Package 'MazamaSpatialUtils'

August 29, 2016

Type Package

Version 0.4.3

Title Mazama Science Spatial Data Download and Utility Functions

Author Jonathan Callahan [aut, cre],
Will Leahy [aut],
Henry Nguyen [aut],
Robin Winstanley [aut],
Ruby Fore [aut]

Maintainer Jonathan Callahan <jonathan.s.callahan@gmail.com>

Depends R (>= 3.1.0)

Imports dplyr, rgdal, rgeos, rvest, sp, stringr, utils

Suggests knitr, maps, testthat

Description A suite of conversion scripts to create internally standardized spatial polygons dataframes. Utility scripts use these datasets to return values such as country, state, timezone, watershed, etc. associated with a set of longitude/latitude pairs. (They also make cool maps.)

License GPL-2

VignetteBuilder knitr

Repository CRAN

LazyData true

NeedsCompilation no

Date/Publication 2016-02-17 09:00:33

R topics documented:

`codeToCode` *Convert Between ISO2 and ISO3 Country Codes*

Description

Converts a vector of ISO 3166-1 alpha-2 codes to the corresponding ISO 3166-1 alpha-3 codes or vice versa.

Usage

```
codeToCode(countryCodes)
```

Arguments

`countryCodes`	vector of country codes to be converted

Value

A vector of ISO country codes

`codeToCountry` *Convert Country Codes to Country Names*

Description

Converts a vector of ISO 3166-1 alpha-2 codes to the corresponding English names.

Usage

```
codeToCountry(countryCodes)
```

Arguments

`countryCodes`	vector of country codes to be converted

Value

A vector of English country names or NA.

`codeToState` *Convert State Codes to State Names*

Description

Converts a vector of ISO 3166-2 alpha-2 state codes to the corresponding English names.

Usage

```
codeToState(stateCodes, countryCodes = NULL, dataset = "NaturalEarthAdm1")
```

Arguments

`stateCodes`	vector of state codes to be converted
`countryCodes`	ISO-3166-1 alpha-2 country codes the state might be found in
`dataset`	name of dataset containing state-level identifiers

Details

For this function to work, you must first run `initializeSpatialData()` to download, convert and install the necessary spatial data.

Value

A vector of English state names or NA.

See Also

convertNaturalEarthAdm1

`convertGADM` *Convert and Regularize Data from the GADM Database*

Description

A SpatialPolygonsDataFrame file is downloaded from the GADM database with additional columns of data added. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`. Dataset and file names are generated like this:

```
paste0('gadm_', countryCode, '_', admLevel)
```

Level 0 will return the national outline. Level 1 will give state/province boundaries. etc.

Usage

```
convertGADM(countryCode = NULL, admLevel = 0, nameOnly = FALSE)
```

Arguments

`countryCode`	ISO-3166-1 alpha-2 country code
`admLevel`	administrative level to be downloaded
`nameOnly`	logical specifying whether to only return the name without creating the file

Value

Name of the dataset being created.

Note

Not all countries have the same number of levels. Many just have two levels while France has five.

References

http://www.gadm.org/country.

Examples

```
## Not run:
convertGADM('DE', 1)

## End(Not run)
```

convertNaturalEarthAdm1 *Convert Level 1 (State) Borders Shapefile*

Description

Returns a SpatialPolygonsDataFrame for a 1st level administrative divisions

Usage

```
convertNaturalEarthAdm1(nameOnly = FALSE)
```

Arguments

nameOnly	logical specifying whether to only return the name without creating the file

Details

A state border shapefile is downloaded and converted to a SpatialPolygonsDataFrame with additional columns of data. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`.

Within the **MazamaSpatialUtils** package the phrase 'state' refers to administrative divisions beneath the level of the country or nation. This makes sense in the United 'States'. In other countries this level is known as 'province', 'territory' or some other term.

Value

Name of the dataset being created.

References

http//www.naturalearthdata.com/download

http://www.statoids.com/ihasc.html

See Also

setSpatialDataDir

getState, getStateCode

convertTMWorldBorders *Convert World Borders Shapefile*

Description

Returns a SpatialPolygonsDataFrame for world divisions

Usage

```
convertTMWorldBorders(nameOnly = FALSE)
```

Arguments

nameOnly	logical specifying whether to only return the name without creating the file

Details

A world borders shapefile is downloaded and converted to a SpatialPolygonsDataFrame with additional columns of data. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`.

