Package 'MazamaSpatialUtils'

January 20, 2025

Type Package

Version 0.8.7

Title Spatial Data Download and Utility Functions

Author Jonathan Callahan [aut, cre],
Rachel Carroll [aut],
Eli Grosman [aut],
Roger Andre [aut],
Tom Bergamaschi [aut],
Tina Chen [aut],
Ruby Fore [aut],
Will Leahy [aut],
Helen Miller [aut],
Henry Nguyen [aut],
Robin Winstanley [aut],
Alice Yang [aut]

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

Description A suite of conversion functions to create internally standardized spatial polygons data frames. Utility functions use these data sets to return values such as country, state, time zone, watershed, etc. associated with a set of longitude/latitude pairs. (They also make cool maps.)

License GPL-2

URL https://github.com/MazamaScience/MazamaSpatialUtils

BugReports https://github.com/MazamaScience/MazamaSpatialUtils/issues

Repository CRAN

Depends R (>= 4.0.0), sf

Imports countrycode, dplyr, magrittr, MazamaCoreUtils (>= 0.4.5), rlang, rmapshaper, stringr

Suggests knitr, markdown, readr, rmarkdown, testthat

Encoding UTF-8

VignetteBuilder knitr

LazyData true

RoxygenNote 7.3.1

NeedsCompilation no

Date/Publication 2024-11-03 05:30:26 UTC

Contents

`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 ISO 3166-1 alpha-2 country codes.

Value

A vector of English country names or NA.

Note

This function is deprecated as of **MazamaSpatialUtils 0.8.7**. Please use countryCodeToName instead.

codeToState *Convert state codes to state nnames*

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.
countryCodes	Vector of 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 install and load the "NaturalEarthAdm1" dataset.

Value

A vector of English state names or NA.

See Also

convertNaturalEarthAdm1

CONUS *CONUS state codes*

Description

State codes for the 48 contiguous states +DC that make up the CONtinental US.

Usage

```
CONUS
```

Format

A vector with 49 elements

convertEEZCountries *Convert Exclusive Economic Zones countries shapefile*

Description

Create a simple features dataframe for combined EEZ/country boundaries.

The full resolution file will be named "EEZCountries.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertEEZCountries()
```

Details

A world EEZ/countries shapefile is converted to a simple features dataframe with additional columns of data. To use this function, the file "EEZ_land_union_v3_202003.zip" must be downloaded into the user's spatial directory which is set with `setSpatialDataDir()`. The resulting file will be created in this same spatial data directory.

Value

Name of the datasetName being created.

Note

For polygons with overlapping claims of sovereignty, we arbitrarily assign the polygon to the country identified in the `ISO_SOV1` field.

The source data is from Version 3 – 2020-03-17.

From the source documentation:

Geographic Information Systems have become indispensable tools in managing and displaying marine data and information. However, a unique georeferenced standard of marine place names and areas was not available, hampering several marine geographic applications, for example the linking of these locations to databases to integrate data. The purpose of Marine Regions is therefore to create a standard, relational list of geographic names, coupled with information and maps of the geographic location of these features. This will improve access and clarity of the different geographic, marine names such as seas, sandbanks, ridges and bays and display univocally the boundaries of marine biogeographic or managerial marine areas.

Marine Regions is an integration of the VLIMAR Gazetteer and the VLIZ Maritime Boundaries Geodatabase. The VLIMAR Gazetteer is a database with geographic, mainly marine names such as seas, sandbanks, seamounts, ridges, bays or even standard sampling stations used in marine research. The geographic cover of the VLIMAR gazetteer is global but initially focused on the Belgian Continental Shelf and the Scheldt Estuary and the Southern Bight of the North Sea. Gradually

more regional and global geographic information was added to VLIMAR and combining this information with the Maritime Boundaries database, representing the Exclusive Economic Zone (EEZ) of the world, led to the creation of marineregions.org.

Marine Regions is managed by the Flanders Marine Institute. Funding for the creation of the VLIMAR gazetteer was provided initially through the EU Network of Excellence MarBEF, but also other European initiatives such as Lifewatch provide the necessary funding for the maintenance and management of Marine Regions.

Marine Regions depends on data and knowledge sharing from global, European, regional and national data providers and relevant experts. By setting up Collaboration Agreements, data providers will benefit from belonging to the Marine Regions partnership as they would get increased visibility, gain access to a variety of data analysis services which will benefit from integration of several distributed spatial datasetNames, as well as enjoying the benefit of the creation of stable unique identifiers. An example template of a Collaboration Agreement can be found here. Please contact info@marineregions.org if your organisation is interested to explore this collaboration.

Citation: Flanders Marine Institute (2020). Union of the ESRI Country shapefile and the Exclusive Economic Zones (version 3). Available online at https://www.marineregions.org/. https://doi.org/10.14284/403 #'

References

https://www.marineregions.org/sources.php#unioneezcountry

See Also

setSpatialDataDir

`convertEPARegions` *Convert EPA Region shapefiles*

Description

Returns a simple features data frame for EPA Regions

The full resolution file will be named "EPA_Regions.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertEPARegions()
```

Details

An EPA region boundary shapefile is converted to a simple features data frame with additional columns of data. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`.

The source data is from 2022.

Value

Name of the datasetName being created.

Note

From the source documentation:

This datasetName represents delineated EPA Region boundaries. EPA has ten regional offices across the country, each of which is responsible for several states and in some cases, territories or special environmental programs.

This Shared Enterprise Geodata and Services (SEGS)datasetName was created by U.S. EPA using 2011 TIGER/Line state boundaries from the U.S. Census Bureau. The core mission of SEGS is to provide a single point of ownership for geospatial datasetNames that are national in extent and of general use to all EPA users and to make those datasetNames available through channels that best meet user needs.

