Package: gisr (via r-universe)

November 5, 2024

Title Geospatial Analytics Utility functions

Version 0.4.0

Description R Spatial functions for HIV/AIDS related Geospatial Analytics.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/USAID-OHA-SI/gisr, https://usaid-oha-si.github.io/gisr/

BugReports https://github.com/USAID-OHA-SI/gisr/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends R (>= 3.6.0)

Imports readr, dplyr (>= 1.0.0), tidyr, magrittr, purrr, stringr, tibble, ggplot2, raster, terra, sf, rnaturalearth, methods, crayon, janitor, jsonlite, zip, glue, fs

Suggests knitr, rmarkdown, testthat (>= 3.0.0), here, rnaturalearthdata, rnaturalearthhires, osmdata, patchwork, glamr, glitr, googledrive, googlesheets4, usethis

Remotes USAID-OHA-SI/glamr, USAID-OHA-SI/glitr, USAID-OHA-SI/grabr, USAID-OHA-SI/gagglr, ropensci/rnaturalearthhires

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev make libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev

Repository https://usaid-oha-si.r-universe.dev

RemoteUrl https://github.com/USAID-OHA-SI/gisr

RemoteRef HEAD

RemoteSha 2a040533ab3eb0658a325acc39ed889a495efcc6

Contents

`admins_map` *Generate administrative boundaries map*

Description

Generate administrative boundaries map

Usage

```
admins_map(countries, add_neighbors = FALSE)
```

Arguments

countries	list of countries to map
add_neighbors	should the map include the neighbor countries

Value

ggplot admin map

Examples

```
## Not run:
 library(gisr)

 admins_map(countries = list("Zambia"))
 admins_map(countries = list("Zambia"), add_neighbors = TRUE)

## End(Not run)
```

---

assess_facilities *Assess facility geo-location reporting levels*

Description

Assess facility geo-location reporting levels

Usage

```
assess_facilities(.data)
```

Arguments

.data	Datim organisation units data frame

Examples

```
## Not run:
assess_facilities(df_sites)
df_sites %>% assess_facilities()

## End(Not run)
```

check_columns *Check columns length*

Description

Check columns length

Usage

```
check_columns(spdf)
```

Arguments

spdf	input spatial data frame

Value

list of columns with more than 10 characters

cntry_polygons *Extract Country Polygons*

Description

Extract Country Polygons

Usage

```
cntry_polygons(spdf, cntry, attrs)
```

Arguments

spdf	VcPolygons data as Spatial Data Frame
cntry	Country name
attrs	Country orgunits, output of `grabr::datim_orgunits(cntry, reshape = TRUE)`

Value

list of spatial data frames

Examples

```
## Not run:

cntry = "Nigeria"

spdf <- gisr::get_vcpolygons(path = glamr::si_path("path_vector"), name = "VcPepfarPolygons.shp")

df_attrs <- grabr::get_attributes(cntry)

cntry_polygons(spdf = spdf, cntry = "Zambia", attrs = df_attrs)


## End(Not run)
```

`download_shapefiles` *Download shapefile zipfile from googledrive*

Description

Download shapefile zipfile from googledrive

Usage

```
download_shapefiles(
  country,
  org_label = "country",
  drive_folder = NULL,
  dest_file = NULL,
  overwrite = TRUE,
  unzip = FALSE
)
```

Arguments

`country`	PEPFAR Countryname
`org_label`	Orgunit label, default is set to country
`drive_folder`	Googledrive id for all PEPFAR Spatial files
`dest_file`	Full file name where to download zipped shapefile
`overwrite`	Should the process overwrite existing files
`unzip`	Should the zipfile be unzipped

Examples

```
## Not run:

  cntry <- "Zambia"

  download_shapefiles(country = cntry)

  download_shapefiles(country = cntry, org_label = "snu1", unzip = TRUE)

  download_shapefiles(country = cntry, org_label = "psnu", unzip = TRUE)

## End(Not run)
```

---

dview *View attributes from simple feature object*

Description

View attributes from simple feature object

Usage

```
dview(geodata, console = FALSE)
```

Arguments

geodata	spatial data
console	view in console? default false

Examples

```
## Not run:
 library(gisr)

 adm0 <- get_admin0("Ghana")

 adm0 %>% dview(console = TRUE)

## End(Not run)
```

explore_facilities *Explore facility locations*

Description

Explore facility locations

Usage

```
explore_facilities(.data, cntry, terr_path = NULL)
```

Arguments

.data	Datim organisation units data frame
cntry	Country name
terr_path	Path to terrain raster dataset