Value

Name of the dataset being created.

References

http://thematicmapping.org/downloads/

See Also

setSpatialDataDir

getCountry, getCountryCode

convertTMWorldBordersSimple *Convert (Simple) World Borders Shapefile*

Description

Returns a SpatialPolygonsDataFrame for a simple world divisions

Usage

```
convertTMWorldBordersSimple(nameOnly = FALSE)
```

Arguments

nameOnly	logical specifying whether to only return the name without creating the file

Details

A world borders shapefile is downloaded and converted to a SpatialPolygonsDataFrame with additional columns of data. The resulting file will be created in the package SpatialDataDir which is set with setSpatialDataDir().

This shapefile is a greatly simplified version of the TMWorldBorders shapefile and is especially suited for spatial searches. This is the default dataset used in getCountry() and getCountryCode(). Users may wish to use a higher resolution dataset when plotting.

Value

Name of the dataset being created.

References

http://thematicmapping.org/downloads/

See Also

setSpatialDataDir

getCountry, getCountryCode

---

convertUSCensusCounties

Convert US County Borders Shapefile

Description

Returns a SpatialPolygonsDataFrame for a US county divisions

Usage

```
convertUSCensusCounties(nameOnly = FALSE)
```

Arguments

nameOnly	logical specifying whether to only return the name without creating the file

Details

A US county borders shapefile is downloaded and converted to a SpatialPolygonsDataFrame with additional columns of data. The resulting file will be created in the spatial data directory which is set with setSpatialDataDir().

Value

Name of the dataset being created.

References

http://www2.census.gov/geo/tiger/GENZ2013

See Also

setSpatialDataDir

getUSCounty

---

convertWBDHUC *Convert USGS Hydrologic Unit Shapefiles*

Description

Previously downloaded shapefiles from the USGS Watershed Boundary Dataset are converted to a SpatialPolygonsDataFrame with additional columns of data. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`.

Usage

```
convertWBDHUC(dsnPath = NULL, level = 8, extension = "",
  nameOnly = FALSE)
```

Arguments

dsnPath	directory where the WBD HUC datasets are found
level	character or integer which must be 2, 4, 6, 8, 10, 12 or 14
extension	character extsion associated with mapshaper simplified files
nameOnly	logical specifying whether to only return the name without creating the file

Details

The full WBD dataset can be downloaded from the USGS with the following command:

```
curl ftp://rockyftp.cr.usgs.gov/vdelivery/Datasets/Staged/WBD/Shape/WBD_National.zip -O
```

Typically, the raw data will be simplified using the command line version of mapshpaper. (Installation instructions are found at this URL.)

With mapshaper, you can reduce the number of vertices in the polygons, greatlyl improving the efficiency of spatial searches. Experimentation at the mapshaper website show that a reduction to 1-2 of the original shapefile size still retains the recognizable shape of polygons, removing only the higher order "crenellations" in the polygons.

An example use of mapshaper would be:

```
mapshaper WBDHU2.shp --simplify 1
```

A full suite of `.shp, .shx, .dbf, .prj` files will be created for the new name `WBDHU2_02`.

Value

Name of the dataset being created.

References

http://nhd.usgs.gov/wbd.html

See Also

setSpatialDataDir

`convertWikipediaTimezoneTable`
Convert Wikipedia Timezone Table to Dataframe

Description

Returns a dataframe version of the Wikipedia timezone table with the following columns:

- timezone – Olson timezone
- UTC_offset – hours between local timezone and UTC
- UTC_DST_offset – hours between local timezone daylight savings and UTC
- countryCode – ISO 3166-2 country code
- longitude – longitude of the Olson timezone city
- latitude – latitude of the Olson timezone city

Usage

```
convertWikipediaTimezoneTable()
```

Details

Older named timezones from the table which are linked to more modern equivalents are not included in the returned dataframe.

Value

Dataframe with 399 rows and 6 columns.

References

http://en.wikipedia.org/wiki/List_of_tz_database_time_zones

convertWorldEEZ *Convert World Exclusive Economic Zones Boundaries Shapefile*

Description

A world EEZ shapefile is downloaded and converted to a SpatialPolygonsDataFrame with additional columns of data. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`.