References

https://hub.arcgis.com/datasets/geoplatform::epa-regions

`convertGACC` *Convert Geographic Area Coordination Center shapefile*

Description

Create a simple features data frame for Geographic Area Coordination Centers (GACCs). These are regions defined by the National Interagency Fire Center (NIFC).

The full resolution file will be named "GACC.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertGACC()
```

Details

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

The source data is from 2020.

Value

Name of the datasetName being created.

Note

From the source documentation:

Although the primary mission of the GACC is logistical coordination, the Center also has support programs in Predictive Services, Intelligence, and in several Center's Fire Information. Predictive Services consists primarily of professional meteorologists who monitor weather and fuel conditions, conduct briefings, produce fire weather related products, liaison with the National Weather Service, and oversee all aspects of the Remote Automated Weather System (RAWS). The Intelligence Section is primarily responsible for collecting and disseminating wildland fire and prescribed fire activity information, monitoring the status of national firefighting resources, maintaining year-to-date and historical fire occurrence data, and managing the Sit Report and ICS-209 programs.

In some GACCs, the Predictive Services and Intelligence sections work as one unit called the Predictive Services Group. The Predictive Services and Intelligence Sections, whether separated or combined, work collaboratively producing Weekly, Monthly, and Seasonal Fire Weather/Fire Danger Outlooks. Each Coordination Center provides additional support to their respective geographic area's wildland fire community through training, workshops, special projects, and other tasks. Except for dispatch of air tankers and lead planes based outside the dispatch center responsibility the fire is located in, the GACC does not have initial-attack dispatch responsibilities. The United States and Alaska are divided into 11 Geographic Areas for the purpose of incident management and mobilization of resources (people, aircraft, ground equipment). Within each Area, an interagency Geographic Area Coordinating Group (GACG), made up of Fire Directors from each of the Federal and State land management agencies from within the Area, is established.

Working collaboratively, the GACG's mission is to provide leadership and support not only for wildland fire emergencies, but to other emergency incidents (i.e. earthquakes, floods, hurricanes, tornadoes, etc), as necessary. Authority for establishment of the GACG is through departmental policy and interagency agreements. Additional agreements are established with cooperators and other organizations in order to facilitate efficient fire management activities within and adjacent to the Area. A cost-effective sharing of resources among public agencies is a key component of the GACG mission and is expected by the public, Congress, and States. All agencies and geographic areas work together under the auspices and direction of the National Interagency Fire Center (NIFC). The Geographic Area Coordination Centers (GACC) is a result of an interagency agreement established by the respective Geographic Area Coordinating Group. The primary mission of the GACC is to serve Federal and State wildland fire agencies through logistical coordination and mobilization of resources (people, aircraft, ground equipment) throughout the geographical area, and with other geographic areas, as necessary. This is generally done through coordinating the movement of resources between the many Dispatch Centers within the geographic area and, as necessary, with the National Interagency Coordination Center (NICC) when resources are unavailable within the Area or when mobilization support is needed in other geographic areas. As you survey each GACC website, it will become obvious they are technical in design and are primarily for use by local and geographic area wildland fire managers and firefighters. For the general public, the GACC website may not meet your needs. If this is the case, please check out the National Fire News website, provided by the National Interagency Fire Center, and the InciWeb website, provided as a guide to large fire incidents throughout the United States.

The National Wildfire Coordinating Group (NWCG) makes no claims, promises, or guarantees about the accuracy, completeness, or adequacy of the content; and expressly disclaims liability for errors and omissions. No warranty of any kind, implied, expressed or statutory is given with respect to the contents.

References

https://hub.arcgis.com/datasets/nifc::national-gacc-boundaries

See Also

setSpatialDataDir

`convertLayer` *Convert shapefile layer to simple features dataframe*

Description

Raw shapefiles are read in using `st_read`. Spatial data are reprojected onto a standard projection (`https://epsg.io/4269`) before being returned.

If shapefiles have no projection information they are assumed to 'geographic' coordinates .

Usage

```
convertLayer(dsn = NULL, layer)
```

Arguments

`dsn`	dsn argument to `sf::st_read()`.
`layer`	layer argument to `sf::st_read()`.

Value

An object of class `sf`.

`convertNaturalEarthAdm1`

Convert Level 1 (state) borders shapefile

Description

Returns a simple features data frame for top level administrative divisions.

The full resolution file will be named "NaturalEarthAdm0.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Returns a simple features data frame for 1st level administrative divisions

The full resolution file will be named "NaturalEarthAdm1.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertNaturalEarthAdm1()

convertNaturalEarthAdm1()
```

Details

A country border shapefile is downloaded and converted to a simple features data frame with additional columns of data. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`. The resulting file will be created in this same spatial data directory.

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

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.

References

https://www.naturalearthdata.com

https://www.naturalearthdata.com

---

convertOSMTimezones *Convert OSM timezone shapefile*

Description

Create a simple features data frame for world timezones.

The full resolution file will be named "OSMTimezones.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertOSMTimezones()
```

Details

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

There are 2 timezones which 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. These are the missing timezones:

```
> OSMTimezones$timezone[is.na(OSMTimezones$countryCode)]
 [1] "America/Nuuk"  "Asia/Qostanay"
```

Value

Name of the datasetName being created.

Note

From the source documentation:

This project aims to stay up-to-date with all of the currently valid timezones that are defined in the timezone database. This project also will attempt to provide the most accurate possible boundaries of timezones according to community input.

The underlying data is downloaded from OpenStreetMap via the overpass turbo API. Various boundaries are assembled together to produce each zone with various geographic operations. In numerous edge cases arbitrary boundaries get created in various zones which are noted in the timezones.json file.

To maintain consistency with the timezone database, this project will only create a new release after the timezone database creates a new release. If there are no new timezones created or deleted in a timezone database release, then this project will only create a release if there have been changes performed to the boundary definitions of an existing zone within this project.

References

https://github.com/evansiroky/timezone-boundary-builder

---

convertTMWorldBorders *Convert world borders shapefile*

Description

Returns a simple features data frame for world divisions

The full resolution file will be named "TMWorldBorders.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertTMWorldBorders()
```

Details

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

Value

Name of the datasetName being created.

