call argument is used to specify the border group dataset for the geographical area. The *MarylandBG* border group dataset supports creating linked micromaps for the 24 counties in the state of Maryland. When the '`bordGrp`' call argument is set to *MarylandBG*, the appropriate name table (county names and abbreviations) and the 24 sub-areas (countries) boundary data is loaded in *micromapST*. The user's data is then linked to the boundary data via the county's name, abbreviated or ID based on the table below.

Usage

```
data(MarylandBG)
```

Details

The *MarylandBG* border group dataset contains the following data.frames:

areaParms - contains specific parameters for the border group

areaNamesAbbrsIDs - containing the names, abbreviations, and numerical identifier for the counties in the state of Maryland.

areaVisBorders - the boundary point lists for each county area in Maryland.

L2VisBorders - the boundaries for an intermediate level. For this border group, this boundary data is not used and set to L3VisBorders as a place holder.

RegVisBorders - the boundaries for an intermediate level. For this border group, this boundary data is not used and set to L3VisBorders as a place holder.

L3VisBorders - the boundary of the state of Maryland

The Maryland county border group contains 24 county sub-areas. Each county has a row in the areaNamesAbbrsIDs data.frame and a set of polygons in the areaVisBorders data.frame datasets. No regions are defined in the Utah county border group, so the L2VisBorders and RegVisBorders data.frames is not used and the '`dataRegionsOnly`' call parameter are is disabled. The L3VisBorders dataset contains the outline of the state of Maryland.

The details on each of these data.frame structures can be found in the "bordGrp" section of this document. The *areaNamesAbbrsIDs* data.frame provides the linkages to the boundary data for each sub-area (county) using the fullname, abbreviation, and numerical identifier for each country to the *<statsDFrame>* data based on the setting of the '`rowNames`' call argument.

A column or the data.frame row.names must match one of the types of names in the *areaNamesAbbrsIDs* data.frame name table. If the data row does not match a value in the name table, an warning is issued and the data is ignored. If no data is present for a sub-area (county) in the name table, the sub-area is mapped but not colored.

The following are a list of the names, abbreviations, and ids for each country in the *MarylandBG* border group.

name	ab	id
Allegany	AL	24001
Anne Arundel	AA	24003
Baltimore	BL	24005
Baltimore City	BC	24510
Calvert	CV	24009
Caroline	CL	24011
Carroll	CR	24013
Cecil	CC	24015
Charles	CH	24017
Dorchester	DR	24019
Frederick	FR	24021
Garrett	GR	24023
Harford	HR	24025
Howard	HW	24027
Kent	KN	24029
Montgomery	MG	24031
Prince George's	PG	24033
Queen Anne's	QA	24035
St. Mary's	SM	24039
Somerset	SS	24037
Talbot	TB	24041
Washington	WH	24043
Wicomico	WC	24045
Worcester	WR	24047

There are no alternate abbreviations or regions assocated with counties in Maryland border group.

The *id* field value is the U. S. state and county FIPS code.

The '`rowNames`' = "alias" or "alt_ab" and the '`regionB`' and '`dataRegionsOnly`' call parameters are not supported in the *MarylandBG* border group.

`mdPopData` *Test data for the Maryland border Group*

Description

This dataset contains the county populations for 1970, 1980, 1990, 2000, and 2010 and the projected estimated populations for 2015, 2020, 2025, 2030, 2035 and 2040 for each Maryland county and Baltimore City.

Usage

```
data(mdPopData)
```

Format

A data frame with 24 rows, 1 for each county, with 12 variables for each county:

Abbr a character vector abbreviation for the county.

County a character vector containing the Maryland county name and Baltimore City.

X1970 a numeric vector of the county's population in 1970.

X1980 a numeric vector of the county's population in 1980.

X1990 a numeric vector of the county's population in 1990.

X2000 a numeric vector of the county's population in 2000.

X2010 a numeric vector of the county's population in 2010.

X2015 a numeric vector of the county's estimated population in 2015.

X2020 a numeric vector of the county's estimated population in 2020.

X2025 a numeric vector of the county's estimated population in 2025.

X2030 a numeric vector of the county's estimated population in 2030.

X2035 a numeric vector of the county's estimated population in 2035.

X2040 a numeric vector of the county's estimated population in 2000. .

Details

This dataset was pulled from the Maryland government website in January, 2015. It contains the population and estimated population for each of Maryland counties and Baltimore City for 1970, 1980, 1990, 2000, and 2010 and estimated populations for years 2015, 2020, 2025, 2030, 2035, and 2040.

---

messages-BG — *micromapST Generated Warning and Error Messages - BuildBorderGroup*

Description

The BuildBorderGroup function verifies the call parameters and the incoming data. It then produces the data structures required by micromapST to draw maps for custom geographic areas. The user must provide a shape file and a name table for the function to product the user's custom border group dataset. This chapter documents the error and warning messages created by the BuildBorderGroup function when a problem or error is discovered before R has a chance to abort the execution of the function.

Like micromapST messages, BuildBorderGroup messages all start with "***" to help quickly find them in the warnings() logs and general output.

The general format of the messages is:

```
***XXXX text of message
```

where the XXXX is the unque four alphanumeric message identifier of the message. This chapter attempts to provide insight into the cause of the message and possible ways to resolve the problem.

Messages with numbers ranging from 3000 to 3999.

Details

Conventions: The user provides the shape file and the name table to the function. The messages attempt to provide specific information on the problem detected. So, parts of the message will be modified to provide more specific information. In the messages below <name> type fields are used to identify the variable information in the messages.

Any values in the message that are replaced by more specific information are shown as <namefield>. The explanation for the message and suggested means of resolution will reference the <namefield> variable to help define what the message is associated with in the operation of the BuildBorderGroup funcition.

The following is a listing of all micromapST generated messages and a description of possible causes and solutions.

3xxx BuildBorderGroup Messages:

310x Debug Call parameter:

3100 debug call parameter is not a numeric value. The default value of <default debug> will be used.

The value provided for the debug call parameter is not a numeric value. It must a number between 0 and 65535. The debug parameter will be set to the default value of 0.

312x checkPointRestart

3120 The checkPointReStart call parameter is not a logical value.

The checkPointReStart call parameter must be a logical variable with a value of TRUE or FALSE. The default value is FALSE.

3122 Check Point Restart has been requested. Check point files will be read from folder: <name table directory>/Checkpoint directory.

The checkPointReStart call parameter was set to TRUE and the BuildBorderGroup function will attempt a restart using the <name table directory> and the Checkpoint subdirectory to locate the check point files. If valid, the files will be reloaded and the BuildBorderGroup process will be continued to completion.

313x General Call Parameter

3130 Require call parameters are missing : <list of parameters> - execution stopped.

BuildBorderGroup requires several call parameters to run. The <list of parameters> string presents to operate properly. The minimum set of call parameters is: NameTableFile, BorderGroupName, and ShapeFile. If no directories are specified either the value of the NameTableDir will be used or the current working directory.

The function must have enough information to locate the initial name table (.csv or spreadsheet format) and the shape file(s) (.shp, .shx, .dbf)

314x Name Table Directory

3141 NameTableDir call parameter is missing or NULL. Correct and re-run.
The NameTableDir parameter must be a character string that provide the directory path information to the directory containing the Name Table file.

3142 NameTableDir call parameter is an 'NA', Empty or not a character string. Value= <NameTableDir> Correct and re-run.
The NameTableDir parameter must contain a character string with length > 1 of an existing path to the directory that contains the Name Table File. The value was found to be NA, ""(empty), or a non-character variable. Correct the incorrect value and retry.

3143 NameTableDir value specified does not exist. Value= <NameTableDir>
The character string provided as the NameTableDir value is not a valid path to the directory containing the Name Table File. The path provided does not exist.

315x BorderGroup Files

3150 BorderGroup directory specified in the call parameter does not exist. It will be created. Value= <BorderGroupName>.
The BorderGroupDir parameter in the function call specifies a non-existing directory. The path provided will be used to create the BorderGroupDir directory.

3152 The required BorderGroupName call parameter is missing.
The BorderGroupName parameter must be provided to be able to create the final border group data set. Provide the required parameter and rerun the function.

3154 BorderGroupName is a 'NA', Empty or is not a character string. Value= <BorderGroupName>.
The BorderGroupName parameter is not a character string or has a value of NA. Neither can be used as the border group name. The value must be a character string that is acceptable as a filename by operating system. Inspect and correct.

3156 BorderGroup directory specified in the call parameter is NA, Empty or not a character vector. The parameter will be ignored and the NameTable directory used. Value=<BorderGroupDir> The provided Border Group Directory is found to be NA, ""(empty), or a non-character variable. The parameter is ignored and the Name Table Directory is used instead. If the Border Group Directory was meant to point to a different location, correct the parameter and retry.

320x NameTable Files

3202 NameTableFile parameter has not been provided. Execution Stopped.
The name table filename for the name table data has not been provided. Add the appropriate NameTableFile call parameter.

3204 The name table passed to function on NameTableFile parameter is not a valid variable type: <class of NameTable data>
The class of the NameTableFile parameter must be a data.frame or a character string. It was found to be <class of NameTable data>. Correct and rerun.

3206 The NameTableFile parameter is set to 'NA', requires a valid file name or structure.

The name table file parameter is set to a NA value. It must be a valid data.frame or a character string indicating the name of the name table file in the name table directory.

3207 The NameTableFile parameter is empty. A name table structure or filename must be provided.
For BuildBorderGroup to function, it must have the name or structure of a valid Name Table. Without this, the border group can not build a name table or complete the construction of a border group dataset. Check on how to build a name table or check for typos in the Name Table File value.

3208 The NameTableLink call parameter is not a character string. Fix and rerun.
The NameTableLink call parameter must be a character string that can be used to reference a data.frame column read in from a .csv, spreadsheet or R .rda and .RData files. Please inspect the link name and make sure it is a valid column in the initial name table.

3209 The NameTable file is not a .csv, Excel, or R .RDA format.
The file containing the initial NameTable data must be a .csv, Excel spreadsheet or R .rda file. Each column must be one of the data columns required to construct the name table. (see name table section.)

321x Name Table Link

3212 The NameTableLink call parameter is an 'NA', empty (\'\' or \' \') or not a character string. Fix and rerun.
The NameTableLink identifies the name table column that will be used to link the name table to the information and polygons in the shape file structure. It was found have a value of 'NA', Empty (\'\' or \' \') or was not a character type value.
The NameTableLink call parameter must be a character string that can be used to reference a data.frame column read in from a .csv, spreadsheet or R .rda file. Please inspect the link name and make sure it is a valid column in the initial name table.

322x Shape File and Directory

3220 Shape File parameter has not been provided or is NA.
The filename of the shape file has not been provided in the BuildBorderGroup function call. Review the function call and make sure the ShapeFile parameter is present and identifies the base filename of the shape file to be used. The extensions of .shp, .shx, or .dbf do not have to be specified. The base filename will be used as the layer to be read. The ShapeFileDir is used as the 'dsn' value on the st_read call.

3221 ShapeFile is a sf structure passed as a call parameter.
The ShapeFile call parameter is a sf structure.

3222 ShapeFile is a SPDF structure passed as a call parameter.
The ShapeFile call parameter is a SPDF structure.

3223 The ShapeFile call parameter is being used to pass a full spatial structure to the function. However, the structure must be a SPDF or a sf class. The data was: <ShapeFile class>
The ShapeFile call parameter must be a character string, sf structure or SPDF structure. The class of the ShapeFile value is <ShapeFile class> and is not usable. Fix and retry.

3224 Shape file directory specified in the ShapeFileDir call parameter does not exist. Value= <ShapeFileDir>

The directory specified in the ShapeFileDir call parameter does not exist. The value was <ShapeFileDir>. Make sure the character string is a valid path to the directory containing the Shape File and that access is permitted. When no value is provided, the directory specified for the NameTableDir will be used. If no NameTableDir is provided, the current working directory is used.

3225 Shape file (dir & name) does not exist. Value= <ShapeFileDir & ShapeFile value>

The path to the shape file (directory and name) does not exist. The either the directory and/or the shape file filename does not exist, is not accessible or does not reference valid ERSI Shapefile collection containing a x.shp file. The extension of .shp is added to the path to validate the directory and filename for ESRI Shapefiles collections. Only the base filename is required in the ShapeFile filename. If present it is removed when the shape file is read. Check the path <ShapeFileDir & ShapeFile value> and make sure it is valid and can be accessed.

3227 The ShapeFile call parameter is an NA, empty(" or ' '), or not a character string. The filename of the Shapefile must be specified.

The provided ShapeFile name has a value of NA, is empty (" or ' '), or is not a character string. A valid filename must be provide for the ShapeFile name. Correct and retry.

323x ShapeLinkName parameter

3230 The ShapeLinkName call parameter is missing. The default value of NAME will be used.

For the shape file polygons to be used by micromapST, a linkage between the name table and shape file must be established. The ShapeLinkName call parameter is used to identify the shape file variable the function can use to match data in the name table. The ShapeLinkName call parameter is missing. The function will attempt to use the default value of NAME to find a variable in the shape file. If this is not correct, provide the ShapeLinkName parameter with a value of the correct variable.

3232 ShapeLinkName value is NA. The default of NAME will be used.

The value in the ShapeLinkName call parameter is an NA value. It must be a character string identifying the Shape File data column/variable to be used to link the polygons to the name table rows.

3234 ShapeLinkName is not a character string. Value= <ShapeLinkName>.

The value in the ShapeLinkName call parameter is not a character string that can be used to access a variable in the shape file header. Review the <ShapeLinkName> character string and correct.

324x Map Headers and parameters

3240 The MapHdr parameter does not contain character strings for use as the column headers. The MapHdr parameter will be ignored.

The MapHdr call parameter does not contain character strings. Inspect and correct. The default of c("","Areas") will be used.

3242 The MapHdr parameter must be a simple vector type.

The MapHdr call parameter must be a simple vector type variable with 1 or 2 elements. Data types of Data.frames, lists, tibbles, and other advanced structures are not allowed. Correct and retry.

3244 The MapHdr parameter has zero or more than 2 elements. Only the first 2 will be used.

The MapHdr call parameter contains more that 2 elements.
For example: c('hdr1','hdr2','hdr3') Only the first 2 elements will be used.

3246 It is suggested the max length of the MapHdr strings be 16 characters.
The MapHdr values (2) should be less than 16 character to keep the map type glyph columns from becoming too wide. Consider shortening the character strings.

325x Map Minimum and Maximum Height

3251 The MapMinH parameter does not contain numeric value. Default Value is used.
The MapMinH (Map Minimum Height) parameter is provided, but does not contain a numeric value. The default value of 1 inch is used.

3252 The MapMinH minimum height value is out of range (0.4 to 2.5 inch). The default will be used.
The MapMinH parameter value must be between 0.4 and 2.5 inches to be usable. The default of 1 inch is used.

3254 The MapMaxH parameter does not contain numeric value. Default Value is used.
The MapMaxH (Map Maximum Height) parameter is provided, but does not contain a numeric value. The default value of 1.75 inches is used.

3255 The MapMaxH maximum height value is out of range (1 to 2.5 inches). The default will be used.
The MapMaxH - value must be in the range from 1 to 2.5 inches. The default value of 1.75 inches will be used.

3257 The MapMinH value must be less than the MapMaxH value. Will swap values.
The MapMaxH value must higher than the MapMinH value. The values are swapped.

326x ID Header and Parameters

3261 The IDHdr parameter does not contain character strings for use as the column headers.
The IDHdr call parameter must be a character string to be used for the ID glyph column headers. The default header of the border group name and Areas will be used.

3262 The IDHdr parameter must be a simple vector type.
The IDHdr parameter value must be a simple one dimensional vector with 2 value, like the MapHdr parameters. The default of the border group name and "Areas" will be used.

3264 The IDHdr parameter has more than 2 elements. Only the first 2 will be used.
The IDHdr parameter has more than 2 values (elements) in the vector. Only the first 2 values will be used for the IDHdr headers.

3266 It is suggested the max length of the IDHdr strings be 12 characters.
To keep the ID glyph column from becoming too wide, it is recommended the IDHdr string be limited to less than 12 characters each.

327x Reduction Percentage Parameter

3272 The ReducePC parameter must be a numeric value. The default of 1.25 % will be used.
The ReducePC (Reduce Percentage) parameter is not a numeric value. The default value of 1.25% will be used. This parameter indicates how much of the original spatial information will be kept when the geometry is simplified by the rmapshaper package.

3274 The ReducePC parameter is not a simple vector. The default value of 1.25 % will be used.
The ReducePC parameter can only be a single value. The default value of 1.25 % will be used.

3276 The ReducePC parameter has more than one value. Only the first value will be used.
More than one value is provided with the ReducePC parameter. Only the first value will be used.

3278 The value of ReducePC is out of range (0.01 to 100 %). The default value of 1.25 % will be used.
The value of the ReducePC parameter must be between 0.01 and 100 percent. It is out of range. The default value of 1.25 % will be used.

328x LabelCex Call Parameter

3282 The LabelCex parameter must be a numeric value. The default of 0.25 will be used.
The value of the LabelCex call parameter must be numeric. The default of 0.25 will be used instead.

3284 The LabelCex parameter must be a simple vector. The default value of 0.25 will be used.
The LabelCex parameter value is not a single value vector but is a more complex data type. The default value of 0.25 will be used.

3286 The LabelCex parameter has more than one value. Only the first value will be used.
The LabelCex parameter contains more than one value. Only the first value will be used.

3288 The value of LabelCex is out of range (0.05 to 10). The default value of 0.25 will be used.
The LabelCex parameter value must be between 0.05 and 10. It is out of range and the default value of 0.25 will be used.

330x proj4 Parameter

3301 No projection provided in the shapefile or the proj4 call parameter,will be set to a Long/Lat projection.
No projection was found in the shapefile or structure provided the function or in the proj4 call parameter. The boundary data is assumed to be longitude/latitude values and the projection in the spatial structure will be set to a Long/Lat projection.

3302 The proj4 call parameter set to NA or Empty(”) or is not a character string. The parameter will be ignored.
The proj4 call parameter must be a valid character string in the proj4 projection syntax. A value of NA is not acceptable and the proj4 call parameter will be ignored.

3304 3304 The proj4 call parameter is not a valid character string or ’crs’ structure for a projection. The proj4 parameter will be ignored. The default AEA projection will be used in needed.
The function was unable to process the provided proj4 projection character string using the st_crs function. The parameter will be ignored. A generic AEA projection based on the center of the map and its height will be used to transform the map for use in the link micromap.

3305 Invalid proj4 parameter value provided. Parameter will be ignored.
When trying to process the proj4 projection string using the sf package st_crs function, a error was raised. The proj4 string can not be process properly and will be ignored.

3306 The proj4 call parameter specifies a long/lat projection.
proj4: <proj4>
The final projection can't be a longlat projection.
The proj4 parameter is ignored and a AEA projection will be created.
The proj4 call parameter specifies a long/lat projection (see <proj4> for details). The final projection of the map for linked micromaps should not be long/lat. Therefore, an Albers Equal Area projection is calculated based on the centroid of the map with secondary latitudes 1/4 the maps height above and below the centroid.

3308 The proj4 call parameter does not have +units=m, changing string to meters.
The proj4 call parameter projection does not have a component of +units=m to set the projection to meters as required. Since this projection will be the final projection of the map, the +units has been changed to m to force the final projection to be in meters.

331x & 332x Shape File Processing and Call Parameter

3310 The Shape file SPDF structure was passed to the function in the call.
This message is an informational message to document the shape file (as a SpatialPolygonsDataFrame) was passed to the function instead of passing the directory and name of the shape file.

3311 Reading shape file from dir: <SFDir> file: <SFName>
The function is going to read the shape file from the directory <SFDir> and layer name <SFName> using rgdal\'s readOGR function with verbose = TRUE.

3312 The sf st_read can not import the shapefile as specified. The errors reported were: <res1> <res2>

In the attempt to read the shapefile, the function attempts to use the dsn= parameter on the st_read for the directory to the shape files and the layer= parameter to specify the specific shape file. This is one of two modes the st_read function can use depending on the driver it selects. If this mode does not succeed, the function will attempt to use the mode where the layer= is not used and the dsn= parameter must contain the entire directory and filename, minus the extentions. If either are successful, this message is generated and the two error messages are reported in <res1> and <res2>. Review the error messages and correct the directory name used, the filename referenced and make sure both exist.

3314 Spatial Driver found in boundary data file read was <SReadDriver>.
The <SReadDriver> was identified during the st_read of the Shape File.

3315 The shape file structure was passed to the function in the call.
The type of data passed to the function in the ShapeFile parameter indicates the user is passing a spatial structure rather than a filename. The type of structure will be validated.

3317 The spatial structure passed to the function via the ShapeFile parameter is not a SPDF or a sf full structure. Please correct and try again.
If a spatial structure is passed to the BuildBorderGroup function, it must be a SPDF or sf structure containing the data information. The data information is required to be able to match the polygons in the structure to the rows in the name table.

3320 The projection field in the shapefile is empty, set to <OrigLongLat>
The proj4string slot in the SpatialPolygonsDataFrame for the shape file is empty. The projection for the shape file is assumped to be a long/lat projection and will be set to a long/lat projection of <OrigLongLat>.

3325 Checking Shape Link Name Column: <ShapeLinkName>
Checking to make sure the shape file data.frame header contain a variable by the name of <ShapeLinkName> as specified on the function call.

3326 The ShapeLinkName provided: <ShapeLinkName> does not exist in the shape file data.
The shape file variable <ShapeLinkName> does not exist in the data.frame header. Check for the existence of the variable and for possible spelling errors and return.

3327 Shape file link variable name is valid, values will be cleaned up and stored in variable X__Link.
The specified Shape File link variable name was found in the SpatialPolygonsDataFrame. Its contents will be cleaned up and saved in variable X__Link for later use.

350x to 355x Name Table General Issues

3510 The Name Table was read from: <NameTableType> <NameTablePath>
The initial name table file will be read from <NameTablePath> containing the directory and filename. The name table file type is <NameTableType>.

3511 The Name Table Path provided to read the Name Table does not exist. Value = <NameTablePath>. The Name Table Directory and Filename to read the Name Table does not exist. Please check the Name Table file's location and try again. The Name Table must be accessible to run the BuildBorderGroup function.

3512 The NameTable in the .rda file is not a data.frame. Please correct and retry.
The initial name table can be constructed and saved in the R .rda format file. When read, the contents of the file is not a data.frame structure and cant be used. Research the cause and correct.

3514 There are more than one data.frame in the .rda file provided for the NameTable. Provide only one data.frame table in the .rda and retry.
When the .rda file was opened for use as the name table data.frame, more than one was found. The .rda should only contain one data.frame for use as the name table.

3215 Name Table Link column is: <NameTableLink>
The function will use the <NameTableLink> column to pair up the name table rows with the collections of polygons in the shape file.

3521 The Name Table has no columns of data.
The name table as read, does not contain any data columns or meaningful column labels. Research and correct.

3522 The Name Table has no rows or areas.
The name table as read, does not contain any area rows. There must be one row per area in the map. Research and correct.

3532 The column specified in the NameTableLink calling parameter does not exist in the loaded Name Table.
The value specified in the NameTableLink call parameter is not the name of a column in the read initial name table. If the Link column exists, it will be used as the NameTableLink column to match the name table areas to the shape file polygons.

3542 At least one of the following columns must be present in the NameTable file: <List of Column Names>

Please correct the spreadsheet and try again.
One of the columns named in <List of Column Names> must exist in the name table. If none of the columns exist, then the initial name table is not valid and needs additional work to permit the building of a border group.

3550 The following columns are not needed and will be deleted from the Name Table:
Extra columns were found in the name table. These columns are listed below and will be deleted. Check for typos to make sure all of the required information is retained.

356x to 359x Name Table Name, Abbr, ID, Alias and Alt_Abr data errors

3562 The <inx> column contains duplicate entries. Correct and retry.
The column identified by <inx> was inspected and found to contain duplicate values (entries). Duplicates are not allowed in location ID columns that must be unique for each row. Research the values in column <inx> and correct.

3564 The <inx> column contains NA or blank values that are not allowed.
The column identified by <inx> contains blank values or NA values. These values are not permitted in location ID type columns. Research and correct the values.

3572 The Abbr column in the Name Table is not included. Will attempt to backfill it from other information.
The Abbr location id should be supplied in the name table whenever possible. If it does not exist, it will be created using other information provided in the the name table.

3573 The Name Table Abbr field is persent - no backfill required.
The name table abbr field was present. Processing continues normally.

3574 Some of the Name Table Abbr values are longer than 5 characters. It is recommended the Abbr values be keep short.
It is recommended the character string values in the Abbr column be kept to 5 characters or less. This keeps the location ID fields in the data smaller and less work for the preparer and keeps the ID glyph abbr label narrower to more statistics type glyphs to be presented.

3576 None of the columns needed are present. The Link and one of the Name, Abbr, and ID column should have been there. This should never happen with the previous checks.
If this error message occurs, then something has happened during the processing. This situation should have been caught earlier. Research the name table and make sure all required columns are present.

3582 Checking ID column in the name table to make sure the values are numeric with leading zeros.
While the ID values of location ids are numeric, a lot of them have leading zero. The IDs are check to ensure they are numeric, then converted to character format and leading zeros added to help in later comparisons.

3584 The ID column is not present. A numerical sequence number has been used to fill the column.
No ID colomn is present in the name table. An ID column consisting of a sequence of 1 to n is used.

3586 The ID data column is not all numeric values. Values will be assigned.
Not all of the ID values are numeric or valid. The rows with bad ID values will be

replaced with new values.

3590 Name Table L2_ID_Name column contains row(s) with NAs or ” values. L2 feature disabled.
All of the rows in the Name Table L2_ID and L2_ID_Name columns must have values for the L2 Feature to operate correctly. Inspect these columns in the name table provided and correct.

3592 Name Table L2_ID column contains row(s) with NAs or ” values. L2 feature disabled.
All of the rows in the Name Table L2_ID and L2_ID_Name columns must have values for the L2 Feature to operate correctly. Inspect these columns in the name table provided and correct.

3596 Name Table regName column contains row(s) with NAs or ” values. Region feature disabled.
All of the rows in the Name Table regID and regName columns must have values for the Region Feature to operate correctly. Inspect these columns in the name table provided and correct.

3598 Name Table regID column contains row(s) with NAs or ” values. Region feature disabled.
All of the rows in the Name Table regID and regName columns must have values for the Region Feature to operate correctly. Inspect these columns in the name table provided and correct.

3599 The Region Feature has been disabled. The number of regions defined is either 1 or is equal to the number of areas.
For the Region feature to operate, there must be more than 1 region and less than the number areas. If equal to 1 or equal to the number areas in the map, the Region function has no value and is disabled.

362x Name Table Modification Parameters

3622 The Name Table in the <inxRN> area row and in the <inx> column is not numeric and has a bad value of: <WrkVal>. Value set to the default. Fix and retry.
The data value in the row named <inxRN> for column <inx> has a value <WrkVal2> that is a non-numeric value. Correct and rerun.

3624 Data in row: <inxRN> for <inx> parameter <WrkVal2> is out of range. (<low> to <high>)
The data value in the row named <inxRN> for column <inx> has a value <WrkVal2> that is out of range. The acceptable range is from <low> to <high>. Correct and rerun.

363x and 364x Map Labels and Coordinates

3630 The MAPL column in the name table is not character data. Labeling will not be done.
The map label must be a character string value. Labeling for the row will not be done.

3631 If the MapL column is present with label(s), then MapX and MapY must be present. One or the other is missing. Labeling is not done.
For each Map Label in the name table, there must be a valid MapX and MapY value specifying the location on the map to draw the label. If all three are not present, the label for that area will not be drawn.

3632 MapL value is present and there are no valid MapX coordinates. Labeling will not be done.
A map label for an area is present, but the MapX coordinate is missing or invalid. The label will not be drawn. Correct the MapX and possibly the MapY values.

3633 MapL is present and there are no valid MapY coordinates. Labeling will not be done.
A map label for an area is present, but the MapY coordinate is missing or invalid. The label will not be drawn. Correct the MapY and possibly the MapX values.

3634 The MapL label - <label> - for area <area name> should be 3 or less characters to be usable.
It is recommended the map label string be shortened to 3 character or less to reduce the space required to draw the label.

3635 No MapLabel content - processing skipped.
The retired MapLabel column in the Name Table does not have any values. Converting the MapLabel column into the MapL, MapX, and MapY columns will be skipped.

3636 Some of the items in the MapLabel entry for <name table row> are NA or blanks. Will be ignored.
The MapLabel entry for the <name table row> is blank or NA and can not be used. The value is ignored.

3637 The MapLabel value for <name table row> is not valid. Must be a character string with three values separated by commas. The value is ignored.
The character string in the MapLabel field must consist of three values separated by commas. The first value is a character string (the label) and the other two columns are numbers representing the X and Y coordintates to draw the label. The X and Y coordinates are in the units and origin of the original shape file.

3638 The label value in the MapLabel entry for <name table row> is > 3 char. Only first 2 characters will be used.
The recommend length of the map labels is 2-3 character. One or more of the labels are greater than 3 character. Adjust the length of the map labels.

3639 One of the coordinates in the MapLabel entry for <name table row> is/are not a number. <MapX> or <MapY>
Check the MapX and MapY coordinates values, they must be numerical values.

3640 One of the MapLabel coordinates for <name table row> are out of range. Entry ignored.
The values for the MapX and MapY coordinates must be within the range of the space covered by the defined map in its original projection. If the values are outside of the map's area, this error will appear. Inspect the coordinates and adjust as needed.

371x Shape File Projection

3711 The projection provided in the Shape File does not have +units=m, modify and setup for re-projection to change to meters.
The final projection must be in meters. The projection in the Shape File or structure is not meters. A new projection based on the original is created with +units=m. This projection will be used for the final transformation of the map.

3712 Found +units=m in projection string of non-longlat projection in the shape file.

The projection in the shapefile is not a long-lat projection and it is already in units of meters. No modifications are required and no transformation will be needed.

372x Boundary data clean up

3720 Cleaning up polygons in spatial structure.
The spatial structure will be passed through sf st_is_valid and st_make_valid functions to clean up the polygon geometries.

3722 Shape File contains invalid polygons. The indexes and associate reasons are\:
This is an informational message indicating the shape file geometry will be passed through the sf package validation and make_valid function to clean up the geometry. S2 functions are not used.

375x Matching Shape File to Name Table

3750 Comparing shape file to name table links
The link values in the shape file identified variable are being compared to the values in the name table identified column. If there are any polygons that do not have a matching name table entry, the polygon(s) will be deleted.

3752 The following Shape File areas are not in Name Table:
<List of Shape File areas not in Name Table>
The areas will be dropped.
Any area (polygon) in the shape file that does not have a row in the Name Table, will be dropped from the final border group map.

376x Linking Name Table to Shape File

3760 Comparing the link values to tie the name table to the shape file.
This is the reverse comparison to comparing shape file links to the name table links. If a name table area/row does not have a matching shape file set of polygons, then the functions execution will be halted. Either provide the polygon(s) for the area included in the name table or remove the name table entry.

3762 The following Name Table areas do not have boundaries in the ShapeFile:
<List of Name Table Rows without polygons>
Correct and retry.
The BuildBorderGroup function can not continue if there is not an area(set of polygons) in the shape file for the Name Table row. Execution stopped.

3764 Some of the polygons in the Shape file still do not belong to areas in the name table. Polygon(s) are ignored.
Not every polygon is linked to an area in the Name Table. These polygons will remain in the map, but will be ignored. If multple ignored polygons share the same boundary, the boundary may not be drawn.

377x Shape File Simplification

3770 Simplifying the shape file boundary data with rmapshaper.
This message is a progress report to indicate the processing has called rmapshaper to simplify the spatialpolygon data.frame (shape file) to the ReducePC specification with a weighting of 0.9. Check the results to see if the map is over simplified or not simplified enough. Change the ReducePC call parameter and rerun until the map becomes usable.

3772 rmapshaper parameters before simplification : Keep= <MS_Keep> Weight= <MS_Weighting>

The rmapshaper ms_simplify function parameters for the MS_Keep and MS_Weighting are listed to document the amount of boundary reduction and smoothing. See the rmapshaper ms_simplify documentation for more details.

3774 rmapshaper processing completed.
The rmapshaper ms_simplify function has completed its work and has returned a modified spatial structure.

3778 Map area aggregation completed.
After the rmapshaper ms_simplify completes its work, the polygons in the spatial structure are aggregated into multipolygons to create one rows per area in the sfc structure. This message signals the aggregation step has been completed. The number of features in the sf structure are now equal to the number of areas in the name table.

379x Area Size Inspection and Name Table Modifications

3793 The following areas may be too small (<0.03%): <ListOfAreas>
In general, if an polygon is less than 0.03 shaded for linked micromaps, it may not be visible to the graphs users. The list of areas that may be too small are provided in the <ListOfAreas> string in the message. Review the list of areas and a plot of the map and determine if the areas should be enlarged or moved or manually manipulated to ensure their shading can be seen easily. checkPointReStart call parameter allows the user to intercept the shape file before the VisBorders data.frame format datasets are created to make custom changes.

3798 Info:No modifications are required to map.
The name table did not contain any shift, scale or rotate parameters for any areas in the maps. No modifications were done.

