Package 'MazamaSpatialUtils'

December 1, 2020

Type Package

Version 0.7.3

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, timezone, 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 (>= 3.5.0), sp

Imports cleangeo, countrycode, dplyr, magrittr, MazamaCoreUtils (>= 0.4.5), rgdal, rgeos, rlang, rmapshaper, stringr, tidyr

Suggests knitr, maps, markdown, readr, rmarkdown, testthat

Encoding UTF-8

VignetteBuilder knitr

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Date/Publication 2020-12-01 20:20:02 UTC

R topics documented:

`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.

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 first run `initializeSpatialData()` to download, convert and install the necessary spatial data.

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

convertCARBAirBasins *Convert California Air Resources Board basin shapefiles*

Description

Returns a SpatialPolygonsDataFrame for CARB air basins,

Usage

```
convertCARBAirBasins(nameOnly = FALSE, simplify = TRUE)
```

Arguments

nameOnly	Logical specifying whether to only return the name without creating the file.
simplify	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A California air basin 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()`.

The source data is from 2004.

Value

Name of the dataset being created.

Note

From the source documentation:

The California Air Basins layer is a polygon shapefile coverage representing the 15 California air basins, as defined in state statute and regulation. See the California Health and Safety Code, Section 39606 et seq. and California Code of Regulations, Title 17, Section 60100 et seq. Shapefile coverage. Projection: Teale Albers, NAD83, Clarke 1866.

Air Basins are designated pursuant to California statute and regulation. Air Basins identify regions of similar meteorological and geographic conditions and consideration for political boundary lines, and are related to air pollution and its transport.

References

https://ww2.arb.ca.gov//ei/gislib/gislib.htm

See Also

setSpatialDataDir

getVariable

`convertEEZCountries` *Convert Exclusive Economic Zones countries shapefile*

Description

Create a SpatialPolygonsDataFrame for combined EEZ/country boundaries

Usage

```
convertEEZCountries(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A world EEZ/countries shapefile is converted to a SpatialPolygonsDataFrame 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 dataset 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 datasets, 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 SpatialPolygonsDataFrame for EPA Regions

Usage

```
convertEPARegions(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

An EPA region boundary 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()`.

The source data is from 2017.

Value

Name of the dataset being created.

Note

From the source documentation:

This dataset 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)dataset 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 datasets that are national in extent and of general use to all EPA users and to make those datasets available through channels that best meet user needs.

References

https://www.arcgis.com/home/item.html?id=c670540796584c72b4f59b676ccabe6a

See Also

setSpatialDataDir

getVariable

`convertGACC` *Convert Geographic Area Coordination Center shapefile*

Description

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

Usage

```
convertGACC(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A GACC 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()`.

The source data is from 2020.

Value

Name of the dataset 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/7dc5f4a286bd47e0aaafa0ab05302fe9_0

See Also

setSpatialDataDir

`convertGADM` *Convert Global Administrative Areas (GADM) SPDF*

Description

Create a SpatialPolygonsDataFrame for Global Administrative Areas.

Usage