References

https://thematicmapping.org/

`convertUSCensusCBSA` *Convert US Core Based Statistical Areas shapefile*

Description

Returns a simple features data frame for US CBSAs

The full resolution file will be named "USCensusCBSA.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertUSCensusCBSA()
```

Details

A US Core Based Statistical Areas (CBSA) shapefile is downloaded and converted to a simple features data frame 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 datasetName being created.

Note

From the source documentation:

Metropolitan and Micropolitan Statistical Areas are together termed Core Based Statistical Areas (CBSAs) and are defined by the Office of Management and Budget (OMB) and consist of the county or counties or equivalent entities associated with at least one urban core (urbanized area or urban cluster) of at least 10,000 population, plus adjacent counties having a high degree of social and economic integration with the core as measured through commuting ties with the counties containing the core. Categories of CBSAs are: Metropolitan Statistical Areas, based on urbanized areas of 50,000 or more population; and Micropolitan Statistical Areas, based on urban clusters of at least 10,000 population but less than 50,000 population.

The CBSA boundaries are those defined by OMB based on the 2010 Census, published in 2013, and updated in 2020.

References

https://www2.census.gov/geo/tiger/TIGER2021/CBSA/

See Also

setSpatialDataDir

getUSCounty

`convertUSCensusCongress`

Convert US congressional districts shapefile

Description

Returns a simple features data frame for US Congressional Districts for the 116th US House of Representatives.

The full resolution file will be named "USCensus116thCongress.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertUSCensusCongress()
```

Details

A US congressional district shapefile is downloaded and converted to a simple features data frame with additional columns of data. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`.

#' The source data is from 2021.

Value

Name of the datasetName being created.

Note

From the source documentation:

Congressional Districts are the 435 areas from which people are elected to the U.S. House of Representatives. After the apportionment of congressional seats among the states based on census population counts, each state is responsible for establishing congressional districts for the purpose of electing representatives. Each congressional district is to be as equal in population to all other congressional districts in a state as practicable. The 116th Congress is seated from January 2019 to 2021. The cartographic boundary files for the District of Columbia, Puerto Rico, and the Island Areas (American Samoa, Guam, the Commonwealth of the Northern Mariana Islands, and the U.S. Virgin Islands) each contain a single record for the non-voting delegate district in these areas. The boundaries of all other congressional districts are provided to the Census Bureau by the states by May 1, 2018.

You can join this file with table data downloaded from American FactFinder by using the AFFGEOID field in the cartographic boundary file.

References

https://www2.census.gov/geo/tiger/GENZ2021/

See Also

setSpatialDataDir

`convertUSCensusCounties` *Convert US county borders shapefile*

Description

Create a simple features data frame for US counties.

The full resolution file will be named "USCensusCounties.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertUSCensusCounties()
```

Details

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

The source data is from 2021.

Value

Name of the datasetName being created.

Note

From the source documentation:

The primary legal divisions of most states are termed counties. In Louisiana, these divisions are known as parishes. In Alaska, which has no counties, the equivalent entities are the organized boroughs, city and boroughs, municipalities, and for the unorganized area, census areas. The latter are delineated cooperatively for statistical purposes by the State of Alaska and the Census Bureau. In four states (Maryland, Missouri, Nevada, and Virginia), there are one or more incorporated places that are independent of any county organization and thus constitute primary divisions of their states. These incorporated places are known as independent cities and are treated as equivalent entities for purposes of data presentation. The District of Columbia and Guam have no primary divisions, and each area is considered an equivalent entity for purposes of data presentation. The Census Bureau treats the following entities as equivalents of counties for purposes of data presentation: Municipios

in Puerto Rico, Districts and Islands in American Samoa, Municipalities in the Commonwealth of the Northern Mariana Islands, and Islands in the U.S. Virgin Islands. The entire area of the United States, Puerto Rico, and the Island Areas is covered by counties or equivalent entities.

You can join this file with table data downloaded from American FactFinder by using the AFFGEOID field in the cartographic boundary file.

References

https://www2.census.gov/geo/tiger/GENZ2021/

`convertUSCensusStates` *Convert US Census state shapefile*

Description

Create a simple features dataframe for US states

The full resolution file will be named "USCensusStates". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertUSCensusStates()
```

Details

A US state borders shapefile is downloaded and converted to a simple features dataframe with additional columns of data. The resulting file will be created in the spatial data directory which is set with setSpatialDataDir.

The source data is from 2021.

Value

Name of the datasetName being created.

Note

From the source documentation:

Cartographic Boundary Files

The cartographic boundary files are simplified representations of selected geographic areas from the U.S. Census Bureau's Master Address File / Topologically Integrated Geographic Encoding and Referencing (MAF/TIGER) Database (MTDB). These boundary files are specifically designed for small-scale thematic mapping. When possible, generalization is performed with the intent to maintain the hierarchical relationships among geographies and to maintain the alignment of geographies within a file set for a given year. To improve the appearance of shapes, areas are represented with fewer vertices than detailed TIGER/Line equivalents. Some small holes or discontiguous parts of

areas are not included in generalized files. Generalized boundary files are clipped to a simplified version of the U.S. outline. As a result, some offshore areas may be excluded from the generalized files.

Limitations

Geographic areas may not align with the same areas from another year. Some geographies are available as nation-based files while others are available only as state-based files.

States and equivalent entities are the primary governmental divisions of the United States. In addition to the fifty states, the Census Bureau treats the District of Columbia, Puerto Rico, and each of the Island Areas (American Samoa, the Commonwealth of the Northern Mariana Islands, Guam, and the U.S. Virgin Islands) as the statistical equivalents of states for the purpose of data presentation.

"Island Areas" are removed in the MazamaSpatialUtils version.