38xx Name Table Modifications

3801 Identifying neighbors for each area.
The function is now searching and identifying the neighbors for each area in the map. This information is used in the color selection algorithm to keep two areas that share the same boundary line from being colored the same color in the examples and any future feature that is added to micromapST.

3804 The list of neighbors for each area are in the name table 'NB' column.
The neighbor list is stored in the 'NB' column of the name table temporarily. This will be save across the checkpoint restart for future use.

3810 Info:No modifications are required to map.
The name table did not contain any modifications for areas in the map.

3811 Area: <area key> will be adjusted using the <modification list> values.
<area key> area has parameters to have shifts, scaling and/or rotate modification done to the boundaries.

3812 Xoffset: <Xoffset> Yoffset: <Yoffset> Scale: <Scale> Rotate: <Rotate> in radians: <radians>
The name table contains the following values for each of the possible modification for the identified area in # 3811.

3813 Re-inserting area polygons for <area key>
The area <area key> has been modified and is being re-inserted into the map and neighboring areas.

3814 plot of original area before modifications in blue:
A screen plot was drawn of the area before modification with blue lines.

3815 plot of main area after modifications in green:
A screen plot was drawn of the area after the modification with green lines.

3818 Original neighbor boundaries in magenta for <neighbor area>
A screen plot was drawn for an neighboring area to the modified area before adjustments were made with magenta lines.

3819 Trimmed neighbor boundaries in seagreen for <neighbor area>
A screen plot was drawn of an neighboring area to the modified area after trimming adjustments were made with seagreen lines.

3822 Name Table modificiations to Shape file are completed.
All of the name table specified area modification has been completed.

384x Area Coverage Analysis

3843 These area coverage estimates were done after the name table modifications. The area's coverage may have be increased or decreased from the original Shape file. This coverage review is done against the current area's coverage after all modifications. If an area's coverage is less than 0.057% of the total map coverage of <TotalArea> * 0.00057 m^2 may not be large enough to be visible in a in a linked micromap graphic. The area(s) that should be reviewed is(are):

3844 <list of areas that may be undersized.> This message provide a list of the areas that may not be visible when willed with the link color in the linked micromap. Investigate and adjust the area size as needed.

3845 All of the area appear to be big enough to show the shading in a linked micromap.
Verify all of the areas can be seen when the linked micromap color shades the area.

386x Final processing and transformation

3860 Transforming projection of Shape file and label points.
This is a progress message to indicate, the function is about to perform any requested or needed project transformations on the shape file and the label points in the name table. If the resulting projection is not correct, check the value provided via the proj4 call parameter and whether the default AEA projection based on the center of the map is correct.

3861 Using user provided projection: <proj4>
This message outputs the projection string provided by the BuildBorderGroup caller via the proj4 calling parameter to help document the border group.

3862 Re-transforming shape file using original projection, with +unit= changed to meters.
The original projection provided on the shape file did not have the +unit= proj4/6 parameter set to meters. BuildBorderGroup assumes the projection of the shape file just prior to the conversion of the boundaries to the VisBorders format is meters. The proj4 call parameter or the projection contained in the shape file was inspected and a new projection created with the +unit= paremeter set to m was created and used in the maps transformation before the VisBorders data.frames are created.

3863 Projecting shape file using created AEA projection.
Since the shape file is a long/lat projection and no projection was provided via the

proj4 call parameter, the BuildBorderGroup function will calculate an Albers Equal Area projection based on the centroid of the mapped area and secondary latitudes at 1/4 the map's height above and below the center latitute. This results in all of the areas being presented with equal area (or shading area).

3865 Transformations completed.
All of the transformation and projection modifications have been completed.

3866 Info:No transformation was done to the map.
No transformations were required on the final map. The original projection is still in use.

3868 The proj4 value character vector is not a valid projection. Must be a value acceptable to st_crs(). proj4 is ignored.
Verify the proj4 call parameter is correct and has the right syntax for the proj4 projection specification. The proj4 parameter will be ignored.

387x Generating Test Plot

3872 The length of name table and the number of areas in the shape file are different.
The number of areas listed in the name table and the number of areas with polygons in the shape file are different. Either the name table has more entried than the shape file or the shape file has more areas then the name table. Correct and re-run.

391x checkPointRestart Writes

3910 The checkpoint files will be store in the Checkpoint directory : <CkptPath>
The directory used to save the check point files is <CkptPath>.

3912 Checkpoint - Name Table RDA Filename: <NTCkpt> Name Table CSV Filename: <NTCkptcsv>
The check point Name Table data.frame is save as a data.frame (.rda) in the <NPCkpt> file and as a .csv file in the <NTCkptcsv> file.

3913 Checkpoint - The shape file spatial data is written to: <SFCkpt>. as a set of ESRI Shapefiles. The RDA image spatial data is in: <SFCkptRDA>
The check point image of the shape file is saved in ERSI Shapefile format in the <SFCkpt> shape files and as a SpatialPolygonsDataFrame in the <SFCkptRDA> RDA file.

3915 Checkpoint - areaParms data.frame in: <APPCkpt> as <APCkpt>
The control areaParms dataset has been checkpointed in the <APPCkpt> directory as the <APCkpt> filename.

3917 BuildBorderGroup has completed writing of the checkpoint files to disk for possible editing and restart. They are located in the following directory : <Check Point Directory>
The border group dataset has been successfully written to directory included in this message and is ready for use, reuse, or editing.

3918 The check point Shape File for the border group is saved to: <Shape File Name>
This message identifies the filename of the saved Check Point Shape File being used to build this border group. It can be editted, but must be returned to the same directory with the same name for the checkPointRestart logic to restart the border group building process.

3919 After editing, the results must be saved back to the same directory and filename.
The check point shape file can be manually edited to make areas more visible in the small micromap. Once the modifications are completed by any external GIS or polygon editor, the final shape file must be save in it's original directory and under the original filename. If you want to save a copy of the shape file producted by the BuildBorderGroup function, save it before making modifications.

392x Checkpoint Reload to Continue

3920 Check Point Restart Process Initiated.
This is an informational message letting the user know the checkPointReStart process has been initiated and all of the working files have been reloaded into the function.

3921 No Border Group or Name Table directory provides. Cannot find restart files. STOP. The name table and border group directories provided for the checkPointReStart could not be located. Make sure to provide the same directories and file names used in the original BuildBorderGroup function call.

3922 NameTable directory: <NameTableDir>
The name table directory being used for the check point restart is <NameTableDir>.

3923 Using the BorderGroup directory: <BorderGroupDir> to locate the checkpoint files.
The name table directory was not provided. Since the target output directory was the BorderGroupDir path, the BorderGroupDir call parameter will be used to locate the checkpoint folder and files.

3925 Reading areaParms dataset: <areaParm.rda file>
The restart logic in the checkpoint feature is loading the areaParms.rda file from the <areaParms.rda file> path.

3926 Reading shape file: dsn=<Shapefile Directory> layer=<Shapefile base name>
The restart logic in the checkpoint feature is loading the saved shape file from the checkpoint folder <ShapFile Directory> and the layer is the <ShapeFile base name>. These are the checkpoint folder locations not the original Shapefile directory and file parameters.

3927 Reading NameTable: <NT filename>
The restart logic in the checkpoint feature is loading the checkpointed version of the name table (areaNamesAbbrsIDs) from the checkpoint folder.

3928 The Shapefile has been modified and the 'X__Key' variable has been removed. Rerun the BuildBorderGroup function to restore the variable.
Then do not remove the variable when editing the shape file.
BuildBorderGroup places a couple of special variable in the shape file to allow the polygons to be matched up with the areas in the name table. It appears the 'X__Key' variable as been removed. Editing is permitted, but none of the 'X__' variables added by the BuildBorderGroup should be removed. Rerun BuildBorderGroup to restore the variables, do the required editing, and then do a '`checkPointReStart`' = TRUE to build the micromapST boundary files.

3Axx Creating micromapST boundaries files

3A02 The shape file variables have been editted. The 'X__Key' variables are missing. Redo the edits and do not delete the 'X__Key' variable.
During the editing of the Shape file, the 'X__Key' variable must be kept the same to

allow the Name Table to maintain a link between the area location ids and the associated polygons. Reedit the Shape File and ensure the X__Key variable is untouched.

3A04 The shape file row.names in the spatial structure do not match the 'X__Key' variable values. Investigate and correct cause. row.names are reset to the 'X__Key' values.

The contents of the X__Key variable/column. If they are changed, the tie between the Name Table and the spatial data is broken. Reedit the Shape file and return the X__Key column to its original value written when the checkpoint file was created.

3A22 Invalid polygon found in <area key> area #:<nz> id:<z>

Message is issued by the BuildVisBorders function as it is process the spatial data and converting it into the data.frame format for micromapST drawing. The message indicates an invalid polygon was identified for the <area key> at row number <nz> with the id of <z>. Identify the bad polygon and fix its geometry.

3A28 Completed conversion to VisBorders format.

All of the boundary data for the individual areas, Layer 2 boundaries (if requested), regional layer boundaries (if requested), and the map outline have been converted to the data.frame point format required by micromapST to draw the miniture maps.

3A30 Creating the 4 micromapST boundary layers (area, L2, L3, and Regions).

The shape file will be copied to the area, L2, Regions, and L3 shape file images and merge based on the spaces layed out in the name table for L2 spaces, Regional spaces, and the outline of the entire map (L3).

3A40 Completed conversion to VisBorders format.

The conversion from spatial structure to micromapST's data.frame format has been completed.

3A53 Writing an images of each Border Group data.frame for <BGBase>

A single R .rda file will be written for each of the 6 data.frames included in the border group <BGBase>. The can be found in the border group directory under the names of <BGBase>_<name of data.frame>.rda.

3A55 Border Group Created - Successfully.

The writing of the border group dataset has been successful. The border group is now ready for use.

3A60 Summary build report of names, abbr, id and other data is written to <RepOutFileName>

Each border group needs documentation to provide the user to let them know what the list of Names, Abbrs, IDs, Aliases, and Alt_Abr location IDs are available in the Name Table for the space mapped by border group. A copy of the key information from the Name Table is printed for use in this documentation in a text file in the border group directory under the name of <RepOutFileName>.

3A69 Border Group: <finalBGroup> is done.

The process of gathering the information, validating it, editing it, and converting in to a format for micromapST has been completed and the dataset written to disk.

3980 The proj4 value character vector is not a valid projection. Must be a value acceptable to st_crs. proj4 is ignored.

The projection character string provided on the proj4 call parameter is not a valid. It can not correctly processed the sf function st_crs to be used as a projection. Make it complient with PROJ4 speciications. Correct and re-run. (convertProj4)

3985 AdjPolygons - Polygons level value is not a Polygons structure.

In processing the SpatialPolygonsDataFrame image of the shape file, the Polygons level below the polygons level is not a valid SpatialPolygons structure and can not be processed. Execution is halted. Review the shape file or SpatialPolygonsDataFrame structure and make sure it is correct.

3997 The number of areas in the map is 1 or less. A border group can not be made.
The number of areas in the name table and the shapefile must be more than 1 polygon or area to be able to build a border group. Check the name table and shapefile contents.

3999 Errors have been found and noted above. Execution stopped. Please fix problem(s) and retry.
Inspect the previous messages and errors to identify the cause of problems that stopped the execution of the function. Fix the problem and rerun.

REPORT MESSAGES PUBLICATION INFORMATION FOR NAME TABLE IN BORDER GROUP
The border group has been created. To make sure the user can use the same location ids that were used in the name table, the following table is printed to provide documentation on what location IDs are available to the user when they are assemblying their data. Each column present in the name table is listed: Name, Abbr, ID, Alias, Alt_Abbr. These are the primary columns used by the data gatherer.

Name Table Regional Values If only the regional space information is present in the name table, then the regID and regName name columns will be displayed in the report for reference.

Name Table Modifications and Map Label Values If Map Labels and name table modifications were specified for any area in the name table, The MapL, MapX, MapY, Xoffset, Yoffset, Scale and Rotate columns of the name table are listed in this section for later reference. If neither Map Labels or modifications were used, this section of the report is not outputed.

Name Table Map Label Values If only Map Labels were implemented in the name table and not name table modifications, only the values for the MapL, MapX, and MapY name table columns are displayed in this section of the report.

Name Table Map Modifications Values If no Map Labels were specified in the name table, but name table modification were specified, then this section of the report will list only the name table modification values for future reference.

Name Table Layer 2 and Regional Values If the name table contains Layer 2 and Regional space information for use by micromapST, then this section of the report will be outputed listing the L2_ID, L2_ID_Name, regID, and regName name table column information for later use. If the L2 and regional information is not present this section of the report is not displayed.

Name Table Layer 2 Values If only the Layer 2 information is present in the name table, then the L2_ID and L2_ID_Names name table columns will be displayed in thei report for reference.

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

messages-MM *micromapST Generated Warning and Error Messages-micromapST*

Description

micromapST verifies as much of its operation and the data provided by the user to try and identify and document to the user problems before they cause an R warning or error exception and throw a R like cryptic message. Each message, warning and error message is documented 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.

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

The general format of the messages is:

```
***XXXX NNNNNN text of message

  or

***XXXX NNNNNN CC text of message
```

where the XXXX is the message alphanumeric identifier, NNNNNN is the name of the function or glyph issuing the message and CC is the glyph graphic column number in the *panelDesc* structure.

The four alphanumeric message identifier following the "***" is a unique message identifer to help find the explanation in this document and to help discuss problems over the email more accurately. It's always best to include the log of the preparation and execution of micromapST as well as the data used when requesting help.

The first two digits/characters of the message identifier indicates which logical segment of the package generated the message:

01xx Main package initialization, startup, call argument validation, user data validation and *panelDesc* structure and contents validation.

02xx panelDesc and glyph parameters and data validation, processing and graphic generation

03xx Border Group data and structure validation and setup,

04xx Internal Operation

05xx Informational Messages

The second field of the message contains the name of the section of code generating the message. If a glyph generated the message, the name of the glyph (type=xxx) and the panel column number are displayed to help you focus on the parameters and data that may need to be inspected. Each message contains specific information (like *panelDesc* named list, checked value, etc.) to help identify the cause of the message and help lead the user to a fix.

The remainder of the message contain a text explanation of the issue and a list of parameters: glyph column number/name, data row or column, variable name, and value related to the issue.

The abbreviation in most messages identifies the processing step in the package the message was produced:

CARG Call Arguments-Parameters

- **<blank>** General call parameter messages
- **-RNC** Row Name Column
- **-RN** Row Name type
- **-AS** Ascend sort order parameter
- **-TL** Title parameter
- **-PN** PlotName parameter
- **-GP** Group Pattern parameter
- **-SC** Scaling parameter
- **-SL** Staggered Label parameter
- **-CS** Column Size parameter
- **-DF** Statistics Data Frame parameter or content
- **-PD** Panel Description data frame parameter or content
- **-PDparm** Panel Description data frame - parm row list
- **-SORT** Sort parameters

Alias Use of the Alias location ID feature

BGBD Border Group Directory parameter

BGBN Border Group Name parameter

BGBG Border Group data set content

BDATA General Border Group boundary data

COLORS color call parameter

DETS details call parameter

PDCOL Panel Description dataframe columns. These messages relate to the columns within the panel description data frame rather than the glyph itself.

PANEL Panel Description dataframe general issues

PDPARM Panel Description dataframe parm list row issues

DMP & INB Internal issues

<glyph name> Messages related to a specific glyph graphic. The <glyph name> would be one of the following: dot, dotconf, dotse, dotsignif, ctrbar, normbar, segbar, map, mapcum, mapmedian, maptail, id, arrow, bar, boxplot, scatdot, ts, or tsconf.

Details

Conventions: The user provides the data.frames for the statsDFrame and panelDesc structures. In this document, there variables are represented by the generic terms of *statsDFrame* and *panelDesc*. In the messages produced by the package, these values are replaced with the names of the user variables provided in the micromapST call.

The following values are also substituted with the real run values at time of execution:

<sortVarCol> List of column names/numbers used in the '`sortVar`' call parameter. Used to identify invalid *statsDFrame* column names and numbers used in the '`sortVar`' call parameter.

<PDvarName> panelDesc variable name. The panelDesc variables include *lab1*, *lab2*, *lab3*, *lab4*, *refVal*, *refTxt*, *col1*, *col2*, *col3*, *type*, and *panelData*.

<PDcol> panelDesc glyph column number. This helps identify which graphic column (glyph) is being processing when the error/warning is signaled. The far left column number is 1.

<PDvarList? a list of variables from the panelDesc data.frame. Used in messages to identify one or more *panelDesc* variables that are invalid.

<detVar> detailed list variable name

<lastVarName> last variable name used. Used to locate a value or absents of a value in a list by using the last good variable name.

<bordGrp> a character string of the name of the border group being referenced. The border group names does not include the .RData or .rda extension used on the data file.

<detVarName> detailed list variable name

<method> The labeling method requested for generating Axis labels.

<list of unused sub-areas> A list of the sub-areas (counties, etc.) that did not have data.

<listRowNames> list of row.names (sub-areas)

<litrowNamesCol> The literal rowNamesCol value submitted by the caller.

<class(xxx)> The class of the variable names "xxx"

<number items> Number of items in list or vector.

<invalid name list> List of invalid names that were unexpected.

<list> a general list of options, variables, or names.

<statsDFrame> The name of the data.frame used as the first call parameter. This data.frame is used for the statsDFrame and provides the numeric values for the plots.

<panelDesc> The name of the data.frame serving as the panelDesc panel column description.

<gName> The name of the glyph being drawn.

<glyphName> The name of the glyhp begin drawn.

<sDFName> The name of statistics data.frame for the linked micromap.

<pdColNum> The panelDesc data.frame column number (glyph column).

<dupList> Is the list of duplicate entries in a list.

<rowNumbers> References the row number in the statsDFrame data.frame.

<list of rows> Is a list of rows related to the message.

<rowNumbers> One or more row numbers related to the message.

<NotUsedNames> Identifies a not used name from name table.

<future> Reserved for future usage.

The following is a listing of all micromapST generated messages and a description of possible causes and solutions.

01yx Initialization, Setup and Call argument validation, *panelDesc* and User data validation. where y indicates the specific argument or area.

01Zx General Operation ("Z"):

01Z0 CARG Key call arguments are missing, NULL, wrong type, or NA, Execution stopped.
The micromapST function requires the *statsDFrame* and *panelDesc* data frames be provided. This is the minimum set of call parameters required. If either or both parameters are missing (the first two in the function call), the execution of the function is stopped.

01Z8 CARG Warnings have been found in the parameters and data. Package continues, but results are unpredictable. Review log and fix errors.
The micromapST function has detected warning in the parameters and data. The function will continue but the results may be unpredictable. Review the errors and warnings, correct the causes and re-run.

01Z9 CARG Errors have been found in parameters and data. Please review program log and fix problems. Packaged stopped.
The micromapST function has detected errors in the data and call parameters that prevent the function from running. Correct the identified error and re-run.

010x *statsDFrame* **call argument (1st argument) (CARG-DF):**

0101 CARG-DF First argument (*statsDFrame***) is missing or NULL or not a data.frame.**
The first calling argument/parameter must be the *statsDFrame* data frame containing the data to be graphed. If the first argument/parameter is missing or NULL. Provide the correctly built *statsDFrame* and re- run. The data structure can not be a list, tibble, matrix, array or simple vector variable.

0102 CARG-DF The First argument (*statsDFrame***) is not a data.frame class.**
The *statsDFrame* variable holding the data for each area (county, district, etc.) in a border group is not a data.frame structure. Each column represents a data variable to micromapST. Each row represents an area being mapped and charted. The area location ids may be located as the data.frame row.names or in a data.frame column itself. The panelDesc structure uses the col1, col2, and col3 columns to specify which column in the *statsDFrame* should be used for the glyph. The sortVar and rowNamesCol call parameters use the statsDFrame column name or number to identify how to sort the graphics or where the area name is located in the data.

0103 CARG-DF The <*sDFName***> *statsDFrame* data.frame must have at least 1 or 2 columns.**
The <sDFName> data.frame provided as the <*statsDFrame*> must have at least one column or two if the rowNameCol call parameter is specified. One data column must be provided and if the row.names feature in the data.frame is not used for the location id, then one additional column is required and the rowNameCol must be specified.

0104 CARG-DF There are duplicate entries in the *statsDFrame* **data.frame. Duplicate entires are ignored.**
The *statsDFrame* data frame contains rows with duplicate names. Only the first row is used, duplicates are ignored. Check the area name values in the data frame and correct as needed to make sure each row represents a unique area. The following message provides a list of the duplicate sub-areas. ***

0105 CARG-DF The duplicate rows are: <listRowNames>
This message provides a list of the duplicate area names, abbreviations or IDs in the *statsDFrame* data frame. Correct the duplicate entries and re-run micromapST. ***

0106 CARG-DF The following row in the <*sDFName***> statsDFrame data.frame are not found in the name table:**

See continued message (0107) for details

0107 CARG-DF <listRowNames>

This is a two line error message. The area names provided in *statsDFrame* data.frame must match the names in the border group name table. This warning provides a list of the araa names not found in the name table. Please select the correct border group or correct the area names in the data.frame and re-run micromapST. Unless requested, the function will stop.

0108 CARG-DF The rows not matched to boundaries will be removed and not mapped.

If a row of data does not match an entry in the name table, the data row in the *statsDFrame* will be deleted. Check the location id used on the data row to make sure it matches the type of names you selected in the name table ("Full", "Ab", "ID", etc.) If the row's loc id does not match, correct and re-run.

0109 CARG-DF Data row names in the ‚sDFName," data.frame must match the location ids in the name table. Call stopped.

If '`ignoreNoMatch`' is FALSE and a data row does not have an entry in the boundary name table then the package can not continue to run. All data must have valid names or abbreviations associated with each row that has sub-area boundaries in the border group. Execution is stopped. Verify the area identifiers in the supplied data, correct and retry.

011x panelDesc call argument (2nd argument) (CARG-PD):

0111 CARG-PD The second call argument, the panelDesc structure, is missing or NULL.

The second call argument/parameter is the *panelDesc* data.frame. Without this data frame, micromapST does not know which glyphs to draw or where the supporting data is located in the *statsDFrame* data.frame. The data frame is missing, NULL, NA or is not a data.frame structure. The data structure can not be a list, tibble, matrix or array variable. Correct and re-run micromapST.

0112 "CARG-DF The second argument, the panelDesc structure, is not a data.frame.

The second argument in the *micromapST* call must be the panelDesc structure as a data.frame. No other type of variable is acceptable. Please correct the structure and retry.

0113 CARG-PD The following named lists in the <*pDName*> panelDesc data.frame are not valid: <*PDVarList*>

The *panelDesc* variables *col1*, *col2*, and *col3* identify the columns in the *statsDFName* data.frame that contain data for each area to be used by the glyph for the column. The column number or name provided in the *panelDesc* structure is not a valid *statsDFrame* column number (out of range) or name. This error message provides a list of the invalid column numbers or names.

0114 CARG-PD The required *type* named list is missing in the <*pDName*> panelDesc data.frame.

The *panelDesc* list named *type* does not exist in the *panelDesc* data frame. This named list is required to identify the type of glyph to be generated for the specific column. Inspect the *panelDesc* data.frame, check for spelling and correct the source of the problem and re-run.

Example: *panelDesc*=list(type=c("map","dot","bar"), ...)

0115 CARG-PD The <*panelDesc*> *type* named list contains one or more invalid glyph name(s): <*pdTypeList*>.

The type named list in the *<panelDesc>* panelDesc data frame contains invalid glyph names. The invalid names are listed in this error message. The valid glyph names are: map, mapcum, mapmedian, maptail, id, arrow, bar, boxplot, dot, dotse, dotconf, dotsignif, segbar, normbar, ctrbar, ts, tsconf, and scatdot.

012x call argument column name/number verification (CARG-xxx):

When a call argument/parameter contains a column name or number referencing the *<statsDFrame>* data.frame, the arguments values are checked during the initialization of the micromapST processing. The following errors may be detected and the function stopped.

The call arguments/parameters that contain column names/numbers are:

```
rowNamesCol      RNC
sortVar          SORT
```

The abbreviated name (*<callArgAbbr>*) listed above for the call argument/parameter ('`<callArg>`') is included in the message to assist in identifying and correcting the problem.

0123 CARG-'`<callArgAbbr>`' A column name in the '`<callArg>`' call argument does not exist in the *<sDFName>* data.frame: *<value>*

One or more of the column names in the '`<callArg>`' call argument/parameter does not exist in the *<statsDFrame>* data.frame. The bad value is displayed in the *<value>* part of the message. Verify that the column name exists in the *<statsDFrame>* data.frame and re-run.

0124 CARG-'`<callArgAbbr>`' The '`<callArg>`' call argument is empty. Argument ignored.

The '`<callArg>`' value was found to have a length of zero - empty. The argument can not be checked and is ignored.

0126 CARG-'`<callArgName>`' The call parameter contains more than one value. Only the first value will be used.

The call parameter <callArgName> can only use a single value. More than one value was specified in the function call. Only the first value will be used.

.

015x '`bordGrp`' and '`bordDir`' parameter (BGBx):

0150 BGBD The directory specified in the bordDir call parameter is not a valid character string. Check the bordDir path name and make sure it is valid.

0151 BGBD The directory specified in the '`bordDir`' argument/parameter does not exist. Value=*<bordDir>*

The path specified does not exist, Processing is stopped. Correct the path name and re-run. Check to make sure the slashes in the path name are correct and that the path exists.

0152 BGBN The value provided as the bordGrp name is not character. Fix and rerun.

The bordGrp call parameter must be a character string of the border group name (and it's file.) If it is not the name of border group provided with the package, it must be the name of the border group file (without the .rda) that is located in the bordDir path.

0153 BGBN When the '`bordDir`' parameter is set to NULL, the '`bordGrp`' must be one contain in the package. *<backageName>*

When the ‘bordDir’ call parameter is not provied, or set to NULL or NA, the value of the ‘bordGrp’ parameter must be one of the internal border groups provided with the package. The border groups provided with the package are:

USStatesBG U.S. States and the District of Columbia

USSeerBG The 18 U.S. Seer Registry areas

KansasBG The counties of the state of Kansas

MarylandBG The counties of the state of Maryland

NewYorkBG The counties of the state of New York

UtahBG The counties of the state of Utah

UKIrelandBG The provinces, counties and cities of the United Kingdom and Ireland

ChinaBG The provinces, special administration areas, metropolitan cities of China

SeuolSKBG The districts of the South Korean city of Seoul

If you are using a custom or private border group, then the ‘bordDir’ parameter must be provide to specify the file directory containing the border group .rda file.
See the appropriate documention section for details on each border group contained in this package.

0154 BGBN The ‘bordGrp’ parameter has not been specified and is required when the *bordDir* is provided.
When the ‘bordDir’ call argument/parameter is specified, the package will load a users or private border group into the package. Therefore, the ‘bordGrp’ call argument/parameter MUST be specified to identify the file in the directory to load. Verify the ‘bordGrp’ filename is present and exists in the ‘bordDir’ directory and re-run. Execution was stopped.

0155 BGBN The ‘bordGrp’ filename must have an ".rda" or ".RData" file extension.
When private border group files are created, they must be saved as ".rda" or ".RData" type "R" files. Please re-create your private border group file and re-run. Execution is stopped.

0156 BGBN The ‘bordGrp’ file in the ‘bordDir’ directory does not exist.
The package attempted to open the ‘bordGrp’ file in the ‘bordDir’ directory, but the file does not exist. If the ‘bordGrp’ filename was specified without a file extension, the package appended an ".rda" file extension. Verify the ‘bordGrp’ filename exists in the ‘bordDir’ directory and re-run. Execution is stopped..

Also see the 01Mx message section for more information on Border Group messages.

017x ‘rowNamesCol’ call argument (CARG-RNC):/cr To be able to link up the *statsDFrame* rows to areas in the border group and its boundary information, the *statsDFrame* must provide a column containing the sub-area names, abbreviations or IDs that match the name table entries in the border group. The values can be supplied via the row.names on the data frame or in a column in the data frame. The ‘rowNamesCol’ call parameter allows the user to specify which *statsDFrame* column to use as the link to the border group name table. The rowNames argument/parameter is used to specify if the link is the name, abbreviation or ID of the sub-area.

0171 CARG-RNC The row names in column ‘rowNamesCol’ of the statsDFrame data.frame contains duplicates. Only one row per area is permitted. Duplicate rows are: <list of rows>. The rowNamesCol will be ignored and the data.frame

row.names values will be used.
The '`rowNamesCol`' value specifies a column with duplicate area names. There can only be one row per area. The rowNamesCol call parameter is ignored and the data.frames row.names information is attempted to be used.

0172 CARG-RNC The '`rowNamesCol`' argument value must have a length = 1. Only first value used.
The '`rowNamesCol`' value must have a length of 1 and should be a simple vector. Only one column can be specified to contain the area names (row names.) If more then one value is provided, only the first value in the data will be used.

0173 CARG The rowColNames is NA. The row.names values on the data.frame will be used.
The '`rowNamesCol`' value provided is NA and cannot be used. Instead the data.frame row.names information will be used.

0174 CARG The rowColNames is NA. The row.names values on the data.frame will be used.
The rowColNames value is NA. It must be a valid statsDFrame column name. The data in the column must contact data that matches one of the location IDs in the Name Table. When no rowColNames is specified, the row.names values on the statsDFrame data.frame are used as the location ID.

0175 CARG The rowColNames value specified does not exist in the data.frame. The row.names values on the data.frame will be used.
The data.frame column name specified in the '`rowColNames`' call parameter does not exist in the data.frame. To continue executing, the data.frames row.names information is used.

018x '`sortVar`' and '`ascend`' call arguments (CARG-AS):
The '`sortVar`' calling argument/parameter specifies the *statsDFrame* column numbers and/or names to be used to sort the data and the rows presented in the linked micromaps. The sorting is executed in the order of the columns listed in the numeric or character vector provided via the '`sortVar`' argument/parameter. The column numbers and names are verified before the sorting is attempted. All references must be valid. The default sort uses the sub-area names as spedified in the '`rowNames`' argument/parameter.
The '`ascend`' flag indicates which direction the sorting will be done. '`ascend`' = TRUE requests ascending order. '`ascend`' = FALSE requests descending order.

0181 CARG-SV The '`sortVar`' parameter is not a numerical or character vector variable. Matrix, arrays, data.frames and tibbles are not supported. Will use the default of alpha sort on area names.
The '`sortVar`' call argument/parameter must be a numeric or character vector with a length less than with 3 values. The values must be valid indexes or names of columns in the statsDFrame data.frame supplied by the user. Only a simple vector is supported. Matrixes, arrays, data.frame and tibbles are not supported.

0186 CARG-AS The '`ascend`' parameter is not a logical variable. Must be TRUE or FALSE.
The '`ascend`' call argument/parameter must be a logical vector with a length of 1. Vectors with multiple values are not allowed. Numeric, Integer or Character vectors are not supported.

019x '`rowNames`' call argument and '`alias`' name matching feature (CARG-RN & ALIAS):
The '`rowNames`' call argument/parameter specifies what type of sub-area name/string

should be used to match the sub-area name in the *statsDFrame* data frame, boxplot and time series data with the border group's name table. The valid rowNames values are: 'full', 'ab', 'alt_ab', 'id', and 'alias'. The default is 'ab'.
When 'alias' is specified, the package attempt to link the *<statsDFrame>* data rows to the boundary data using a partial match between the sub-area names in the *<statsDFrame>* data.frame and the *alias* column in the name table. This feature was implemented to handle special cases were the data source does not use the regularly used names or abbreviations for the sub-areas. See the discussion on the USSeerBG border group for more details and an example. At this time, only the USSeerBG border group makes use of this feature.

0190 CARG-RN Invalid 'rowNames' argument value. The value must be 'ab', 'alt_ab', 'id', 'alias', or 'full'. The default of 'ab' will be used.
The 'rowNames' call argument/parameter contains an invalid value. The default of 'ab' is used.

0191 CARG-RN rowNames='alias' is not supported for this bordGrp. The default of 'ab' will be used.
The rowNames value of 'alias' is only supported in border groups containing the an alias column in the name table and have the border group areaParm variable *enableAlias* set to TRUE. The only border group supplied with the package that supports the alias name matching is the the USSeerBG border group.

0192 CARG-RN rowNames='seer' is only supported for the USSeerBG bordGrp. The default of 'ab' will be used.
Select the correct border group or use another rowNames value.

0193 The rowNames parameter is NULL, NA or Missing. The default of 'ab' will be used.
rowNames must be provided to define the type of location id to use. The default is "ab". The other options are "full", "alias", and "alt_ab".

0195 ALIAS Alias Names(s) in the data no not match the name table for the area. The unmatched data rows are: <listRowNames>
The rowNames value of 'alias' was specified. During partial wildcard matching of the "alias" column in the border group name table with the data column in the *statsDFrame* data frame specified by the 'rowNamesCol' call argument/parameter, one or more data rows did not successfully match the name table. The <listRowNames> part of the message lists the data row values that did not match. Review the *statsDFrame* values used for sub-area names and the documentation of the alias strings provided in the selected border group and make adjustments as required.

0196 ALIAS Sub-area names in the data have duplicate name in rows: <listRowNames> Only one row per area is permitted.
There are multiple rows in the *statsDFrame* that match a single sub-area in the border group name table. Only the first matching data row will be used. The <listRowNames> section of the message contains a list of the duplicate rows. Correct row names to eliminate any duplicates and re-run.