Usage

```
convertWorldEEZ(nameOnly = FALSE)
```

Arguments

`nameOnly`	logical specifying whether to only return the name without creating the file

Value

Name of the dataset being created.

References

http://www.marineregions.org/downloads.php

See Also

setSpatialDataDir

getCountry, getCountryCode

convertWorldTimezones *Create Timezone Dataset*

Description

A world timezone shapefile is downloaded from http://efele.net/maps/tz/world/ and converted to a SpatialPolygonsDataFrame with additional columns of data. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`.

Usage

```
convertWorldTimezones(nameOnly = FALSE)
```

Arguments

`nameOnly`	logical specifying whether to only return the name without creating the file

Value

Name of the dataset being created.

Note

The following list of timezones have polygons but the associated rows in the dataframe have no data. These timezones also have no `countryCode` assigned. We hope to rectify this in a future release.

```
> WorldTimezones@data$timezone[is.na(WorldTimezones$countryCode)]
[1] "Europe/Zagreb"         "Europe/Vatican"        "America/Coral_Harbour"
[4] "Arctic/Longyearbyen"   "uninhabited"           "America/Kralendijk"
[7] "Europe/Jersey"         "Europe/Bratislava"     "America/St_Barthelemy"
[10] "Europe/Ljubljana"      "Europe/Mariehamn"      "Europe/Podgorica"
[13] "Europe/Isle_of_Man"    "Europe/Guernsey"       "Europe/San_Marino"
[16] "Europe/Skopje"         "Europe/Sarajevo"       "America/Lower_Princes"
[19] "America/Marigot"       "Africa/Juba"
```

See Also

setSpatialDataDir

convertWikipediaTimezoneTable

`countryToCode` *Convert Country Names to Country Codes*

Description

Converts a vector of English country names to the corresponding ISO 3166-1 alpha-2 codes.

Usage

```
countryToCode(countryNames)
```

Arguments

`countryNames`	vector of country names to be converted

Value

A vector of ISO 3166-1 alpha-2 codes or NA.

getCountry *Return Country Names at Specified Locations*

Description

Uses spatial comparison to determine which country polygons the locations fall into and returns the country name for those polygons.

If `allData=TRUE`, additional data is returned.

Usage

```
getCountry(lon, lat, dataset = "SimpleCountries", countryCodes = NULL,
  allData = FALSE, useBuffering = FALSE)
```

Arguments

`lon`	vector of longitudes in decimal degrees
`lat`	vector of latitudes in decimal degrees
`dataset`	name of spatial dataset to use
`countryCodes`	vector of countryCodes
`allData`	logical specifying whether a full dataframe should be returned
`useBuffering`	logical flag specyfing the use of location buffering to find the nearest polygon if not target polygon is found

Value

Vector of country names in English.

References

http://www.naturalearthdata.com/downloads/10m-cultural-vectors/

See Also

SimpleCountries

getSpatialData

Examples

```
lon <- seq(0,50)
lat <- seq(0,50)
getCountry(lon,lat)
```

getCountryCode *Return Country ISO Codes at Specified Locations*

Description

Uses spatial comparison to determine which country polygons the locations fall into and returns the country code strings for those polygons.

If `allData=TRUE`, additional data is returned.

Usage

```
getCountryCode(lon, lat, dataset = "SimpleCountries", countryCodes = NULL,
  allData = FALSE, useBuffering = FALSE)
```

Arguments

`lon`	vector of longitudes in decimal degrees
`lat`	vector of latitudes in decimal degrees
`dataset`	name of spatial dataset to use
`countryCodes`	vector of countryCodes
`allData`	logical specifying whether a full dataframe should be returned
`useBuffering`	logical flag specyfing the use of location buffering to find the nearest polygon if not target polygon is found

Value

Vector of ISO-3166-1 alpha-2 country codes.

References

http://www.naturalearthdata.com/downloads/10m-cultural-vectors/

See Also

SimpleCountries

getSpatialData

Examples

```
lon <- seq(0,50)
lat <- seq(0,50)
getCountryCode(lon,lat)
```

getCountryName *Return Country Names at Specified Locations*

Description

Uses spatial comparison to determine which country polygons the locations fall into and returns the country name for those polygons.

If `allData=TRUE`, additional data is returned.