These files were specifically created to support small-scale thematic mapping. To improve the appearance of shapes at small scales, areas are represented with fewer vertices than detailed TIGER/Line Shapefiles. Cartographic boundary files take up less disk space than their ungeneralized counterparts. Cartographic boundary files take less time to render on screen than TIGER/Line Shapefiles. You can join this file with table data downloaded from American FactFinder by using the AFFGEOID field in the cartographic boundary file. If detailed boundaries are required, please use the TIGER/Line Shapefiles instead of the generalized cartographic boundary files.

References

https://www2.census.gov/geo/tiger/GENZ2021/

`convertWBDHUC` *Convert USGS hydrologic unit geodatabase*

Description

Create a simple features data frame for USGS watershed boundaries

Usage

```
convertWBDHUC(
  gdbDir = "~/Data/WBD/WBD_National_GDB.gdb",
  level = 2,
  simplify = TRUE
)
```

Arguments

`gdbDir`	Directory containing the geodatabase.
`level`	Character or integer which must be 2, 4, 6, 8, 10, 12 or 14.
`simplify`	Logical specifying whether to perform simplification

Details

A USGS Watershed Boundary Dataset geodatabase is converted to a simple features data frame with additional columns of data. To use this function, the WBD geodatabase must be downloaded into a directory which is identified with `gdbDir`. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`.

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

```
curl https://prd-tnm.s3.amazonaws.com/StagedProducts/Hydrography/WBD/National/GDB/WBD_National_GDB.z
```

The source data was downloaded on 2023-03-23.

Value

Name of the datasetName being created.

Note

From the source documentation:

The Watershed Boundary Dataset (WBD) is a comprehensive aggregated collection of hydrologic unit data consistent with the national criteria for delineation and resolution. It defines the areal extent of surface water drainage to a point except in coastal or lake front areas where there could be multiple outlets as stated by the "Federal Standards and Procedures for the National Watershed Boundary Dataset (WBD)" "Standard" (https://pubs.usgs.gov/tm/11/a3/). Watershed boundaries are determined solely upon science-based hydrologic principles, not favoring any administrative boundaries or special projects, nor particular program or agency. This dataset represents the hydrologic unit boundaries to the 12-digit (6th level) for the entire United States. Some areas may also include additional subdivisions representing the 14- and 16-digit hydrologic unit (HU). At a minimum, the HUs are delineated at 1:24,000-scale in the conterminous United States, 1:25,000-scale in Hawaii, Pacific basin and the Caribbean, and 1:63,360-scale in Alaska, meeting the National Map Accuracy Standards (NMAS). Higher resolution boundaries are being developed where partners and data exist and will be incorporated back into the WBD. WBD data are delivered as a dataset of polygons and corresponding lines that define the boundary of the polygon. WBD polygon attributes include hydrologic unit codes (HUC), size (in the form of acres and square kilometers), name, downstream hydrologic unit code, type of watershed, non-contributing areas, and flow modifications. The HUC describes where the unit is in the country and the level of the unit. WBD line attributes contain the highest level of hydrologic unit for each boundary, line source information and flow modifications.

The intent of defining Hydrologic Units (HU) within the Watershed Boundary Dataset is to establish a base-line drainage boundary framework, accounting for all land and surface areas. Hydrologic units are intended to be used as a tool for water-resource management and planning activities particularly for site-specific and localized studies requiring a level of detail provided by large-scale map information. The WBD complements the National Hydrography Dataset (NHD) and supports numerous programmatic missions and activities including: watershed management, rehabilitation and enhancement, aquatic species conservation strategies, flood plain management and flood prevention, water-quality initiatives and programs, dam safety programs, fire assessment and management, resource inventory and assessment, water data analysis and water census.

References

https://www.usgs.gov/national-hydrography/watershed-boundary-dataset

See Also

setSpatialDataDir

`convertWeatherZones` *Convert NWS Public Forecast Zones Shapefile.*

Description

Create a simple features data frame for NWS weather forecast zones.

The full resolution file will be named "WeatherZones.rda". In addition, "_05", _02" and "_01" versions of the file will be created that that are simplified to 5%, 2% and 1%. Simplified versions will greatly improve the speed of both searching and plotting.

Usage

```
convertWeatherZones()
```

Details

A weather forecast zone shapefile is downloaded and converted to a simple features data frame with additional columns of data. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`.

The source data is from 2022-09-13.

Value

Name of the datasetName being created.

Note

Records with a duplicated `zoneID` column (typically representing coastal land and its watery inlets separately) are combined so that `zoneID` becomes a unique identifier.

From the source documentation:

The NWS issues forecasts and some watches and warnings for public zones which usually are the same as counties but in many cases are subsets of counties. Counties are subset into zones to allow for more accurate forecasts because of the differences in weather within a county due to such things as elevation or proximity to large bodies of water.

References

https://www.weather.gov/gis/PublicZones

`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
- status – either 'Canonical', 'Alias' or 'Deprecated'
- notes – typically specifying the target of an 'Alias'

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 388 rows and 10 columns.

References

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

countryConversion *Conversion functions for country names, codes and FIPS codes.*

Description

Converts a vector of country names or codes from one system to another returning NA where no match is found.

Usage

```
countryCodeToName(countryCode = NULL)

countryCodeToFIPS(countryCode = NULL)

countryFIPSToName(countryFIPS = NULL)

countryFIPSToCode(countryFIPS = NULL)

countryNameToCode(countryName = NULL)

countryNameToFIPS(countryName = NULL)
```

Arguments

countryCode	Vector of ISO 3166-1 alpha-2 codes.
countryFIPS	Vector of two-character FIPS codes.
countryName	Vector of English language country names.

Value

A vector of country names or codes.