```
convertGADM(
  countryCode = NULL,
  admLevel = 0,
  nameOnly = FALSE,
  baseUrl = "https://biogeo.ucdavis.edu/data/gadm3.6/Rsp"
)
```

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.
`baseUrl`	Base URL for data queries.

Details

A pre-generated Global Administrative Areas SpatialPolygonsDataFrame is downloaded and amended with additional columns of data. The resulting file will be created in the spatial data directory which is set with `setSpatialDataDir()`.

The `@data` slot of each SpatialPolygonsDataFrame is both simplified and modified to adhere to the **MazamaSpatialUtils** internal standards.

Value

Name of the dataset being created.

Note

Unlike other `convert~()` functions, no checks, cleanup or simplification is performed.

From the source documentation:

GADM has always had multiple unique IDs associated with a record. In the future there will be only be the GID and associated tables that link the GID to other ID systems such as ISO, FIPS, and HASC. The GID starts with the three letter ISO 3166-1 alpha-3 country code. If there are subdivisions these are identified by a number from 1 to n, where n is the number of subdivisions at level 1. This value is concatenated with the country code, using a dot to delimit the two. For example, AFG.1, AFG.2, ..., AFG.n. If there are second level subdivisions, numeric codes are assigned within each first level subdivision and these are concatenated with the first level identifier, using a dot as delimiter. For example, AFG.1.1, AFG.1.2, AFG.1.3, ..., and AFG.2.1, AFG.2.2, ... And so forth for the third, fourth and fifth levels. Finally there is an underscore followed by a version number appended to the code. For example, AFG.3_1 and AFG.3.2_1. The GID codes are presistent after version 3.6 (there were errors in the codes in version 3.4). If an area changes, for example if it splits into two new areas, two new codes will be assigned, and the old code will not be used any more. The version only changes when there is a major overhaul of the divisions in a country, for example when a whole new set of subdivsions is introduced.

References

https://gadm.org/data.html

https://gadm.org/metadata.html

---

convertNaturalEarthAdm1

Convert Level 1 (State) Borders Shapefile

Description

Returns a SpatialPolygonsDataFrame for 1st level administrative divisions

Usage

```
convertNaturalEarthAdm1(nameOnly = FALSE, simplify = TRUE)
```

Arguments

nameOnly	Logical specifying whether to only return the name without creating the file.
simplify	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

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/downloads/

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

See Also

setSpatialDataDir

getVariable

---

convertNWSFireZones *Convert NWS Public Forecast Zones Shapefile*

Description

Create a SpatialPolygonsDataFrame for NWS weather forecast zones

Usage

```
convertNWSFireZones(nameOnly = FALSE, simplify = TRUE)
```

Arguments

nameOnly	Logical specifying whether to only return the name without creating the file.
simplify	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A NWS forecast zone 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()`.

The source data is from 2020.

Value

Name of the dataset being created.

Note

From the source documentation:

This data set is used to delineate the Fire Weather Zones that are used by NWS in the fire weather forecast program.

References

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

See Also

setSpatialDataDir

getVariable

---

convertOSMTimezones *Convert OSM Timezone Shapefile*

Description

Create a SpatialPolygonsDataFrame for world timezones.

Usage

```
convertOSMTimezones(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A world timezone 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()`.

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

Value

Name of the dataset 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

See Also

setSpatialDataDir

getVariable

`convertStateLegislativeDistricts` *Convert US state legislative districts shapefile*

Description

Create a SpatialPolygonsDataFrame for US State Legislative Districts of a specified state

Usage

```
convertStateLegislativeDistricts(
  stateCode,
  house = "Upper",
  nameOnly = FALSE,
  simplify = TRUE
)
```

Arguments

`stateCode`	ISO 3166-2 alpha-2 state code.
`house`	Character specifying either "Upper" or "Lower" house.
`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A US State Legislative District 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.

Note

From the source documentation:

The 2019 cartographic boundary shapefiles 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. 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.

SLDL stands for State Legislative District Lower Chamber...

SLDU stands for State Legislative District Upper Chamber.

State Legislative Districts (SLDs) are the areas from which members are elected to state legislatures. The SLDs embody the upper (senate) and lower (house) chambers of the state legislature. Nebraska has a unicameral legislature and the District of Columbia has a single council, both of which the Census Bureau treats as upper-chamber legislative areas for the purpose of data presentation; there are no data by SLDL for either Nebraska or the District of Columbia. A unique three- character census code, identified by state participants, is assigned to each SLD within a state. In Connecticut, Illinois, Louisiana, Maine, Maryland, Massachusetts, Michigan, Ohio, and Puerto Rico, the Redistricting Data Program (RDP) participant did not define the SLDs to cover all of the state or state equivalent area. In these areas with no SLDs defined, the code "ZZZ" has been assigned, which is treated as a single SLD for purposes of data presentation.

The boundaries of the 2018 state legislative districts were provided by state- level participants through the RDP and reflect the districts used to elect members in or prior to the November 2018 election

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://www.census.gov/geographies/mapping-files/time-series/geo/cartographic-boundary.html

See Also

setSpatialDataDir

`convertTerrestrialEcoregions` *Convert Terrestrial Ecoregion Shapefile*

Description

Create a SpacialPolygonsDataFrame for Terrestrial Ecoregions.