0197 The rowNames parameter is not a character string.
The rowNames parameter must be provided and must be a character vector. If it is not, the function is terminated.

01Ax 'title' call argument (CARG-TL):
The 'title' calling argument/parameter is used to specify a one or two line title for the linked micromap. If not 'title' is provided, the titles are left empty - blank.

01A0 CARG-TL The '`title`' argument contains more than 2 items. Only the first two will be used.

The '`title`' call argument/parameter contains more that 2 character strings. Only two title lines are supported. Only the first two values (lines) are used.

01A1 CARG-TL The typeof/class of the '`title`' parameter is not character. Only character vectors are supported. The '`title`' argument is ignored.

The type and class of the '`title`' call argument/parameter must be a character vector. Numeric, integer and logical vectors are not supported.

01A2 CARG-TL The '`title`' argument/parameter is empty. Recommend providing a title for the linked micromap.

The '`title`' call argument/parameter is empty (length of 0). It's recommend a set of titles be specified for the linked micromap. Please provide a '`title`' character vector to help identify the linked micromap and re-run.

01Bx '`plotNames`' call argument (CARG-PL):

The '`plotNames`' calling argument/parameter specifies the type of area name to be used in the ID glyph. The allowed values are '`ab`' and '`full`'.

01B0 CARG-PN Invalid '`plotNames`' argument value. The value must be '`ab`' or '`full`'.

The value of the '`plotNames`' call argument/parameter is not valid, The value must be '`ab`' or '`full`'. The default value of '`ab`' is used. Check the spelling and re-run.

01B2 CARG-PN Invalid plotNames argument value. The value must be 'ab' or 'full'. The default of 'ab' will be used.

The '`plotNames`' argument value must be 'ab' (abbreviation) or 'full' (full name). The default of 'ab' will be used. If the data contails full names, correct and re-run.

01Cx '`grpPattern`' call argument (CARG-GP):

The package creates a pattern of how many sub-area rows will be placed in each glyph group row based on the total number of sub-area with data. The user can override this generated pattern by providing their own pattern in the form of an integer vector. Each entry represents one glyph group row. The maximum value for an entry is 5. The values must be in decending order toward the median sub-area and must sum to the number of sub-areas in the data. For example: grpPattern = c(5,5,4,3,4,5,5) for 31 sub-areas. If these rules are not followed, the following messages are generated and the package created pattern is used.

01C0 CARG-GP The '`grpPattern`' call parameter must be an integer vector.

The vector provided must be an numeric or integer vector. The package created pattern is used. Correct and re-run.

01C1 CARG-GP The total of the rows per group must equal the number of rows in the <*statsDFrame*> data.frame. grpPattern ignored.

The sum of the sub-area rows in the grpPattern vector is not equal to the number of sub-areas present in the *statsDFrame* data.frame. For example, if the <*statsDFrame*> data.frame has 51 rows after duplicates are removed, then the sum of the grpPattern vector must be 51. A grpPattern of c(5,5,5,5,5,1,5,5,5,5,5) would be valid, but c(4,4,4,4,4,1,4,4,4,4,4) would be invalid. The package will use a calculated pattern. Correct and re-run.

01C2 CARG-GP The '`grpPattern`' call parameter vector must be <= 5 (rows per group). A value of <x> was found.

Each element in the grpPattern vector represents the number of sub-area rows per glyph group/row. The maximum number of sub-area per group is 5. The package will use a calculated pattern. Correct and re-run.

01C3 CARG-GP The '`grpPattern`' call parameter is not properly ordered. The number of rows per group must be in desending order to the median sub-area.
The number of rows per glyph group must be arranged in descending order from the ends to the median sub-area row in the middle of the list. For example: grgPattern = c(5,5,4,3,4,5,5). A grpPattern of c(3,4,5,5,4,3) is acceptable. The package will created and use a calculated pattern. Correct and re-run.

01C4 CARG-GP The one of the values in the grpPattern call parameter is non-numeric or an NA. grpPattern ignored.
The grpPattern is a vector of integers specifying the number of areas in each group/row or perceptional row. One of the values in the vector is a non-numeric value or an NA. Correct and re-run.

01Dx '`axisScale`' call argument (axisScale & CARG-SC):
The '`axisScale`' calling argument/parameter is a single value character vector that specifies the type of axis scaling the micromapST should use on the glyphs. The acceptable values are:

o original axis labeling provided by the pretty function. Scaling is done in the same way as previous releases of micromapST.

e labeling is done using the extended algorithm from the labeling package.

s labeling is done using the extended algorithm and label scaling by applying a multiplier of hundreds, thousands, etc and a subtitle used to identify the multiplier units.

```
Example:
  before:      1000  2000  3000  4000

  after:            in hundreds
                 1     2     3     4
```

sn labeling is done using the extended algorithm and each label is scaled by applying a multiplier of hundreds, thousands, etc. and a character identifying the scaling units postpended to the label.

```
Example:
   before:    0    5000  10000  15000  20000

   after:     0     5K    10K    15K    20K
```

'`axisScale`' must be a character vector with a single item.

01D0 CARG-SC The '`axisScale`' argument is invalid, '`axisScale`' can only be set to '`o`', '`e`', '`s`', or '`sn`'. The default value of '`e`' will be used.
The '`axisScale`' call argument/parameter does not contain a valid value. It must be '`o`', '`e`', '`s`', or '`sn`'. The default of '`e`' will be used.

01D1 CARG-SC The '`axisScale`' argument is not a character vector of length 1.
The '`axisScale`' call argument/parameter must be a character vector containing only one value. Numerical, integer and logical vectors are not supported. Only simple vector structures are support. Correct the value assigned with '`axisScale`' and re-run.

01D3 CARG-AX The value for axisMethod internal variable is out of range 1-5 : <locAxisMethod>
The '`axisScale`' call argument/parameter is translated into the axisMethod variable internally. The value should be between 1 and 6. If out of range an internal programming problem has occured.

01Ex '`staggerLab`' call argument (CARG-SL):

01E0 CARG-SL The '`staggerLab`' argument is not a logical value. Setting '`staggerLab`' to FALSE.
The '`staggerLab`' parameter must be a TRUE or FALSE value. If it is not a logical variable, the call parameter is ignored and the default is used. ***check***

01E4 CARG-SL The maxAreasPerGrp argument is not a numeric value. Setting to the default of 5.
The '`maxAreasPerGrp`' argument must be a numeric value from 1 to 5. The default value is 5.

01E5 CARG-SL The maxAreasPerGrp call parameter is not 5 or 6. Value set to 5.
The '`maxAreasPerGrp`' argument must be either 5 or 6. At the current time only 5 is supported and will enforced.

01Fx '`colSize`' call argument (CARG-CS):
This feature is not implemented in the version. The argument name is reserved for future use and development.

01F1 CARG-CS The '`colSize`' parameter in *pDName* contains NA values in the colSize column: <BadValues>. Values must be numeric and > 0. The columns in the panelDesc data.frame (*panelDesc*) are not value width numbers. <BadValues> lists the values in the column. The width of the glyph column will be calculated. Correct if needed and re-run.

01F2 CARG-CS The '`colSize`' parameter in *pDName* has values for fixed width glyphs. Value(s): <BadValues>. Value(s) are ignored and set to NA. The '`colSize`' parameter in the panelDesc data.frame cannot be used to assign the column width on a glyph column that has already been assign a column width. The value can be NA, "", " " or 0. If any other value that is not numeric, it is invalid. The value is ignored. <BadValues> contains the list of values that will be ignored.

01F3 CARG-CS The '`colSize`' parameter in <pDName> does not contain numeric values : <BadValues> The <pDName> panelDesc data.frame has an a non-numeric value, character, logical, or a numeric with a value of "Inf" in the '`colSize`' parameter column. The <BadValues> list identifies the problem. The value is ignored. Correct and re-run.

01F4 CARG-CS The '`colSize`' entries in *pDName* are out of range (<= 0 or > 200). Values are: <BadValues> The one or more values specified for '`colSize`' in the *pDName* data.frame are out of range. The '`colSize`' value will be ignored.

01F5 CARG-CS The reviewed '`colSize`' parameter in *pDName* has bad values (reported above) and have been replaced by the mean of the good values: <meanColSize>, Bad Values: <BadValues>
The "colSize" panelDesc parameter row as a bad value, see message <BadValues>.

01F6 CARG-CS The '`colSize`' parameter in *pDName* contains no useful information and will be ignored. The information provided in the colSize argument in the panelDesc data.frame has no usable values or information. The argument will be ignored.

01Gx Regional Features:

01G0 CARG-RB The regionsB call argument is not a logical variable. The default of FALSE will be used.
The regionB call parameter must be TRUE or FALSE. FALSE indicates the region boundaries should not be drawn, while TRUE indicate the should be drawn as required.

01G5 ARG-DRO The dataRegionsOnly call argument is not a logical variable. The default of FALSE will be used.
The dataRegionsOnly call parameter must be TRUE or FALSE. FALSE indicates the all regions will be used and drawn. TRUE indicates only the regions with areas with data in the statsDFrame data.frame will be drawn. Regions the do not have areas with data will not be drawn. This is a means of including multiple regions in a border group and only drawn sub-regions as required by the data.

01Kx ‘`colors`’ call argument (COLORS):

01K0 COLORS An invalid single value is provided for the ‘`colors`’ argument. It must be ’BW’, ’greys’, or ’grays’. The argument is ignored.
In the micromapST function call, a single value was provided for the ‘`colors`’ parameter. The value must be *BW*, *greys*, or *grays*. If not, the parameter is ignored.

01K1 COLORS The colors vector has the incorrect number of items. It must have 1 or 24 items. <number items> provided.
In the micromapST function call, a single value or a vector 24 values can be supplied. If a different number of values are provided, the parameter is ignored and the default colors are used.

01K2 COLORS The colors vector type is invalid. It must be a character vector.
In the micromapST function call, the ‘`colors`’ parameter contains non-character values. The ‘`colors`’ contained in the vector must be character strings identifying the colors to be used for the 24 color parameters used by micromapST. Numerical values related to the current color palette are not acceptable.

01Mx Boundary datasets:

01M0 BGBG System error encountered when loading the border group. See error message:
The package attempted to load the user/private border group. The R system load() function failed and returned an error message. The error message is logged following this message using the same message indicator. Inspect the file and resolve the R system load() error mesage and re-run. Execution was stopped.
The second like of error message details the loading error issued by R.

01M1 The areaNamesAbbrsIDs (Name Table) is missing from the border group.
The border group must contain the name table (areaNamesAbbrsIDs) to be useful. Determine why the name table is missing and retry.

01M2 The areaVisBorders boundary data set is missing from the border group.
The border group must contain the boundary data for all of the areas in the name table. In this case, the areaVisBorders data.frame containing the area boundary point is missing. Determine why the table is missing and retry.

01M3 The areaVisBorders boundary data set has a NULL value from the border group.
The border group must contain the boundary data for all of the areas in the name table. In this case, the areaVisBorders data.frame containing the area boundary point

has a NULL value. Determine why the areaVisBorders data set is NULL, repair and retry.

01M4 The L3VisBorders boundary dataset is missing from the Border Group.
The Level 3 (map outline) boundary table is an optional boundary points data.frame and is missing from the border group. The package will disable drawing the L3 borders.

01M5 The L2Borders is set to be drawn, but no L2 border data is present. L2 Feature disabled.
The L2VisBorders boundary point data.frame is missing from the border group when the L2Feature is enabled. The L2Feature will be disabled. If you require the L2 boundaries determine why the data.frame is missing and retry.

01M6 The RegBorders is set to be drawn, border data is not present. Reg Feature disabled.
The RegVisBorders boundary point data.frame is missing from the border group when the RegFeature is enabled. The RegFeature will be disabled. dataRegionsOnly options is also disabled. If you require the Reg boundaries determine why the data.frame is missing and retry.

01M7 The L3VisBorders is NULL. Ploting the L3 outline is disabled.
The L3 boundary data found in the border group is empty. The drawing of the L3 outline boundary has been disabled. This may be a normal situation and only needs attention if the L3 boundary data was expected to be in the border group. Correct the problem and recreate the border group.

01M8 BGBN After loading <BorderGroupName> border group, the following optional objects are missing: <list of data.frames>
After loading the border group specified in the function call (<BorderGroupName>), the following optional objects (data.frames) were missing in the border group: <list of data.frames>. These are optional and any use of them has been disabled. If you need one of these boundary data.sets, contact the border group builder and determine why these data.frames were excluded. The optional boundary data.frames are: L3VisBorders, L2VisBorders, and RegVisBorders.

01M9 BGBN After loading <BorderGroupName> border group, the following critical objects are missing: <list of data.frames>
After loading the border group specified in the function call (<BorderGroupName>), the following critical objects (data.frames) were missing in the border group: <list of data.frames>. These boundary data.frames are required for the micromapST package to run. The execution of the package has been stopped. Contact the border group builder and determine why these data.frames were excluded. The optional boundary data.frames are: areaVisBorders, areaNamesAbbrsIDs, and areaParms.

01MA L2Border was requested, but the Name Table does not have a L2_ID column. Disable drawing of L2.
The border group indicates a Level 2 boundary should be drawn where appropriate. However, the name table does not have the L2_ID column to like the Level 2 boundaries to the basic areas. Correct the border group's name table, rebuild the border group and try again.

01MB Regional boundaries have been requested, but the Name Table has no regID column. The feature is disabled.
The border group indicates the Regional boundary and the regional mapping fea-

ture has been requested, but the Name Table has no regID column. This feature is disabled.

01Nx 'details' call argument (DETS):

The 'details' call argument/parameter contains a named list of micromapST parameters values the user wants to override or change. Most of the parameters relate to minor functional variations in how the glyphs are drawn. Several other control the details on how the graphic page is layed out and should not be changed under normal situations. Changes of these variable can cause unpredictable results.

Future: The *panelDesc* data frame has been extended to support glyph parameter modifications on a per column per glyph basis and is the preferred way of working with the glyphs variables. Any changes made via the 'details' parameter are global changes and will effect all of the glyphs.

01N0 DETS The <tag> does not have a valid value: <value> Check type <method> used.

The 'details' call argument/parameter must be a named list structure and the type of each named variable on the list must match the master table. If it is not, 'details' call parameter is ignored.

01N1 DETS The 'details' parameter is not a list.

The 'details' call argument/parameter must be a list structure. If it is not, 'details' value is ignored.

01N2 DETS variable: <name> not found in master variable list. Name is not valid, skipped.

All of the variables used in the 'details' call argument/parameter name list must be on the master detail variable list. See the documentation section on the micromapDefSets and the micromapSTSetDefaults for the variable description. If the variable is not present, it will be deleted and ignored.

01N3 DETS Zero length variable name found in the 'details' list after the *<lastVarName>* **variable.**

This error frequently occurs when the 'details' parameter list contain two commas and no name/value. The position of the two commas is preceded by the <lastVarName> variable in the list. The empty variable/list will be ignored, but please remove the extra comma for future runs.

01NA DETS The *Id.Dot.pch* **variable is can only be set to a value in the range from 1 to 25. Using the default of 22.**

The *IdDotpch* variable is the symbol code for the character preceeding the labels in the id glyph. Only pch values of 1 to 25 are supported.. The variable is ignored and the default character value of 22 is used.

Over time each more 'details' variable will be checked and additional error message will be provided.

*** What other 'details' variable should be validated?

The *panelDesc* is being enhanced to allow appropriate variables to be defined on a glyph graphic column basis. As this feature becomes available the documentation will be updated.

02xx panelDesc and glyphs data messages:

As micromapST setups to create the linked micromaps, each glyphs preforms additional validate on the *panelDesc* variables and the associated *statsDFrame* data.

The message identifier (4 alphanumeric) in the message is followed by the glyph name (type) and the *panelDesc* panel column number (1 through "N"). This helps the user quickly identify which glyphs column in the *panelDesc* definition is related to and therefore the data in the *statsDFrame*. The text of the message with specific information to pin point the problem.

For the map, mapcum, mapmedian, maptail and id glyphs there are no additional data from the user and no additional validation is required.

For the arrow, bar, dot, dotsignif, dotse, dotconf, ctrbar, normbar, segbar, and scatdot glyphs uses data in the *statsDFrame* data.frame and use the *col1*, *col2*, and *col3* named lists for the panel column to identify the *statsDFrame* columns with the required data. The *col1*, *col2*, and *col3* pointers are validated and the data in the *statsDFrame* data.frame is inspected based on the type of glyph requested. The messages use the 3rd digit to identify which *panelDesc* variable was validated. 1 = *col1*, 2 = *col2*, 3 = *col3*. The name of the *panelDesc* named list is also included in the message.

For the segmented stacked bar glyphs (CTRBAR, SEGBAR, NORMBAR), *col1* and *col2* specify the first and last *statsDFrame* column of the contiguous set of columns to use for the data for the segment values/sizes. The message related to the inspection of these columns use the 3rd digit of the message number to indicate the relative column number, 1 being the first column of data up to a maximum of 9 for the last column of data. The column name is also included in the messages to help.

The following is a summary of the usage of the *col1*, *col2*, and *col3* variables by each glyph:

Variable Usage by glyph:

glyph		col1	col2	col3
dot	=	dotvalue	-	-
bar	=	bar length/value	-	-
arrow	=	start value	end value	-
dotsignif	=	dot value	p_value	-
dotse	=	dot value	standard error	-
scatdot	=	x coordinates	y coordinates	-
ctrbar	=	first data column in set	last data column in set	-
normbar	=	first data column in set	last data column in set	-
segbar	=	first data column in set	last data column in set	-
dotconf	=	dot value	lower confidence	upper confidence

The *col1*, *col2*, and *col3* variables are not used by the map, mapcum, maptail, mapmedian, id, boxplot, ts, and tsconf glyphs.

The 3rd character in the message identifier *panelDesc* data column (*col1*, *col2*, or *col3*) pointer related to the warning message. The glyph graphic column number associated with the warning is provided in the <cc> text of the message.

02yx *panelDesc* **parameter checks for** *panelDesc* **variables:**

This top level checks for the values provided in the *col1*, *col2*, *col3* variables are done by the internal function CheckColx and include:

a) If does not exist, create dummy vector to for variable

b) Verify type of vector is numeric, integer, or character. All other types not permitted.

c) Fill empty string items with "NA" where appropriate.

The rest of the checking is done in the glyphs based on which variables needed.
*** Note, the use of <PDvar> appears to be a duplication of the *xx* variable in the message, is this true? The following warnings relate to the *panelDesc* variables in general:

0205 <gNameList> The length of the glyph type list is different the length of the variables list. The number of glyph graphs indicated in the panelDesc$Type column is different than the number of data columns for each glyph variable (col1, col2, etc.) Even if the data column is not used by the glyph, the variable columns all must be the same length as the panelDesc$Type column.

020A <gName> <pdColNum> The first column name/number (<stColName1>) must proceed the last column name/number (<stColName2>) in the <sDFName> data frame. *** More Information to be provided later. ***

020B <gName> <pdColNum> The first column name/number (<stColName1>) must proceed the last column name/number (<stColName2>) in the <sDFName> data frame. *** More Information to be provided later. ***

02y0 <gName> <pdColNum> The first column name/number (<stColName1>) must proceed the last column name/number (<stColName2>) in the <sDFName> data frame. *** More Information to be provided later. ***

02y1xx <gName> <pdColNum> The number of segments is <value>. It must be between 2 and 9. If over 9, only the first 9 will be used. *** More Information to be provided later. ***

02y1 PDCOL <glyph> cc The column name/number of <PDColOrig> in <PDVar> does not exist in the statsDFname data.frame. The column name or number of <PDColOrig> in <PDVar> does not exist as a column name in the statsDFrame data.frame, is a negative or zero valued integer, is not an integer, or is an integer with a value greater then the number of columns in the statsDFrame data.frame.

It appears the "y" is the col"y" indicator and "cc" is the glyph column number. If so, the "y" is redundent.
The following warning deal with the column name or number specified in the *panelDesc* data.frame variables *col1*, *col2*, and *col3*:

02y4 <glyph> cc No *<statsDFrame>* **column was specified in** *<PDVarName>* **in the** *<panelDesc>* **panelDesc data.frame. A data column name/number is required.** *<usage>*
The <glyph> function for column *<cc>* requires a valid *<statsDFrame>* data.frame column be specified in the *panelDesc* *<PDVarName>* variable. The value is missing. Correct the *<PDVarName>* for column *cc* and re-run.

02y5 <glyph> cc The required panelDeac variable *<pdVarName>* **is missing from the** *<pdDFname>* **data frame.** *<usage>*
When using the "<glyph>" in glyph column "cc", requires the *<pdVarName>* variable to locate the data in the *<statsDFrame>* data frame. This is general related to the *col1*, *col2* and *col3* variable lists in the *panelDesc* data frame. Check the glyph descriptions and make sure all of the panelDesc variables required for the glyph are provided.

02y6 <glyph> cc The specified column name or number in *<pdVarName>* **panelDesc variable (<stColName>) does not exist in** *<statsDFrame>* **data frame or is out of range.** *<usage>*
The <glyph> determined the *statsDFrame* column name or number provided in the *<pdVarName>* does not exist or is out of range. If a column name was provided,

it did not match any column names in the *<statsDFrame>* data frame. If a column number was provided, it has a value < 0 or greater than the number of columns in the *<statsDFrame>* data frame. Verify the column name or number in the panelDesc column variable and re-run.

02y7 <glyph> cc The data provided in the *<stColName>* **column of the** *<statsDFrame>* **data frame contains one or more non-numeric entries.** *<usage>*
For the glyph <glyph> in panel column *cc*, the data provided in the *<stColName>* are character vectors and one or more values contain non-numeric characters. This may result in an improper translation to numerics. The entries are treated as NA values. Inspect the data in column *<stColName>* and correct.

02y8 <glyph> cc The data provided in the *<stColName>* **column in the** *<statsDFrame>* **data frame contains one or more entries that could not be converted to numeric values.** *<usage>*
For the glyph *<glyph>* in column *cc*, an attempt was made to convert the character data in the *<stColName>* column of the *<statsDFrame>* to numeric values. One or more entries did not convert correctly resulting in missing values. Inspect the data in the *<stColName>* column of the *<statsDFrame>* data frame and correct.

02y9 <glyph> cc The *<stColName>* **data column in** *<statsDFrame>* **data frame is not a character or numeric vector.** *<usage>*
The *<stColName>* data column in the *<statsDFrame>* is not a character or numeric column. The glyph requires numerical values to generate the glyph. Inspect the data and correct. Logical. List and Matrix type data is not supported. If the column is character vectors, the character vectors must only contains numerical images and must be able to be converted to numerics to draw the glyph. Inspect the data column and correct.

02yA <glyph> cc The *<stColName>* **data column in** *<statsDFrame>* **data frame does not contain any numerical data. No rows will be drawn.** *<usage>*
The *<statsDFrame>* data frame column *<stColName>* does not contain ANY numerical data to permit drawing of the requested glyph. Inspect the data column and re-run.

02yB <glyph> Col: <cc> The *<pdVarName>* **data column in** *<statsDFrame>* **data frame contains one or more missing values and will not be graphed. <usage>. The following <pdUsage> rows will not be drawn:**
In the <glyph> for column *cc*, the data column in the user provided *statsDFrame* pointed to by the *<PDVarName>* *panelDesc* variable contains a *NA* value. The glyph for that data row will not be drawn. If this is an error, correct the data in the *statsDFrame* data.frame and re-run. The line following this message contains a list of the lines in the *statsDFrame* that contained the missing (NA) values.

02yD <glyph> cc The *<stColName>* **data column in** *<statsDFrame>* **data frame does not contain any data. Data vector has length of zero.** *<usage>*
The *<statsDFrame>* data frame column *<stColName>* has a length of zero. If the *<statsDFrame>* is properly constructed with at least one sub-area, then this should not happen. If this error occurs, check the *<statsDFrame>* to make sure it has not been corrupted and has rows for each sub-area being graphed and mapped.

The following warning related to the data pointed to by the *panelDesc col1*, *col2*, and *col3* variable in the *statsDFrame* data.frame. The "y" indicates which panelDesc col (1, 2, or 3) is being referenced.

02yC <glyph> cc The *<stColName>* **column in** *<statsDFrame>* **data frame contains one or more missing values.** *<usage>*
*** More Information to be provided later. *** Description needed. **checked** arrow, bar, dot, dotsignif, dotconf, ***check***

The following special case that do not reference a particular panelDesc column variable. These are generated by the stacked bar glyphs with regard to the *<statsDFrame>* data and the panelDesc col1 and col2 specifications.:

020A <glyph> cc The first column name/number (*<stColName>***) must proceed the last column name/number (***stColName***>) in the** *<statsDFrame>* **data frame.**
In segbar, normbar and ctrbar stacked bar glyphs, the user provides the location of the first and last data column in the *<statsDFrame>* data frame. Each data column between the first and last columns are used to create the stacked bar glyph. The processing is done from lower column number to higher column number. Therefore, the first column must preceed the last column in the data frame. That is must have a lower column number or a column name that preceeds the last column name in the data frame.

020B <glyph> cc The number of segments is <numSegs>. It must be between 2 and 9. If over 9, only the first 9 will be used.
Stacked bar glyphs use 2 to 9 data columns from the *statsDFrame* data.frame to create segmented stacked bar graphs (CTRBAR, SEGBAR, NORBAR). Each data column represents the length of one segment and is validated for the glyph prior to use. If only 1 segment is specified, no stacked bar glyph is drawn. If more than 9 segments exists, only the first 9 will be used in the segbar, normbar and ctrbar glyph.

020C <glyph> cc The number of segments is <numSegs> At least 2 data columns must be defined.
****Duplicate of 020B Used by segbar, normbar, ctrbar glyphs. ***check*** *** More Information to be provided later. ***

020D <glyph> cc The data provided cannot be centered around the value of <center value>.
The ctrbar glyph is designed to center the stacked bars around the value of zero. The data provided is all higher than or lower than zero. A future feature is planned to allow the center value for the ctrbar glyphs to be specified. Until then the center value is set to 0 (zero). Used by ctrbar glyphs. ***check***

The following warnings are specific to the dot significance glyph:

022P <glyph> cc One or more P_Value data entries in *<stColName>* **are missing.**
The data in the *<statsDFrame>* pointed to for use by the dotsignif glyph (*col2*) for the significance P Value is not in the range of 0 to 1 probability. Review the data and correct the values to a range of 0 to 1. This is only used by the DOTSIGNIF glyph.

022Q <glyph> cc One or more P_Value data entries in *<stColName>* **are out of the range.**
The data in the *<statsDFrame>* pointed to for use by the dotsignif glyph (*col2*) for the significance P Value is not in the range of 0 to 1 probability. Review the data and correct the values to a range of 0 to 1. This is only used by the DOTSIGNIF glyph.

The following warnings are issued during the processing of the sub-area identifiers in the *<statsDFrame>* data frame and the border group name table.

BoxPlot specific processing messages (boxplot data list):

02B1 BOXPLOT cc The *panelData* **value of** *<pDataValue>* **in the** *<panelData>* **data frame does not exist or is not accessible.**

The *panelData* variable in the *<panelDesc>* data frame is used to pass completed data structures to glyphs that cannot be contained in the *<statsDFrame>* data frame. In this case, the variable name provided *<pDataValue>* does not exist or is not accessible by the package. Make sure the name is spelled correctly and exists in the .GlobalEnv (calling) environment. This message is used by the BOXPLOT, TS and TSCONF glyphs. **checked** boxplot,

02B3 BOXPLOT cc The *<pdDataName>* data for the boxplot is not is list.
The data structure must be a list (one per area) of boxplot data lists. With the name of each area the attribute or name of each list entry.

02B4 BOXPLOT cc The *<pdDataName>* structure does not have any name attributes for the boxplot data.
No names, no way to link to the boundary data. Each list in this structed must be named using the area's name that matches the name table.

02B5 BOXPLOT cc The *<pdDataName>* boxplot data is not a valid structure. Must contain 6 boxplot sub lists.
The *<pdDataName>* structure must contain the 6 data lists for each area. Less than 6 or greater than 6 lists were found in the area's list structure. Correct the structure of the list and retry.

02B6 BOXPLOT cc The *<pdDataName>* boxplot data does not contain all of the lists of boxplot function output. Invalid structure.
The *<pdDataName>* structure must contain the following 6 names list for each area: stats, names, out, group, n, and conf. If these named lists are not found, the package considers the data structure invalid and will not attempt to draw the boxplot glyph. Verify the *<pdDataName>* is the correct variable and was properly created by the boxplot function.

02B7 BOXPLOT cc In the *<PDDataName>* boxplot data, the '$name' named list contains one or more missing values (NA).
The '$name' list is used to link the boxplot data to the boundary data. If the data cannot be match with a area, the areas glyph row will not be drawn. Check the boxplot data and ensure each individual boxplot has a valid sub-area name associated with it.

02B8 BOXPLOT cc There are duplicate sets of boxplot data in <boxnam> for the same area. Only the first one will be used.
The '$name' list in the boxplot data contain duplicates sub-area names. Only one boxplot can be drawn per area. The first set of data will be used to draw the boxplot. Any other data with the same name will be ignored. If a area does not have any boxplot data, that area's glyph will be omitted.

02BA BOXPLOT cc The $stats matrix in the <boxnam> boxplot data does not have 5 values per area list is not the same length as the $name list.
Need description

02BB BOXPLOT cc The $stats matrix in the *<pcDataName>* boxplot data must *<bpNumNames>* elements.
*** More Information to be provided later. ***

02BC BOXPLOT cc The $stats matrix in the *<pcDataName>* boxplot data has missing values. Areas with missing values will not be drawn.
*** More Information to be provided later. ***

02BD BOXPLOT cc The sub-area names/abbrevations in the *<pdDataName>* boxplot data $names values do not match the border group names: *<nameList>*.

*** More Information to be provided later. ***

02BE BOXPLOT cc 02BE BOXPLOT cc There are one or more of rows in the *<statsDFrame>* that does not have matching boxplot data, (*<pdDataName>*) entries.

*** More Information to be provided later. ***

02BF BOXPLOT cc 02BE BOXPLOT cc The missing sub-areas are: *<AreaList>*

*** More Information to be provided later. ***

For the time series (TS) and time series with confidence band (TSCONF), data for the graphics is provided through a separate data structure via the panelData column in the *panelDesc* data.frame. The *statsDFrame* data.frame is not used and the *colx* indexes are not used.

02D1 <glyphs> cc The *panelData* value of *<pDataValue>* in the *<panelData>* data frame does not exist or is not accessible.

The *panelData* variable in the *<panelDesc>* data frame is used to pass completed data structures to glyphs that cannot be contained in the *<statsDFrame>* data frame. In this case, the variable name provided *<pDataValue>* does not exist or is not accessible by the package. Make sure the name is spelled correctly and exists in the .GlobalEnv (calling) environment. This message is used by the BOXPLOT, TS and TSCONF glyphs.

The following warnings are issued during the processing of the Scatter Dot glyphs using the *<statsDFrame>* data from the x and y coordinates of the points for each geographic area. These points are also used to determine the slope of the "DIAGONAL" line and the "LOESS" line included in the graphic.

02N1 The following parameter names were presented in the 'panel=list()' for column cc for SCATDOT : <List of Invalid Names>. Ignoring bad entries. Suggest correcting.

Only "line", "line.col", "line.lty", "line.lwd" and "f" labels are permitted in the parm list entry for a Scatter Dot glyph. Any other label will be ignored and processing will continue. Please remove or correct invalid names. The format for specifying these options is list(line="LOWESS", line.col="red", f=.5).

02N2 The SCATDOT line style specified <Line Style> is not "NOLINE", "DIAGONAL", or "LOWESS". The default value of <Default Line Style> will be used.

The line style specified in the parm=list() list for this scatdot glyph is not "NOLINE", "DIAGONAL", or "LOWESS". One of there values must be specified. The default of "DIAGONAL" will be used.

02P3 <glyph> cc There are no links in the *<PDDataName>* data to the following *statsDFrame* areas: <list of *statsDFrame* areas.

Sub-areas in the *statsDFrame* do not have data in *<PDDataName>*. *** More Information to be provided later. ***

02P4 <glyph> cc The following area links in *<PDDataName>* data do not link to *<statsDFrame>* areas and will not be used: <listOfNames>

Areas in the *<PDDataName>* do not have links to the *<statsDFrame>* data.frame. *** More Information to be provided later. ***

02P5 <glyph> cc There are no sub-area links provided in *<PDDataName>*. Cannot link data to the *statsDFrame* data.frame.

<PDDataName> does not have values to link to *<statsDFrame>* or the name table. *** More Information to be provided later. ***

02P6 <glyph> cc There are duplicate sub-areas in the *<PDDataName>* structure. Only the first match will be used.
<PDDataName> has duplicate entries. *** More Information to be provided later. ***

Only validate panelData entries for boxplot, ts and tsconf glyphs. For boxplot, datatype must be list. For ts and tsconf, datatype must be 3 dimensional array.
Implementation: *statsDFrame* and the sorted link table drives the glyph generation. Must do this to match up with the other columns. If no data is in *<PDDataName>* for the *statsDFrame* link, no glyphs is drawn for the row.
If duplicates exist in *<PDDataName>* only the first will be used.
If extra entries exist in *<PDDataName>* that don't match the *statsDFrame*, even if they match the name table, they are not used.
TS and TSConf specific processing:

02T1 TSxxxx-15 cc The variable name specified in the panelData row does not exist.
The name of the array containing the area information, the series and the point information (X, Y, HighY and LowY) values does not exist. The glyph will not be drawn. Check for a misspelled variable or incorrect name and retry.