Examples

```
library(MazamaSpatialUtils)

# FIPS codes are different!

countryNameToCode("Germany")
countryNameToFIPS("Germany")

countryCodeToName("CH")
countryFIPSToName("CH")

countryCodes <- sample(SimpleCountries$countryCode, 30)

data.frame(
  name = countryCodeToName(countryCodes),
```

```
  code = countryCodes,
  FIPS = countryCodeToFIPS(countryCodes)
)
```

`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 English language country names.

Value

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

Note

This function is deprecated as of **MazamaSpatialUtils 0.8.7**. Please use countryNameToCode instead.

`dissolve` *Aggregate shapes in a simple features data frame*

Description

Aggregate shapes in a simple features dataframe. This is a convenience wrapper for `ms_dissolve`.

Usage

```
dissolve(SFDF, field = NULL, sum_fields = NULL, copy_fields = NULL, ...)
```

Arguments

`SFDF`	Object of class simple features data frame.
`field`	Name of the field to dissolve on.
`sum_fields`	Names of fields to sum.
`copy_fields`	Names of fields to copy. The first instance of each field will be copied to the aggregated feature
`...`	arguments passed to `rmapshaper::ms_dissolve()`

Value

A simple features dataframe with aggregated shapes.

Examples

```
regions <- dissolve(SimpleCountries, field = "UN_region", sum_fields = "area")
plot(regions)
dplyr::glimpse(regions)
```

`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.

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

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

Usage

```
getCountry(
  longitude = NULL,
  latitude = NULL,
  datasetName = "SimpleCountriesEEZ",
  countryCodes = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

`longitude`	Vector of longitudes in decimal degrees East.
`latitude`	Vector of latitudes in decimal degrees North.
`datasetName`	Name of spatial dataset to use.
`countryCodes`	Vector of ISO 3166-1 alpha-2 country codes.
`allData`	Logical specifying whether a full dataframe should be returned.
`useBuffering`	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.

Value

Vector of English language country names.

References

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

See Also

SimpleCountries

getSpatialData

Examples

```
library(MazamaSpatialUtils)

longitude <- seq(0, 50)
latitude <- seq(0, 50)

getCountry(longitude, latitude)
```

---

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.

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

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

Usage

```
getCountryCode(
  longitude = NULL,
  latitude = NULL,
  datasetName = "SimpleCountriesEEZ",
  countryCodes = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

`longitude`	Vector of longitudes in decimal degrees East.
`latitude`	Vector of latitudes in decimal degrees North.
`datasetName`	Name of spatial dataset to use.
`countryCodes`	Vector of ISO 3166-1 alpha-2 country codes.

allData	Logical specifying whether a full dataframe should be returned.
useBuffering	Logical flag specifying the use of location buffering to find the nearest polygon if no 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

```
library(MazamaSpatialUtils)

longitude <- seq(0, 50)
latitude <- seq(0, 50)

getCountryCode(longitude, latitude)
```

---

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.

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

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

Usage

```
getCountryName(
  longitude = NULL,
  latitude = NULL,
  datasetName = "SimpleCountriesEEZ",
  countryCodes = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

longitude	Vector of longitudes in decimal degrees East.
latitude	Vector of latitudes in decimal degrees North.
datasetName	Name of spatial dataset to use.
countryCodes	Vector of ISO 3166-1 alpha-2 country codes.
allData	Logical specifying whether a full dataframe should be returned.
useBuffering	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.

Value

Vector of English language country names.

See Also

SimpleCountries

getSpatialData

Examples

```
library(MazamaSpatialUtils)

longitude <- seq(0, 50)
latitude <- seq(0, 50)

getCountryName(longitude, latitude)
```

---

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.

Specification of `HUCs` limits spatial searching to the specified HUCs and greatly improves performance. For instance, if searching for level 10 HUCs in the Upper Columbia basin, it would make sense to first search WBDHU4_01 to learn that the level 4 HUC is `1702`. Then you can greatly improve search times for higher level HUCs by specifying: `HUCs = c("1702")`.

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

Usage

```
getHUC(
  longitude = NULL,
  latitude = NULL,
  dataset = NULL,
  HUCs = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

longitude	Vector of longitudes in decimal degrees East.
latitude	Vector of latitudes in decimal degrees North.
dataset	Name of spatial dataset to use.
HUCs	Vector of Hydrologic Unit Codes used to limit searches.
allData	logical specifying whether a full dataframe should be returned
useBuffering	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.

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.

Specification of HUCs limits spatial searching to the specified HUCs and greatly improves performance. For instance, if searching for level 10 HUCs in the Upper Columbia basin, it would make sense to first search WBDHU4_01 to learn that the level 4 HUC is 1702. Then you can greatly improve search times for higher level HUCs by specifying: HUCs = c("1702").

If allData = TRUE, additional data is returned.

Usage

```
getHUCName(
  longitude = NULL,
  latitude = NULL,
  dataset = NULL,
  HUCs = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

longitude	Vector of longitudes in decimal degrees East.
latitude	Vector of latitudes in decimal degrees North.
dataset	Name of spatial dataset to use.
HUCs	Vector of Hydrologic Unit Codes used to limit searches.
allData	Logical specifying whether a full dataframe should be returned
useBuffering	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.

Value

Vector of HUC names.

See Also

getSpatialData

---

getPolygonID *Get polygonID from SFDF of interest*

Description

Extracts the the vector of unique polygon identifiers from SFDF.

This function is useful when writing code to aggregate data by polygon and calculate per-polygon statistics. Each unique simple features data frame will have a different set of data columns but each is guaranteed to have a column named polygonID that uniquely identifies each polygon.

This allows us to write code that aggregates by polygon without having to know whether the polygons represent, countries, timezones or HUCs, etc.

Usage

```
getPolygonID(SFDF)
```

Arguments

SFDF	Spatial polygons dataset of interest.

Value

Vector of polygon identifiers.

---

getSpatialData *Return spatial data associated with a set of locations*

Description

All locations are first converted to SpatialPoints objects. The st_intersects function is then used to determine which polygon from SFDF each location falls in. The dataframe row associated with each polygon is then associated with each location.

Usage