Examples

```
## Not run:
explore_facilities(df_sites, cntry = "saturn")
df_sites %>% explore_facilities("saturn")

## End(Not run)
```

export_spdf *Save spatial data as shapefile*

Description

Save spatial data as shapefile

Usage

```
export_spdf(spdf, name)
```

Arguments

spdf	sf object
name	filename with full path

Value

boolean

Examples

```
## Not run:
 library(gisr)
 library(sf)

 shp <- get_admin0(countries = "Nigeria")

 export_spdf(spdf = shp, name = "./GIS/nga_country_boundaries")
 export_spdf(spdf = shp, name = "./GIS/nga_country_boundaries.shp")

## End(Not run)
```

`extract_attributes` *Extract Orgunit Boundaries Attributes*

Description

Extract Orgunit Boundaries Attributes

Usage

```
extract_attributes(
  country,
  username,
  password,
  folderpath,
  prefix = "orghierarchy",
  baseurl = NULL
)
```

Arguments

`country`	OU/country
`username`	Datim username
`password`	Datim password
`folderpath`	Local directory of files
`prefix`	Prefix for filename
`baseurl`	Datim URL

Value

df

Examples

```
## Not run:
 library(gisr)

 extract_attributes(country = "Nigeria")
 extract_attributes(country = "Nigeria", folderpath = glamr::si_path("path_vector"))

## End(Not run)
```

`extract_boundaries` *Extract PEPFAR Orgunit Boundaries*

Description

PEPFAR VcPolygons are shared with orgunit uids only, making hard for analysts to identify specific polygon each orgunit level. This function extract orgunt attributes from Datim and append them to the global shapefile, allowing analysts to filter and work only with subset.

Usage

```
extract_boundaries(
  spdf,
  country,
  level = 3,
  username,
  password,
  export = FALSE,
  name = NULL
)
```

Arguments

`spdf`	PEPFAR Global Shapefile
`country`	Country name
`level`	Orgunit level
`username`	Datim username
`password`	Datim password
`export`	Export extract as shapefile?
`name`	Export filename

Value

sf object with orgunit attributes

extract_facilities *Extract facility sites*

Description

Extract facility sites

Usage

```
extract_facilities(.data)
```

Arguments

.data	Datim organisation units data frame - with label and coordinates columns

Examples

```
## Not run:
  grabr::datim_pull_hierarchy(...) %>%
    extract_facilities()

## End(Not run)
```

extract_locations *Extract location data*

Description

Extract location data

Usage

```
extract_locations(
  country,
  username,
  password,
  level = NULL,
  add_geom = TRUE,
  baseurl = NULL
)
```

Arguments

country	PEPFAR Operating Unit or Country name
username	Datim Account Username
password	Datim Account password
level	PEPFAR Org Level, optional
add_geom	Include geometry column, default value is true
baseurl	Datim URL

Value

A dataframe or Null if not match

Examples

```
## Not run:

extract_locations("<saturn>", "<my_username>", "<my_password>")

## End(Not run)
```

---

extract_raster *Extract raster data for an AOI (Countries)*

Description

Extract raster data for an AOI (Countries)

Usage

```
extract_raster(countries, ras, mask = FALSE, buffer = 0.1)
```

Arguments

countries	List of the country names or sf object
ras	RasterLayer or Path to raster file
mask	Should the extracted data match the exact boundary limits?
buffer	Extend AOI extent by x

Value

spdf spatial dataframe

Examples

```
## Not run:
 library(gisr)
 library(sf)

 get_terrain(countries = list("Zambia"))
 get_terrain(countries = list("Zambia"), mask = TRUE)
 get_terrain(countries = list("Zambia"), buffer = .5, terr = "../../HDX_Data")

## End(Not run)
```

`extract_roads` *Extract Road Network data from OSM*

Description

Extract Road Network data from OSM

Usage

```
extract_roads(aoi, radius = NULL, clip = FALSE)
```

Arguments

`aoi`	Area of Interest as sf object
`radius`	Buffer redius in meters, default = 1000m
`clip`	Should the output be clipped to the AOI? Default is FALSE

Examples

```
## Not run:

 library(gisr)

 adm_zmb <- get_admin1(countries = "Zambia") %>%
   dplyr::select(name) %>%
   dplyr::filter(name == 'Lusaka')

 adm_zmb %>%
   extract_roads() %>%
   gview()

## End(Not run)
```

gattributes *Get attributes from feature class*

Description

Get attributes from feature class

Usage

```
gattributes(geodata)
```

Arguments

geodata	spatial data

Value

attribures as data frame

Examples

```
## Not run:
 library(gisr)

 # Admin level 0 [country] geodata
 adm0 <- get_admin0("Ghana")

 # Extract attrs from geodata
 .df <- gattributes(geodata = adm0)

 head(.df)

## End(Not run)
```

generate_facilities_report *Report locations data completeness*

Description

Report locations data completeness

Usage

```
generate_facilities_report(
  cntry,
  user,
  pass,
  terr_path = NULL,
  output_folder = NULL
)
```