02T2 TSxxxx-02 cc The data structured passed in the panelData field is not an array. Structure name = <dataNam>.
The data structure provided the additional information to construct the time series graphs MUST be a 3 dimension array. Any other type of structure is not acceptable.

02T3 TSxxxx-03 cc The time serial array's 1st dimension is not <numRows> areas to match statsDFrrame. It is <dimDArr[1]>.
The time series array does have enough 1st dimensional values to match the number of rows in the statsDFrame and the Name Table.

02T4 TSxxxx-04 cc The time serial array's 2nd dimension must have at least 2 values (time periods 2 to "n"). It is <dimDArr[2]>.
The time series must cover at least 2 or more time periods. So, the 2nd dimension must have a dimension of at least 2.

02T5 TSxxxx-05 cc The time series array's 3rd dimension is not 4 values for TSConf (X, Y, LowY and HighY). It is <dimDArr[3]>.
See note on message "02TA" below.

02T6 TSxxxx-06 cc The time series array's 3rd dimension must be 2 or 4. It is <dimDArr[3]>. (TS: X & Y, TSCONF: X, Y, LowY, HighY.)
See notes on message "02TA" below.

02T7 TSxxxx-10 cc rowNames on array do not match area ID list. The bad area IDs are: <List of Name Table Keys>.
In processing the time series array, several location IDs cannot be found in the Name Table. They are listed in the <List of Name Table Keys> string. Determine the reason for the incorrect values, correct and retry.

02TA TSxxxx-12 cc The time series array\'s 3rd dimension must be at least 2. It is <number>.
The 1st dimension represents the geographical areas of the Time Service. The 2nd dimension represents the time of the observation (0 to "N"). The area's points are sorted by the X value provided in the 3rd dimension. The 3rd dimension of the array are is the observed value on the graph for the area at the time. For standard TS graphs this is x and y values on the chart. For a TS-conf graph, the 3rd dimension contains

the x and y values and the confidence interval HighY and Low Y values. Correct the content of the 3rd dimension and retry.

02TB TSxxxx-14 cc The time series array does not have rownames (location IDs) assigned to the 1st dimension. Data cannout be paired up with areas.
The row.names tied to the 1st dimension of the Time Series array are required to be able to link each time series with it geographic area. Make sure the values used for the 1st Dimension match the select type of location IDs in the Name Table.

03xx border group elements:

0310 Missing the areaVisBorders dataset in the border group specified. In the border group specified, there is not area boundary data information provided. Cannot draw the area boundaries. Check the contents of the border group, rebuild or repair.

0311 Missing the L2VisBorders dataset in the border group specified. In the border group specifeid, there is no level 2 boundary data information provided. Cannot draw the level 2 boundaries. Check the contents of the border group, rebuild or repair.

0312 Missing the RegVisBorders dataset in the border group specified. In the border group specified, there is no regional boundary data information provided. Cannot draw any regional boundaries. Check the contents of the border group, rebuild or repair.

0313 Missing the L3VisBorders dataset in the border group specified. In the border group specified, there is no outer boundary data information provided. Cannot draw the outer boundaries. Check the contents of the border group, rebuild or repair.

04xx panel layout calculations:

042x column sizing calculations:

0420 PANEL Calculated column widths is less than minimum <colSizeMin> inches - too many columns specified.
*** More Information to be provided later. ***

0421 PANEL Column width is to small to be useful, Package stopped.
*** More Information to be provided later. ***

043x row sizing calculations:

0430 PANEL panelLayout - The calculated width of <calculated-width> is too large for the available space of <space-width>.
*** More Information to be provided later. ***

0431 PANEL panelLayout - The calculated GrpRow Height is too small to be used.
*** More Information to be provided later. ***

0432 PANEL panelLayout - The calculated GrpRow Height is <GrpRowHeight> inches. The minimum size limit is: <minimum row height>.
*** More Information to be provided later. ***

045x panel functions:

0450 PANEL panelLengthen - invalid vector length. < 2
*** More Information to be provided later. ***

0451 PANEL panelSelect - Dimension error. Program error - index 'i' or 'j' is out of bounds.
*** More Information to be provided later. ***

0452 PANEL panelSelect - Bad label region name. Must be left, right, top or bottom.
*** More Information to be provided later. ***

049x Internal Package Messages:

0490 DMP Error in axisMethod selection in Dot and DotSignif code.
*** More Information to be provided later. ***

0491 INB is.between.r The r range value is not a vector with length of 2. FALSE returned.
Function is.between.r — Need description **checked** *** More Information to be provided later. ***

05xx Informational:

050x Panel Messages:

0501 PANEL Number of parameters overlaid = <numOverlaid.
*** More Information to be provided later. ***

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

`micromapGDefaults` *micromapGDefaults data.frame structure*

Description

The micromapGDefaults data.frame provides all of the detailed structure, colors, sizing, font sizes, separation distances, line weights and types, spacing, etc. required to physically construct the requested micromapST graphic in portrait or landscape modes from a letter size (8.5 x 11) up to a tabloid (11 x 17) page. The data.frame is mainly used internal to micromapST, but a copy can be obtained by a user when a large number of changes are required. This is not recommended. The primary purpose of this section is to provide a list and description of many of the variables in the details list that can be used to enable or disable functions of micromapST and its glyphs. These internal variable are identified by a "*" after the variable name. These are the only variables that generally safe to be modified by the user.

Details

The data.frame contains two lists: colors and details.

colors The colors vector is the name of a color palette or a vector of 12 or 24 color names or values ("#xxxxxx" or name). The first twelve (12) colors are used to link the areas to the glyphs. The second 12 colors are used with the Time Series glyphs when transparent colors are required for the confidence band. The vector defines the 12 colors and their transparent equal are:

- The 6 colors in each group for the states/areas and symbols in the glyphcs. One color per row (area). The 6th color is not used at this time.
- 1 color for the median state and glyphics and is generally black,
- 1 foreground color for highlighted states in the map. This is used to highlight states already referenced previously or have meaning depend on the type of map requested. The usage is as follows:

```
"map"       - not used.
"mapcum"    - highlight states previously
              referenced above (a previous
              group/row).
"maptail"   - highlight states previously
              referenced above the median row
              and highlight remaining states not
              featured below the median row.
"mapmedian" - highlight all states not featured
              above the median in maps above the
              median row and highlight all states
              not featured below the median in maps
              below the median row.
```

- 2 colors to represent non-featured areas above the median row and below the median row.
- 1 color to fill in non-referenced areas on the map. These are areas in the border group, but the user has not provided any data row in the statsDFrame supplied in the *micromapST* function call.
- 1 color to fill in non-active area. That is an area that in not referenced in the name table and can't be matched to any user data.

The additional 12 colors are the same colors defined above but modified 20% tranparency to provide a set of "transparent" colors for confidence graphs like the time series. This is done via the adjustcolors(colors,0.2) function. Only the first 6 of the transparency colors are used. The other 6 colors are reserved for future requipments.

If colors parameter can also be set to a single value to enable black and white color schemes. The acceptable values are; "greys" or "grays" or "bw". When specified, the entire plot will be done using the packages standard black/white/gray shades designed to support b&w duplication, color blindness and non-color publication.

Additional color palettes may be supported in future releases.

The package default 24 colors will be used:

- 5 state colors: "red", "orange", "green", "greenish blue", "lavender", "magenta",
- 1 median state color: "black",
- 1 highlighted area: "light yellow",
- 2 above and below median highlighted areas: light red, light blue,
- 1 color for non refernce (used) areas in the data,
- 1 color for non-active areas in the border group,
- 12 translucent colors using the above colors at 20%.

It is strongly recommended to use the default. When changing the colors list, then the entire list must be specified.

details is a list structure that contains the internal variables and values used by micromapST to create the graphics structure layout and guide the operations of the *micromapST* function. The details internal variables provide a way to tune the look of the created link micromap and its glyphs. These internal variables are divided into two groups: General and Advanced. The general variable don't affect how the panels are constructed, but allow you to change the looks of the graphics: dot, shapes, colors, line weights, etc. The advanced variable affect the

structure of the panels and how areas are presented. The following internal variable accessible through the details named list are grouped by general usage and their glyph types.

To change the values of items in the details list, only the variable(s) requiring change need to be specified as a list for the details=list() parameter in the call. In general, the beginning of the variable names indicates the glyph or glyph group the variable is associated, in most cases.

```
Arrow.    -> arrow glyph
Bar.      -> bar glyph
BoxP      -> boxplot glyph
CBar      -> ctrbar glyph
CSNBar    -> ctrbar, segbar and normbar glyphs.
Dot.      -> dot, dotconf, and dotse glyphs
Dot.conf. -> dotconf glyph
Dot.Signf -> dot with significance overlay.
Grid      -> grid elements of all glyphs
Id.       -> id glyph
Map.      -> map glyphs
Panel     -> general glyph panel
Rank      -> area ranking glyphs.
Ref       -> Reference text and line
SCD.      -> scatter dot glyph
SNBar     -> segbar and normbar glyphs
Title     -> page and column labels and titles
TS        -> ts and tsconf glyph
TSconf    -> tsconf glyph
```

For example: to turn off the midpoint dot in the segmented bar glyphics, all that is required is:

```
details = list(SNBar.Middle.Dot=FALSE)
```

General Variables: The following are the internal variables for the XAxis, Grid, Panels, Reference Line, and the Glyphs.

X-Axis variables: **XAxis.Sp.mcex** = 0.2 Size used for XAxis space between lines of labels

XAxis.indent = 10 in 1000th of an inch. First and Last label indents from edge.

XAxis.nGridpIn = 3.4 labels per inch in XAxis - initial objective.

XAxis.gapPC = 0.75 (*100 for precentage). Percentage of the width of a space used to determine label overlaps.

Y-Axis variables: **YAxis.width** = 0.2 inches. Extra column gap size required to support drawing an Y-Axis for TS and ScatDot glyphics

Grid and Panel Variables: **Grid.Line.col** = "white" Grid line color

Grid.Line.lwd = 1, Grid line width

Panel.Fill.col = "#676767FF", defaults to light gray

Panel.Outline.col = "black", color of panel outlines

Reference Line and Text: The following variable related to the reference text and line feature.

Ref.Val.lty = "dashed", set reference value line to dashed
Ref.Val.lwd = 1.5, line width of reference line
Ref.Val.col = "midgreen", color of reference line when color is used.
Ref.Val.BW.col = "black", color of reference line when grays are used.
Ref.Text.col = "black", color of reference line text when color is used.
Ref.Text.BW.col = "black", color of reference line text when grays are used.

Arrow Glyph: The following variables are used by the '`arrow`' glyph.

Arrow.lwd = 2.5, line width of arrow.
Arrow.cex = 0.08, size of arrow (not implemented)
Arrow.Head.length = 0.08, length of arrow.
Arrow.Dot.pch = 21, arrow-dot symbol 19-25.
Arrow.Dot.pch.size = 0.9 cex, arrow-dot size.
Arrow.Dot.pch.lwd = 0.5, line weight used on filled arrow-dot symbols.
Arrow.Dot.Outline =FALSE, include dot outline when filled. FALSE=NO, TRUE=YES
Arrow.Dot.Outline.col = "black", color used for arrow-dot outline.
Arrow.Dot.Outline.lwd = 0.5, line weight for arrow-dot outline.

Bar Glyph: The following variables are used by the '`bar`' glyph.

Bar.barht = 2/3, fraction of line height for bar. Should never be > .90. Usable range is 0.333 to 0.90
Bar.Outline.col = "black", color of bar outline.
Bar.Outline.lwd = 0.5, line width for bar outline
Bar.Outline.lty = "solid", line type for bar outline

Boxplot Glyph: The following variables are used by the '`boxplot`' glyph.

BoxP.thin = 0.2, line width of box.
BoxP.thick = 0.6, thick line width.
BoxP.Use.Black = FALSE, whether to outline the outlier points.

BoxP.Median.col = "black", color of median box
BoxP.Median.Line = 0.80, line width of median line.

BoxP.Median.Dot.lwd = 2, line width for median.

BoxP.Outlier.lwd = 0.4, line width of outlier outlines.
BoxP.Outlier.cex = 0.7, size of outlier dots.
BoxP.Outlier.BW.col = "#4c4c4cFF" color of outliner lines when greys used.

Dot, Dotconf, and DotSE glyphs: The following variables are used by the '`dot`', '`dotconf`', '`dotsignif`', and '`dotse`' glyphs.

Dot.pch = 21, solid circle (S compatible).
Dot.pch.size = 0.9 cex, size of dot.
Dot.pch.lwd = 0.5, linen weight for dot outline when 0:18 dot used.

Dot.Outline =FALSE, whether to outline the dots.
Dot.Outline.col = "black", color of dot outline.

Dot.Outline.lwd = 0.5, line width of dot outline.

Dotconf Glyph: In addition to the variables listed above, the '`dotconf`' glyph also has the following variable.

Dot.Conf,pch = 21, solid circle (S compatible).

Dot.Conf.pch.size = 0.9 cex, size of dot.

Dot.Conf.pch.lwd = 0.5, linen weight for dot outline when 0:18 dot used.

Dot.Conf.Outline =FALSE, whether to outline the dots.

Dot.Conf.Outline.col = "black", color of dot outline.

Dot.Conf.Outline.lwd = 0.5, line width of dot outline.

Dot.Conf.lwd = 2, line width of confidence interval lines.

Dotse Glyph: In addition to the variable defined above for the '`dot`', '`dotconf`', '`dotsignif`' and '`dotse`' glyphs, the '`dotse`' glyph also has the following variable define.

Dot.SE,pch = 21, solid circle (S compatible).

Dot.SE.pch.size = 0.9 cex, size of dot.

Dot.SE.pch.lwd = 0.5, linen weight for dot outline when 0:18 dot used.

Dot.SE.Outline =FALSE, whether to outline the dots.

Dot.SE.Outline.col = "black", color of dot outline.

Dot.SE.Outline.lwd = 0.5, line width of dot outline.

Dot.SE = 95, percent confidence interval

Dot.SE.lwd = 2, line width of confidence interval lines.

dotsignif Glyph: In addition to the variable defined above for the '`dot`', '`dotconf`', '`dotsignif`' and '`dotse`' glyphs, the '`dotsignif`' glyph also has the following variable define.

Dot.Signif.pch = 4, overprint character "x" when not significance.

Dot.Signif.pch.size = 0.9*1.2 cex size of the overprint character.

Dot.Signif.pch.lwd = 0.5, linen weight for dot outline when 0:18 dot used.

Dot.Signif.pch.col = "black", color of overlaid symbol on DOT.

Dot.Signif.Outline =FALSE, whether to outline the dots.

Dot.Signif.Outline.col = "black", color of dot outline.

Dot.Signif.Outline.lwd = 0.5, line width of dot outline.

Dot.Signif.pvalue = 0.05, p-value for testing significance.

Dot.Signif.range = c(0,1), valid range for significant test data for p-value

id Glyph: The following variables are used by the '`id`' glyph.

Id.Cex.mod = 1, fudge adjustment for ID Text./cr

Id.Title.1.pos = 0.9 inches, top panel 1st line id title placement above the first panel, used with lab1

Id.Title.2.pos = 0.1 inches, top panel 2nd line id title placement above the first panel, used with lab2

Id.Text.cex = 0.65, text side of ID column

Id.Dot.pch = 22, pch symbol value to plot next to state name/abbrev.

Id.Dot.cexm = 1.5, size of dot symbol for state ID

Id.Dot.lwd = 0.8, size of solid dot symbol for state ID

Id.Dot.width = 0.1 inches. With of the ID symbol (box)

Id.Space = 0.03125, width of a space in inches.

Id.Start = 0.055, offset from left for start of ID column.

map, mapcum, maptail, and mapmedian Glyphs: The following variables are used by all of the "map" type glyphs.

Map.Area.Spec.cex = 0.32, font size for state labels

Map.Bg.col = "#262626FF", (grey(0.88)) color of background (not active) sub-areas fill in maps

Map.Bg.Line.col = "white", background of maps

Map.Bg.Line.lwd = 0.3, line weight for map background boundaries

Map.Fg.Line.col = "black", foreground color of maps

Map.Fg.Line.lwd = 0.3, line weight for map foreground boundaries

Map.L2.Line.col = "lighter grey", color of Layer 2 outline in maps

Map.L2.Line.lwd = 0.35. line weight for Layer 2 boundaries

Map.L3.Line.col = "black", color of Layer 3 (national) outline in maps

Map.L3.Line.lwd = 0.4. line weight for Layer 3 (national) boundaries

Map.Lab.Box.Width = 0.09, width in inches of box symbols using in titles for maps

Map.Max.width = 2.5 inches, maximum width of each map

Map.Min.width = 1.5 inches, minimum width of each map

Map.Median.text = "Median for Sorted Panel", text used in single row median panel instead of a map.

Rank Glyph: The following variable is used by the '`rank`' glyph.

Rank.width = 0.25 inches - column fixed width

Rank.method = 1, rank method - to be defined.

Scatter Dot Glyph: The following variables are used by the scatter dot glyph ('`scatdot`').

SCD.Axis,cex = 0.52, font size for Y axis labels for scatter dots

SCD.Bg.pch = 21, type of point/symbol to be used for background data points (not active) - state's dots.

SCD.Bg.pch.fill = "transparent", fill color for not selected state's dots.

SCD.Bg.pch.col = "black", border color used for non-active data points.

SCD.Bg.pch.lwd = 0.6, line width of outline of point/symbol used as non-active data points.

SCD.Bg.pch.size = 0.75, size of point/symbol used as non-active data points.

SCD.Fg.pch = 21, type of point/symbol for active data points.

SCD.Fg.pch.col = "black", border color for foreground dots.

SCD.Fg.pch.lwd = 0.6, Scatter dot symbol outline line weight for active data points.

SCD.Fg.pch.size = 1, size of point/symbol for active data points in scatter dots.

SCD.Median.pch = 21, shape of filled symbol for median value - scatter dots.

SCD.Median.pch.col = "black", border color for median symbol - scatter dots.

SCD.Median.pch.lwd = 0.6, line width used on the median symbol - scatter dots.

SCD.Median.pch.size = 1, symbol size median value - scatter dots.

SCD.Median.pch.fill = "black", color of filled symbol for median value - scatter dots.

SCD.hGrid = FALSE, whether or not to include horizontal grid lines in panel.

SCD.xsc = 1.08, x range multiplier to keep dots from being clipped

SCD.ysc = 1.12, y range multiplier to keep dots from being clipped

SCD.DiagLine = TRUE, whether or not to include x=y sloped line

SCD.DiagLine.col = colGrid, color of sloped line, default, grid line color. SCD.DiagLine must be TRUE.

SCD.DiagLine.lwd = 1.25, line weight of sloped line, default, grid line color. SCD.DiagLine must be TRUE.

SCD.DiagLine.lty = "solid", line type of sloped line, default, grid line color. SCD.DiagLine must be TRUE.

Segmented Stacked Bar Glyphs: micromapST support three types of stacked bar graphs: Centered, Segmented, and Normalized. The following section describes the internal variable used by all of these glyphs and then the unique variables used by each type.

Centered, Segmented, and Normal Stacked Bar Glyphs: The following variables are used by all of the horizontal stacked bar glyphs (‘`ctrbar`’, ‘`segbar`’ and ‘`normbar`’). Any variable for a specific type of stacked bar are listed following this section. The CSN at the beginning of the name of each variable indicates they are part of this group.

CSNBar.barht = 0.66667, fixed height of bar when variable height bars are is not used. Should never be greater than 0.90. Usable range is 0.333 to 0.90.

CSNBar.First.barht = 0.3333, height of first bar when variable height bars are used. Must be less than SBar.Last.barht and in the range of 0.333 to 0.6667, SNBar.varht or CBar.varht must be TRUE for this option of function.

CSNBar.Last.barht = 0.80, height of last bar when variable height bars are used. Must be greater than CSNBar.First.barht and in the range of 0.6667 to 0.90. CSNBar.varht or CBar.varht must be TRUE for this option of function.

CSNBar.Outline.col = "black", color of stacked bar outlines.

CSNBar.Outline.lwd = 0.75, line weight for bar segment outline in segmented bar plots.

CSNBar.Outline.lty = "solid", line type for bar segment outline in segmented bar plots.

Centered Stacked Bar Glyph: The following variables are only used by the ‘`ctrbar`’ glyph.

CBar.varht = FALSE, enables variable height bars.

CBar.two.ended = FALSE, request two ended variable height bars be used

CBar.Center.Line.enable = FALSE, request a line is drawn at center point.

CBar.Center.value = 0, value of the center stacked bar glyph. def=0

CBar.Zero.Line.col = "white", centered bar zero vertical line color

CBar.Zero.Line.lwd = 1, line width for centered bar zero vertical line

CBar.Zero.Line.lty = "dotted", type of centered bar zero line.

SegBar, and NormBar Glyphs: The following variables are used by the ‘`segbar`’ and ‘`normbar`’ glyphs:

SNBar.varht = FALSE, enables variable height bars from SBar.First.barht to SBar.Last.barht.

SNBar.two.ended = FALSE, request two ended variable height bars be used (small to large to small).(Not implemented)

SNBar.Middle.Dot =FALSE, request a dot be draw in at the mid point in the segmented bars.

SNBar.MDot.pch =21, type of point/symbol used as the mid point dot. SNBar.Middle.Dot must be TRUE for this parameter to function.

SNBar.MDot.pch.fill ="white", color of the point/symbol used as the mid point dot. SNBar.Middle.Dot must be TRUE for this parameter to function.

SNBar.MDot.pch.size =0.3, size of point/symbol used as the mid point dot. SNBar.Middle.Dot must be TRUE for this parameter to function.

SNBar.MDot.pch.border.lwd =NA, line width of outline of point/symbol used as the mid point dot. SNBar.Middle.Dot must be TRUE for this parameter to function.

SNBar.MDot.pch.border.col =NA, color of outline of point/symbol used as the mid point dot. SNBar.Middle.Dot must be TRUE for this parameter to function.

Time Series and Time Series with Confidence bands Glyphs: The following variables are used by the time series glyphs ('`ts`' and '`tsconf`'):

TS.lwd = 1.1, time series line weight

TS.Axis.cex = 0.52, font size for Y axis labels

TS.hGrid = FALSE, whether or not to include horizontal grid lines in panel

Advanced Internal Variables: The advanced internal variable should not be modified unless you have lot of time. If any are modify, the operation of the package may become unpredicatable and can't be supported. However, through the use of gentle changes and experimentation, you can modify the look of the resulting micromapST output. While not recommended, the authors felt access to these internal variable will help a user in some unexpected situations.

These variables should not be modified unless absolutely necessary. IF there area modified, the outcome can not be predicted and can't be supported. Modify at your own risk:

Page and Panel Layout: topMar = 1.2, top margin in inches

botMar = 0.5, bottom margin in inches

botMarLegend = 0.75, bottom margin for legend

botMardif = 0.2, bottom margin difference

leftMar = 0.0, left margin in inches

leftMarAxis = 0.2, left margin when Y axis labels and title are required

rightMar = 0.0, right margin in inches

borderSize = 0.5 inches. The border space between the page edges and the margins. This value is used for the top, bottom, left and right border spacing.

Margins and Axis Layout: mgpLeft = c(0.75, 0.1, 0), Left Y axis margin line for axis labels and axis line.

Scaling, Separation, and Padding: pad = 0.67, y axis padding for integer plotting locations

padex = 0.34 inches, total panel padding (i.e., 0.17 at top and bottom of panel)

padMinus = 0.63 inches, spacing to keep reference line off panel edge

Row and Column Parameters: rowSepGap = 0.075 inches. Space between panel groups at or around the median panel. There are 7 units per panel. The average unit is between 1/10 and 1/8 inches.

rowSizeMn = 0.5 inches. Minimum panel group height.

rowSizeMx = 1.25 inches. Maximum panel group height.

rowSizeMin = 1.65 units. Minor panel group height in units for single area panel.

rowsizeMaj = 7 units. Major panel group height for all panels, except single area panel.

colSepGap = 0.75 inches. Glyphics column separator space.

colSizeMin = 0.75 inches. Minimum Glyph column size.

colSizeMax = 2.0 inches. Maximum Glyph column size.

Title Variables: **Title.Line.1.pos** = 1.27 inches, top panel 1st line placement above the first panel, used with lab1

Title.Line.2.pos = 0.64 inches, top panel 2nd line placement above the first panel, used with lab2

Title.Line.2x.pos = 0.01 inches, top panel X-Axis line placement above the first panel

Title.Line.3x.pos = 0.01 inches, bottom panel X-Axis line placement below the last panel

Title.Line.3.pos = 0.64 inches, bottom panel line placement below the last panel, used with lab3

Title.Line.4.pos = 1.27 inches, reference line legend below last panel, used with reftext

Title.Line.5.pos = 0.35 inches, Y axis label placement (to the left of panel), used with lab4

Title.cex = 1.0, text size of title, used with title

Debug Variable: The following variable is reserved for package testing only and should not be used.

MST.Debug = 0, disabled. Do not use.

Unused Variables: The following variables are not implemented and reserved for future use.

(small to large to small). (Not implemented)

rcRatioMin = 0.25, minimum row size to col size ratio permited. (not implemented)

rcRatioMax = 2, maximum row size to col size ratio permited. (not implemented)

mgpTop = c(3.2,0.1,0), Top margin line for X axis labels and axis line.

mgpBottom = c(3.2,0.1,0), Bottom margin line for X axis labels and axis line.

padjBottom = -0.35, Axis tick label placement adjustment

sc = 1.08, x axis scale expansion factor. Applied to the data range to calculate the graph\'s range.

XAxis.staggered = TRUE, enable staggered label feature - NOT USED

XAxis.L.mcex = 0.888889 - actually 0.6667 Size used for large XAxis labels

XAxis.M.mcex = 0.777778 - actually 0.5833 Size used for medium XAxis labels

XAxis.S.mcex = 0.666667 - actually 0.5 Size used for small X Axis labels For line labels - Normal = .75, space = .15 (20%), small space = .15 .5 = .075

XAxis.offset = 0.0 inches. X Axis offset

YAxis.cex = 0.33332 cex size of Y Axis labels

YAxis.offset = 0.0 lines. Offset of Y Axis labels from panel edge.

YAxis.nGridpIn = 5 labels. Number of labels per inche for Y Axis - initial goal.

YAxis.staggered = TRUE. Enable staggered labels on Y Axis - NOT USED.

Ref.Text.cex = 0.75, size of reference line text

Arrow.Shadow.lwd = 4.0, line width of arrow shadow to create outline (not implemented)

Arrow.Shadow.col = "black", arrow shadow color (not implemented)

BoxP.Outlier.pch = 20, symbol for outlier - 19-25.

BoxP.Outline.col = "#262626FF", boxplot outline color

BoxP.Median.Dot.pch = 19, solid circle symbol.

BoxP.Median.Dot.col = "white", color of median dot.

BoxP.Median.Dot.cex = 0.95, size of circles.

Dot.Conf.size = 0.55, size of confidence interval

Dot.SE.size = 0.55, size of confidence interval

Id.Dot.Outline.col = "dark gray", color of outlines of ID symbols

Id.Dot.Outline.lwd = 0.8, line weight for outlines of ID symbols

Map.Bg.Line.lty = "solid", line type for map background boundaries

Map.Fg.Line.lty = "solid", line type for map foreground boundaries

Map.L2.Line.lty = "solid", line type for Layer 2 boundaries

Map.L3.Line.lty = "solid", line type for Layer 3 (national) boundaries

Map.Panel.col = "white", background color of map panels

Map.Unu.col = "lightest grey", color of sub-areas not referenced in maps

staggered = FALSE. Position of last staggered label. FALSE = low, TRUE = high.

Text.cex = 0.75 cex, general text size

The micromapGDefaults data.frame is built by the micromapGSetDefaults function when the micromapST package is called. Once built it cannot be changed. To change one or two (a few) variables, construct a list of these variables and pass it to micromapSEER via the details parameter in the call. To do large scale customization, call the micromapGSetDefaults function to get a copy of the entire data.frame and modify this copy. This is not recommended.

Author(s)

Daniel B. Carr, George Mason University, Fairfax VA, with contributions from Jim Pearson and Linda Pickle of StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapGSetDefaults, micromapGSetPanelDef

`micromapGSetDefaults` *function to build the micromapGDefault data.frame*

Description

The micromapGSetDefaults function generates a data.frame containing two lists: the colors and details list. Each list contains the operational parameters to instruct micromapST on how to physically construct a micromapST graphic.

The colors list contains 24 rgb colors:

The 12 basic colors are:

6 colors for the state maps in each group of 6 areas that make up one panel row. Currently the max is 5 areas per group. The 6th color is reserved for future use.

1 color for the median state (in the middle)

1 color for the cumulative hightlight color for the areas already presented in other panel rows.

2 colors for the when highlighting areas above and below the median.

1 color for use to shade not referenced areas in the user data.

1 color for use to shade not-active areas.

and a repeat of the 12 rgb colors with an alpha transparency value of 10%.

The details list contains the spacing, margins, text size, etc. information to guide the construction of the micromapST graphic. These lists may be modified by the user, but this is not recommended. See the micromapSTDefault description for more information on these lists.

Usage

```
micromapGSetDefaults()
```

Details

The default colors in the colors list are: red, orange, green, greenish-blue, lavender and magenta for the area colors, black for the median state color, and light yellow for the cummulative hightlighting, vary pale red and green for highlight areas above and below the median, the lightest and lighter gray to shade not referenced and not-active areas on the map. The default values in the details list can be found in the micromapGDefaults documentation.

This function is primarily used internally by micromapST. However, if the user wants to see values of all of the internal variables or wants to make wholesale changes to the layout and operational changes to the colors and details list, this function can be used to create a copy of the full colors and details lists.

The return value is a variable equal to list(colors, details) where both colors and details are named lists.

Value

micromapGDefaults

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

---

micromapGSetPanelDef *function to build the micromapGPanelDef data.frame*

Description

The micromapGSetPanelDef function generates a data.frame representing the layout of the columns and row groups in the total graphic. It include the location of each individual graphic, the spacing of the columns and row groups and spacing for the title column headers and trailers and x axis when needed. The resulting structure along with the values from micromapGSetDefaults are copies into the package's memory and used by the "panel functions" to manage the panels during the setup and glyphics creation.

Usage

```
micromapGSetPanelDef(nRows,rSizeMaj,rSizeMin,rSepGap,MaxRows,UGrpPattern)
```

Arguments

`nRows`	for the number of data rows to include in the map.
`rSizeMaj`	the maximum size of a row group in inches. A row group represents a number of data rows as one horizontal set of graphs.
`rSizeMin`	the minimum size of a row group in inches.
`rSepGap`	the size of the spacing between row groups in the output graphic.
`MaxRows`	the maximum number of rows.
`UGrpPattern`	a vector of the number of rows per group. This is specified by the user as a substitute for the calculated row group pattern. The number of rows must equal the number of rows in the data that match the mapping data and must have the rows grouped so that the top and bottom groups have the most rows. The pattern can then slowly dimenish to 2 rows as the group approach the middle (median) row group.

Details

Minor modifications can be made to the presentation structure through the details= call parameter, but this is not recommended. See the micromapGDefault description for more information on these lists.

Value

micromapGSetPanelDef

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

`micromapSEER` *A function to create linked micromaps for the 20 U. S. Seer Registries*

Description

The *micromapSEER* function or the *micromapST* function with the '`bordGrp`' set to *USSeerBG* can be used to create linked micromaps for the 20 U. S. Seer Registries.

Usage

```
micromapSEER(statsDFrame,panelDesc,...)
```

Arguments

`statsDFrame`	- data frame of the data required for creating the graphical glyphics with one row per Seer Area being mapped. Each row is linked to the boundary data using a Seer Area abbreviation or name.
`panelDesc`	- data frame containing information and pointers for each glyphic column to be generated. The data frame specifies the type of glyphic and the columns in the *statsDFrame* data frame that contain the data for the glyphic or the external data structure via the panelData list.
`...`	- the remaining parameters required and used by the *micromapST* function call.

Details

More details to follow.

Value

None

Author(s)

Daniel B. Carr, George Mason University, Fairfax VA, and Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`micromapST` *Linked Micromap Graphics Package*

Description

Provides a easy and quick means of creating Linked Micromaps for any collection of geographically associated areas. The micromapST package uses the standard graphics and RColorBrewer packages to rapidly create highly readable linked micromap plots. This gives the user the ability to explore different views of their data quickly.

micromapST uses the border and name information contained in border group datasets to define the geographical areas used in creating the linked micromaps.

The *micromapSEER* function is included to help users of the specialized *micromapST* for NCI Seer areas called *micromapSEER* migrate to this package. The *micromapSEER* function, calls *micromapST* with the border group set to *USSeerBG* to generate linked micromaps for the 20 U. S. Seer Areas.