```
getSpatialData(
  longitude = NULL,
  latitude = NULL,
  SFDF = NULL,
  useBuffering = FALSE,
  verbose = FALSE
)
```

Arguments

longitude	Vector of longitudes in decimal degrees East.
latitude	Vector of latitudes in decimal degrees North.
SFDF	sf dataframe with MULTIPOLYGON geometry.
useBuffering	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.
verbose	Logical flag controlling detailed progress statements.

Details

For coastal locations, locations may lie just outside the boundaries of an individual polygon, especially if it is of low resolution. To account for this any location that remains unassociated after the first pass is checked to seee if it is with a specific distance of any polygon. The set of distances is gradually increased until a polygon is reached or the maximum distances is encountered. Distances include: 1km, 2km, 5km, 10km, 20km, 50km, 100km, 200km. If a location is more than 200km away from any polygon, a data frame record with all NAs is returned for that location.

Missing or invalid values in the incoming longitude or latitude vectors result in records with all NAs at those positions in the returned data frame.

Value

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(
  longitude = NULL,
  latitude = NULL,
  datasetName = "NaturalEarthAdm1",
  countryCodes = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

longitude	Vector of longitudes in decimal degrees East.
latitude	Vector of latitudes in decimal degrees North.
datasetName	Name of spatial dataset to use.
countryCodes	Vector of ISO 3166-1 alpha-2 country codes.
allData	Logical specifying whether a full dataframe should be returned.
useBuffering	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.

Value

Vector of English language state names.

See Also

getSpatialData

Examples

```
## Not run:
library(MazamaSpatialUtils)
setSpatialData("~/Data/Spatial_0.8")

loadSpatialData("NaturalEarthAdm1")

longitude <- seq(-140, -90)
latitude <- seq(20, 70)
getState(longitude, latitude)

## 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(
  longitude = NULL,
  latitude = NULL,
  datasetName = "NaturalEarthAdm1",
  countryCodes = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

longitude	Vector of longitudes in decimal degrees East.
latitude	Vector of latitudes in decimal degrees North.
datasetName	Name of spatial dataset to use.
countryCodes	Vector of ISO 3166-1 alpha-2 country codes.
allData	Logical specifying whether a full dataframe should be returned.
useBuffering	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.

Value

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

See Also

getSpatialData

Examples

```
## Not run:
library(MazamaSpatialUtils)
setSpatialData("~/Data/Spatial_0.8")

loadSpatialData("NaturalEarthAdm1")

longitude <- seq(-140, -90)
latitude <- seq(20, 70)
getStateCode(longitude, latitude)

## 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(
  longitude = NULL,
  latitude = NULL,
  datasetName = "NaturalEarthAdm1",
  countryCodes = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

`longitude`	Vector of longitudes in decimal degrees East.
`latitude`	Vector of latitudes in decimal degrees North.
`datasetName`	Name of spatial dataset to use.
`countryCodes`	Vector of ISO 3166-1 alpha-2 country codes.
`allData`	Logical specifying whether a full dataframe should be returned.
`useBuffering`	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.

Value

Vector of English language state names.

See Also

getSpatialData

Examples

```
## Not run:
library(MazamaSpatialUtils)
setSpatialData("~/Data/Spatial_0.8")

loadSpatialData("NaturalEarthAdm1")

longitude <- seq(-140, -90)
latitude <- seq(20, 70)
getStateName(longitude, latitude)

## 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(
  longitude = NULL,
  latitude = NULL,
  datasetName = "SimpleTimezones",
  countryCodes = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

`longitude`	Vector of longitudes in decimal degrees East.
`latitude`	Vector of latitudes in decimal degrees North.
`datasetName`	Name of spatial dataset to use.
`countryCodes`	Vector of ISO 3166-1 alpha-2 country codes.
`allData`	Logical specifying whether a full dataframe should be returned.
`useBuffering`	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.

Value

Vector of Olson timezones.

References

https://github.com/evansiroky/timezone-boundary-builder

Examples

```
library(MazamaSpatialUtils)

longitude <- seq(-120, -60, 5)
latitude <- seq(20, 80, 5)

getTimezone(longitude, latitude)
```

---

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(
  longitude = NULL,
  latitude = NULL,
  dataset = "USCensusCounties",
  stateCodes = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

`longitude`	Vector of longitudes in decimal degrees East.
`latitude`	Vector of latitudes in decimal degrees North.
`dataset`	Name of spatial dataset to use.
`stateCodes`	Vector of US state codes used to limit the search.
`allData`	Logical specifying whether a full dataframe should be returned.
`useBuffering`	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.

Value

Vector of English language county names.

References

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

See Also

getSpatialData

Examples

```
## Not run:
library(MazamaSpatialUtils)
setSpatialDataDir("~/Data/Spatial_0.8")

loadSpatialData("USCensusCounties")

longitude <- seq(-140, -90)
latitude <- seq(20, 70)
getUSCounty(longitude, latitude)

## End(Not run)
```

---

getVariable *Return SFDF 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(
  longitude = NULL,
  latitude = NULL,
  dataset = NULL,
  variable = NULL,
  countryCodes = NULL,
  allData = FALSE,
  useBuffering = FALSE
)
```

Arguments

longitude	Vector of longitudes in decimal degrees East.
latitude	Vector of latitudes in decimal degrees North.
dataset	Name of spatial dataset to use.
variable	Name of dataframe column to be returned.
countryCodes	Vector of ISO 3166-1 alpha-2 country codes.
allData	Logical specifying whether a full dataframe should be returned.
useBuffering	Logical flag specifying the use of location buffering to find the nearest polygon if no target polygon is found.

Value

Vector or dataframe.

See Also

getSpatialData

---

installedSpatialData *List locally installed spatial datasets*

Description

Searches the directory set with setSpatialDataDir for locally installed spatial data and returns a list of dataset names that can be used with loadSpatialData.

If verbose = TRUE, a brief description is provided for each locally installed dataset.

Usage