Arguments

cntry	Country name
user	Datim account username
pass	Datim account password (glamr::mypwd is recommended)
terr_path	Path to terrain raster data
output_folder	Output folder

Examples

```
## Not run:
generate_sites_report(cntry = "saturn", mer_sites = sites)

## End(Not run)
```

---

geo_fence *Generate a buffer around an Area of Interest*

Description

Generate a buffer around an Area of Interest

Usage

```
geo_fence(aoi, radius = 1000, append = TRUE)
```

Arguments

aoi	Area of Interest as sf object
radius	Buffer redius in meters, default = 1000m
append	Should the buffered area be appended to the AOI? Default is TRUE

Value

simple feature class

Examples

```
## Not run:
 library(gisr)

 adm <- get_admin0(countries = "Zambia")

 adm %>% geo_fence(radius = 5000, append = TRUE) %>% gview()

 adm %>% geo_fence(radius = 5000, append = FALSE) %>% gview()

## End(Not run)
```

`geo_neighbors` *Get neighbors of a given contry*

Description

Get neighbors of a given contry

Usage

```
geo_neighbors(src, countries, var = "sovereignt", crs = 4326, crop = FALSE)
```

Arguments

`src`	Source spatial data frame
`countries`	countries of interest
`var`	Variable name
`crs`	Coordinates reference system, default is WGS84 (EPGS:4326)
`crop`	Crop sfc to focus countries extent?

Value

`simple feature class`

Examples

```
## Not run:
 library(gisr)

 cntry <- "Zambia"

 shp_ne <- get_necountries()

 # Country + neighbors boundaries
 cntries <- geo_neighbors(src = shp_ne, countries = cntry)
```

```
 # Country + neighbors boundaries: crop to country extent
 cntries <- geo_neighbors(src = shp_ne, countries = cntry, crop = TRUE)

 cntries %>% gview()

## End(Not run)
```

`get_admin0` *Get admin level 0 boundary*

Description

sf boundaries data for a given country

Usage

```
get_admin0(countries, scale = c("medium", "large", "small"), crs = 4326)
```

Arguments

`countries`	list of country names
`scale`	spatial resolution of the geodata
`crs`	coordinates reference system

Value

```
simple feature class
```

Examples

```
## Not run:
  library(gisr)

  get_admin0(countries = list("Zambia"))

## End(Not run)
```

get_admin1 *Get admin level 1 boundaries*

Description

sf boundaries data for a given country

Usage

```
get_admin1(countries, crs = 4326)
```

Arguments

countries	list of country names
crs	coordinates reference system

Value

simple feature class

Examples

```
## Not run:
 library(gisr)

 get_admin1(countries = list("Zambia"))

## End(Not run)
```

get_attributes *Get Attributes Data for Orgunit Boundaries*

Description

Get Attributes Data for Orgunit Boundaries

Usage

```
get_attributes(
  country,
  username,
  password,
  folderpath = NULL,
  search = "orghierarchy",
  baseurl = NULL
)
```

Arguments

country	OU/country
username	Datim username
password	Datim password
folderpath	Local directory of files
search	Search keyword
baseurl	Datim URL

Value

```
OU Orgunit level as df
```

Note

This will attempt to read data from local directory when folderpath is not set to null. If null, username and password will be required

Examples

```
## Not run:
 library(gisr)

 get_attributes(country = "Nigeria")

## End(Not run)
```

---

get_basemap *Get Basemap*

Description

Get Basemap

Usage

```
get_basemap(spdf, terr, country = NULL, add_admins = FALSE)
```

Arguments

spdf	PEPFAR ORGs Spatial Data
terr	RasterLayer
country	OU or Country Name
add_admins	Should the sub-admins be added? Default is false

Value

ggplot plot of base map

Examples

```
## Not run:
library(gisr)
library(sf)

 shp <- get_pepfar_shp(shp_path = glamr::si_path("path_vector"), add_attr = TRUE)
 ras <- get_raster(terr_path = glamr::si_path("path_raster"))

 get_basemap(spdf = shp, country = "Nigeria", terr = ras)

## End(Not run)
```

---

get_grids *Create square grids*

Description

Create square grids

Usage

```
get_grids(spdf, size = 15000, clip = TRUE)
```

Arguments

spdf	input spatial data frame
size	size of each hex bin in meters, default set to 15K meters (15KM)
clip	Should the output be cliped to the input boundaries? Default is TRUE