The *micromapST* contains border group datasets with the boundary and name information for the following::

- *USStatesBG* - data from the original *micromapST* package for . the U. S. 50 states and the District of Columbia.
- *USSeerBG* - data for the 21 U. S. Seer Areas.
- *KansasBG* - data for the 105 counties in the state of Kansas.
- *NewYorkBG* - data for the 62 counties in the state of New York.
- *MarylandBG* - data for the 24 counties in the state of Maryland.
- *UtahBG* = data for the 29 counties in the state of Utah
- *ChinaBG* - 34 administrative areas in the country of China.
- *UKIrelandBG* - 218 administrative areas in UK, Ireland and Isle of Man.
- *SeoulKoreaBG* - 25 districts in Seoul S. Korea.
- *AfricaBG* - 52 countries of the Africa continent.

Each plot row represents a single sub-area (state, county or province) within the border group area. Each column can be defined to present a different graphical representations of the user's data.
For linked micromaps, the primary columns are a MAP type, ID (sub-area name) and one or more glyphics. The statistical data is presented in the glyphics columns as one of the following glyph types:

- arrows,
- bars,
- boxplots,
- dots,
- dots with confidence intervals,
- dots with standard error,

- dots with a significance marker,
- time series line plots with or without confidence bands,
- scatter plots and optionally diaganol line or lowesse line, and
- horizontal stacked (segmented, centered, and normalized) bars.

All border groups are distrbuted with the package as .rda datasets.

Usage

```
micromapST ( statsDFrame,
                     panelDesc,
                     rowNamesCol = NULL,
                     rowNames   = NULL,
                     sortVar    = NULL,
                     ascend     = TRUE,
                     title      = c("", ""),
                     plotNames  = NULL,
                     axisScale  = NULL,
                     staggerLab = NULL,
                     bordGrp    = NULL,
                     bordDir    = NULL,
                     dataRegionsOnly = NULL,
                     regionsB   = NULL,
                     grpPattern = NULL,
                     maxAreasPerGrp = NULL,
                     ignoreNoMatches = FALSE,
                     colors     = NULL,
                     details    = NULL)
```

Arguments

`statsDFrame` a data.frame containing data used with the following plots/glyphs:. *arrow*, *bar*, *segbar*, *normbar*, *ctrbar*, *dot*, *dotse*, *dotconf*, *dotsignif*, and *scatdot* plots. The data for the *boxplot* and time series plots (*TS* and *TSConf*) is more complex and multi-dimensional and is passed to the glyph generation routines via the *panelData* parameter (see below for more details.)
The *row.names* of *statsDFrame* data.frame are used as the link identifier between the data row to the map boundary data. For U.S. state data, the link identifier must be the the state's 2 character abbreviations, full names, or 2-digit US FIPS codes as the *id*. For Seer Areas, the area identifier is the Seer Area abbreviates as defined in the *USSeerBG* document. Refer to the documentation on each border group for the exact names, abbrevviations and ids defined for each sub-area.
The data columns in the statsDFrame are associated with each graphic using the *col1*, *col2*, and *col3* vectors in the panelDesc data.frame by column name or number.

`panelDesc` a data.frame that defines the description of each column: types, associated data columns in the *stateFrame* data.frame, column titles (top and bottom), reference values and text, and names of additional data.frames for complex glyphics (time series and boxplots). See section on panelDesc data.frame. for more details.

`rowNames` defines the type of value used as the row.names in the *stateFrame*. The options are:

- '`auto`' - micromapST will scan the data in the *rowNamesCol* against the border group's name table ("Abbr", "Name", "ID", and "Alt-Abbr" columns) and select the name column containing the most matched. If the '`auto`' does not properly select the expected location id column, specify the desired column in the *rowNames* call parameter.;
- '`ab`' - an abbreviation for the sub-area: 2 character state ID, postal code abbreviation, ISO abbreviation, or a generally accepted abbreviation for the sub-area.;
- '`id`' - sub-area ID, numerical integer identifiers, In the U.S. border groups, the state and county FIPs codes are used. An alias for '`id`' is '`FIPS`'.;
- '`full`' - the full sub-area names.;
- '`alias`' - partial match aliases. Used only with the Seer Registry sub-area registry names as outputted by the SeerStat programs; or
- '`alt_ab`' - an alternate sub-area abbreviation. In some areas, there area two accepted set of abbreviations. If the border group has an alternate abbreviation defined, this option allows the alternative abbreviation to be used as the *rowNames* in the data.

See the documentation on each border group for details on the full names, abbreviatations, optional alternate abbreviations and numeric identifiers defined for each area in the particular border group in the *areaNamesAbbrsIDs* R object.
The default is '`ab`'.

If a border group does not have one of these sets of name data, then the corresponding option will not be available.

The linkage between data and boundaries is accomplished using the strings in the statsDFrame column identified by *rowNamesCol* or if no *rowNamesCol* is specified in the `row.names` information of the *statsDFrame*,

The linkage values are validated to against the *areaNamesAbbrsIDs* data.frame for the specified border group for the *micromapST* call. If a sub-area in the border group is not referenced in the data, it is outlined on the map, but not colored. If the sub-area link cannot be matched, a warning message is generated and the execution of the package is stopped. Unless, the user has set the '`noMatchIgnore`' call argument to TRUE and the data will be ignored.

The '`alias`' option is implemented to support with the *USSeerBG* border group and data generated by the Seer Stat program. The registry column identifying the Seer Area contains the Seer Area name and additional information. This option along with the *alias* field in the border group allows the package to use the registry column as the linkage to the boundary data using a wildcard or partial string allows the registry column generated by Seer Stat to be used as the linkage. This option is controlled the *enableAlias* variable in the border group's *areaParms* data.frame.

`rowNamesCol` allows the user to specify the data.frame column that contains the sub-area string to be used to link the data row to the boundary data for the sub-area. The *rowNames* option above specifies which name information (full name, abbreviation, id, alternate abbreviations) the sub-area string is matched to in the border group name information. (see the border group documentation for more details.) The *rowNamesCol* value must be a column number or column name within the *statsDFrame* data.frame. The default value for *rowNamesCol* is to use row.names of the passed data.frame.

`sortVar` defines the column name(s) or number(s) in the *statsDFrame* data.frame to be used to sort the *statsDFrame* data.frames before creating the state micromap. A vector of column names or numbers can be used sort on multiple columns and to break ties.

For Example: `sortVar=c(4,5)` where columns 4 and 5 in the *statsDFrame* are used in the sort. If the user needs to sort the data based on information in the boxplot or time-series data, the best practice is to copy the data into the *statsDFrame*.

`ascend` a logical value. If *TRUE*, *sortVar* will be sorted in ascending order. If *FALSE*, *sortVar* will be sorted in descending order. The default value is *TRUE*.

`title` A character vector with one or two character strings to be used as the title of the overall micromap plot page.

```
For example:
    \code{title = "micromapST Title"} or
    \code{title = c("title line 1","title line 2")}
```

`plotNames` defines the type of state names to be displayed when an *id* glyph column requested. The options are: '`ab`' or '`full`'. '`ab`' will display the area abbreviations in uppercase as defined in the border groups name information.
'`full`' will display the full area name as properly capitalized strings as defined in the border group name information.
(For the U.S. Stata name information, the full name for District of Columbia is shown as Dist. of Col.. because of space limitations).
The default is '`ab`'.

`axisScale` defines the type of axis labels to be used and if scaling is applied. The acceptable values are '`o`', '`w`', '`s`', and '`sn`'. '`o`' is the original method using the pretty function limiting the values to the range of the data. '`w`' is the default and uses the Wilkinson algorithm to generate the axis labels for a set of data, '`s`' uses the wilkinson algorithm, adjusts the range to cover the labels, and determines a scaling to apply to the labels. If 1000 is the scaling, "in the thousands" is added to the column titles. '`sn`' uses the wilkinson algorithm, adjusts the range to cover the labels, and checks each value to determine if it should be scaled. If it is scales, the scale identifier is added as a suffix. (e.g., 1240334 become 1.24M) "w" is the wilkinson algorithm without any scaling.

`staggerLab` is a logical variable that controls whether the axis labels are staggered and drawn on on two lines or non-staggered in one line. If set to FALSE (the default), the labels are not staggered. If TRUE, two lines are used to draw the axis labels, alternating labels on each line.

`bordDir` specifies the path to a border group dataset (.rda) that is external to the *micromapST* package. The name of the border group is specified in the *bordGrp* parameter.

`bordGrp` specifies which preloaded border group to use for the area names, abbreviations, numeric identifier, and area boundary data. The supported border groups are *USStatesBG* and *USSeerBG*. The default *bordGrp* value is *USStatesBG*. For more information on building your own border group refer to the section in this manual on the *BuildBorderGroup* function.

`dataRegionsOnly`
specifies to only map regions containing data when *aP_Regions* is set to *TRUE*. This indicates the name table contains the information to draw partial area maps of regions. When set to *TRUE*, only regions within the area are drawn that contain at least one sub-area with data provided by the caller. If *FALSE*, the entire area is drawn. The default is *FALSE*. Retional boundaries are not required, but suggested. See border group documentation for more details.

`regionsB` when regional boundaries are provided, controls whether to overlay the boundaries on the map. If dataRegionsOnly is set to *TRUE* and only a subset of regions will be mapped and regional boundaries are provided, regionsB is et to *TRUE*. The default is *FALSE*. Independent of '`dataRegionsOnly`', if set to *TRUE*, if present, regional boundaries are overlayed on the map, If set to *FALSE*, no regional boundaries are drawn. See border group documentation for more details.

`grpPattern` The micromapST package generates the pattern of rows to panel groups automatically. The pattern is based on a maximum of 5 rows per group and number of rows can only decend from the edges toward the median row. The pattern generated by the package can be overriden by using the *grpPattern* to specify the pattern to use as a numeric vector. The provided vector must pass the following checks: a) must be a numeric vector, b) the sum of the values in the vector must equal the number of rows in the user supplied statsDFrame data frame c) the maximum number of rows per panel group is 5. d) the number of rows per panel must be integers and decending from the outside to the middle of the vector.

`maxAreasPerGrp` This parameter defines the maximum number of areas to be represented by a group-row in the resulting linked micromap. The value can be 5 or 6.

`ignoreNoMatches`
The *micromapST* package will automatically handle situations where there is no data for a sub-area in an area. However, it will stop processing if there is data for a non-existing area. To instruct the package to ignore rows of data for sub-areas that do not have boundary information in the border group, set the call argument '`ignoreNoMatches`' to *TRUE* to get the package to detail the data rows from the processing.

`colors` is a vector containing a vector of 12 or 24 color names or values ("#xxxxxx") or the name of a color palette. The vector of 12 or 24 color names or "#xxxxxx" values are used to define the colors used for:

- The 6 colors in each group for the states and symbols in the glyphs. one color per row (state).
- 1 color for the median state and glyphics,

- 1 foreground color for highlighted states in the map. This is used to highlight states already referenced previously or have meaning depend on the type of map requested. The usage is as follows:

```
"map"      - not used for this type of micromap.
"mapcum"   - highlight areas previously referenced
             above.
"maptail"  - highlight areas previously referenced
             above the median row and highlight
             remaining states below the median row.
"mapmedian"- highlight all areas above the median
             in maps above the median row and
             highlight all areas below the median
             in maps below the median row.
```

- 2 accent colors for "mapmedian" sub-area colors for above median and below median.
- When an 12 additional colors are specified, they are used as the translucent colors in the '`tsconf`' confidence intervals bands. If only 12 colors are provided, the additional 12 translucent colors are generated using a 20% transparent version of the original color.

```
e.g., \code{adjustcolors(colors,0.2)}
```

The only color palette support is a gray palette to permit publication of the linked micromaps using a gray scale instead of color. By setting *colors* = "greys", "grays", or "bw", the entire plot will be generated using gray scale that has been balanced to maintain readability and reproduction without the use of color printing. Additional color palettes may be supported in future releases.

If a *colors* vector is not provided, the package default colors will be used:

- 6 state colors: "red", "orange", "green", "greenish blue", "lavender"
- 1 median state color: "black"
- 1 highlighed states: "light yellow" for "map", "mapcum", "maptail"
- 2 highlighed states: "light red" and "light green" for "mapmedian" and
- 12 translucent colors using the above colors at 20%.

It is strongly recommended to use the default.

`details` defines the spacing, line widths and many other details of the plot layout, structure and content; see *micromapGDefaults$details* for more details. Generally *details* does not need to be specified, the default values will always be used and are strongly recommended. However, in a few cases, it may be desireable to turn off or disable a feature. In these cases, the user can specify just the specific variable and value in a list and pass it to *micromapST* via the *details* parameter. For example:

```
details=list(SNBar.Middle.Dot=FALSE,SBar.varht=FALSE)
```

The entire details variable list does not have to be passed. See the section on the *micromapGDefaults$details* for more details.

Details

The *micromapST* function creates a linked micromap plot for data referencing a collection of geographic related areas, like the 50 US States and DC geographical areas or U.S. Seer Areas. The function provides links from a US state map to several forms of graphical charts: dot ('`dot`'), dot with confidence intervals ('`dotconf`'), dot standard error ('`dotse`'), dot with significance mark ('`dotsignif`'), arrow ('`arrow`'), bar chart ('`bar`'), time series ('`ts`'), time series with a confidence band ('`tsconf`'), horizontal stacked (segmented) bar ('`segbar`'), normalized bar ('`normbar`'), centered bar charts ('`ctrbar`'), scattered dot ('`scatdot`'), and box plots ('`boxplot`'). The data values for each column of graphs and each area are provided in the *statsDFrame* data.frame. The panelDesc data.frame specifies the type of chart, the column numbers in the *statsDFrame* with the statistics for the chart, column titles, reference values, etc. Additional data for boxplots and time series plots are provided through the panelData data.frame column.

The following sets of data have been included in the package to support the examples and provide samples of data the micromapST package can utilize.

Dataset contained in the package to support the examples provides are:

statePop2010 U. S. State population data for 2010.

TSdata To be Added

Educ8thData To be Added

stateData To be Added

wflung00and95 White Female Lung (wflung) data from 1995 and 2000

wflung00cnty Wflung US data for 2000 by county.

wflung00and95US Wflung US data for 2000 and 1995.

wflung5070 Wflung data for 50s to 70s

wflung5070US Wflung data for 50s to 70s US rates

KanPopInc Kansas state population by county.

mdPopData Maryland state population by county.

nyPopData New York state population by county.

UtahPopData Utah state population by county.

AfricaPopData Africa population by country.

SeoulPopData Seoul City population by district.

cnPopData China population by province and adminstrative district.

Value

None

Author(s)

Daniel B. Carr, George Mason University, Fairfax VA, with contributions from Jim Pearson and Linda Pickle of StatNet Consulting, LLC, Gaithersburg, MD

References

Daniel B. Carr and Linda Williams Pickle, Visualizing Data Patterns with Micromaps, CRC Press, 2010

Linda Williams Pickle, James B. Pearson Jr., Daniel B. Carr (2015), micromapST: Exploring and Communicating Geospatial Patterns in US State Data., Journal of Statistical Software, 63(3), 1-25., https://www.jstatsoft.org/v63/i03/

See Also

micromapST, micromapSEER

Examples

```
###
#
#   micromapST - Example # 01 - map with no cumulative shading,
#     2 columns of statistics: dot with 95% confidence interval,
#     boxplot sorted in descending order by state rates, using
#     the default border group of "USStatesBG", with default symbols.
###

# load sample data, compute boxplot
TDir<-"c:/projects/statnet/"  # my private test PDF directory exist, don't use temp.
if (!dir.exists(TDir)) {TDir <- paste0(tempdir(),"/") }  # get a temp directory for the output
                                # PDF files for the example.
cat("TempDir:",TDir,"\n")

   # replace this directory name with the location if you want to same
   # the output from the examples.

utils::data(wflung00and95,wflung00and95US,wflung00cnty,envir=environment())

wfboxlist = graphics::boxplot(split(wflung00cnty$rate,wflung00cnty$stabr),
                      plot=FALSE)

# set up 4 column page layout

panelDesc01 <- data.frame(
  type=c("map","id","dotconf","boxplot"),
  lab1=c("","","State Rate","County Rates"),
  lab2=c("","","and 95% CI","(suppressed if 1-9 deaths)"),
  lab3=c("","","Deaths per 100,000","Deaths per 100,000"),
  col1=c(NA,NA,1,NA),col2=c(NA,NA,3,NA),col3=c(NA,NA,4,NA),
  refVals=c(NA,NA,NA,wflung00and95US[1,1]),
  refTexts=c(NA,NA,NA,"US Rate 2000-4"),
  panelData= c("","","","wfboxlist")
  )
panelDesc <- panelDesc01
# set up PDF output file, call package

ExTitle <- c("Ex01-US White Female Lung Cancer Mortality, 2000-2004",
               "State Rates & County Boxplots")
```

```
grDevices::pdf(file=paste0(TDir,"Ex01-US-WFLung-2000-2004-St-DotCf-Co-Box.pdf"),
     width=7.5,height=10)

micromapST(wflung00and95, panelDesc01, sortVar=1, ascend=FALSE,
           title=ExTitle
         )

x <- grDevices::dev.off()
#
### End Example 01

###
#
#   micromapST - Example # 02 - map with cumulative shading
#                 from top down (mapcum), arrow and bar charts,
#                 sorted in descending order by starting
#                 value of arrows (1950-69 rates) using default
#                 border group of "USStatesDF".  This
#                 example also provides custom colors for the
#                 linked micromaps, highlights, etc.
#
###

# Load example data from package.
utils::data(wmlung5070,wmlung5070US,envir=environment())

panelDesc02 <- data.frame(
   type=c("mapcum","id","arrow","bar"),
   lab1=c("","","Rates in","Percent Change"),
   lab2=c("","","1950-69 and 1970-94","1950-69 To 1970-94"),
   lab3=c("MAPCUM","","Deaths per 100,000","Percent"),
   col1=c(NA,NA,"RATEWM_50","PERCENT"),
   col2=c(NA,NA,"RATEWM_70",NA)
 )

colorsRgb = matrix(c(                    # the basic 7 colors.
 213,  62,  79,   #region 1: red     #D53E4F - Rust Red
 252, 141,  89,   #region 2: orange     #FC8D59 - Brn/Org
 253, 225, 139,   #region 3: green     #FEE08B - Pale Brn
 153, 213, 148,   #region 4: greenish blue  #99D594 - med Green
  50, 136, 189,   #region 5: lavendar      #3288BD - Blue
 255,   0, 255,   #region 6                 #FF00FF - Magenta
 .00, .00, .00,   #region 7: black for median #000000 - Black
 230, 245, 152,   #non-highlighted foreground #E6F598 - YellowGreen
 255, 174, 185,   # alternate shape upper   #FFAEB9 - Mauve
 191, 239, 255,   # alternate shape lower   #BFEFFF - Cyan
 242, 242, 242,   # lightest grey for non-referenced sub-areas  #F2F2F2
 234, 234, 234),  # lighter grey for bkg - non-active sub-areas. #EAEAEA

  ncol=3,byrow=TRUE)

xcolors = c( grDevices::rgb(colorsRgb[,1],colorsRgb[,2],colorsRgb[,3],
```

```
                              maxColorValue=255),
                # set solid colors
              grDevices::rgb(colorsRgb[,1],colorsRgb[,2],colorsRgb[,3],64,
                              maxColorValue=255))
                # set translucent colors for time series.

# set up reference names for color set
names(xcolors) =c("rustred","orange","lightbrown","mediumgreen",
                  "blue","magenta", "black","yellowgreen",
                  "mauve","cyan","lightest grey","lighter grey",
                  "l_rustred","l_orange","vlightbrown","lightgreen",
                  "lightblue","l_black","l_yelgreen","l_mauve",
                  "l_cyan","l_lightest grey","l_lighter grey")

ExTitle <- c("Ex02-US Change in White Male Lung Cancer Mortality Rates",
                   "from 1950-69 to 1970-94-Diff colors")

grDevices::pdf(file=paste0(TDir,"Ex02-US WmLung50-70-Arrow-Bar.pdf"),width=7.5,height=10)

micromapST(wmlung5070,panelDesc02,sortVar=1,ascend=FALSE,
            title=ExTitle, colors=xcolors
            )

x <- grDevices::dev.off()
#
### End Example 02

## Not run:
###
#
#   micromapST - Example # 03 - Time Series Line Plots with
#     Confidence Bands maptail option highlights states from extremes
#     to middle state read in time series data set example using the
#     default border group of "USStatesDF".
#
###

# Load example data from package.
utils::data(TSdata,envir=environment())
temprates     <-  data.frame(TSdata[,,2])

# TSdata structure is array of size c(51,15,4),
# dimensions = 51 states, 15 years, (year label, point value, low limit,
# high limit)

panelDesc03   <- data.frame(
    type=c("maptail","id","tsconf","dot"),
    lab1=c("","","Time Series","Female"),
    lab2=c("","","Annual Rate per 100,000","Most Recent Rate (2010)"),
    lab3=c("","","Years","Deaths per 100,000"),
    lab4=c("","","Rate",""),
    col1=c(NA,NA,NA,15),
    panelData =c(NA,NA,"TSdata",NA)
```

```
    )

ExTitle <- c("Ex03-US Time Series with Confidence bands",
               "Annual Female Lung Cancer Mortality Rates, 1996-2010")

grDevices::pdf(file=paste0(TDir,"Ex03-US Time-Series-with-Conf.pdf"),
    width=7.5,height=10)

micromapST(temprates,panelDesc03,sortVar="P15",ascend=FALSE,
           title=ExTitle)

x <- grDevices::dev.off()
#
### End Example 03

## End(Not run)

###
#
#   micromapST - Example # 03a - Time Series Line Plots with
#     Confidence Bands maptail option highlights states from extremes
#     to middle state read in time series data set example using the
#     default border group of "USStatesDF".
#
#   Specify the x-Axis values are dates and to format them as dates.
###

# Load example data from package.
utils::data(TSdata,envir=environment())
temprates      <-  data.frame(TSdata[,,2])   # y rate

# In the original package TS data, the x data was not
# a date value, it was the year number. To be able to demonstrate
# the X-Axis Date format labeling, these were changed to Date values
# by effectively substracting 1970-1-1 from the year value.

####
#
#  Example 3a - Building TS conf array and converting years
#  into date values for the X-Axis and labels.
#
# example of build a TS Conf array.
#
#  Using the old TSdata array as a starting point and source of data,
#  but build an entirely new TS array structure in a similar manner
#  that might be used to build your own time series array.
#

data(TSdata)   # get old array
TSAreas      <- row.names(TSdata)  # one per area (index 1)
NewArray     <- array(dim=c(51,15,4),dimnames=list(TSAreas))
  # this is for 51 states, 15 samples/observations, and 4 values per sample.
```

```
for (inx in seq(1,length(TSAreas)))  { # loop once per area

  Samp     <-  TSdata[inx,,]            # samples for an area
            # each sample has 15 observations of 4 values.
            # value 1 is the X axis data or the DATE of the observation
  Samp[,1]  <- as.Date(paste0(as.character(Samp[,1]),"-01-01"))
    # convert simple year number to date
  NewArray[inx,,] <- Samp

}

# setting the attribute "xIsDate" on array to TRUE, signals micromapST
# the user wants to see the x-axis values as dates.

## Not run:
attr(NewArray,"xIsDate") <- TRUE

# TSdata and NewArray structures are arrays of size c(51,15,4),
# dimensions = 51 states, 15 years, (year label, point value, low limit, high limit)

panelDesc03a   <- data.frame(
    type=c("maptail","id","tsconf","dot"),
    lab1=c("","","Time Series (YYYY-MM)","Female"),
      # recommend adding to the column title a note about the date format used.
    lab2=c("","","Annual Rate per 100,000","Most Recent Rate (2010)"),
    lab3=c("","","Years","Deaths per 100,000"),
    lab4=c("","","Rate",""),
    col1=c(NA,NA,NA,15),
    panelData =c(NA,NA,"NewArray",NA)
    )

ExTitle <- c("Ex03a-US Time Series with Confidence bands with time (yyyy-mm)",
               "Annual Female Lung Cancer Mortality Rates, 1996-2010")

grDevices::pdf(file=paste0(TDir,"Ex03a-US Time-Series-with-Conf wDates.pdf"),
    width=7.5,height=10)

micromapST(temprates,panelDesc03a,sortVar="P15",ascend=FALSE,
           axisScale="s",
           title=ExTitle)

x <- grDevices::dev.off()

## End(Not run)
#
### End Example 03a

###
#
#   micromapST - Example 04 - dot followed by a scatter dot columns
#     use same data as Example 3 to compare 1996 & 2010 rates
#     mapmedian option shades states above or below the median
#     (light yellow) using the default border group of "USStatesBG"
```

```
#
#   USES data loaded for Example 03 (temprates).
#
####
TDir<-"c:/projects/statnet/"  # my private test PDF directory exist, don't use temp.
if (!dir.exists(TDir)) {TDir <- paste0(tempdir(),"/") }  # get a temp directory for the output

# Load example data from package.
utils::data(TSdata,envir=environment())
temprates      <-  data.frame(TSdata[,,2])   # y rate

panelDesc04 <- data.frame(
    type=c("mapmedian","id","dot","scatdot"),
    lab1=c("","","Female Lung Cancer Mortality","Comparison of Rates"),
    lab2=c("","","Rate in 1996 (Sort Variable)",
                      "in 1996 (x axis) and 2010 (y axis)"),
    lab3=c("","","Deaths per 100,000","Deaths per 100,000 in 1996"),
    lab4=c("","","","Rate in 2010"),
    col1=c(NA,NA,1,1),
    col2=c(NA,NA,NA,15)
    )
wparm <- list(NA,NA,NA,list(line="LOWESS",f=.77,line.col="blue",line.lwd=1,line.lty="solid"))
panelDesc04$parm <- wparm

ExTitle <- c("Ex04-US Dot Plot for 1996, Scatter Plot Comparing 1996 to 2010",
               "Female Lung Cancer Mortality Rates, LOWESS Line")

FName <- paste0(TDir,"Ex04-US FLCMR Scatter-Dots-1996-2010.pdf")
grDevices::pdf(file=FName,width=7.5,height=10)

micromapST(temprates,panelDesc04,sortVar=1,ascend=FALSE,title=ExTitle)

x <- grDevices::dev.off()
#
### End Example 04

###
#
#   micromapST - Example 05 - horizontal stacked (segmented) bars
#     segbar plots the input data, normbar plots percent of total
#     package computes the percents from input data
#     input for the categories for each state must be in consecutive
#     columns of the input data.frame using the default border group
#     of "USStatesBG"
####

# Load example data from package.
utils::data(statePop2010,envir=environment())
utils::data(Educ8thData,envir=environment())

JData <- cbind(statePop2010,Educ8thData)

panelDesc05 <- data.frame(
```

```
    type=c("map","id","segbar","normbar","ctrbar"),
    lab1=c("","","Stacked Bar","Normalized Stacked Bar","Centered Bar"),
    lab2=c("","","Counts","Percent","Education"),
    col1=c(NA,NA,"Hisp","Hisp","PctBelowBasic"),
    col2=c(NA,NA,"OtherWBH","OtherWBH","PctAdvanced")
    )
ExTitle <- c("Ex05-Seg Norm Ctr Bars-2010 Census Pop by Race Sorted by Cnt Other Race",
          "Cat-L to R: Hispanic, non-Hisp White, Black, Educ with ctrbar all with varbar")

grDevices::pdf(file=paste0(TDir,"Ex05-US-Seg-Norm-Ctr-Bar-with-varheight.pdf"),
     width=7.5,height=10)

micromapST(JData, panelDesc05, sortVar="OtherWBH", ascend=FALSE,
           title= ExTitle,
           details=list(SNBar.varht=TRUE,CBar.varht=TRUE), axisScale="sn" )

x <- grDevices::dev.off()
#
### End Example 05

## Not run:
###
#
#   micromapST - Example 06 - horizontal stacked (segmented) bars
#     segbar plots the input data, normbar plots percent of total
#     package computes the percents from input data
#     input for the categories for each state must be in consecutive
#     columns of the input data.frame using the default border group
#     of "USStatesBG".
#
#     Turning off the variable bar height and the midpoint dot features
#     in the horizontal stacked bars (segmented)
#
#  USES data loaded for Example 05 above - statePop2010.
#
###

# Reuse data loaded for Example 5 above.

panelDesc06= data.frame(
      type=c("map","id","segbar","normbar"),
      lab1=c("","","Stacked Bar","Normalized Stacked Bar"),
      lab2=c("","","Counts","Percent"),
      col1=c(NA,NA,"Hisp","Hisp"),
      col2=c(NA,NA,"OtherWBH","OtherWBH")
    )

ExTitle <- c("Ex06-Stacked Norm Bars: 2010 Census Pop by Race, Sorted by Other Race",
              "Cat-L to R: Hisp, non-Hisp White, Black, Other,ID-diamond")

grDevices::pdf(file=paste0(TDir,"Ex06-Stkd-Norm-Bar-fixedheight-nodot.pdf"),
    width=7.5,height=10)
```

```
micromapST(statePop2010,panelDesc06,sortVar=4,ascend=FALSE,
            title= ExTitle,
            details=list(SNBar.Middle.Dot=FALSE,SNBar.varht=FALSE,Id.Dot.pch=23)
     )
x <- grDevices::dev.off()
#
### End Example 06

## End(Not run)

###
#
#   micromapST - Example 07 - centered (diverging) stacked bars
#
#     National 8th grade Math Proficiency NAEP Test Scores Data for 2011
#     source: National Center for Education Statistics,
#     http://nces.ed.gov/nationsreportcard/naepdata/
#     bar segment values - % in each of 4 categories:
#           % < Basic, % at Basic, % Proficient, % Advanced
#     using the default border group of "USStatesBG".
####

# Load example data from package.
utils::data(Educ8thData,envir=environment())

# columns = State abbrev, State name, Avg Score, %s \<basic,
#           basic, proficient, advanced

panelDesc07 <- data.frame(
    type=c("map","id","dot","ctrbar"),
    lab1=c("","","Avg. Scores","Math Proficiency"),
    lab2=c("","","","<Basic, Basic, Proficient, Advanced"),
    lab3=c("","","","% to Left of 0           | % to Right"),
    col1=c(NA,NA,"avgscore","PctBelowBasic"),
    col2=c(NA,NA,NA,"PctAdvanced")
  )

ExTitle <- c("Ex07-US Dot Stkd Bars:Educational Progress (NAEP) in Math-2011, 8th Grade",
             "Centered at Not-Prof vs. Prof")

grDevices::pdf(file=paste0(TDir,"Ex07-US Dot-Centered-Bar Educ.pdf"),width=7.5,height=10)

micromapST(Educ8thData,panelDesc07,
             sortVar=3,
             title=ExTitle)

x <- grDevices::dev.off()
#
### End of example 07

## Not run:
###
#
```

```
#  Example 08 - use of state.x77 data table as data source
#     Data does not contain a row for DC, a missing sub-area.
#     Example also uses a smaller then 7.5 x 10 graphic space.
#
###

utils::data(state,envir=environment())

stateData <- as.data.frame(state.x77)

rownames(stateData) <- state.abb

panelDesc08 <- data.frame(type = c("maptail", "id", "dot"),
                  lab1 = c("", "", "Murder"),
                  lab3 = c("", "", "Murders per 100K Population"),
                  col1 = c(NA, NA, 5))

ExTitle <- c("Ex08-US LM Plot of Murders in the United States",
                      "No DC row entry.")

grDevices::pdf(file = paste0(TDir,"Ex08_US state.x77_no_DC.pdf"),
    width = 5, height = 9)

micromapST(stateData, panelDesc08,
           sortVar = 5, ascend = FALSE,
           title = ExTitle)

x <- grDevices::dev.off()

#
### End Example 08

## End(Not run)

###
#
#  Example 09 - US state map based on data from state.x77 table with
#     DC row added to complete data.frame, but with missing values (NAs).
#     The DC row will be sorted to the bottom of the list size
#     it does not contain any data.
#
#   Used data and the panelDesc data.frames (stateData and panelDesc16)
#   used in example 09.
#
###

panelDesc09 <- data.frame(type = c("maptail", "id", "dot"),
                  lab1 = c("", "", "Murder"),
                  lab3 = c("", "", "Murders per 100K Population"),
                  col1 = c(NA, NA, 5))

# add DC as 51st state with missing data "NA" to stateData.
```

```
rm(state)

utils::data(state,envir=environment())

stateData <- as.data.frame(state.x77)
rownames(stateData) <- state.abb

stateData <- rbind(stateData, DC = rep(NA, 8))
                  # missing values for DC row.

ExTitle <- c("Ex09-US-LM Plot of Murders in the United States",
                  "DC row added with NA, decending.")

grDevices::pdf(file =paste0(TDir,"Ex09_US_state.x77_DCasNA_D.pdf"),
     width = 5, height = 9)

micromapST(stateData, panelDesc09,
           sortVar = 5, ascend = FALSE,
           title = ExTitle)

x <- grDevices::dev.off()
#
### End Example 09

###
#
#  Example # 10 - Maps Seer Registries using the micromapST function
#   with the bordGrp = "USSeerBG".
#
###

# Load example data from package.
utils::data(Seer18Area,envir=environment())

# set up 4 column page layout

panelDesc10 = data.frame(
  type=c("mapcum","id","dotsignif","arrow")
  ,lab1=c("","","Rate Trend APC", "Rate Change")
  ,lab2=c("","","Dot-Signif","2002-06 to 2007-11")
  ,lab3=c("","","","")
  ,col1=c(NA,NA,"RateTrendAPC","Rate20022006")
  ,col2=c(NA,NA,"pValue", "Rate20072011")
  )

ExTitle    <- c("Ex10-SeerStat Data-2002-6 and 2007-11",
                     "Dot with Signif., Arrow and Bar")

grDevices::pdf(file=paste0(TDir,"Ex10-SeerStat-DotSignif.pdf"),width=7.5,height=10)

micromapST(Seer18Area,panelDesc10,
        sortVar="Rate20022006",ascend=FALSE,
        title=ExTitle,
```

```
        rowNames="alias",rowNamesCol='Registry',
        bordGrp="USSeerBG",
        plotNames="ab")

x <- grDevices::dev.off()
#
#  Both calls are effectively identical.
#
### End of example 10

###
#
#  Example # 11 - Counties in Kansas on an 11 x 17 page
#
###

# Load example data from package.
utils::data(KansPopInc,envir=environment())

# Four Column Layout:  Map, ID, Dot, and Dot
panelDesc11 = data.frame(
  type=c("map","id","dot",        "dot")
  ,lab1=c("",     "",  "Population", "Average Inc.")
  ,lab2=c("",     "",  "in 2000",    "per year")
  ,lab3=c("",     "",  "People",     "")
  ,col1=c(NA,     NA,  "Pop",        "AvgInc")
  )

ExTitle    <- c("Ex11-Kansas Pop data 11x17",
                  "Current Pop and Average Inc - scaling=e")

grDevices::pdf(file=paste0(TDir,"Ex11-Kansas_Population_and_Income-11x17.pdf"),
     width=10, height=16)
          # tabloid size page (11x17) to handle 105 counties.

#  Use default scaling = "e" and no staggered labels,
#  Use full county names for data to boundary matching,
#  but presented abbreviated county names
#  in "id" glyphic column with large page.

micromapST(KansPopInc, panelDesc11,
        sortVar=c("AvgInc","Pop"), ascend=FALSE,
        title=ExTitle,
        rowNames="full", rowNamesCol='County',
        bordGrp="KansasBG",
        plotNames="ab")

x <- grDevices::dev.off()
#
### End Example 11

###
#
```

```
#   micromapST - Example 12 - A linked micromap of the counties of
#      the state of Maryland using the border group "MarylandBG".
#      The MarylandPopInc data is shown using two dot glypics - current
#       population and average increase per county.
#      A "maptail" state map is used to show the counties in relationship
#      to the median county as sorted by the 1970 population.
###

utils::data(mdPopData,envir=environment())

# set up 5 column page layout

panelDesc12 = data.frame(
  type=c("maptail","id","dot","dot","arrow")
  ,lab1=c("","","Population", "Population","Change")
  ,lab2=c("","","in 1970","in 2000", "from 1970 to 2000")
  ,lab3=c("","","","","")
  ,col1=c(NA,NA,"X1970","X2010","X1970")
  ,col2=c(NA,NA,"","","X2010")
 )

ExTitle       <- c("Ex12-Maryland Population-map",
                      "1970 and 2010 Pop and Change,stag,sn")

grDevices::pdf(file=paste0(TDir,"Ex12-MD Pop 1970 and 2010 plus change-map.pdf"),
     width=7.5, height=10.5)

micromapST(mdPopData, panelDesc12,
        sortVar=2, ascend=FALSE,
        title=ExTitle,
        rowNames="full", rowNamesCol='County',
        bordGrp="MarylandBG",
        axisScale="sn", staggerLab=TRUE,
        plotNames="ab")

x <- grDevices::dev.off()
#
### End Example 12

###
#
#   micromapST - Example 13 - A linked micromap of the counties
#      of the state of New York state using the border group
#      "NewYorkDF".  The pop/inc data is shown using two dot glyphs,
#      an arrow and bar glyph (2010 Population, an arrow showing the
#      change in population from 2000 to 2010, Population in 2000,
#      and a bar showing the amount of the change.)
#
###

# Load example data from package.
utils::data(nyPopData,envir=environment())
```

```
nyPopData$Dif00_10 <- nyPopData$Pop_2010 - nyPopData$Pop_2000

# set up 6 column page layout with colSize

panelDesc13 <- data.frame(
  type=c("map","id","dot",           "arrow",        "dot",     "bar")
  ,lab1=c("",  "",  "Population in", "Increase from","Pop 2005","Incre")
  ,lab2=c("",  "",  "2010",          "2000",         "",        "2000to2010")
  ,lab3=c("",  "",  "",              "",             "",        "")
  ,col1=c(NA,  NA,  "Pop_2010",      "Pop_2000",     "Pop_2000","Dif00_10")
  ,col2=c(NA,  NA,  "",              "Pop_2010",     "",        "")
  ,colSize=c(NA,NA, 15,              20,             5,        20)
  )

ExTitle   <- c("Ex13-New York Population data",
                "2010 Pop and since 2000-colSize,sn,stag")

grDevices::pdf(file=paste0(TDir,"Ex13-New York Pop 2010 and Change-sn colSize.pdf"),
     width=7.5, height=10.5)

micromapST(nyPopData, panelDesc13,
        sortVar="Pop_2000", ascend=FALSE,
        title=ExTitle,
        rowNames="full",rowNamesCol="Area",
        axisScale="sn", staggerLab=TRUE,
        bordGrp="NewYorkBG"
      )

x <- grDevices::dev.off()
#
#### End Example 13

###
#
#   micromapST - Example 14 - A linked micromap of the counties in the
#      state of Utah.  The UtahPopData data is shown using two dot glypics
#      - current population and average increase per area.
#
###

# Load example data from package.
utils::data(UtahPopData,envir=environment())

#
#  Get population differences from 2011 to 2001 and 1991.
#   Data contains ",".  The comma's must be removed and values are
#     converted to numbers.
#   If data is factors, need to add "as.character()" function
#     to the formula below.

UtahPopData2 <- as.data.frame(sapply(UtahPopData,
                        function(x) gsub(",","",x)),stringsAsFactors=FALSE)
```

```
# Calculate the differenct between 2001 and 2011 population.
UtahPopData2$Del1101 <- as.numeric(UtahPopData2$X2011)
                        - as.numeric(UtahPopData2$X2001)

# Calculate the difference between 1991 and 2001 population.
UtahPopData2$Del0191 <- as.numeric(UtahPopData2$X2001)
                        - as.numeric(UtahPopData2$X1991)

# set up 5 column page layout

panelDesc14 = data.frame(
  type=c("map","id","dot","arrow","arrow")
  ,lab1=c("","","Population", "2001-2011","Chg 1991-2001")
  ,lab2=c("","","in 2011","pop change","pop change")
  ,lab3=c("","","","","")
  ,col1=c(NA,NA,"X2011","X2011","X2001")
  ,col2=c(NA,NA,NA,"X2001","X1991")
  )

ExTitle    <- c("Ex14 - Utah county population 2011",
                 " and changes last two decades,sn")

grDevices::pdf(file=paste0(TDir,"Ex14-Utah Population.pdf"),
     width=7.5,height=10.5)

micromapST(UtahPopData, panelDesc14,
        sortVar="X2011", ascend=FALSE,
        title=ExTitle,
        rowNames="full",rowNamesCol='County',
        axisScale="sn",
        bordGrp="UtahBG",
        plotNames="ab"
      )

x <- grDevices::dev.off()
#
### End Example 14

###
#
#   micromapST - Example 15 - A linked micromap of the provinces,
#      municipalities, autonomous regions and special administrative
#      regions of China using the border group of "ChinaDF".
#      The ChinaPopInc data is shown using two dot glypics - current
#      population and average increase per area.
#
###

utils::data(cnPopData,envir=environment())

# set up 4 column page layout

panelDesc15 = data.frame(
```

```
  type=c("map","id","dot","bar")
  ,lab1=c("","","Population", "Population")
  ,lab2=c("","","in 2013","in 2013")
  ,lab3=c("","","","")
  ,col1=c(NA,NA,"pop2013","pop2013")
  )

ExTitle       <- c("Ex14-China Population",
                     "in 2013 by area")

grDevices::pdf(file=paste0(TDir,"Ex15-China 2013 Population.pdf"),
    width=7.5, height=10.5)

micromapST(cnPopData, panelDesc15,
        sortVar="pop2013", ascend=FALSE,
        title=ExTitle,
        rowNames="full", rowNamesCol='area',
        bordGrp="ChinaBG",
        plotNames="full")

x <- grDevices::dev.off()
#
### End Example 15

###
#
#   micromapST - Example 16 - A linked micromap of the districts
#      of the city Seoul South Korea, using the border group of
#      "SeoulSKoreaBG". The included SeoulPopData dataset provides
#      population and district area statistics for 2012.
#      The micromapST generates two glyphics, a sorted dot
#      glyphic based on the population and a bar graph based on
#      the area.
#
###

# Load example data from package.
utils::data(SeoulPopData,envir=environment())

# set up 4 column page layout

panelDesc16 = data.frame(
  type=c("map","id","dot","bar")
  ,lab1=c("","","Population", "Area")
  ,lab2=c("","","in 2012","in 2012")
  ,lab3=c("","","","sqkm")
  ,col1=c(NA,NA,"Pop.2012","Area")
  )

ExTitle       <- c("Ex16-Seoul Population",
                     "in 2012 by district")

grDevices::pdf(file=paste0(TDir,"Ex16-Seoul 2012 Population.pdf"),
```

```
     width=7.5, height=10.5)

micromapST(SeoulPopData,panelDesc16,
        sortVar=3, ascend=FALSE,    # sort based on the population
        title=ExTitle,
        rowNames="full", rowNamesCol='District',
        bordGrp="SeoulSKoreaBG",
        plotNames="full"
      )

x <- grDevices::dev.off()
#
### End Example 16

###
#
#  Example 17 - use of Africa population data as data source
#      Demonstrates support for vertical oriented geographical
#       areas.
#
###

# Load example data from package.
utils::data(AfricaPopData,envir=environment())

panelDesc17 <- data.frame(
              type = c("map", "id", "dot",      "dot",          "dot"),
              lab1 = c("",      "","Population","Percentage Of","Est x2 Time"),
              lab3 = c("",      "","People",    "Total",        "Years"),
              col1 = c(NA,      NA,"Projection","PercOf",       "Est2Time")
            )

ExTitle <- c("Ex17-Africa Population Data",
               "Sorted by Population on 11x17")

grDevices::pdf(file = paste0(TDir,"Ex17-Africa Micromap-11x17.pdf"),
     width = 11, height = 17)

micromapST(AfricaPopData, panelDesc17,
           sortVar = "Projection", ascend = TRUE,
           title = ExTitle,
           rowNames = "ab", rowNamesCol = "Abbr",
           bordGrp = "AfricaBG" )

x <- grDevices::dev.off()
#
### End  Example 17



###

unlink(TDir)
```