```
installedSpatialData(verbose = TRUE)
```

Arguments

verbose	Logical specifying whether or not to print dataset descriptions.

Value

Invisibly returns a vector of dataset names().

installSpatialData *Install spatial datasets*

Description

Install spatial datasets found at `url` into the directory previously set with `setSpatialDataDir()`.

If `pattern = NULL` (default), available datasets will be displalyed..

Usage

```
installSpatialData(
  dataset = NULL,
  urlBase = "http://data.mazamascience.com/MazamaSpatialUtils/Spatial_0.8"
)
```

Arguments

`dataset`	Name of spatial dataset to install.
`urlBase`	Location of spatial data files.

Value

If `pattern = NULL` a vector of dataset names.

iso2ToIso3 *Convert from ISO2 to ISO3 country codes*

Description

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

Usage

```
iso2ToIso3(countryCodes)
```

Arguments

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

Value

A vector of ISO3 country codes

iso3ToIso2 *Convert from ISO3 to ISO2 country codes*

Description

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

Usage

```
iso3ToIso2(countryCodes)
```

Arguments

countryCodes	Vector of ISO 3166-1 alpha-3 codes.

Value

A vector of ISO2 country codes

loadSpatialData *Load spatial datasets*

Description

Load datasets found in the directory previously set with `setSpatialDataDir()`. Only files matching `pattern` will be loaded. By default, all `.rda` files are matched.

Core datastes available for the package include:

- `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`)

These can be installed with `installSpatialData()`.

Usage

```
loadSpatialData(pattern = "*\\.rda")
```

Arguments

pattern	Regular expression used to match file names.

Value

Invisibly returns a vector of spatial dataset names loaded into the global environment.

See Also

setSpatialDataDir

installSpatialData

`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` *Simplified spatial dataset of country boundaries.*

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 can be used with `getCountry()`, `getCountryCode()` and `getCountryName()` when restricting searches to land-based locations.

Usage

```
SimpleCountries
```

Format

A simple features data frame with 246 records and 7 columns of data.

Details

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

See Also

convertTMWorldBordersSimple

This dataset was generated on 2022-11-04 by running:

```
library(MazamaSpatialUtils)
setSpatialDataDir("~/Data/Spatial_0.8")

convertTMWorldBorders()

loadSpatialData("NaturalEarthAdm0_05")

columnNames <- c("countryCode", "countryName", "ISO3", "FIPS",
                 "UN_region", "polygonID")
SimpleCountries <- NaturalEarthAdm0_05[, columnNames]
save(SimpleCountries, file = "data/SimpleCountries.rda")
```

SimpleCountriesEEZ *Simplified spatial dataset of EEZ/country combined boundaries.*

Description

SimpleCountriesEEZ is a simplified world borders dataset with a 200 mile coastal buffer corresponding to Exclusive Economic Zones, suitable for quick spatial searches. This dataset is distributed with the package and is used by default in `getCountry()`, `getCountryCode()` and `getCountryName()`.

Usage

```
SimpleCountriesEEZ
```

Format

A simple features data frame with 319 records and 5 columns of data.

Details

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

See Also

convertEEZCountries

This dataset was generated on 2022-11-03 by running:

```
library(MazamaSpatialUtils)
setSpatialDataDir("~/Data/Spatial_0.8")

convertEEZCountries()

loadSpatialData("EEZCountries_05")

SimpleCountriesEEZ <- EEZCountries_05[,c("countryCode", "countryName", "polygonID")]
save(SimpleCountriesEEZ, file = "data/SimpleCountriesEEZ.rda")
```

SimpleTimezones *Simplified spatial dataset of world timezones.*

Description

This dataset is used by default in the `getTimezones()` function and contains the following columns of data:

- `timezone` – Olson timezone
- `UTC_offset` – offset from UTC (hours)
- `UTC_DST_offset` – offset from UTC during daylight savings (hours)
- `countryCode` – ISO 3166-1 alpha-2 country code
- `longitude` – longitude of the timezone polygon centroid
- `latitude` – longitude of the timezone polygon centroid
- `status` – one of 'Canonical', 'Alias' or 'Deprecated'
- `notes` – typically specifying the target of an 'Alias'
- `polygonID` – unique identifier (= `timezone`)

This dataset was generated on 2022-11-03 by running:

```
library(MazamaSpatialUtils)
setSpatialDataDir("~/Data/Spatial_0.8")

convertOSMTimezones()

loadSpatialData("OSMTimezones_02")

SimpleTimezones <- OSMTimezones_02
save(SimpleTimezones, file = "data/SimpleTimezones.rda")
```

Usage

```
SimpleTimezones
```

Format

A simple features data frame with 423 records and 9 columns of data.

`simplify` *Simplify simple features data frame*

Description

Simplify a simple features dataframe. This is a convenience wrapper for `ms_simplify`

Usage

```
simplify(SFDF, keep = 0.05, ...)
```

Arguments

`SFDF`	Object of class simple features data frame.
`keep`	Proportion of points to retain (0-1; default 0.05)
`...`	Arguments passed to `rmapshaper::ms_simplify()`

Value

A simplified simple features dataframe.

Examples

```
## Not run:
library(MazamaSpatialUtils)
FR <-
  SimpleCountries %>%
  dplyr::filter(countryCode == "FR")
par(mfrow = c(3, 3), mar = c(1, 1, 3, 1))
for (i in 9:1) {
  keep <- 0.1 * i
  geom <-
    FR %>%
    simplify(keep) %>%
    sf::st_geometry()
  plot(geom, main=paste0("keep = ", keep))
}
layout(1)
par(mar = c(5,4,4,2)+.1)

## End(Not run)
```

simplifyAndSave *Save simplified versions of a spatial features dataframe*

Description

Creates and saves "_05", _02" and "_01" versions of SFDF that are simplified to 5%, 2% and 1%. .

Usage