Usage

```
getCountryName(lon, lat, dataset = "SimpleCountries", countryCodes = NULL,
  allData = FALSE, useBuffering = FALSE)
```

Arguments

`lon`	vector of longitudes in decimal degrees
`lat`	vector of latitudes in decimal degrees
`dataset`	name of spatial dataset to use
`countryCodes`	vector of countryCodes
`allData`	logical specifying whether a full dataframe should be returned
`useBuffering`	logical flag specyfing the use of location buffering to find the nearest polygon if not target polygon is found

Value

Vector of country names in English.

References

http://www.naturalearthdata.com/downloads/10m-cultural-vectors/

See Also

SimpleCountries

getSpatialData

Examples

```
lon <- seq(0,50)
lat <- seq(0,50)
getCountryName(lon,lat)
```

getHUC *Return HUCs at Specified Locations*

Description

Uses spatial comparison to determine which HUC polygons the locations fall into and returns the HUC identifier strings for those polygons.

If `allData=TRUE`, additional data is returned.

Usage

```
getHUC(lon, lat, SPDF, HUCs = NULL, allData = FALSE)
```

Arguments

`lon`	vector of longitudes in decimal degrees
`lat`	vector of latitudes in decimal degrees
`SPDF`	spatial polygons dataset of HUCs
`HUCs`	vector of Hydrologic Unit Codes
`allData`	logical specifying whether to return a full dataframe

Value

Vector of HUC identifiers.

See Also

getSpatialData

getHUCName *Return HUC Names at Specified Locations*

Description

Uses spatial comparison to determine which HUC polygons the locations fall into and returns the HUC names for those polygons.

If `allData=TRUE`, additional data is returned.

Usage

```
getHUCName(lon, lat, dataset = "WBDHU10-ms", HUCs = NULL, allData = FALSE)
```

Arguments

lon	vector of longitudes in decimal degrees
lat	vector of latitudes in decimal degrees
dataset	name of spatial dataset to use
HUCs	vector of Hydrologic Unit Codes
allData	logical specifying whether to return a full dataframe

Value

Vector of HUC names.

See Also

getSpatialData

---

getSpatialData *Return Spatial Data Associated with a Set of Locations*

Description

All locations are first converted to SpatialPoints objects. The **sp::over()** function is then used to determine which polygon from SPDF each location falls in. The dataframe row associated with each polygon is then associated with each ocation.

Usage

```
getSpatialData(lon, lat, SPDF, useBuffering = FALSE, verbose = FALSE)
```

Arguments

lon	vector of longitudes in decimal degrees
lat	vector of latitudes in decimal degrees
SPDF	object of class SpatialPolygonsDataFrame
useBuffering	logical flag specyfing the use of location buffering to find the nearest polygon if not target polygon is found
verbose	logical flag controlling detailed progress statements

Details

For coastal locations it can often happen that the precise coordinates of the location lie outside the boundaries of low resolution SpatialPolygonsDataFrame. To account for this, locations that remain unassociated after the first pass are then buffered to create small circles. All polygons are then checked to see if there is any intersection with the now buffered locations. A buffering loop increases buffer size though the following radii until an intersecting polygon is found: 1km, 2km, 5km, 10km, 20km, 50km, 100km, 200km.

If a buffered location is more than 200km away from any polygon, a value of NA (or data frame row with all NAs) is returned for that location.

Missing or invalid values in the incoming `lon` or `lat` vectors result in NAs at those positions in the returned vector or data frame.

Value

Vector or dataframe of data.

`getSpatialDataDir` *Get Package Data Directory*

Description

Returns the package data directory where spatial data is located.

Usage

```
getSpatialDataDir()
```

Value

Absolute path string.

See Also

dataDir

setSpatialDataDir

getState *Return State Names at Specified Locations*

Description

Uses spatial comparison to determine which 'state' polygons the locations fall into and returns the ISO 3166-2 2-character state code strings for those polygons.

Specification of `countryCodes` limits spatial searching to the specified countries and greatly improves performance.

If `allData=TRUE`, additional data is returned.

Usage

```
getState(lon, lat, dataset = "NaturalEarthAdm1", countryCodes = NULL,
  allData = FALSE, useBuffering = FALSE)
```

Arguments

`lon`	vector of longitudes in decimal degrees
`lat`	vector of latitudes in decimal degrees
`dataset`	name of spatial dataset to use
`countryCodes`	vector of country codes
`allData`	logical specifying whether to return a full dataframe
`useBuffering`	logical flag specyfing the use of location buffering to find the nearest polygon if not target polygon is found

Value

Vector of state names in English.