NewYorkBG *NewYorkBG border group datasets to supports building micromaps for the 62 counties in the state of New York.*

Description

The *micromapST* function has the ability to generate linked micromaps for any geographical area. To specify the geographical area, the 'bordGrp' call argument is used to specify the border group dataset for the geographical area. The *NewYorkBG* border group dataset supports creating linked micromaps for the 62 counties in the state of New York. When the 'bordGrp' call argument is set to *NewYorkBG*, the appropriate name table (county names and abbreviations) and the 62 sub-areas (countries) boundary data is loaded in *micromapST*. The user's data is then linked to the boundary data via the county's name, abbreviated or ID based on the table below.

Usage

```
data(NewYorkBG)
```

Details

The *NewYorkBG* border group dataset contains the following data.frames:

areaParms - contains specific parameters for the border group

areaNamesAbbrsIDs - containing the names, abbreviations, numerical identifier and alias matching string for each of the 62 counties in New York

areaVisBorders - the boundary point lists for each area.

L2VisBorders - the boundaries for an intermediate level. For this border group, this boundary data.frame is not used and set to L3VisBorders as a place holder.

RegVisBorders - the boundaries for an intermediate level. For this border group, this boundary data.frame is not used and set to L3VisBorders as a place holder.

L3VisBorders - the boundary of the state of New York.

The New York county border group contains 62 county sub-areas. Each county has a row in the areaNamesAbbrsIDs data.frame and a set of polygons in the areaVisBorders data.frame datasets. No regions are defined in the New York county border group, so the L2VisBorders dataset is not used and the regions option is disabled. The L3VisBorders dataset contains the outline of the state of New York.

The details on each of these data.frame structures can be found in the "bordGrp" section of this document. The *areaNamesAbbrsIDs* data.frame provides the linkages to the boundary data for each sub-area (county) using the fullname, abbreviation, and numerical identifier for each country to the *<statsDFrame>* data based on the setting of the 'rowNames' call argument.

A column or the data.frame row.names must match one of the types of names in the *areaNamesAbbrsIDs* data.frame name table. If the data row does not match a value in the name table, an warning

is issued and the data is ignored. If no data is present for a sub-area (county) in the name table, the sub-area (county) is mapped but not colored.

The following are a list of the names, abbreviations, and IDs for each country in the *NewYorkBG* border group.

name	ab	id
Albany	ALBA	36001
Allegany	ALLE	36003
Bronx	BRON	36005
Broome	BROO	36007
Cattaraugus	CATT	36009
Cayuga	CAYU	36011
Chautauqua	CHAU	36013
Chemung	CHEM	36015
Chenango	CHEN	36017
Clinton	CLIN	36019
Columbia	COLU	36021
Cortland	CORT	36023
Delaware	DELA	36025
Dutchess	DUTC	36027
Erie	ERIE	36029
Essex	ESSE	36031
Franklin	FRAN	36033
Fulton	FULT	36035
Genesee	GENE	36037
Greene	GREE	36039
Hamilton	HAMI	36041
Herkimer	HERK	36043
Jefferson	JEFF	36045
Kings	KING	36047
Lewis	LEWI	36049
Livingston	LIVI	36051
Madison	MADI	36053
Monroe	MONR	36055
Montgomery	MONT	36057
Nassau	NASS	36059
New York	NEWY	36061
Niagara	NIAG	36063
Oneida	ONEI	36065
Onondaga	ONON	36067
Ontario	ONTA	36069
Orange	ORAN	36071
Orleans	ORLE	36073
Oswego	OSWE	36075
Otsego	OTSE	36077
Putnam	PUTN	36079
Queens	QUEE	36081
Rensselaer	RENS	36083

Richmond	RICH	36085
Rockland	ROCK	36087
St. Lawrence	STLA	36089
Saratoga	SARA	36091
Schenectady	SCHE	36093
Schoharie	SCHO	36095
Schuyler	SCHU	36097
Seneca	SENE	36099
Steuben	STEU	36101
Suffolk	SUFF	36103
Sullivan	SULL	36105
Tioga	TIOG	36107
Tompkins	TOMP	36109
Ulster	ULST	36111
Warren	WARR	36113
Washington	WASH	36115
Wayne	WAYN	36117
Westchester	WEST	36119
Wyoming	WYOM	36121
Yates	YATE	36123

There are no alternate abbreviations or regions assocated with counties in New York border group.

The *id* field value is the U. S. state and county FIPS code.

The '`rowNames`' = "alias" or "alt_ab", the '`regionB`' and '`dataRegionsOnly`' features are not supported in the *NewYorkBG* border group.

`nyPopData` *Test data for the New York border Group*

Description

This dataset contains the 2014 population and average income per person in each of the New York counties.

Usage

```
data(nyPopData)
```

Format

A data frame with 63 rows, 1 for each county and 14 variables per county:

Area a character vector containing the New York county and buorgh name.

Est_2000 a numeric vector of the county's estimated population for 2000.

Est_2001 a numeric vector of the county's estimated population for 2001.

Est_2002 a numeric vector of the county's estimated population for 2002.

Est_2003 a numeric vector of the county's estimated population for 2003.

Est_2004 a numeric vector of the county's estimated population for 2004.

Est_2005 a numeric vector of the county's estimated population for 2005.

Est_2006 a numeric vector of the county's estimated population for 2006.

Est_2007 a numeric vector of the county's estimated population for 2007.

Est_2008 a numeric vector of the county's estimated population for 2008.

Est_2009 a numeric vector of the county's estimated population for 2009.

Est_2010 a numeric vector of the county's estimated population for 2010.

Pop_2000 a numeric vector of the county's actual population for 2000.

Pop_2010 a numeric vector of the county's actual population for 2010.

Details

This dataset was pulled from the New York government website in January, 2015. It contains the actual and estimate populations of each county from 2000 to 2010.

---

panelDesc *micromapST panel description data.frame structure*

Description

The panelDesc data.frame provides the *micromapST* function with the information required to process the *statsDFrame* data and panelData data.frames and to generate the required linked micromap plot.
It specifies which columns in the *statsDFrame* data.frame contain the data for each glyph column, the column types, labels, reference values and text, and when more complex data is needed by a glyph (boxplot and time series) what the name of the data structure..

```
Example
  panelDesc = data.frame(
      type=c("mapcum","id","dotconf","dotconf"),
      lab1=c("","","White Males","White Females"),
      lab2=c("","","Rate and 95% CI","Rate and 95% CI"),
      lab3=c("","","Deaths per 100,000","Deaths per 100,000"),
      col1=c(NA,NA,"Rate",9),
      col2=c(NA,NA,4,11),
      col3=c(NA,NA,5,12),
      colSize=c(NA,NA,5,5),
      refVals=c(NA,NA,NA,wflungUS[,1]),
      refTexts=c(NA,NA,NA,"US Rate"),
      panelData=c("","","",""),
      parm=list(NA,NA,NA,list=(a=9))
```

The panelDesc data.frame (which does not have to be named "panelDesc", any name will do) provides the means of defining how many columns to create, the type of glyph per column, where the data required by the glyph is located in the *statsDFrame* (column number or name) or the name of a supplemental data structure when the glyph is boxplots or time series (via the *panelData* list entry), the column titles, and the column's reference value and label for the link micromap generation.

In the following description the term "AREA" represents the geographic unit being mapped and associated with data in the *statsDFrame*. The naming used must match the border group specified. If the border group of "USStatesDF" is used, the areas are U.S. States and DC and 51 data rows must be present. If the border group of "USSeerDF" is used, the areas are U.S. Seer areas as defined by NCI and the number of data rows can be 9, 11, 13, 17 or 18. In all cases, the abbreviations and names defined in the border group dataset must be used in preparing the statsDFrame and panelData structures.

Glyph Types

The *type* vector defines the type of glyph to be used for each column. The available glyphs are:

Map types: "map", "mapcum","maptail","mapmedian"

State or Area ID and/or Name: "id"

Ranking: "rank"

Graphical Type: "dot", "dotse","dotconf", "dotsignf", "bar", "arrow", "ts", "tsconf","scatdot", "segbar", "normbar", "ctrbar", "boxplot"

The following provides a description of each panel type:

map - US map with active areas colored

mapcum - US map with active areas colored and previously active area highlighted generating an accumulation from top to bottom

maptail - US map with active areas colored and previously active area highlighted until the median area, then the reverse to the end (areas that have not been active are highlighted.)

mapmedian - US map with active areas colored. Maps above the median area have areas with values above the median highlighted. Maps below the median area have areas with values below the median highlighted. This helps define the above and below median area groups.

id - generates a column with a colored identifier (a square) and the area or area name or abbreviation.

rank - number the area in rank order, sequentially.

arrow - an arrow between two values with a head.

bar - a single bar chart.

boxplot - a boxplot per area with box, upper and lower whiskers and outliers.

dot - a dot for a single value.

dotse - a dot for a single value and its standard error.

dotconf - a dot for a single value and its confidence interval.

dotsignf - a dot for a single value with an indicator of its significants.

ts - a time series line for up to 30 sets "x" and "y" values for each area. The TS glyph can have X-Axis labels formated as numbers or dates.

tsconf - a time series line for a up to 30 sets of "x", "y" and upper "y" and lower "y" values as a confidence interval band for each area. The TSConf glyph can have X-Axis labels formated as numbers or dates.

segbar - a horizontal stacked (segmented) bar plot starting at 0 for 2 to 9 bars.

normbar - a stacked bar plot where the data is normalized for each area by dividing the bar segment values by the sum of the values for all of the bars. Up to 9 bars are supported.

ctrbar - a stacked bar plot where the bar segments are centered around the 0.Up to 9 bars are supported.

scatdot - a set of points for each area with an "x" and "y" value.

Labels (Column Headers and Footers)

micromapST supports up to 3 column labels or titles: *lab1*, *lab2* and *lab3*, where *lab1* and *lab2* are header titles for the column. *lab3* is the footer title for the column. All titles are optional. *lab3* is used to indicate the unit of measure at the bottom of the columns, but is not limited to this use. For example:

```
 lab1=c("Col1-Title", "Col2=Title", "Col3-Title" ) # 1st title for columns
 lab2=c("Col1-Sub",   "Col2-Sub",   "Col3-Sub"   ) # 2nd title for columns
lab3=c("Col1-Footer","Col2-Footer","Col3-Footer") # Footer title for columns
```

lab4 is used only when time series or scatter dot glyphs are used to provide a Y axis title for the column. All label-title vectors are optional and only required when an title or label is needed.

Data References

Depending on the type of glyph selected for the column, 1 to 3 data values for each area may be required: The *col1*, *col2* and *col3* vectors serve as indexes to columns in the *statsDFrame* data.frame passed in the arguments of the *micromapST* function call. The values can be either the numeric number of the row in *statsDFrame* data.frame or the column name. If no index is required, the entry should be set to *NA*.

If the glyph requires one value, then only the *col1* index is used and the *col2* and *col3* indexes are set to *NA* if present . If 2 values are required, then *col1* and *col2* indexes are used and the *col3* index is set to *NA*, if present. If 3 values are required, then *col1*, *col2*, and *col3* indexes are used.

The *statsDFrame* column indexes can be provided as an integer or the column name. If the integer value is less than 1 or greater than the number of columns in *statsDFrame* or a column name is used that does not exist in statsDFrame, the *micromapST* function will stop and generate an error message.

Glyph Name	Meaning	col1	col2	col3	panelData
arrow	Arrow	Beginning Values	Ending Values (arrow head)	NA	NA

bar	Horizontal bar	Bar end values (length)	NA	NA	NA
segbar	Horizontal stacked bar	Values for first (left -most) segment (length)	Values for the last (right-most) bar segment (length)	NA	NA
normbar	Horizontal stacked bar, normalized to total 100%	Values for first (left-most) bar segment (length)	Values for last (right-most,bar segment (length)	NA	NA
ctrbar	Horizontal stacked bar, centered on the middle bar	Values for first (left-most) bar segment (length)	Values for last (right-most,bar segment (length)	NA	NA
boxplot	Horizontal box plot	NA	NA	NA	Name of output list from call to boxplot(...plot=F)
dot	Dot	Values for dots	NA	NA	NA
dotconf	Dot with confidence interval line	Values for dots	Values of lower limits	Values for upper limits tab	NA
dotse	Dot with line length +/- standard error	Values for dots	Standard errors	NA	NA

dotsignf	Dot overprinted if not significant	Values for dots	P value associated with dot	NA	NA
scatdot	Scater plot of dots	Values on horizontal (x) axis	Values on vertical (y) axis	NA	NA
ts	Time Series (line) plot	NA	NA	NA	Name of array with dimensions of c(51,t,2), where t = # of time points (max 15), x values in [,,1], y values in [,,2]
tsconf	Time Series (line) plot with confidence limits	NA	NA	NA	Name of array with dimensions of c(51,t,4), as ts lower limit is [,,3] amd the upper limit is [,,4]

The panelData data.frame is only used when a glyph requires more data per area than can be provided by the *statsDFrame* columns. Only glyphs using this vector are boxplots and time series.

In the case of the '`boxplot`' glyph, the boxplot function with plot=F is used to generate the boxplot statistical details for each area. The name of the resulting list of 51 sets of boxplot statistics (one for each area) is placed in the panelData data.frame element for the boxplot column.

For the time series and time series with confidence interval, the glyphs require a 3 dimensional array of data. The first dimension ([*area*,,]) represents the areas. The second dimension ([,*t*,]) ranges from 2 to *n*. There is no upper limit, but 200-250 samples is a practical limit. One for each data point. The third dimension ([,,*v*]) provides the values at data point *t* for area *st*. [,,*1*] is the *x* axis value. For time series, is usually just the value 1 to *n* to order the *y* values. [,,*2*] is the median *y* value. For time series with confidence intervals: [,,*3*] is the lower value *y* and [,,*4*] is the upper value *y*.

Reference Lines

Reference lines can be created in arror, bar, dot, dotconf, dotse, and segbar glyphs by specifying the reference values in the *RefVal=* vector. A label appearing at the bottom of the column can be specified using the *RefTxt=* vector in the *panelDesc* data.frame.

Format

The parameters in the panelDesc data.frame structure are:

type= The types of graphics for each column of panels can be specified by the following keywords in the "type variable":

The following are the type of glyphs that can be specified in the type vector:

Map types: "map", "mapcum","maptail","mapmedian"

State ID and/or Name: "id"

Glyph Type: "dot", "dotse","dotconf", "dotsignf", "bar", "arrow", "ts", "tsconf","scatdot", "segbar", "normbar", "ctrbar", "boxplot"

The following provides a description of each panel type:

map - a non-highlighted map

mapcum - maps show the accumulated areas top to bottom

maptail - maps show the accumulated areas from the top and bottom toward median area.

mapmedian - the maps above the median highlight the areas above the median area and maps below the median highlight areas below the median area based on the sorting variable.

id - generates a column with a color identifier (a filled in square) and the area abbreviation or name. The plotNames parameter in the *micromapSEER* call controls whether the area's full name or 2 character abbreviation is displayed.

rank - sequentially number areas from 1 (highest rank) to "n" (lowest rank)

arrow - an arrow from value 1 to value 2 with value 2 the head of the arrow.

bar - a bar for a single set of values, The values can be positive or negative.

boxplot - a boxplot for each area using a data.frame generated by the boxplot function with plot=F. The name of the boxplot data.frame is passed to *micromapSEER* using the *panelData* vector.

dot - a dot for a single value using one set of values.

dotse - a dot for a single value and its standard error using two values.

dotconf - a dot for a single value and its confidence interval using three values.

dotsignf - a dot for a single value overlaid if value is not significant using two values: value for dot and P value.

ts - a time series line plot for each area. The glyph use the *panelData* vector to get the name of a three (3) dimensional array the data for the plot. The array contains one entry per area, 1 to 30+ data points and the *x* and *y* values. See section on *panelData* below for more details. A reasonable upper limit to the number of points is between 200-300. Only a few will be selected to be used as X-Axis labels. The format of the X-Axis label is controled by the "xIsDate" attribute on the array being set to TRUE. If the "xIsDate" attribute is not set to TRUE, the X-Axis will be formated as numeric and axisScaling can be preformed. If the "xIsDate" attribute is TRUE, the default date format of " or less than 90, a short date format will be used of " The x-axis date feature will override the specification of the axisScale call parameters on time series glyph columns.

tsconf - a time series line and confidence interval band for each area. The glyph use the *panelData* vector to get the name of a three (3) dimensional array the data for the plot. The array contains one entry per area, 1 to *n* data points and the *x*, *y*, *lower y* and *upper y* values. See section on *panelData* below for more details. A reasonable upper limit to the number of points is between 200-300. The format of the X-Axis label is controled by

the "xIsDate" attribute on the array. If the "xIsDate" is set to TRUE, the X-Axis values will be format using the default date format of " date format of " TRUE, the X-Axis will be formated as numeric and axisScaling can be preformed. The x-axis date feature will override the specification of the axisScale call parameters on time series glyph columns.

segbar - a horizontal stacked (segmented) bar plot starting at 0 using data in the statsDFrame data.frame. The *col1* and *col2* columns are used to indicate the first and last columns in the statsDFrame data.frame that contain the contiguous bar segment values (lengths). For example: The data for a 5 segment bar glyph is in columns 4 through 8 in the *statsDFrame* (5 columns). *col1* is set to 4 to identify the first column and *col2* is set to 8 to identify the last column in the sequence. Column names may be used, but the column identified in *col1* must preceed the column identified in *col2*.

normbar - a stacked bar plot where the data is normalized for each area by dividing the bar segment values by the sum of the values for all of the bars. The stacked bar plot for each area then ranges from 0 to 100% (edge to edge). The *col1* and *col2* columns are used to identify the first and last columns for bar data in the *statsDFrame* in the same way as for the "segbar" glyph (see above.)

ctrbar - a stacked bar plot where the bar segments are centered around the middle of the data. If there is an even number of segments, the 0 point is between the lower half and the upper half of the segments. If there is an odd number of segments, the center is the midpoint of the middle segment. The other segments are plotted to the left and right of the center point. The *col1* and *col2* columns are used to indicate the first and last columns in the *statsDFrame* data.frame that contain the contiguous bar segment values. (See "segbar" type above for more information.)

scatdot - a set of 51 points with an *x* and *y* value per area. All points are plotted in each panel with the key areas in the panel highlighted. *col1* indicates *statsDFrame* column containing the *x* values and *col2* indicates the column containing the *y* values.

Example: `type=c("id","map","rank", "boxplot")` To specify a *micromapSEER* with three columns, left to right, containing the area label, a map and a boxplot.

col1=, *col2=*, *col3=* Vectors of index numbers or names of columns in *statsDFrame* data.frame to be used as data for graphics. The uses of these three vectors are defined below:

any "map" type, id, boxplots, ts, and tsconf glyphs do not use the *col1*, *col2*, or *col3* vectors to locate data in the *statsDFrame* data.frame. If these vectors are present, the corresponding entires should be `NA` for the respective columns.

dot uses *col1* to specify a single data column in *statsDFrame* data.frame to be ploted.

bar uses *col1* to specify the data column in *statsDFrame* data.frame for the length of the bar. The data value can be positive or negative.

dotse uses *col1* and *col2* to specify the data columns in *statsDFrame* data.frame to be used as the estimate and standard error values, respectively.

dotsignf uses *col1* and *col2* to specify the data columns in *statsDFrame* data.frame to be used as the value for the dot and its associated P value.

arrow uses *col1* and *col2* to specify the data columns in *statsDFrame* data.frame for the beginning and end values of the arrow.

segbar, normbar, ctrbar uses *col1* and *col2* to specify the first and last columns in the *statsDFrame* data.frame. The *statsDFrame* data.frame columns from *col1* to *col2* are used for the length values of each bar in the glyph. *col1* must preceed *col2* in the *statsDFrame* data.frame. The minimum number of data columns is 2 columns with a maximum of 9 columns.

scatdot uses *col1* and *col2* to specify the *x* and *y* values respectivefully for a dot for each of the 51 areas and DC in a scatter dot plot.

dotconf uses *col1*, *col2*, and *col3* to specify the data columns in *statsDFrame* data.frame for the estimate value, lower confidence interval, and upper confidence interval values.

See the table above.

colSize= A numeric vector used to specify the proportional width size of a glyph column in relation to all other glyph columns. If used, values must be included for all glyph columns except for the map and id glyphs, which are fixed width columns. The width of a glyph column is determined by summing all of the colSize values and dividing the sum into the value for each glyph column to yield a percentage of the available width to be allocated to each column. For example: colSize=c(NA,NA,10,10,5,15), does not affect columns 1 and 2. The percentages for columns 3 through 6 are 25%, 25%, 12.5% and 37.5%. If 4 inches of space is available, the columns will be allocated: 1, 1, 0.5, and 1.5 inches. The column widths are still regulated by the minimum and maximum column widths set in the package. If a value is missing for non-map or id glyph, the package will a value equal to the average of the provided values.

lab1=, *lab2=* Character vectors provide the two column labels (titles) lines at the top of each column. If no label is required, use `""` for a blank line.

lab3= Character vector used as a label at the bottom of each column. This is typically used to show units of measure. If no label is required, use `""` for a blank line.

lab4= Character vector used as the vertical (y) axis label for *ts*, *tsconf*, and *scatdot* glyphs. If no label is required, use `""` for a blank line.

refVals= Is a list of object names providing the reference values for each graphic column. The reference value is displayed as a dashed vertical line for each panel in the specified column.

refTexts= Is a list of 1 or 2 labels to be displayed at the bottom of each column to identify the reference value.

parm= list of lists containing named items to be passed to a glyph. It was originally added to pass control parameters to the *scatdot* glyph to request a "LOWESS" line and to provide the line graphic parameters (type, width, and color). It use will be expanded as needed to pass other parameters to other glyphs. Since it is a list and not a vector, it must be added to the panelDesc data.frame in a different manner:

```
panelDesc <- data.frame(type=c("map","id","dot","scatdot"),  ... )
wparm     <- list(NA,NA,NA,list(line="LOWESS",line.col="blue"))
panelDesc$parm <- wparm
```

panelData= List of object names containing the boxplot data list and/or an array of time series data for each area. If boxplot and time series data are not used in a column, then associated object names should be `NA`.

For '`boxplot`' data, each row name in the boxplot list must be the area abbreviation (2 character) for the area associated with the data. There must be the same number of rows as in the name table and statsDFrame table. Each row must be data produced by the '`boxplot`' function. The area location identifier used in the statsDFrame data and must be placed in the boxplot$names (*names*) attribute for that set of boxplot data to be able to associate the individual boxplots to each area.