Usage

```
convertTerrestrialEcoregions(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A Terrestrial Ecoregions 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.

Note

From the source documentation:

This map depicts the 825 terrestrial ecoregions of the globe. Ecoregions are relatively large units of land containing distinct assemblages of natural communities and species, with boundaries that approximate the original extent of natural communities prior to major land-use change. This comprehensive, global map provides a useful framework for conducting biogeographical or macroecological research, for identifying areas of outstanding biodiversity and conservation priority, for assessing the representation and gaps in conservation efforts worldwide, and for communicating the global distribution of natural communities on earth. We have based ecoregion delineations on hundreds of previous biogeographical studies, and refined and synthesized existing information in regional workshops over 10 years to assemble the global dataset. Ecoregions are nested within two higher-order classifications: biomes (14) and biogeographic realms (8). Together, these nested classification levels provide a framework for comparison among units and the identification of representative habitats and species assemblages.

References

https://www.worldwildlife.org/publications/terrestrial-ecoregions-of-the-world

See Also

setSpatialDataDir

getVariable

`convertTMWorldBorders` *Convert World Borders Shapefile*

Description

Returns a SpatialPolygonsDataFrame for world divisions

Usage

```
convertTMWorldBorders(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A world borders shapefile is downloaded and converted to a SpatialPolygonsDataFrame with additional columns of data. The resulting file is 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 simple world divisions

Usage

```
convertTMWorldBordersSimple(nameOnly = FALSE, simplify = TRUE)
```

Arguments

nameOnly	Logical specifying whether to only return the name without creating the file.
simplify	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

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 simplified version of the TMWorldBorders shapefile. 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

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

Description

Returns a SpatialPolygonsDataFrame for US CBSAs

Usage

```
convertUSCensusCBSA(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A US Core Based Statistical Areas (CBSA) 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.

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 2018

References

https://www2.census.gov/geo/tiger/TIGER2019/CBSA/

See Also

setSpatialDataDir

getUSCounty

`convertUSCensusCongress` *Convert US congressional districts shapefile*

Description

Returns a SpatialPolygonsDataFrame for US Congressional Districts for the 116th US House of Representatives.

Usage

```
convertUSCensusCongress(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A US congressional district 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()`.

#' The source data is from 2019.

Value

Name of the dataset 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/GENZ2019/

See Also

setSpatialDataDir

`convertUSCensusCounties` *Convert US county borders shapefile*

Description

Create a SpatialPolygonsDataFrame for US counties.

Usage

```
convertUSCensusCounties(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

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()`.

The source data is from 2019.

Value

Name of the dataset 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/GENZ2019/

See Also

setSpatialDataDir

getUSCounty

`convertUSCensusStates` *Convert US Census State Shapefile*

Description

Create a SpatialPolygonsDataFrame for US states

Usage

```
convertUSCensusStates(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A US state 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()`.

The source data is from 2019.

Value

Name of the dataset being created.

Note

From the source documentation:

The 2019 cartographic boundary shapefiles 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. 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/GENZ2019/

See Also

setSpatialDataDir

getState

getCode

getName

convertUSCensusUrbanAreas *Convert US Census Urban Areas shapefiles*

Description

Create a SpatialPolygonsDataFrame for US states

Usage

```
convertUSCensusUrbanAreas(nameOnly = FALSE, simplify = TRUE)
```

Arguments

nameOnly	Logical specifying whether to only return the name without creating the file.
simplify	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

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()`.

The source data is from 2019.

Value

Name of the dataset being created.

Note

From the source documentation:

After each decennial census, the Census Bureau delineates urban areas that represent densely developed territory, encompassing residential, commercial, and other nonresidential urban land uses. In general, this territory consists of areas of high population density and urban land use resulting in a representation of the "urban footprint." There are two types of urban areas: urbanized areas (UAs) that contain 50,000 or more people and urban clusters (UCs) that contain at least 2,500 people, but fewer than 50,000 people (except in the U.S. Virgin Islands and Guam which each contain urban clusters with populations greater than 50,000). Each urban area is identified by a 5-character numeric census code that may contain leading zeroes.