See Also

getSpatialData

Examples

```
## Not run:
lon <- seq(-140,-90)
lat <- seq(20,70)
getState(lon,lat)

## End(Not run)
```

getStateCode *Return State ISO Codes at Specified Locations*

Description

Uses spatial comparison to determine which 'state' polygons the locations fall into and returns the ISO 3166 2-character state code strings for those polygons.

Specification of `countryCodes` limits spatial searching to the specified countries and greatly improves performance.

If `allData=TRUE`, additional data is returned.

Usage

```
getStateCode(lon, lat, dataset = "NaturalEarthAdm1", countryCodes = NULL,
  allData = FALSE, useBuffering = FALSE)
```

Arguments

`lon`	vector of longitudes in decimal degrees
`lat`	vector of latitudes in decimal degrees
`dataset`	name of spatial dataset to use
`countryCodes`	vector of country codes
`allData`	logical specifying whether to return a full dataframe
`useBuffering`	logical flag specyfing the use of location buffering to find the nearest polygon if not target polygon is found

Value

Vector of ISO-3166-2 alpha-2 state codes.

See Also

getSpatialData

Examples

```
## Not run:
lon <- seq(-140,-90)
lat <- seq(20,70)
getStateCode(lon,lat)

## End(Not run)
```

getStateName *Return State Names at Specified Locations*

Description

Uses spatial comparison to determine which 'state' polygons the locations fall into and returns the ISO 3166-2 2-character state code strings for those polygons.

Specification of `countryCodes` limits spatial searching to the specified countries and greatly improves performance.

If `allData=TRUE`, additional data is returned.

Usage

```
getStateName(lon, lat, dataset = "NaturalEarthAdm1", countryCodes = NULL,
  allData = FALSE, useBuffering = FALSE)
```

Arguments

`lon`	vector of longitudes in decimal degrees
`lat`	vector of latitudes in decimal degrees
`dataset`	name of spatial dataset to use
`countryCodes`	vector of country codes
`allData`	logical specifying whether to return a full dataframe
`useBuffering`	logical flag specyfing the use of location buffering to find the nearest polygon if not target polygon is found

Value

Vector of state names in English

See Also

getSpatialData

Examples

```
## Not run:
lon <- seq(-140,-90)
lat <- seq(20,70)
getStateName(lon,lat)

## End(Not run)
```

getTimezone *Return Olson Timezones at Specified Locations*

Description

Uses spatial comparison to determine which timezone polygons the locations fall into and returns the Olson timezone strings for those polygons.

Specification of `countryCodes` limits spatial searching to the specified countries and greatly improves performance.

If `allData=TRUE`, additional data is returned.

Usage

```
getTimezone(lon, lat, dataset = "SimpleTimezones", countryCodes = NULL,
  allData = FALSE, useBuffering = FALSE)
```

Arguments

`lon`	vector of longitudes in decimal degrees
`lat`	vector of latitudes in decimal degrees
`dataset`	name of spatial dataset to use
`countryCodes`	vector of countryCodes
`allData`	logical specifying whether to return a full dataframe
`useBuffering`	logical flag specyfing the use of location buffering to find the nearest polygon if not target polygon is found

Value

Vector of Olson timezones.

References

http://efele.net/maps/tz/

See Also

SimpleTimezones

getSpatialData

Examples

```
lon <- seq(-120,-60,5)
lat <- seq(20,80,5)
getTimezone(lon,lat)
```

getUSCounty *Return US County Name at Specified Locations*

Description

Uses spatial comparison to determine which county polygons the locations fall into and returns the county name strings for those polygons.

Specification of `stateCodes` limits spatial searching to the specified states and greatly improves performance.

If `allData=TRUE`, additional data is returned.

Usage

```
getUSCounty(lon, lat, dataset = "USCensusCounties", stateCodes = NULL,
  allData = FALSE)
```

Arguments

`lon`	vector of longitudes in decimal degrees
`lat`	vector of latitudes in decimal degrees
`dataset`	name of spatial dataset to use
`stateCodes`	vector of stateCodes used to limit the search
`allData`	logical specifying whether a full dataframe should be returned

Value

Vector of county names in English.