For the time series glyph (ts), the data must be a three (3) dimensional array. The first dimension `[`*st*`,,]` represent one entry for each area (1 to 51). The second dimension `[,`*t*`,]` indexes

up to 30+ data points for the area. The third dimension `[,,v]` are the data point values at each data point. `[,,var{1}]` is the *x* value and `[,,2]` is the median *y* value for the data point. The rownames associated with the first dimension must be the area location ids used in the statsDFrame table to link the elements of this structure the presentation order of the areas.

For the time series with confidence intervals glyph (tsconf), the array is extended to include: `[,,3]` and `[,,4]` for the *lower y* and *upper y* values.

For time series data, the order of the first dimension of the array must match the area order in the *statsDFrame*. For example, the data in `dataArray[1,,]` is the the area identified in `statsDFrame[1,]`

The Date feature allows the caller to request the TS X-Axis labels be formated at dates. This requires the data in the TS array has valid date data as the X data. These are numbers based on 1970-1-1 being day zero in the computer calendar. There are many functions in R to convert to and from characters and date variable. In the past, before this feature, users had to do work-a-rounds by using year numbers or year and faction numbers. Once you have inserted the date X values into the array [,,1], modify the class of the array to add the "Date" class. micrpmapST will inspect the array and find the "Date" class, flag it for internal operations and remove it. The date format of " If the time series contains <= 90 days, the date format will be changed to " to provide a more useful label. The date feature is only available on the Time Series Glyphs.

If axisScale is set to "s" or "sn", they will be ignored for any TS glyph using the date feature.

Details

The panelDesc data.frame is used to describe the content of the micromapST plot to the function. It contains the index of the data in the `statsDFrame` data.frame, the types of graphics to be used in each column, titles, column headers, reference values and labels, etc.

Note

A descriptor may be omitted if none of the panel plots need it.

Author(s)

Daniel B. Carr, George Mason University, Fairfax VA, with contributions from Jim Pearson and Linda Pickle of StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`Plot_Vis` *A function to plot the micromapST boundary format called VisBorders*

Description

The *Plot_Vis* function will plot the boundary data contained in a border group. The border group boundary information is stored as a data.frame with an "x", "y", "hole", and "key" columns. The "key" column identifies the name table entry the boundary point and polygon belongs to. The "x" and "y" are the coordinates for the boundary vertex in the polygon, and the "hole" column identifies if the polygon is a hole in the area identified by the "key". At the end of each sequence of vectex for a polygon, the first vertex and the last vertex must be the same point. This is followed by a "x" and "y" coordinates of c(NA,NA) to tell the turn off drawing and wait for the next point. It's primary purpose is to visualize boundary data that has been prepared for micromapST and assist in developing new border groups for micromapST.

Usage

```
Plot_Vis( VisBrd, VisCol, xTitle=NULL, xAxes=FALSE, xLwd=0.05)
```

Arguments

`VisBrd` - is a data frame containing a VisBorders formated collection of boundaries to be plotted. Refer to the section under the discussion on the *areaVisBorder* data.frame and it's format. In a border group, this would be the areaVisBorder, L2VisBorder, RegVisBorder, or L3VisBorders data.frame.

`VisCol` - is a vector of the colors to fill each polygon in the VisBorder data frame. One color value is required for each "NA" (end of polygon) in the VisBorders data.frame. This vector must have the same number of elements at the VisBorder has polygons (point sets ending in NA.)

If multiple polygons represent a single area, the function caller must compensate for this in the VisCol vector. The usual strategy is to assign a color to each row in the Name Table, then compare the keys in the Name Table and the keys in the VisBorder and assign the color for that Key/Polygon.

Warning: R's polygon function does not advance to the next color in this list, if the vectors between the "NA" rows represent a point or a line, R's polygon function will not advance to the next color, nothing to fill. This will affect which colors shade each polygon.

`xTitle` - is character string value to be used for the title of the VisBorders plot. The default for this call parameter is NULL. Only a single string can be used.

`xAxes` - is a logical (FALSE or TRUE) indicator as to whether or not the X and Y Axis should be drawn. The default for this call parameter is FALSE.

`xLwd` - is a numerical value of the boundary line width to use when drawing the areas in the map. The value may range from .1 up to 3 times the standard width. The default value is 0.5.

Details

More details to follow.

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

`Seer18Area` *Test data for 18 U.S. Seer Registries (of 20) for general dot, arrow, and bar glyphics.*

Description

Randomly generated statistics to create rate, count and population data for the 18 U.S. Seer registries for testing of micromapSEER or micromapST using the "USSeerBG" border group. The data was generated by the SeerStat program and exported as a CSV file. The non-data lines at the bottom of the file must be removed.

Usage

```
data(Seer18Area)
```

Format

A data frame with 18 observations, 1 for each Seer registries (sub-areas), on the following 11 variables.

Registry a character vector of the name of the Seer Registry

Rate20022006 a numeric vector of the incident rates from 2002 to 2006

Count20022006 a numeric vector of the incident counts

Pop20022006 a numeric vector of the population

Rate20072011 a numeric vector of the incident rates from 2007 to 2011

Count20072011 a numeric vector of the incident counts

Pop20072011 a numeric vector of the population

RateTrendAPC a numeric vector of the rate trends per Seer registry

pValue a numeric vector of the P Value related to the Rate Trend

LowerCI a numeric vector of the lower confidence interval

UpperCI a numeric vector of the upper confidence interval

Details

This dataset is a randomly generated collection of data from 2002 to 2011 Seer Registry data to support testing of micromapST functions for the 18 U.S. Seer registries. The data has no relationship to the Seer registries and is only for test purposes. The row names are the Seer registry abbreviations. This dataset is used in the micromapST examples.

Author(s)

Linda W. Pickle and Jim Pearson of StatNet Consulting, LLC, Gaithersburg, MD

`SeoulPopData` *Test data for the Seoul South Korea city district border Group*

Description

This dataset contains the 2012 population and area data for each of the districts in city of Seoul.

Usage

```
data(SeoulPopData)
```

Format

A data frame with 25 observations, 1 for each district, on the following 5 variables.

District a character vector containing the full name of the Seoul district.

City a character vector containing the city name => Seoul.

Pop.2012 a numeric vector of the number of the district's 2012 population.

Area a numeric vector of the area square kilometers for the district.

Founded a character vector containing the dates each district was founded.

Details

This dataset was pulled from the China ??? government website on Dec. 2014. It contains the population and average income per person for each of Kansas' 105 counties.

`SeoulSKoreaBG` *Seoul South Korea border group datasets to support creating micromaps for the 25 districts in the Korean city of Seoul.*

Description

The *micromapST* function has the ability to generate linked micromaps for any geographical area. To specify the geographical area, the '`bordGrp`' call argument is used to specify the border group dataset for the geographical area. The *SeoulSKoreaBG* border group dataset supports creating linked micromaps for the 25 districts in the city of Seoul South Korea. When the '`bordGrp`' call argument is set to *SeoulSKoreaBG*, the appropriate name table (county names and abbreviations) and the boundary data for the 25 districts are loaded in *micromapST*. The user's data is then linked to the boundary data via the district's name, abbreviated or ID based on the table below.

Usage

```
data(SeoulSKoreaBG)
```

Details

The *SeoulSKoreaBG* border group contains the following data.frames::

areaParms - contains specific parameters for the border group

areaNamesAbbrsIDs - containing the names, abbreviations, and numerical identifier for the districts in the city of Seoul.

areaVisBorders - the boundary point lists for each district in the city of Seoul Korea.

L2VisBorders - the boundaries for an intermediate level. For this border group, this boundary data is not used and set to L3VisBorders as a place holder.

RegVisBorders - the boundaries for an regional level. For this border group, this boundary data is not used and set to L3VisBorders as a place holder.

L3VisBorders - the boundary of the city of Seoul South Korea

The Seoul district border group contains 25 district sub-areas. Each district has a row in the areaNamesAbbrsIDs data.frame and a set of polygons in the areaVisBorders data.frame datasets. No regions or L2 boundaries are defined for the Seoul district border group. The RegVisBorders and and L2VisBorders data.frames are set the L3VisBorders data.frame. The regional feature is disabled. The L3VisBorders dataset contains the outline of the city of Seoul.

The details on each of these data.frame structures can be found in the "bordGrp" section of this document. The *areaNamesAbbrsIDs* data.frame provides the linkages to the boundary data for each sub-area (district) using the fullname, abbreviation, and numerical identifier for each country to the *<statsDFrame>* data based on the setting of the '`rowNames`' call parameter.

A column or the data.frame row.names must match one of the types of names in the *areaNamesAbbrsIDs* data.frame name table. If the data row does not match a value in the name table, an warning is issued and the data is ignored. If no data is present for a sub-area (district) in the name table, the sub-area is mapped but not colored.

The following are a list of the names, abbreviations, and ids for each country in the *SeoulSKoreaBG* border group.

name	ab	id
Dobong	DO	11100
Dongdaemun	DN	11060
Dongjak	DG	11200
Eunpyeong	EU	11120
Gangdong	GA	11090
Gangbuk	GN	11250
Gangnam	GG	11230
Gangseo	GS	11160
Geumcheon	GE	11180
Guro	GU	11170
Gwanak	GW	11210
Gwangjin	GJ	11050

Jongro	JO	11010
Jung	JU	11070
Jungnang	JN	11020
Mapo	MA	11140
Nowon	NO	11110
Seocho	SE	11220
Seodaemun	SD	11130
Seongbuk	SN	11080
Seongdong	SG	11040
Songpa	SO	11240
Yangcheon	YA	11150
Yeongdeungpo	YE	11190
Yongsan	YO	11030

The *id* field value is the ISO numerical code for the district.

The ‘`rowNames`’ = "alias" or "alt_ab" and the ‘`regions`’ features are not supported in the *SeoulSKoreaBG* border group.

`statePop2010` *US State Population for 2010*

Description

US State 2010 population data by race and Hispanic ethnicity.

Usage

```
data(statePop2010)
```

Format

A data frame with 51 observations (one per state) on the following 6 variables per state:

Hisp an integer count of the Hispanic population

white an integer count of the white population

black an integer count of the black population

OtherWBH an integer count of the population other than white, black or Hispanic

pctHisp a numeric percentage of the Hispanic population to the total population

pctOtherWBH a numeric percentage of the population other than white, black, or Hispanic

Each row has the state 2 character abbreviation as its row name.

Details

The dataset contains 51 records, one for each state. The data represents the population count or percentage of the total population by race and Hispanic ethnicity within the state. This dataset is used by the micromapSEER examples with the "USStatesDF" border group.

Source

United States Census Bureau, Population Total by State, by Race, Combinations of Two Races, and not Hispanic or Latino, 2010 (Summary File 1, Table QT-P4), URL = http://factfinder2.census.gov/.

`SynTable` *This data set contains a synonym table to help translate common incorrect location id strings*

Description

When the user supplied data does not contain the exact Name or Abbr string to match the name table information. The data row cannot be linked or map to the micromapST graphics. An example is the many ways the District of Columbia in the U. S. is identified. Originally special code was used to identify these mismatches and correct them. The SynTable dataset now provide a open method to address the problem. The common string representing an area can be entered into the table and the correct Name and Abbr equivalent. When a data row's name does not match the name table, the Synonym Table is referenced to see it there is an alternative value to use.

Usage

```
data(SynTable)
```

Format

A data frame with 51 observations (one per state) on the following 7 variables.

`locid` a character string representing the name of the row in the user's provided data.frame.

`Name` a character string containing the equivalent name table Name column value.

`Abbr` a character string containing the equivalent name table Abbr column value.

Author(s)

Jim Pearson, Statnet Consulting, Gaithersburg, MD 20877

See Also

micromapST

`TSdata` *Time Series Example Dataset*

Description

Data for Time Series Examples

The data are age-adjusted (2000 U.S. standard) female lung cancer mortality rates (per 100,000 population) for each year from 1996 to 2010.

Usage

```
data(TSdata)
```

Format

This dataset is an array with dimensions of 51, 15, 4. The rownames of the array are the 51 state and DC abbreviations (2 characters). TSdata[,,1:4] contains the x (time) value, followed by the value for the line, then the lower 95% confidence limit, and finally the upper 95% confidence limit value.

Details

The first dimension [*st*,,] of 51 elements contains each state or DC. This dimension is referenced by the rownames of the array.

The second dimension [,*t*,] of *n* elements in this case are the time periods in the time series. Our example uses the years 1996 to 2010 as the time period values. A reasonable number of points is between 20 and 30.

The third dimension [,,*v*] of 2 or 4 elements is the x or y values during the time period. If no confidence data is provided, the third dimension is 2:

- data[,,*1*] is the X value
- data[,,*2*] is the mid-Y value (Y)

If a confidence band is being plotted in '`\var{tsconf}`' graphs then there are 4 elements.

- data[,,*1*] is the X value
- data[,,*2*] is the mid-Y value (Y)
- data[,,*3*] is the low-Y value
- data[..*4*] is the high-Y value

For example, the x,y coordinates for year=1996 (time period = 1) for the first state (*AK*) is TSdata[*1*,*1*,c(1,2)].

This approach was done to allow a data matrix built for the "tsconf" glyphs to be used for a *ts* glyphs.

This data is used by micromapSEER with the "USStatesDF" border group.

```
# how to create a new time series data set
tempTS <-read.table("...yourfilename.csv",sep=",",header=T)
yrmat  <-matrix(rep(1996:2010,51),nrow=51,ncol=15,byrow=T)  # year labels
ratemat<-as.matrix(
       tempTS[,c(8,13,18,23,28,33,38,43,48,53,58,63,68,73,78)]
       )
locimat<-as.matrix(
       tempTS[,c(9,14,19,24,29,34,39,44,49,54,59,64,69,74,79)]
       )
hicimat<-as.matrix(
       tempTS[,c(10,15,20,25,30,35,40,45,50,55,60,65,70,75,80)]
       )
workmat<-cbind(yrmat,ratemat,locimat,hicimat)
TSdata <-NULL
TSdata <-array(workmat,dim=c(51,15,4))
# change state ab from factors to characters.
rownames(TSdata)<-as.character(tempTS$stab)
```

Source

Surveillance, Epidemiology, and End Results (SEER) Program (www.seer.cancer.gov) SEER*Stat Database: Mortality - All COD, Aggregated With State, Total U.S. (1969-2010) (Katrina/Rita Population Adjustment), National Cancer Institute, DCCPS, Surveillance Research Program, Surveillance Systems Branch, released April 2013.
Underlying mortality data provided by NCHS (www.cdc.gov/nchs).

`UKIrelandBG` *UKIrelandBG border group datasets contains the boundary information for the United Kingdom and Ireland.*

Description

The *micromapST* function has the ability to generate linked micromaps for any geographical area. To specify the geographical area, the '`bordGrp`' call argument is used to specify the border group dataset for the geographical area. The *UKIrelandBG* border group dataset supports creating linked micromaps for the all of the United Kingdom including Northern Ireland and the Isle of Man and Ireland. When the '`bordGrp`' call argument is set to *UKIrelandBG*, the appropriate name table (county names and abbreviations) and the 219 sub-areas (counties, etc.) boundary data is loaded into *micromapST*. The user's data is then linked to the boundary data via the name, abbreviation, or alternate abbreviation for each sub-area (county, etc.).

The United Kingdom and Ireland information was pulled from the UK and Ireland public web sites in March of 2015.

The UKIreland border group was constructed to provide an area with more than 100 sub-areas for testing *micromapST* and enhancing it's ability to handle a large number of sub-areas and generate usable linked micromaps.

Usage

```
data(UKIrelandBG)
```

Details

The *UKIrelandBG* border group dataset contains the following data.frames:

areaParms - contains specific parameters for the border group

areaNamesAbbrsIDs - containing the names, abbreviations, alternate abbreviation, or numerical identifier for each of the United Kingdom or Ireland counties.

areaVisBorders - the boundary point lists for each sub-area.

L2VisBorders - the boundaries for an intermediate level. For the United Kingdom and Ireland border group, this boundary point list is not used and is set to equal L3VisBorders data.frame for the border group.

RegVisBorders - the boundaries for the 4 United Kingdom and Ireland regions or realms: England, Wales, Scotland, Northern Ireland, Ireland, and Isle of Man.

L3VisBorders - the boundary of the United Kingdom and Ireland area.

The UKIreland border group contains 219 sub-areas (counties, etc.) Each registry has a row in the *areaNamesAbbrsIDs* data.frame and a set of polygons in the areaVisBorders data.frame datasets. Regions are defined in this border group as the 6 country and kingdom regions in the UK and Ireland. The regions feature is enable. The siz (6) regions are: England, Scotland, Wales, Northern Ireland, Ireland and Isle of Man. The names, abbreviations, alternate abbreviations and IDs for the counties in the UKIreland border group are:

Name	ab	id	alt_ab	region
Aberdeen	GB.ABE	826139	ABE	SCT
Aberdeenshire	GB.ABD	826140	ABD	SCT
c	GB.AGY	826171	AGY	WLS
Angus	GB.ANS	826141	ANS	SCT
Antrim	GB.ANT	826113	ANT	NIR
Ards	GB.ARD	826114	ARD	NIR
Argyll and Bute	GB.AGB	826142	AGB	SCT
Armagh	GB.ARM	826115	ARM	NIR
Ballymena	GB.BLA	826116	BLA	NIR
Ballymoney	GB.BLY	826117	BLY	NIR
Banbridge	GB.BNB	826118	BNB	NIR
Barking and Dagenham	GB.BDG	826001	BDG	ENG
Bath and North East Somerset	GB.BAS	826002	BAS	ENG
Bedfordshire	GB.BDF	826003	BDF	ENG
Belfast	GB.BFS	826119	BFS	NIR
Berkshire	GB.BRK	826004	BRK	ENG
Bexley	GB.BEX	826005	BEX	ENG
Blackburn with Darwen	GB.BBD	826006	BBD	ENG
Blaenau Gwent	GB.BGW	826172	BGW	WLS
Bournemouth	GB.BMH	826007	BMH	ENG
Brent	GB.BEN	826008	BEN	ENG

Bridgend	GB.BGE	826173	BGE	WLS
Brighton and Hove	GB.BNH	826009	BNH	ENG
Bristol	GB.BST	826010	BST	ENG
Bromley	GB.BRY	826011	BRY	ENG
Buckinghamshire	GB.BKM	826012	BKM	ENG
Caerphilly	GB.CAY	826174	CAY	WLS
Cambridgeshire	GB.CAM	826013	CAM	ENG
Camden	GB.CMD	826014	CMD	ENG
Cardiff	GB.CRF	826175	CRF	WLS
Carlow	IE.CW	372001	CW	IRE
Carmarthenshire	GB.CMN	826176	CMN	WLS
Carrickfergus	GB.CKF	826120	CKF	NIR
Castlereagh	GB.CSR	826121	CSR	NIR
Cavan	IE.CN	372002	CN	IRE
Ceredigion	GB.CGN	826177	CGN	WLS
Cheshire	GB.CHS	826015	CHS	ENG
Clackmannanshire	GB.CLK	826143	CLK	SCT
Clare	IE.CE	372003	CE	IRE
Coleraine	GB.CLR	826122	CLR	NIR
Conwy	GB.CWY	826178	CWY	WLS
Cookstown	GB.CKT	826123	CKT	NIR
Cork	IE.CO	372004	CO	IRE
Cornwall	GB.CON	826016	CON	ENG
Craigavon	GB.CGV	826124	CGV	NIR
Croydon	GB.CRY	826017	CRY	ENG
Cumbria	GB.CMA	826018	CMA	ENG
Darlington	GB.DAL	826019	DAL	ENG
Denbighshire	GB.DEN	826179	DEN	WLS
Derby	GB.DER	826020	DER	ENG
Derbyshire	GB.DBY	826021	DBY	ENG
Derry	GB.DRY	826125	DRY	NIR
Devon	GB.DEV	826022	DEV	ENG
Donegal	IE.DL	372005	DL	IRE
Dorset	GB.DOR	826023	DOR	ENG
Down	GB.DOW	826126	DOW	NIR
Dublin	IE.D	372006	D	IRE
Dumfries and Galloway	GB.DGY	826144	DGY	SCT
Dundee	GB.DND	826145	DND	SCT
Dungannon	GB.DGN	826127	DGN	NIR
Durham	GB.DUR	826024	DUR	ENG
Ealing	GB.EAL	826025	EAL	ENG
East Ayrshire	GB.EAY	826146	EAY	SCT
East Dunbartonshire	GB.EDU	826147	EDU	SCT
East Lothian	GB.ELN	826148	ELN	SCT
East Renfrewshire	GB.ERW	826149	ERW	SCT
East Riding of Yorkshire	GB.ERY	826026	ERY	ENG
East Sussex	GB.ESX	826027	ESX	ENG
Edinburgh	GB.EDH	826150	EDH	SCT

Eilean Siar	GB.ELS	826151	ELS	SCT
Enfield	GB.ENF	826028	ENF	ENG
Essex	GB.ESS	826029	ESS	ENG
Falkirk	GB.FAL	826152	FAL	SCT
Fermanagh	GB.FER	826128	FER	NIR
Fife	GB.FIF	826153	FIF	SCT
Flintshire	GB.FLN	826180	FLN	WLS
Galway	IE.G	372007	G	IRE
Glasgow	GB.GLG	826154	GLG	SCT
Gloucestershire	GB.GLS	826030	GLS	ENG
Greenwich	GB.GRE	826031	GRE	ENG
Gwynedd	GB.GWN	826181	GWN	WLS
Hackney	GB.HCK	826032	HCK	ENG
Halton	GB.HAL	826033	HAL	ENG
Hammersmith and Fulham	GB.HMF	826034	HMF	ENG
Hampshire	GB.HAM	826035	HAM	ENG
Haringey	GB.HRY	826036	HRY	ENG
Harrow	GB.HRW	826037	HRW	ENG
Hartlepool	GB.HPL	826038	HPL	ENG
Havering	GB.HAV	826039	HAV	ENG
Herefordshire	GB.HEF	826040	HEF	ENG
Hertfordshire	GB.HRT	826041	HRT	ENG
Highland	GB.HLD	826155	HLD	SCT
Hillingdon	GB.HIL	826042	HIL	ENG
Hounslow	GB.HNS	826043	HNS	ENG
Inverclyde	GB.IVC	826156	IVC	SCT
Isle of Wight	GB.IOW	826044	IOW	ENG
Islington	GB.ISL	826045	ISL	ENG
Kensington and Chelsea	GB.KEC	826046	KEC	ENG
Kent	GB.KEN	826047	KEN	ENG
Kerry	IE.KY	372008	KY	IRE
Kildare	IE.KE	372009	KE	IRE
Kilkenny	IE.KK	372010	KK	IRE
Kingston upon Hull	GB.KHL	826048	KHL	ENG
Kingston upon Thames	GB.KTT	826049	KTT	ENG
Lambeth	GB.LBH	826050	LBH	ENG
Lancashire	GB.LAN	826051	LAN	ENG
Laoighis	IE.LS	372011	LS	IRE
Larne	GB.LRN	826129	LRN	NIR
Leicester	GB.LCE	826052	LCE	ENG
Leicestershire	GB.LEC	826053	LEC	ENG
Leitrim	IE.LM	372012	LM	IRE
Lewisham	GB.LEW	826054	LEW	ENG
Limavady	GB.LMV	826130	LMV	NIR
Limerick	IE.LK	372013	LK	IRE
Lincolnshire	GB.LIN	826055	LIN	ENG
Lisburn	GB.LSB	826131	LSB	NIR
London	GB.LND	826056	LND	ENG

Longford	IE.LD	372014	LD	IRE
Louth	IE.LH	372015	LH	IRE
Luton	GB.LUT	826057	LUT	ENG
Magherafelt	GB.MFT	826132	MFT	NIR
Manchester	GB.MAN	826058	MAN	ENG
Mayo	IE.MO	372016	MO	IRE
Meath	IE.MH	372017	MH	IRE
Medway	GB.MDW	826059	MDW	ENG
Merseyside	GB.MSY	826060	MSY	ENG
Merthyr Tydfil	GB.MTY	826182	MTY	WLS
Merton	GB.MRT	826061	MRT	ENG
Middlesbrough	GB.MDB	826062	MDB	ENG
Midlothian	GB.MLN	826157	MLN	SCT
Milton Keynes	GB.MIK	826063	MIK	ENG
Monaghan	IE.MN	372018	MN	IRE
Monmouthshire	GB.MON	826183	MON	WLS
Moray	GB.MRY	826158	MRY	SCT
Moyle	GB.MYL	826133	MYL	NIR
Neath Port Talbot	GB.NTL	826184	NTL	WLS
Newham	GB.NWM	826064	NWM	ENG
Newport	GB.NWP	826185	NWP	WLS
Newry and Mourne	GB.NYM	826134	NYM	NIR
Newtownabbey	GB.NTA	826135	NTA	NIR
Norfolk	GB.NFK	826065	NFK	ENG
North Ayrshire	GB.NAY	826159	NAY	SCT
North Down	GB.NDN	826136	NDN	NIR
North East Lincolnshire	GB.NEL	826066	NEL	ENG
North Lanarkshire	GB.NLK	826160	NLK	SCT
North Lincolnshire	GB.NLN	826067	NLN	ENG
North Somerset	GB.NSM	826068	NSM	ENG
North Yorkshire	GB.NYK	826069	NYK	ENG
Northamptonshire	GB.NTH	826070	NTH	ENG
Northumberland	GB.NBL	826071	NBL	ENG
Nottingham	GB.NGM	826072	NGM	ENG
Nottinghamshire	GB.NTT	826073	NTT	ENG
Offaly	IE.OY	372019	OY	IRE
Omagh	GB.OMH	826137	OMH	NIR
Orkney Islands	GB.ORK	826161	ORK	SCT
Oxfordshire	GB.OXF	826074	OXF	ENG
Pembrokeshire	GB.PEM	826186	PEM	WLS
Perthshire and Kinross	GB.PKN	826162	PKN	SCT
Peterborough	GB.PTE	826075	PTE	ENG
Plymouth	GB.PLY	826076	PLY	ENG
Poole	GB.POL	826077	POL	ENG
Portsmouth	GB.POR	826078	POR	ENG
Powys	GB.POW	826187	POW	WLS
Redbridge	GB.RDB	826079	RDB	ENG
Redcar and Cleveland	GB.RCC	826080	RCC	ENG

Renfrewshire	GB.RFW	826163	RFW	SCT
Rhondda, Cynon, Taff	GB.RCT	826188	RCT	WLS
Richmond upon Thames	GB.RIC	826081	RIC	ENG
Roscommon	IE.RN	372020	RN	IRE
Rutland	GB.RUT	826082	RUT	ENG
Scottish Borders	GB.SCB	826164	SCB	SCT
Shetland Islands	GB.ZET	826165	ZET	SCT
Shropshire	GB.SHR	826083	SHR	ENG
Sligo	IE.SO	372021	SO	IRE
Somerset	GB.SOM	826084	SOM	ENG
South Ayrshire	GB.SAY	826166	SAY	SCT
South Gloucestershire	GB.SGC	826085	SGC	ENG
South Lanarkshire	GB.SLK	826167	SLK	SCT
South Yorkshire	GB.SYK	826086	SYK	ENG
Southampton	GB.STH	826087	STH	ENG
Southend-on-Sea	GB.SOS	826088	SOS	ENG
Southwark	GB.SWK	826089	SWK	ENG
Staffordshire	GB.STS	826090	STS	ENG
Stirling	GB.STG	826168	STG	SCT
Stockton-on-Tees	GB.STT	826091	STT	ENG
Stoke-on-Trent	GB.STE	826092	STE	ENG
Strabane	GB.STB	826138	STB	NIR
Suffolk	GB.SFK	826093	SFK	ENG
Surrey	GB.SRY	826094	SRY	ENG
Sutton	GB.STN	826095	STN	ENG
Swansea	GB.SWA	826189	SWA	WLS
Swindon	GB.SWD	826096	SWD	ENG
Telford and Wrekin	GB.TFW	826097	TFW	ENG
Thurrock	GB.THR	826098	THR	ENG
Tipperary	IE.TA	372022	TA	IRE
Torbay	GB.TOB	826099	TOB	ENG
Torfaen	GB.TOF	826190	TOF	WLS
Tower Hamlets	GB.TWH	826100	TWH	ENG
Tyne and Wear	GB.TWR	826101	TWR	ENG
Vale of Glamorgan	GB.VGL	826191	VGL	WLS
Waltham Forest	GB.WFT	826102	WFT	ENG
Wandsworth	GB.WND	826103	WND	ENG
Warrington	GB.WRT	826104	WRT	ENG
Warwickshire	GB.WAR	826105	WAR	ENG
Waterford	IE.WD	372023	WD	IRE
West Dunbartonshire	GB.WDU	826169	WDU	SCT
West Lothian	GB.WLN	826170	WLN	SCT
West Midlands	GB.WMD	826106	WMD	ENG
West Sussex	GB.WSX	826107	WSX	ENG
West Yorkshire	GB.WYK	826108	WYK	ENG
Westmeath	IE.WH	372024	WH	IRE
Westminster	GB.WSM	826109	WSM	ENG
Wexford	IE.WX	372025	WX	IRE

Wicklow	IE.WW	372026	WW	IRE
Wiltshire	GB.WIL	826110	WIL	ENG
Worcestershire	GB.WOR	826111	WOR	ENG
Wrexham	GB.WRX	826192	WRX	WLS
York	GB.YOR	826112	YOR	ENG
Isle of Man	IM	833000	IMN	IMN

When compiling the abbreviations for this border group, multiple sets of abbreviations were found. The two most common sets are included in the border group as "ab" and "alt_ab" types of '`rowNames`'.

The L3VisBorders dataset contains the outline of the UK and Ireland.

The details on each of these data.frame structures can be found in the "bordGrp" section of this document. The *areaNamesAbbrsIDs* data.frame provides the linkages to the boundary data for each sub-area (registry) using the fullname, abbreviation, alternate abbreviation and numerical identifier for each county/provence to the *<statsDFrame>* data based on the setting of the '`rowNames`' call argument.

A column or the data.frame row.names must match one of the types of names in the *areaNamesAbbrsIDs* data.frame name table. If the data row does not match a value in the name table, an warning is issued and the data is ignored. If no data is present for a sub-area in the name table, the sub-area is mapped but not colored.

The '`dataRegionsOnly`' call parameter instructs the package to only map the regions with Seer registers with data. The regions are the four census regions: England, Scotland, Wales, Isle of Man, Northern Irelend and Ireland.

Source

NCI

References

???? Retrieved 2013-01-10.

`UKIrelandPopData` *Test data for the UK-Ireland border Group*

Description

The UKIrelandPopData and UKIrelandPopData2 datasets contain the county populations and area statistics for the UK and Ireland..

Usage

```
data(UKIrelandPopData)
```

Format

A data frame with 218 rows, 1 for each county, with 5 variables for each county:

Name a character vector containing the UK-Ireland county name.

Abbr a character vector containing the ISO 2 character abbreviation for the UK and Ireland country. Warning: the UKIreland border group uses the newer ISO 3 character notation for it\'s abbreviations.

Pop a numeric vector of the county's population.

Area.M a numeric vector of the county's area in square miles.

Area.Km a numeric vector of the county's area in square kilometers.

Details

This dataset was pulled from several UK and Ireland website in January, 2015. The difference between UKIrelandPopData and UKIrelandPopData2 is incorrect labeling of Anglesey as Anglesey, Isle of.

USSeerBG — *USSeerBG border group datasets to support use with U.S. 20 Seer areas/registries*

Description

The *micromapST* function has the ability to generate linked micromaps for any geographical area. To specify the geographical area, the '`bordGrp`' call argument is used to specify the border group dataset for the geographical area. The *USSeerBG* border group dataset supports creating linked micromaps for the 20 Seer registries in the U. S. When the '`bordGrp`' call argument is set to *USSeerBG*, the appropriate name table (county names and abbreviations) and the 20 sub-areas (Seer registries) boundary data is loaded in *micromapST*. The user's data is then linked to the boundary data via the Seer registry's name, abbreviated, alias match or ID based on the table below.

The 20 U. S. Seer registries are the accepted registries as of January 2010 funded by NCI.

Usage

```
data(USSeerBG)
```

Details

The *USSeerBG* border group dataset contains the following data.frames:

areaParms - contains specific parameters for the border group

areaNamesAbbrsIDs - containing the names, abbreviations, numerical identifier and alias matching string for each of the 20 Seer registries.

areaVisBorders - the boundary point lists for each area.

L2VisBorders - the boundaries for an intermediate level. For Seer registry border group, L2VisBorders contains the boundaries for the 51 states and DC in the U. S to help provide a geographical reference of the registries to the states.

RegVisBorders - the boundaries for the 4 U. S. Census regions in the U. S in support of the region feature.

L3VisBorders - the boundary of the U. S.

The Seer Registries border group contains 20 Seer Registry sub-areas. Each registry has a row in the *areaNamesAbbrsIDs* data.frame and a set of polygons in the areaVisBorders data.frame datasets.