References

https://www2.census.gov/geo/tiger/TIGER2019/UAC/

`convertUSIndianLands` *Convert Indian Lands Shapefile*

Description

Create a SpatialPolygonsDataFrame for Native American land.

Usage

```
convertUSIndianLands(nameOnly = FALSE, simplify = TRUE)
```

Arguments

`nameOnly`	Logical specifying whether to only return the name without creating the file.
`simplify`	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A Native American land 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()`.

The source data is from 2014.

Value

Name of the dataset being created.

Note

From the source documentation:

This map layer shows Indian lands of the United States. For the most part, only areas of 320 acres or more are included; some smaller areas deemed to be important or significant are also included. Federally-administered lands within a reservation are included for continuity; these may or may not be considered part of the reservation and are simply described with their feature type and the administrating Federal agency. Some established Indian lands which are larger than 320 acres are not included in this map layer because their boundaries were not available from the owning or administering agency.The USIndianLands shapefile represents lands administered by the Bureau of Indian Affairs, ie. Indian reservations

These data are intended for geographic display and analysis at the national level, and for large regional areas. The data should be displayed and analyzed at scales appropriate for 1:1,000,000-scale data. No responsibility is assumed by the National Atlas of the United States in the use of these data. and is compiled by the National Atlas of the United States of America.

References

https://www.sciencebase.gov/catalog/item/5d150464e4b0941bde5b7658

See Also

setSpatialDataDir

getVariable

---

convertWBDHUC *Convert USGS hydrologic unit shapefiles*

Description

Create a SpatialPolygonsDataFrame for USGS watershed boundaries

Usage

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

Arguments

gdbDir	Directory containing the geodatabase.
level	Character or integer which must be 2, 4, 6, 8, 10, 12 or 14.
nameOnly	Logical specifying whether to only return the name without creating the file.
simplify	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.
cleanTopology	Logical specifying to use the **cleangeo** package to clean topological errors in the shapefiles.

Details

A USGS Watershed Boundary Dataset geodatabase is converted to a SpatialPolygonsDataFrame 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 dataset 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
```

Value

Name of the dataset being created.

Note

If processing takes too long you can greatly speed things up with: `simplify = FALSE, cleanTopology = FALSE`

The source data is from Version 2.3 – 2020-11-19.

From the source documentation:

The Watershed Boundary Dataset (WBD) is a seamless, national hydrologic unit dataset. Simply put, hydrologic units represent the area of the landscape that drains to a portion of the stream network. More specifically, a hydrologic unit defines the areal extent of surface water drainage to an outlet point on a dendritic stream network or to multiple outlet points where the stream network is not dendritic. A hydrologic unit may represent all or only part of the total drainage area to an outlet point so that multiple hydrologic units may be required to define the entire drainage area at a given outlet. Hydrologic unit boundaries in the WBD are determined based on topographic, hydrologic, and other relevant landscape characteristics without regard for administrative, political, or jurisdictional boundaries. The WBD seamlessly represents hydrologic units at six required and two optional hierarchical levels.

The hydrologic units (HU) in the WBD form a standardized system for organizing, collecting, managing, and reporting hydrologic information for the nation. The HU in the WBD are arranged in a nested, hierarchical system with each HU in the system identified using a unique code. Hydrologic unit codes (HUC) are developed using a progressive two-digit system where each successively smaller areal unit is identified by adding two digits to the identifying code the smaller unit is nested within. WBD contains eight levels of progressive hydrologic units identified by unique 2- to 16-digit codes. The dataset is complete for the United States to the 12-digit hydrologic unit. The 14- and 16-digit hydrologic units are optional and are not complete for the nation. Efforts are ongoing to complete 10- and 12-digit unit delineations within 8-digit hydrologic units extending across the U.S. – Canada border. Additional information about this effort and access to data is linked on the "resources" section on this page. A similar effort is complete for the 10- and 12-digit units extending across the U.S. – Mexico border.

References

https://www.usgs.gov/core-science-systems/ngp/national-hydrography/watershed-boundary-dataset

See Also

setSpatialDataDir

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

Description

Create a SpatialPolygonsDataFrame for NWS weather forecast zones.

Usage