References

http://www.naturalearthdata.com/downloads/10m-cultural-vectors/

See Also

getSpatialData

Examples

```
## Not run:
lon <- seq(-140,-90)
lat <- seq(20,70)
getUSCounty(lon,lat)

## End(Not run)
```

getVariable *Return SpatialDataframe Variable at Specified Locations*

Description

Uses spatial comparison to determine which polygons the locations fall into and returns the variable associated with those polygons.

If `allData=TRUE`, the entire dataframe is returned.

Usage

```
getVariable(lon, lat, dataset = NULL, variable = NULL,
  countryCodes = NULL, allData = FALSE)
```

Arguments

`lon`	vector of longitudes in decimal degrees
`lat`	vector of latitudes in decimal degrees
`dataset`	name of spatial dataset to use
`variable`	name of dataframe column to be returned
`countryCodes`	vector of countryCodes
`allData`	logical specifying whether a full dataframe should be returned

Value

Vector or dataframe.

See Also

getSpatialData

Examples

```
## Not run:
loadSpatialData('NaturalEarthAdm1')
lon <- seq(0,50)
lat <- seq(0,50)
getVariable(lon,lat,'NaturalEarthAdm1','gns_lang')

## End(Not run)
```

initializeSpatialData *Install Core Datasets*

Description

Four core datasets can be installed to enhance the base the functionality in **MazamaSpatialUtils**. Running `initializeSpatialData()` will install these datasets in the the directory specified by `setSpatialDataDir()`.

The core datastes are:

- `TMWorldBorders` – high resolution country polygons (higher resolution than `SimpleCountries`)
- `NaturalEarthAdm1` – state/province polygons throughout the world
- `USCensusCounties` – county polygons in the United States
- `WorldTimezones` – high resolution timezone polygons (higher resolution than `SimpleTimezones`)

Usage

```
initializeSpatialData()
```

Value

Nothing.

See Also

setSpatialDataDir

installSpatialData *Install a Named Spatial Dataset*

Description

Install a named spatial dataset.

Usage

```
installSpatialData(dataset = NULL, verbose = FALSE, ...)
```

Arguments

`dataset`	name of dataset
`verbose`	logical flag controlling detailed progress statements
`...`	additional arguments needed by some `convert~` functions

Value

Character name of the installed dataset.

loadSpatialData *Load a Named Spatial Dataset into the Global Environment*

Description

Load a named spatial dataset. The package `SpatialDataDir` will be searched for the specified dataset. Set this location with `setSpatialDataDir()`.

Usage

```
loadSpatialData(dataset = NULL)
```

Arguments

`dataset`	name of dataset

See Also

getSpatialDataDir

setSpatialDataDir

MazamaSpatialUtils *Mazama Science spatial data and utility functions.*

Description

This package contains code to convert various spatial datasets into .RData files with uniformly named identifiers including:

- countryCode – ISO 3166-1 alpha-2
- countryName – Country name
- stateCode – ISO 3166-2 alpha-2
- timezone – Olson timezone
- longitude – degrees East
- latitude – degrees North
- area – m^2

The parameters listed above will be found in the @data slot of each spatial dataset whose source data has an equivalent field. The only field guaranteed to exist in every dataset is `countryCode`.

The following additional standards are applied during the data conversion process:

- all spatial data are converted to a purely geographic projection (`CRS("+proj=longlat +ellps=GRS80 +datum=NAD83 +`
- no duplicated rows in the dataframe (conversion to **multi-**polygons)

- lowerCamelCase, human readable names replace original parameter names
- redundant, software-internal or otherwise unuseful data columns may be dropped
- parameters may be added to the @data dataframe
- latitude and longitude of polygon centroids may be added

Utility functions allow users to determine the country, state, county and timezones associated with a set of locations, e.g. environmental monitoring sites.

The uniformity of identifiers in the spatial datasets also makes it easy to generate maps with data from any dataset that uses standard ISO codes for countries or states.

History

version 0.4.2 – patch

- Added `encoding` argument to converLayer().
- Modified convertUSCensusCounties() to use `encoding='latin1'`.

version 0.4.1 – patch

- Added `useBuffering` to get-Sate,CountryTimezone functions.

version 0.3.2 – patch

- getSpatialData() no longer fails on invliad/missing locations, now returns dataframe rows with all NA.