Regions are defined in this border group as the 4 census regions in the U. S. The regions feature is enable. The four census regions are: NorthEast, South, MidWest, and West. The states and Seer registries in each region are:

state	Seer Registries	region
Alabama	<none>	South
Alaska	Alaska Natives	West
Arizona	Arizona Natives	West
Arkansas	<none>	South
California	California-LA, California-Other, California-SF, California-SJ	West
Colorado	<none>	West
Connecticut	Connecticut	NorthEast
Delaware	<none>	South
District of Columbia	<none>	South
Florida	<none>	South
Georgia	Georgia-Atlanta, Georgia-Other, Georgia-Rural	South
Hawaii	Hawaii	West
Idaho	<none>	West
Illinois	<none>	MidWest
Indiana	<none>	MidWest
Iowa	Iowa	MidWest
Kansas	<none>	MidWest
Kentucky	Kentucky	South
Louisiana	Louisiana	South
Maine	<none>	NorthEast
Maryland	<none>	South
Massachusetts	<none>	NorthEast
Michigan	Michigan-Detroit	MidWest
Minnesota	<none>	MidWest
Mississippi	<none>	South
Missouri	<none>	MidWest
Montana	<none>	West
Nebraska	<none>	MidWest
Nevada	<none>	West

New Hampshire	<none>	NorthEast
New Jersey	New Jersey	NorthEast
New Mexico	New Mexico	West
New York	<none>	NorthEast
North Carolina	<none>	South
North Dakota	<none>	MidWest
Ohio	<none>	MidWest
Oklahoma	Oklahoma-Cherokee	South
Oregon	<none>	West
Pennsylvania	<none>	NorthEast
Rhode Island	<none>	NorthEast
South Carolina	<none>	South
South Dakota	<none>	MidWest
Tennessee	<none>	South
Texas	<none>	South
Utah	Utah	West
Vermont	<none>	NorthEast
Virginia	<none>	South
Washington	Washington-Seattle	South
West Virginia	<none>	South
Wisconsin	<none>	MidWest
Wyoming	<none>	West

The L3VisBorders dataset contains the outline of the United States.

The details on each of these data.frame structures can be found in the "bordGrp" section of this document. The *areaNamesAbbrsIDs* data.frame provides the linkages to the boundary data for each sub-area (registry) using the fullname, abbreviation, and numerical identifier for each country to the *<statsDFrame>* data based on the setting of the '`rowNames`' call argument.

A column or the data.frame row.names must match one of the types of names in the *areaNamesAbbrsIDs* data.frame name table. If the data row does not match a value in the name table, an warning is issued and the data is ignored. If no data is present for a sub-area (registry) in the name table, the sub-area (registry) is mapped but not colored.

The following are a list of the names, abbreviations, alias and IDs for each country in the *USSeerBG* border group.

Name	ab	alias string	id	counties	region
Alaska Natives	AK-NAT	ALASKA NATIVES	18	all	West
Arizona Natives	AZ-NAT	ARIZONA NATIVES	20	all	West
California-LA	CA-LA	LOS ANGELES	4	Los Angeles	West
California-SF	CA-SF	SAN FRANCISCO	2	Alameda, Contra Costa, Marin, San Francisco, San Mateo	West
California-SJ	CA-SJ	SAN JOSE	3	Montersey San Benito,	West

				Santa Clara, Santa Cruz	
California-Other	CA-OTH	CALIFORNIA EXCLUDING	5	all other counties	West
Connecticut	CT	CONNECTICUT	1	all	NorthEast
Georgia-Atlanta	GA-ATL	ATLANTA	6	Clayton, Cobb, DeKalb, Fulton, Gwinnett	South
Georgia-Rural	GA-RUR	RURAL GEORGIA	8	Glascock, Greene, Hancock, Jasper, Jefferson, Morgan, Putnam, Taliaferro, Warren, Washington	South
Georgia-Other	GA-OTH	GREATER GEORGIA	7	all other counties	South
Hawaii	HI	HAWAII	9	all	West
Iowa	IA	IOWA	10	all	MidWest
Kentucky	KY	KENTUCKY	14	all	South
Michigan-Detroit	MI-DET	DETROIT	15	Macomb, Oakland, Wayne	MidWest
New Jersey	NJ	NEW JERSEY	11	all	NorthEast
New Mexico	NM	NEW MEXICO	12	all	West
Oklahoma-Cherokee	OK-CHE	OKLAHOMA	19	Adair, Cherokee, Craig, Delaware, Mayes, McIntosh, Muskogee, Nowata, Ottawa, Rogers, Seqouyah, Tulsa, Wagnorer, Washington	South
Utah	UT	UTAH	16	all	West
Washington-Seattle	WA-SEA	SEATTLE	17	Clallam, Grays Harbor, Island, Jefferson, King, Kitsap, Mason, Pierce, San Juan, Skagit, Snohomish, Thurston, Whatcom	South

The ‘rowNames’ = *alias* and the ‘regions’ = *TRUE* features are enabled in the *USSeerBG* border group.

The *alias* option is designed to allow the package to match the registry labels created by the Seer Stat website when exporting Seer data for analysis. The alias match is a "contains" match, so the registry field in the user data must "contain" the "alias" values listed in the above table. To help generalize the match, the user’s registry value is stripped of any punctuation, control characters and multiple spaces (blanks, tabs, cr, lf) are reduced to a single blank and the string is converted to all upper case. Then the wild card match is performed.

The ‘dataRegionOnly’ call parameter (when set to *TRUE*) instructs the package to only map the regions with Seer registers with data. The regions used are the four census regions: NorthEast, South, MidWest and West. The RegVisBorders data.frame contains the outline of each of these regions. For example: if Seer registry data is provided for the only the New Mexico, Utah and California Registries in the West region, then only the states and regional boundary for the West region are drawn.

The *USSeerBG* border group does not contain or support an alternate set of abbreviations. If ‘rowNames’ is set to *alt_ab*, an warning is generated and the standard Seer registry abbreviations are used.

The following steps should be used to export data for *micromapST*’s use from the SEER*Stat Website:

1. Log on to the SEER^Stat website.
2. Create the matrix of results you want in SEER*Stat.
3. Click on Matrix, Export, Results as Text File (if you created multiple matrices of results, make sure that the one you want to export is highlighted)
4. In the Matrix Export Options window, click on:
   (a) Output variables as Labels without quotes
   (b) Remove all thousands separators
   (c) Output variable names before data
   (d) Preserve matrix columns & rename fields
   (e) Leave defaults clicked for Line delimiter, Missing Character, and Field delimiter
5. Change names and locations of text and dictionary files from defaults to the appropriate name and directory location.

To read the resulting text file into R use the read.delim function with ‘header’ = *TRUE*. Follow the read.delim call with a str function to verify the data was read correctly.

```
dataT <- read.delim("c:\datadir\seerstat.txt",header=FALSE)
str(dataT)
```

Source

NCI

References

United States National Cancer Institute Seer Website at www.seer.cancer.gov; Seer Software at seer.cancer.gov/seerstat.; United States Census Bureau, Geography Division. "Census Regions and Divisions of the United States" (PDF). Retrieved 2013-01-10.

`USStatesBG` *USStatesBG border group datasets to support use with U.S. States amd D.C. Areas*

Description

The *micromapST* function has the ability to generate linked micromaps for any geographical area. To specify the geographical area, the '`bordGrp`' call argument is used to specify the border group dataset for the geographical area. When *micromapSt* function is used to micromap the U.S. States and DC areas, no border group needs to be specified. The *USStatesBG* border group is the same as the original sub-areas (states and DC) and boundaries used by the all previous versions of the *micromapST* package. By default *micromapST* loads the area fullnames, abbreviations, IDs and boundaries files for 50 U. S. states and the District of Columbia for the processing of user data and the creation of the requested linked micromap.

Usage

```
data(USStatesBG)
```

Details

The *USStatesBG* border group contains in the following data.frames:

areaParms - specific parameters associated with this border group

areaNamesAbbrsIDs - containing the names, abbreviations, numerical identifier and alias matching string for each of the 51 U. S. States and D.C.

areaVisBorders - the boundary point lists for each of the 51 States and D.C..

L2VisBorders - the boundaries for the U. S. states and DC. It is identical to *areaVisBorders* data.frame.

RegVisBorders - the boundaries for the 4 U. S. census regions

L3VisBorders - the boundary of the U.S

Refer to the section on the border group data.frames for a detailed discussion on the formats and usage of each of the above data.frame.

In this border group, there are 51 areas (states and DC) and information and names in the areaNamesAbbrIDs data.frame and there boundaries in the areaVisBorders data.frame. The L2VisBorders data.frame contains a copy of the areaVisBorders data.frame to allow heavier overlaying of the state and DC boundaries during mapping. The RegVisBorders data.frame contains the information and boundaries for the 4 U. S. census regions. The L3VisBorders dataset contains the information on the boundaries of the U.S.

Alaska and Hawaii are relocated to below California and Alaska is reduced in size by 50

The names, abbreviations, id, and assigned U. S. regions used in the areaNamesAbbrsIDs data.frame the the 50 U. S. States and DC are as follows:

name	ab	id	region
Alabama	AL	01	South
Alaska	AK	02	West
Arizona	AZ	04	West
Arkansas	AR	05	South
California	CA	06	West
Colorado	CO	08	West
Connecticut	CT	09	NorthEast
Delaware	DE	10	South
DC	DC	11	South
Florida	FL	12	South
Georgia	GA	13	South
Hawaii	HI	15	West
Idaho	ID	16	West
Illinois	IL	17	MidWest
Indiana	IN	18	MidWest
Iowa	IA	19	MidWest
Kansas	KS	20	MidWest
Kentucky	KY	21	South
Louisiana	LA	22	South
Maine	ME	23	NorthEast
Maryland	MD	24	South
Massachusetts	MA	25	NorthEast
Michigan	MI	26	MidWest
Minnesota	MN	27	MidWest
Mississippi	MI	28	South
Missouri	MO	29	MidWest
Montana	MT	30	West
Nebraska	NE	31	MidWest
Neveda	NV	32	West
New Hampshire	NH	33	NorthEast
New Jersey	NJ	34	NorthEast
New Mexico	NM	35	West
New York	NY	36	NorthEast
North Carolina	NC	37	South
North Dakota	ND	38	MidWest
Ohio	OH	39	MidWest
Oklahoma	OK	40	South
Oregon	OR	41	West
Pennsylvania	PA	42	NorthEast
Rhode Island	RI	44	NorthEast
South Carolina	SC	45	South
South Dakota	SD	46	MidWest
Tennessee	TN	47	South

Texas	TX	48	South
Utah	UT	49	West
Vermont	VT	50	NorthEast
Virginia	VA	51	South
Washington	WA	53	West
West Virginia	WV	54	South
Wisconsin	WI	55	MidWest
Wyoming	WY	56	West

All data must be tagged with the name, abbreviation or id strings to be able to find the associated boundaries information. With the *USStatesBG* only, the package will accept several variations on the full name for DC. They include: Washington, D.C., District of Columbia, and D. C. All can be with or without punctuation and upper and lower case. When detected, the name is translated to "DC".

The '`dataRegionsOnly`' call parameter can be set to *TRUE* to request the package to limit the mapping to only states in regional areas where data is being mapped and omit states and regions that do not contain data. This allows the caller to focus the micromaps on one of the four (4) census regions: NorthEast, South, Midwest, or West.

The '`rowNames`' = *alias*, '`rowNames`' = *alt_ab* are not support in the '`USStatesBG`' border group.

Source

NIST - Federal Information Processing Standards and U. S. Census website.

References

NIST FIPS 6-4 Standards

`UtahBG` *UtahBG border group datasets to support creating micromaps for the counties in the state of Utah*

Description

The *micromapST* function has the ability to generate linked micromaps for any geographical area. To specify the geographical area, the '`bordGrp`' call parameter is used to specify the border group dataset for the geographical area. The *UtahBG* border group dataset is contained within this package and supports creating linked micromaps for the 29 counties in the state of Utah. When the '`bordGrp`' call parameter is set to *UtahBG*, the appropriate name table (county names and abbreviations) and the 29 sub-areas (countries) boundary data is loaded in *micromapST*. The user's data is then linked to the boundary data via the county's name, abbreviation, alternate abbreviation, or ID based on the table below.

Usage

```
data(UtahBG)
```

Details

The *UtahBG* border group dataset contains the following data.frames:

areaParms - contains specific parameters for the border group

areaNamesAbbrsIDs - containing the names, abbreviations, and numerical identifier for the 29 counties in the state of Utah.

areaVisBorders - the boundary point lists for each county area in Utah.

L2VisBorders - the boundaries for an intermediate level. For this border group, this boundary data.frame is not used and set to L3VisBorders as a place holder.

RegVisBorders - the boundaries for an intermediate level. For this border group, this boundary data.frame is not used and set to L3VisBorders as a place holder.

L3VisBorders - the boundary of the state of Utah

The Utah county border group contains 29 county sub-areas. Each county has a row in the areaNamesAbbrsIDs data.frame and a set of polygons in the areaVisBorders data.frame datasets. No regions are defined in the Utah county border group, so the L2VisBorders dataset is not used and the regions option is disabled. The L3VisBorders dataset contains the outline of the state of Utah.

The details on each of these data.frame structures can be found in the "bordGrp" section of this document. The *areaNamesAbbrsIDs* data.frame provides the linkages to the boundary data for each sub-area (county) using the fullname, abbreviation, and numerical identifier for each country to the *<statsDFrame>* data based on the setting of the '`rowNames`' call argument.

A identified column (*idCol*) or the data.frame row.names must match one of the types of names in the border group's *areaNamesAbbrsIDs* data.frame name table. If the data row does not match a value in the name table, an warning is issued and the data is ignored. If no data is present for a sub-area (county) in the name table, the sub-area is mapped but not colored.

The following are a list of the names, abbreviations, alternate abbreviations and IDs for each country in the *UtahBG* border group.

name	ab	alt_ab	id
Beaver	BV	BEA	49001
Box Elder	BE	BOX	49003
Cache	CH	CAC	49005
Carbon	CA	CAR	49007
Daggett	DAG	DAG	49009
Davis	DAV	DAV	49011
Duchesne	DU	DUC	49013
Emery	EM	EME	49015
Garfield	GA	GAR	49017
Grand	GR	GRA	49019
Iron	IR	IRO	49021
Juab	JB	JUA	49023
Kane	KN	KAN	49025
Millard	MI	MIL	49027
Morgan	MO	MOR	49029
Piute	PT	PIU	49031
Rich	RH	RIC	49033

Salt Lake	SL	SAL	49035
San Juan	SJ	SNJ	49037
Sanpete	SP	SNP	49039
Sevier	SV	SEV	49041
Summit	SU	SUM	49043
Tooele	TO	TOO	49045
Uintah	UI	UIN	49047
Utah	UT	UTA	49049
Wasatch	WS	WST	49051
Washington	WA	WSH	49053
Wayne	WN	WAY	49055
Weber	WB	WEB	49057

When compiling the information for the *UtahBG* border group, it was not clear what was the standard accepted abbreviation for each county. Therefore, two sets of abbreviations for each county are included. The first abbreviation set can be referenced by setting the '`rowNames`' call parameter to "ab". The second (alternate) abbreviation set can be used by setting the '`rowNames`' to "alt_ab".

The *id* field value is the 5 digit U. S. state and county FIPS code.

The '`rowNames`' = "alias" and the '`regions`' features are not supported in the *UtahBG* border group.

`UtahPopData` *Test data for the Utah state border Group*

Description

This dataset contains the Utah county populations for each year from 1940 to 2011.

Usage

```
data(UtahPopData)
```

Format

A data frame with 29 rows, 1 for each county, with 73 variables for each county:

County a character vector containing the Utah county full name.

X1940 a numeric vector of the county's population in 1940.

X1941 a numeric vector of the county's population in 1941.

X1942 a numeric vector of the county's population in 1942.

X1943 a numeric vector of the county's population in 1943.

X1944 a numeric vector of the county's population in 1944.

X1945 a numeric vector of the county's population in 1945.

X1946 a numeric vector of the county's population in 1946.

X1947 a numeric vector of the county's population in 1947.
X1948 a numeric vector of the county's population in 1948.
X1949 a numeric vector of the county's population in 1949.
X1950 a numeric vector of the county's population in 1950.
X1951 a numeric vector of the county's population in 1951.
X1952 a numeric vector of the county's population in 1952.
X1953 a numeric vector of the county's population in 1953.
X1954 a numeric vector of the county's population in 1954.
X1955 a numeric vector of the county's population in 1955.
X1956 a numeric vector of the county's population in 1956.
X1957 a numeric vector of the county's population in 1957.
X1958 a numeric vector of the county's population in 1958.
X1959 a numeric vector of the county's population in 1959.
X1960 a numeric vector of the county's population in 1960.
X1961 a numeric vector of the county's population in 1961.
X1962 a numeric vector of the county's population in 1962.
X1963 a numeric vector of the county's population in 1963.
X1964 a numeric vector of the county's population in 1964.
X1965 a numeric vector of the county's population in 1965.
X1966 a numeric vector of the county's population in 1966.
X1967 a numeric vector of the county's population in 1967.
X1968 a numeric vector of the county's population in 1968.
X1969 a numeric vector of the county's population in 1969.
X1970 a numeric vector of the county's population in 1970.
X1971 a numeric vector of the county's population in 1971.
X1972 a numeric vector of the county's population in 1972.
X1973 a numeric vector of the county's population in 1973.
X1974 a numeric vector of the county's population in 1974.
X1975 a numeric vector of the county's population in 1975.
X1976 a numeric vector of the county's population in 1976.
X1977 a numeric vector of the county's population in 1977.
X1978 a numeric vector of the county's population in 1978.
X1979 a numeric vector of the county's population in 1979.
X1980 a numeric vector of the county's population in 1980.
X1981 a numeric vector of the county's population in 1981.
X1982 a numeric vector of the county's population in 1982.
X1983 a numeric vector of the county's population in 1983.

X1984 a numeric vector of the county's population in 1984.

X1985 a numeric vector of the county's population in 1985.

X1986 a numeric vector of the county's population in 1986.

X1987 a numeric vector of the county's population in 1987.

X1988 a numeric vector of the county's population in 1988.

X1989 a numeric vector of the county's population in 1989.

X1990 a numeric vector of the county's population in 1990.

X1991 a numeric vector of the county's population in 1991.

X1992 a numeric vector of the county's population in 1992.

X1993 a numeric vector of the county's population in 1993.

X1994 a numeric vector of the county's population in 1994.

X1995 a numeric vector of the county's population in 1995.

X1996 a numeric vector of the county's population in 1996.

X1997 a numeric vector of the county's population in 1997.

X1998 a numeric vector of the county's population in 1998.

X1999 a numeric vector of the county's population in 1999.

X2000 a numeric vector of the county's population in 2000.

X2001 a numeric vector of the county's population in 2001.

X2002 a numeric vector of the county's population in 2002.

X2003 a numeric vector of the county's population in 2003.

X2004 a numeric vector of the county's population in 2004.

X2005 a numeric vector of the county's population in 2005.

X2006 a numeric vector of the county's population in 2006.

X2007 a numeric vector of the county's population in 2007.

X2008 a numeric vector of the county's population in 2008.

X2009 a numeric vector of the county's population in 2009.

X2010 a numeric vector of the county's population in 2010.

X2011 a numeric vector of the county's population in 2011. .

Details

This dataset was pulled from the Utah government website in January, 2015.

`wflung00and95` *Lung cancer mortality data for white females, 2000-4 and 1995-9*

Description

Counts and rates of age-adjusted (2000 U.S. standard) lung cancer mortality data among white women, aggregated for 1995-9 and 2000-4.

Usage

```
data(wflung00and95)
```

Format

A data frame with 51 observations, 1 for each state + DC, on the following 12 variables.

Rate.00 a numeric vector of age-adjusted rates by state during 2000-4 for white females

Count.00 a numeric vector of the number of white female lung cancer deaths during 2000-4

Lower.00 a numeric vector of the 95% confidence interval lower bound for white female 2000-4 rates

Upper.00 a numeric vector of the 95% confidence interval upper bound for white female 2000-4 rates

Pop.00 a numeric vector of the white female population during 2000

StdErr.00 a numeric vector of the standard error of the white female 2000-4 rates

Rate.95 a numeric vector of age-adjusted rates by state during 1995-9 for white females

Count.95 a numeric vector of the number of white female lung cancer deaths during 1995-9

Lower.95 a numeric vector of the 95% confidence interval lower bound for white female 1995-9 rates

Upper.95 a numeric vector of the 95% confidence interval upper bound for white female 1995-9 rates

Pop.95 a numeric vector of the white female population estimates for 1995

StdErr.95 a numeric vector of the standard error of the white female 1995-9 rates

Details

The rates on this file are directly age adjusted to the US 2000 standard population and are expressed as the number of deaths per 100,000 person-years. The row names are the 2 character postal codes for the states. The data represents the rates for two periods of time: 2000 to 2004 and 1995 to 1999. This dataset is used in the *micromapSEER* examples using the border group of "USStatesDF".

Author(s)

Linda W. Pickle and Jim Pearson of StatNet Consulting, LLC, Gaithersburg, MD

Source

Surveillance Research Program, National Cancer Institute SEER*Stat software (https://www.seer.cancer.gov/seerstat), November 2007 data submission, released April 2008. Data originally provided to NCI by the National Center for Health Statistics.

`wflung00and95US` *wflung 2000 to 2004 and 1995 to 1999 US data*

Description

Counts and age-adjusted rate of white female lung cancer for the total U.S. for the aggregated periods 1995-9 and 2000-4.

Usage

```
data(wflung00and95US)
```

Format

A data frame with 1 observation on the following 13 variables.

Rate.00 a numeric vector, state rate for 2000-2004

Count.00 a numeric vector, state number of cases for 2000-2004

Lower.00 a numeric vector, lower end point for 95% confidence interval

Upper.00 a numeric vector, upper end point for 95% confidence interval

Pop.00 a numeric vector, state population fro 2000-2004

StdErr.00 a numeric vector, state standard error

Rate.95 a numeric vector, state rate for 1995-1999

Count.95 a numeric vector, state number of cases for 1995-1999

Lower.95 a numeric vector, lower end point for 95% confidence interval

Upper.95 a numeric vector, upper end point for 95% confidence interval

Pop.95 a numeric vector, state population for 2000-2004

StdErr.95 a numeric vector, state standard error

Details

See documentation for wflung00and95 for more details. The row name is the associated state abbreviation - 2 characters. This dataset is used in the *micromapSEER* examples using a border group of "USStatesDF"..

Author(s)

Linda W. Pickle and Jim Pearson of StatNet Consulting, LLC, Gaithersburg, MD

Source

Surveillance Research Program, National Cancer Institute SEER*Stat software (https://www.seer.cancer.gov/seerstat), November 2007 data submission, released April 2008. Data originally provided to NCI by the National Center for Health Statistics.

References

none

---

wflung00cnty *Lung cancer mortality data for white females, by county, 2000-4*

Description

Counts and rates of lung cancer mortality data among white women, aggregated for 2000-4 by county.

Usage

```
data(wflung00cnty)
```

Format

A data frame with 2577 observations on the following 6 variables.

fips a numeric vector of 5 digit fips codes identifying the state and the county

rate a numeric vector of age-adjusted rates by county during 2000-4 for white females

count a numeric vector of the number of white female lung cancer deaths during 2000-4 by county

pop a numeric vector of the white female population in the county during 2000

stcode a numberic vector of the 2 digit state fips code

stabr a character vector of the 2 character state postal code

Details

The rates on this file are directly age adjusted to the US 2000 standard population and are expressed as the number of deaths per 100,000 person-years. Counties with from 1 to 9 deaths are suppressed (deleted from the file). This dataset is used by the *micromapSEER* examples using the border group of "USStatesDF".

Author(s)

Linda W. Pickle and Jim Pearson of StatNet Consulting, LLC, Gaithersburg, MD

Source

Surveillance Research Program, National Cancer Institute SEER*Stat software (http://www.seer.cancer.gov/seerstat), November 2007 data submission, released April 2008. Data originally provided to NCI by the National Center for Health Statistics.

References

FIP 6-4 Codes

`wmlung5070` *Lung cancer mortality data for white males, 1950-69 and 1970-94*

Description

Counts and rates of lung cancer mortality data among white men by state, aggregated for 1950-1969 and 1970-1994

Usage

```
data(wmlung5070)
```

Format

A data frame with 51 observations, 1 for each state + DC, on the following 5 variables.

RATEWM_50 a numeric vector, state age-adjusted rates during 1950-69

COUNTWM_50 a numeric vector, the number of lung cancer deaths during 1950-69

RATEWM_70 a numeric vector, state age-adjusted rates during 1970-94

COUNTWM_70 a numeric vector, the number of lung cancer deaths during 1970-94

PERCENT a numeric vector of the percent change in rate from 1950-69 to 1970-94

Details

The rates on this file are directly age adjusted to the US 1970 standard population and are expressed as the number of deaths per 100,000 person-years. The row names are the 2 character postal codes for the states. Note that the data currently available on the NCI web site are from a later data submission and so may differ slightly (in first decimal place) from the rates provided here due to corrections to the dataset after its first publication. The name of each row is the state abbreviation - 2 characters. This dataset is used by the *micromapSEER* examples using the border group of "USStatesDF".

Author(s)

Linda W. Pickle and Jim Pearson of StatNet Consulting, LLC, Gaithersburg, MD

Source

Surveillance Research Program, National Cancer Institute SEER*Stat software (https://www.seer.cancer.gov/seerstat), November 2007 data submission, released April 2008. Data originally provided to NCI by the National Center for Health Statistics.

References

Devesa SS, Grauman DJ, Blot WJ, Pennello GA, Hoover RN, Fraumeni, JF Jr. Atlas of cancer mortality in the United States: 1950-94, NIH Publication 99-4564, Bethesda, MD: National Cancer Institute

---

`wmlung5070US` *U.S. lung cancer mortality data for white males, 1950-1969 and 1970-1994*

Description

Count and age-adjusted rate of lung cancer mortality among white men for the total U.S., aggregated for 1950-69 and 1970-94.

Usage

```
data(wmlung5070US)
```

Format

A data frame with 1 observations on the following 5 variables.

RATEWM_50 a numeric vector, US age adjusted mortality rates for 1950-1969

COUNTWM_50 a numeric vector, US number of cases from 1950-1969

RATEWM_70 a numeric vector, US age adjusted mortality rates for 1970-1994

COUNTWM_70 a numeric vector, US number of cases from 1970-1994

PERCENT a numeric vector, change from 1950-1969 to 1970-1994 US rates.

Details

see wmlung5070 for further details. The row name is always *US* indicating US rates. This dataset is used by the *micromapSEER* examples using the border group of "USStatesDF".

Author(s)

Linda W. Pickle and Jim Pearson of StatNet Consulting, LLC, Gaithersburg, MD

References

None

`X-Axis` *How the X-Axis is created and user controls.*

Description

How the X-Axis is created is important to almost every glyph built by *micromapST*. It provides the detailed scaling information on the data in the glyph columns. The X-Axis needs to provide as much information to the reader of the overall graphic as possible. While the space is limited, *micromapST* has made several features available to help the user manage how the X-Axis presents the data information.

Details

Tools that are available:

No modification Layout the X-Axis numbers in a straight line with no modification. Try to present as many ticks as possible.

Reduce Size If the X-Axis is very crowed, the package may reduce the font size of all of the X-Axis's to get more information visible to the user. This is generally done in a manner similar to each column casting a vote to reduce the font size.

Overlap Protection When number are presented at the edges of the panels, they often stretch past the ends of the panel and into the next panel's space. micromap ST attempt to catch these problems and may elect to not present a number in that position or vote for a reduction in the font size.

High/Low When there is overlap at the edges of panels and when not enough space to present a resaonable number of X-Axis label (points), micromapST may present all of the X-Axis labels in a one up and one down pattern. In this way the numbers on the edge of panels can overlap labels on another panel by present one value in a low position next to the panel line and the next panel present it's X-Axis value in an high position avoiding over printing its neighboring value. When this is enable, all of the numbers in the X-Axis will alternate being presented in a high/low position sequence. This also allows more numbers to be presented since all neighboring label have more room on the axis. Generally the font is also reduced so that two line of labels can be presented in about 1.5 lines of space.

Scaling Depending on the magitude of the X-Axis numbers, scaling the number can save a lot of space and present a more readable X-Axis. micromapST support two types of scaling. a) reduce the magnitude of the number by 100, 1000, 10,000, etc and add a second line indicating what the divisor was (tens, hundreds, thousands, etc.) b) again reduce the magnitude of the numbers by factors of 10s and add to the end of each number a scaling letter (H=hundread, T=thousands, M=millions, B=billions, etc.) This technique is used along with the High/Low and Overlap Protection options when possible.

These options are all targeted at improving the readability of the X-Axis. To enhance the label placement, the *labeling* package is used to determine the best placement for the X-Axis labels. micromapST makes use of the "wilkinson" function and the "extended" function based on "wilkinson".

Value

None

Author(s)

Jim Pearson, StatNet Consulting, LLC, Gaithersburg, MD

See Also

micromapST

Index

∗ **data structure**
micromapGDefaults, 111
panelDesc, 150
∗ **datasets**
AfricaBG, 9
AfricaPopData, 11
bordGrp, 12
ChinaBG, 32
cnPopData, 35
detailsVariables, 36
Educ8thData, 37
KansasBG, 62
KansPopInc, 65
LOWESSData, 66
MarylandBG, 67
mdPopData, 68
messages-BG, 69
messages-MM, 88
NewYorkBG, 147
nyPopData, 149
Seer18Area, 160
SeoulPopData, 161
SeoulSKoreaBG, 161
statePop2010, 163
SynTable, 164
TSdata, 165
UKIrelandBG, 166
UKIrelandPopData, 172
USSeerBG, 173
USStatesBG, 178
UtahBG, 180
UtahPopData, 182
wflung00and95, 185
wflung00and95US, 186
wflung00cnty, 187
wmlung5070, 188
wmlung5070US, 189
∗ **functions**
ClnStrName_MST, 34
glyph-arrow, 38
glyph-bar, 39
glyph-boxplot, 40
glyph-ctrbar, 41
glyph-dot, 43
glyph-dotconf, 44
glyph-dotse, 46
glyph-dotsignif, 47
glyph-id, 49
glyph-mapxxxx, 50
glyph-normbar, 52
glyph-rank, 54
glyph-scatdot, 55
glyph-segbar, 58
glyph-ts, 60
micromapGSetDefaults, 121
micromapGSetPanelDef, 122
micromapSEER, 123
Plot_Vis, 158
X-Axis, 190
∗ **messages**
messages-BG, 69
messages-MM, 88
∗ **panelDesc**
panelDesc, 150

AfricaBG, 9
AfricaPopData, 11
areaNamesAbbrsIDs *(bordGrp)*, 12
areaParms *(bordGrp)*, 12
areaVisBorders *(bordGrp)*, 12
arrow *(glyph-arrow)*, 38

bar *(glyph-bar)*, 39
bordGrp, 12
boxplot *(glyph-boxplot)*, 40
BuildBorderGroup, 18, *35*

ChinaBG, 32
ClnStrName_MST, 34

cnPopData, 35
ctrbar (glyph-ctrbar), 41

detailsVariables, 36
dot (glyph-dot), 43
dotconf (glyph-dotconf), 44
dotse (glyph-dotse), 46
dotsignif (glyph-dotsignif), 47

Educ8thData, 37

glyph-arrow, 38
glyph-bar, 39
glyph-boxplot, 40
glyph-ctrbar, 41
glyph-dot, 43
glyph-dotconf, 44
glyph-dotse, 46
glyph-dotsignif, 47
glyph-id, 49
glyph-mapxxxx, 50
glyph-normbar, 52
glyph-rank, 54
glyph-scatdot, 55
glyph-segbar, 58
glyph-ts, 60
glyph-tsconf (glyph-ts), 60

id (glyph-id), 49

KansasBG, 62
KansPopInc, 65

L2VisBorders (bordGrp), 12
L3VisBorders (bordGrp), 12
LOWESSData, 66

map (glyph-mapxxxx), 50
mapcom (glyph-mapxxxx), 50
mapmedian (glyph-mapxxxx), 50
maptail (glyph-mapxxxx), 50
MarylandBG, 67
mdPopData, 68
messages-BG, 69
messages-MM, 88
micromapGDefaults, 111, *121*
micromapGSetDefaults, *120*, 121
micromapGSetPanelDef, *120*, 122, *122*
micromapSEER, *28*, 123, *131*
micromapST, *28*, *35*, *39–41*, *43*, *44*, *46–49*, *52*, *54*, *58*, *60*, *62*, *123*, 124, *131*, *158*, *160*, *164*, *191*
micromapST-package, 4

NewYorkBG, 147
NORMBAR (glyph-normbar), 52
nyPopData, 149

panelDesc, 150
Plot_Vis, 158

RANK (glyph-rank), 54
RegVisBorders (bordGrp), 12

SCATDOT (glyph-scatdot), 55
Seer18Area, 160
SEGBAR (glyph-segbar), 58
SeoulPopData, 161
SeoulSKoreaBG, 161
statePop2010, 163
SynTable, 164

ts (glyph-ts), 60
TSdata, 165

UKIrelandBG, 166
UKIrelandPopData, 172
UKIrelandPopData2 (UKIrelandPopData), 172
USSeerBG, 173
USStatesBG, 178
UtahBG, 180
UtahPopData, 182

wflung00and95, 185
wflung00and95US, 186
wflung00cnty, 187
wmlung5070, 188
wmlung5070US, 189

X-Axis, 190