```
convertWeatherZones(nameOnly = FALSE, simplify = TRUE)
```

Arguments

nameOnly	Logical specifying whether to only return the name without creating the file.
simplify	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A weather forecast zone 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()`.

The source data is from 2020.

Value

Name of the dataset being created.

Note

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

See Also

setSpatialDataDir

getVariale

---

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

---

convertWorldTimezones *Convert Timezone Shapefile*

Description

Create a SpatialPolygonsDataFrame for world timezones.

Usage

```
convertWorldTimezones(nameOnly = FALSE, simplify = TRUE)
```

Arguments

nameOnly	Logical specifying whether to only return the name without creating the file.
simplify	Logical specifying whether to create "_05", _02" and "_01" versions of the file that are simplified to 5%, 2% and 1%.

Details

A world timezones 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()`.

The source data is from 2020 – release 2020d.

Value

Name of the dataset being created.

Note

From the source documentation:

Each shape or geojson object has a single attribute or property respectively called tzid. The tzid corresponds to the timezone name as defined in the timezone database (for example: America/Los_Angeles or Asia/Shanghai).

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.

References

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

See Also

setSpatialDataDir

getTimezone

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 English language country names.

Value

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

dissolve *Aggregate shapes in a SpatialPolygonsDataFrame*

Description

Aggregate shapes in a spatial polygons dataframe. This is a convenience wrapper for `rmapshaper::ms_dissolve()`

Usage

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

Arguments

`SPDF`	Object of class SpatialPolygonsDataFrame.
`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 spatial polygons dataframe with aggregated shapes.

Examples

```
regions <- dissolve(SimpleCountries, field = "UN_region", sum_fields = "area")
plot(regions)
regions@data
```

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(
  longitude,
  latitude,
  dataset = "SimpleCountriesEEZ",
  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.
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.

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

Usage

```
getCountryCode(
  longitude,
  latitude,
  dataset = "SimpleCountriesEEZ",
  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.
`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.

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

Usage

```
getCountryName(
  longitude,
  latitude,
  dataset = "SimpleCountriesEEZ",
  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.
`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)

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.

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

Usage

```
getHUC(
  longitude,
  latitude,
  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.
`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 not 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.

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

Usage

```
getHUCName(
  longitude,
  latitude,
  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.
`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 not target polygon is found.

Value

Vector of HUC names.

See Also

getSpatialData

getPolygonID *Get polygonID from SPDF of interest*

Description

Extracts the the vector of unique polygon identifiers from SPDF.

This function is useful when writing code to aggregate data by polygon and calculate per-polygon statistics. Each unique SpatialPolygonsDataFrame 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(SPDF)
```

Arguments

SPDF	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 **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 location.

Usage

```
getSpatialData(
  longitude,
  latitude,
  SPDF,
  useBuffering = FALSE,
  verbose = FALSE
)
```

Arguments

longitude	Vector of longitudes in decimal degrees East.
latitude	Vector of latitudes in decimal degrees North.
SPDF	Object of class SpatialPolygonsDataFrame.
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

Occasionally for coastal locations the precise coordinates lie outside the boundaries of a low resolution SpatialPolygonsDataFrame. To account for this any location that remains unassociated after the first pass is then buffered to create a small circle around the original location. All polygons are then checked to see if there is any intersection with the now larger buffered locations. Each point is then checked for an intersecting polygon at the following radii: 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 longitude or latitude 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(
  longitude,
  latitude,
  dataset = "NaturalEarthAdm1",
  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.
`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")

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,
  latitude,
  dataset = "NaturalEarthAdm1",
  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.
`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")

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,
  latitude,
  dataset = "NaturalEarthAdm1",
  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.
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")

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,
  latitude,
  dataset = "SimpleTimezones",
  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.
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