```
simplifyAndSave(
  SFDF = NULL,
  datasetName = NULL,
  uniqueIdentifier = NULL,
  dataDir = getSpatialDataDir()
)
```

Arguments

SFDF	Simple features dataframe.
datasetName	Base name used for SFDF.
uniqueIdentifier	Name of column containing unique polygon identifiers.
dataDir	Spatial data directory set with setSpatialDataDir().

Value

Writes data files to disk.

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`	Vector of state names to be converted.
`countryCodes`	Vector of 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 install and load the "NaturalEarthAdm1" dataset.

Value

A vector of ISO 3166-2 codes or NA.

See Also

convertNaturalEarthAdm1

Examples

```
## Not run:
stateToCode("Washington")
stateToCode("Barcelona")
stateToCode("Shandong")

## End(Not run)
```

summarizeByPolygon *Summarize values by polygon*

Description

Given vectors of longitudes, latitudes and values, this function will summarize given values by spatial polygon using the FUN and return a dataframe with polygon IDs and summary values.

Usage

```
summarizeByPolygon(
  longitude,
  latitude,
  value,
  SFDF,
  useBuffering = FALSE,
  FUN,
  varName = "summaryValue"
)
```

Arguments

longitude	vector of longitudes
latitude	vector of latitudes
value	vector of values at the locations of interest
SFDF	simple features data frame with polygons used for aggregating
useBuffering	passed to MazamaSpatialUtils::getSpatialData()
FUN	function to be applied while summarizing (e.g. mean, max, etc.)
varName	variable name assigned to the summary variable

Value

A dataframe with the same rows as 'SFDF' but containing only two columns: 'polygonID' and the summary value.

Note

This function has not been thoroughly tested and is included in the package for experimental use only.

US_52 *US state codes*

Description

State codes for the 50 states +DC +PR (Puerto Rico).

Usage

```
US_52
```

Format

A vector with 52 elements

US_countyCodes *Dataframe of US county codes*

Description

US_countyCodes The following columns for US states and territories:

- `stateCode` – ISO 3166-2 alpha-2
- `stateFIPS` – 2-digit FIPS code
- `countyName` – English language county name
- `countyFIPS` – five-digit FIPS code (2-digit state and 3-digit county combined to create a unique identifier)

This dataset was generated on 2022-11-04 by running:

```
library(MazamaSpatialUtils)
setSpatialDataDir("~/Data/Spatial_0.8")
loadSpatialData("USCensusCounties_02")

US_countyCodes <-
  USCensusCounties_02
  dplyr::select(stateCode, stateFIPS, countyName, countyFIPS)

US_countyCodes$geometry <- NULL

save(US_countyCodes, file = "data/US_countyCodes.rda")
```

Usage

```
US_countyCodes
```

Format

A dataframe with 3197 rows and 4 columns of data.

US_countyConversion *Conversion functions for US county names and FIPS codes.*

Description

Converts a vector of US county names or FIPS codes from one system to another returning NA where no match is found.

Usage

```
US_countyNameToFIPS(state = NULL, countyName = NULL)

US_countyFIPSToName(state = NULL, countyFIPS = NULL)
```

Arguments

state	Vector of state codes, names or FIPS codes. Values will be evaluated to determine the type of input.
countyName	Vector of English language county names.
countyFIPS	Vector of two-digit FIPS codes.

Value

A vector of US county names or FIPS codes.

Examples

```
library(MazamaSpatialUtils)

US_countyNameToFIPS("Washington", "King")

# If a single state is provided, it will be recycled
US_countyNameToFIPS("Washington", c("King", "Okanogan"))

# Normally, equal length vectors are provided
US_countyNameToFIPS(c("WA", "WA"), c("King", "Okanogan"))

# You cannot mix codes!
US_countyNameToFIPS(c("WA", "Washington"), c("King", "Okanogan"))

# No 'Okanogan' county in Texas
US_countyNameToFIPS(c("WA", "TX"), c("King", "Okanogan"))

# But there is a 'King' county in Texas
US_countyNameToFIPS(c("TX", "WA"), c("King", "Okanogan"))
```

```
US_countyNameToFIPS(c("TX", "WA"), c("King", "King"))

# The US_countyFIPSToName() function is included for symmetry but a
# more typical usage of a 5-digit county FIPS would be to extract it from
# the US_countyCodes package dataset:

US_countyCodes %>% dplyr::filter(countyFIPS == 53033)
```

`US_stateCodes` *Dataframe of US state codes*

Description

US_stateCodes the following columns for US states and territories:

- `stateName` – English language state name
- `stateCode` – ISO 3166-2 alpha-2
- `stateFIPS` – two-digit FIPS code

This dataset was generated on 2020-06-11 by running:

Usage

```
US_stateCodes
```

Format

A dataframe with 52 rows and 3 columns of data.

`US_stateConversion` *Conversion functions for US state names, codes and FIPS codes.*

Description

Converts a vector of US state names or codes from one system to another returning `NA` where no match is found.

Usage

```
US_stateCodeToName(stateCode = NULL)

US_stateCodeToFIPS(stateCode = NULL)

US_stateFIPSToName(stateFIPS = NULL)

US_stateFIPSToCode(stateFIPS = NULL)

US_stateNameToCode(stateName = NULL)

US_stateNameToFIPS(stateName = NULL)
```

Arguments

stateCode	Vector of ISO 3166-2 alpha-2 codes.
stateFIPS	Vector of two-digit FIPS codes.
stateName	Vector of English language state names.

Value

A vector of US state names or codes.

Examples

```
library(MazamaSpatialUtils)

US_stateNameToCode("Washington")
US_stateNameToFIPS("Washington")

postalCodes <- sample(US_stateCodes$stateCode, 30)

data.frame(
  name = US_stateCodeToName(postalCodes),
  code = postalCodes,
  FIPS = US_stateCodeToFIPS(postalCodes)
)
```

Index