#' version 0.3.1 – addition of buffered search and WorldEEZ polygons

- Updated included datasets to use `"+proj=longlat +ellps=GRS80 +datum=NAD83 +no_defs"`.
- Addition of buffered search so that locations can find nearby polygons.
- Addition of convertWorldEEZ() function.

version 0.2.4 – patch

- Updated default projection from `"+proj=longlat"` to `"+proj=longlat +ellps=GRS80 +datum=NAD83 +no_defs"` to support libproj >= 4.9.1

version 0.2.3 – patch

- Removed unneeded test that fails with sp version 1.1-0.

version 0.2.2 – minor tweaks to 0.2.1

- User specification of `SpatialDataDir` is now required.
- Minor documentation improvements.

version 0.2.1 – addition of GADM and USGS HUC8

- Converts for GADM administrative boundaries and and USGS watershed datasets.
- Addition of code-name, name-code and code-code conversion utilities.
- Addition of organizePolygons() function.

version 0.1 – initial release

setSpatialDataDir *Set Package Data Directory*

Description

Sets the package data directory where spatial data is located. If the directory does not exist, it will be created.

Usage

```
setSpatialDataDir(dataDir)
```

Arguments

dataDir	directory where spatial datasets are created

Value

Silently returns previous value of data directory.

See Also

SpatialDataDir

getSpatialDataDir

SimpleCountries *World Country Polygons*

Description

SimpleCountries is a simplified world borders dataset suitable for global maps and quick spatial searches. This dataset is distributed with the package and is used by default whenever a dataset with country polygons is required.

Format

A SpatialPolygonsDataFrame with 246 elements and 7 columns of data.

Details

This dataset is equivalent to TMWorldBordersSimple but with fewer columns of data.

See Also

convertTMWorldBordersSimple

SimpleTimezones *World Timezone Polygons*

Description

SimpleTimezones is a simplified world timezones dataset suitable for global maps and quick spatial searches. This dataset is distributed with the package and is used by default whenever a dataset with timezone polygons is required.

Format

A SpatialPolygonsDataFrame with 1106 elements and 6 columns of data.

Details

This dataset is a simplified version of WorldTimezones. It was simplified with http://mapshaper.org.

See Also

convertWorldTimezones

SpatialDataDir *Directory for Spatial Data*

Description

This package maintains an internal directory location which users can set using `setSpatialDataDir()`. All package functions use this directory whenever datasets are created or loaded.

The default setting when the package is loaded is `getwd()`.

Format

Absolute path string.

See Also

getSpatialDataDir

setSpatialDataDir

stateToCode — *Convert State Names to State Codes*

Description

Converts a vector of state names to an ISO 3166-2 two character state codes.

Usage

```
stateToCode(stateNames, countryCodes = NULL, dataset = "NaturalEarthAdm1")
```

Arguments

stateNames	state names to be converted
countryCodes	ISO 3166-2 alpha-2 country codes the state might be found in
dataset	name of dataset containing state-level identifiers

Details

For this function to work, you must first run `initializeSpatialData()` to download, convert and install the necessary spatial data.

Value

A vector of ISO 3166-2 codes or NA.

See Also

convertNaturalEarthAdm1

subsetHUC — *Subset pre-formatted HUC files into smaller groupings.*

Description

A SpatialPolygons Dataframe is broken into smaller pieces based on HUC code or state. The SpatialPolygons Dataframe must have the required fields 'stateCode', 'HUC', and 'allStateCodes' and is intended to come from the convertUSGSHUC function. The difference between stateCode and allStateCodes is that stateCode has just one two-digit ISO code while allStateCodes can have more than one. This allows us to include in the subset HUCs where part of the watershed is in the specified state even though the centroid is in a different state.

Usage

```
subsetHUC(SPDF, parentHUCs = NULL, stateCodes = NULL,
  allStateCodes = NULL)
```

Arguments

SPDF	a spatial polygons dataframe created using the convertUSGSHUC function
parentHUCs	a character vector specifying one or more containing HUCs
stateCodes	a character vector specifying one or more containing states
allStateCodes	similar to stateCode, but will also include HUCs who touch the state but whose centroid is in a different state.

Value

a SpatialPolygons Dataframe subsetted to the appropriate specifications.

Index