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

See Also

SimpleTimezones

getSpatialData

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,
  latitude,
  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")

loadSpatialData("USCensusCounties")

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

## End(Not run)
```

`getVariable` *Return SPDF 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,
  latitude,
  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

Examples

```
library(MazamaSpatialUtils)

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

getVariable(longitude, latitude, "SimpleCountries", "UN_region")
```

---

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

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_0.7"
)
```

Arguments

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

Value

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

`installSpatialData_0.6`
Install version 0.6 spatial datasets

Description

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

Usage

```
installSpatialData_0.6(
  urlBase = "http://data.mazamascience.com/MazamaSpatialUtils/Spatial",
  file = "mazama_spatial_files-0.6.tar.gz"
)
```

Arguments

`urlBase`	location of spatial data files
`file`	name of the tar.gz file containing spatial datasets

Value

Nothing.

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, only `.RData` and `.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 = "*\\.[rR][dD]a?t?a")
```

Arguments

`pattern`	Regular expression used to match filenames.

Value

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

See Also

setSpatialDataDir

installSpatialData

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_defs")`)
- 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.

`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 SpatialPolygonsDataFrame with 246 records and 6 columns of data.

Details

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

See Also

convertTMWorldBordersSimple

This dataset was generated on 2020-11-19 by running:

```
library(MazamaSpatialUtils)
setSpatialDataDir("~/Data/Spatial")

convertTMWorldBordersSimple()

loadSpatialData("TMWorldBordersSimple")
```

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 SpatialPolygonsDataFrame 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 2020-11-18 by running:

```
library(MazamaSpatialUtils)
setSpatialDataDir("~/Data/Spatial")

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 in the `@data` slot:

- `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 2020-11-12 by running:

```
library(MazamaSpatialUtils)
setSpatialDataDir("~/Data/Spatial")

convertWorldTimezones()

loadSpatialData("WorldTimezones_02")

SimpleTimezones <- WorldTimezones_02
save(SimpleTimezones, file = "data/SimpleTimezones.rda")
```

Usage

```
SimpleTimezones
```

Format

A SpatialPolygonsDataFrame with 423 records and 9 columns of data.

simplify *Simplify SpatialPolygonsDataFrame*

Description

Simplify a spatial polygons dataframe. This is a convenience wrapper for `rmapshaper::ms_simplify()`

Usage

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

Arguments

SPDF	Object of class SpatialPolygonsDataFrame.
keep	Proportion of points to retain (0-1; default 0.05)
...	Arguments passed to `rmapshaper::ms_simplify()`

Value

A simplified spatial polygons dataframe.

Examples

```
## Not run:
FR <- subset(SimpleCountries, countryCode == 'FR')
par(mfrow = c(3, 3), mar = c(1, 1, 3, 1))
for (i in 9:1) {
  keep <- 0.1 * i
  plot(simplify(FR, keep), main=paste0("keep = ", keep))
}
layout(1)
par(mar = c(5,4,4,2)+.1)

## End(Not run)
```

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 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

Examples

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

## End(Not run)
```

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 the subset to include HUCs where part of the watershed is in the specified state even though the centroid is in a different state.

Usage

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

Arguments

`SPDF`	a spatial polygons dataframe created using the convertUSGSHUC function
`parentHUCs`	Character vector specifying one or more containing HUCs.
`stateCodes`	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.

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,
  SPDF,
  useBuffering = FALSE,
  FUN,
  varName = "summaryValue"
)
```

Arguments

longitude	vector of longitudes
latitude	vector of latitudes
value	vector of values at the locations of interest
SPDF	SpatialPolygonsDataFrame 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 'SPDF@data' but containing only two columns: 'polygonID' and the summary value.

Note

This function has not been thoroughly tested and should only be 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 state 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 2020-10-26 by running:

```
library(MazamaSpatialUtils)
setSpatialDataDir("~/Data/Spatial")
loadSpatialData("USCensusCounties_02")

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

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

Usage

```
US_countyCodes
```

Format

A dataframe with 3196 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 reuturning `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 reuturning 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