Value

country hex polygon as feature class

Examples

```
## Not run:
 library(gisr)

 shp <- get_admin0(countries = "Nigeria")

 get_grids(shp, 10000)

## End(Not run)
```

get_hexbins *Create hexagonal grids*

Description

Create hexagonal grids

Usage

```
get_hexbins(spdf, size = 15000, clip = TRUE)
```

Arguments

spdf	input spatial data frame
size	size of each hex bin in meters, default set to 15K meters (15KM)
clip	Should the output be cliped to the input boundaries? Default is TRUE

Value

country hex polygon as feature class

Examples

```
## Not run:
 library(gisr)

 shp <- get_admin0(countries = "Nigeria")

 get_hexbins(shp, 10000)

## End(Not run)
```

get_nepolygons *Get Natural Earth Polygons*

Description

Get Natural Earth Polygons

Usage

```
get_nepolygons(
  scale = c("large", "small", "medium"),
  type = c("countries", "map_units", "sovereignty", "tiny_countries")
)
```

Arguments

scale	Scale of the map - options are 'large', 'small', 'medium'
type	country type - options are 'countries', 'map_units', 'sovereignty', 'tiny_countries'

Value

world countries as sf object

Examples

```
## Not run:
 library(gisr)

 shp_ne <- get_nepolygons()
 shp_ne <- get_nepolygons(scale = "large", type = "countries")

## End(Not run)
```

---

get_raster *Get Terrain Raster dataset*

Description

Get Terrain Raster dataset

Usage

```
get_raster(folderpath, name = NULL, rename = FALSE, ...)
```

Arguments

folderpath	Path to raster file, default will be `si_path('path_raster')`
name	Name of the raster file (with extension), default is set to terrain raster `SR_LR.tif`
rename	Should the RasterLayer be renamed? If yes, the name is changed to `value`
...	Additional arguments to be passed to `base::list.files`. Eg: `Use ignore.case = TRUE` for non case sensitive search

Value

RasterLayer

Examples

```
## Not run:
 library(glamr)
 library(gisr)

 get_raster()

 get_raster(name = "sample.tif")
 get_raster(path = "./geodata/raster", name = "sample.tif")

## End(Not run)
```

`get_terrain` *Get terrain data for an AOI (Countries)*

Description

Get terrain data for an AOI (Countries)

Usage

```
get_terrain(countries, terr, mask = FALSE, buffer = 0.1)
```

Arguments

`countries`	List of the country names or sf object
`terr`	RasterLayer or Path to terrain raster file
`mask`	Should the extracted data match the exact boundary limits?
`buffer`	Extend AOI extent by x

Value

`spdf spatial dataframe`

Note

`get_terrain()` will evantually be replaced by `extract_raster()`

Examples

```
## Not run:
 library(gisr)
 library(sf)

 get_terrain(countries = list("Zambia"))
 get_terrain(countries = list("Zambia"), mask = TRUE)
 get_terrain(countries = list("Zambia"), buffer = .5, terr = "../../HDX_Data")
```

```
## End(Not run)
```

`get_vcpolygons` *Get PEPFAR Visual Crossing Polygons*

Description

Get PEPFAR Visual Crossing Polygons

Usage

```
get_vcpolygons(folderpath, name = NULL)
```

Arguments

`folderpath`	Path to PEPFAR Global Shapefile
`name`	Name or pattern of shapefile

Value

sf object

Examples

```
## Not run:
 library(gisr)

 shp_pepfar <- get_vcpolygons()
 shp_pepfar <- get_vcpolygons(folderpath = glamr::si_path("path_vector"))
 shp_pepfar <- get_vcpolygons(folderpath = "./GIS", name = "VcPepfarPolygons.shp")

## End(Not run)
```

`gview` *Plot sf features*

Description

Plot sf features

Usage

```
gview(geodata, ...)
```

Arguments

geodata	spatial data as sf object
...	arguments passed to geom_sf

Examples

```
## Not run:
 library(gisr)

 adm0 <- get_admin0("Ghana")

 adm0 %>%
  dplyr::select(name) %>%
  gview()

## End(Not run)
```

---

spdf_export *Export spatial data as shapefile with flags*

Description

Export spatial data as shapefile with flags

Usage

```
spdf_export(spdf, name)
```

Arguments

spdf	sf object
name	filename with full path

Value

boolean

Examples

```
## Not run:
 library(gisr)
 library(sf)

 shp <- get_admin0(countries = "Nigeria")

 export_spdf(spdf = shp, name = "./GIS/nga_country_boundaries")
 export_spdf(spdf = shp, name = "./GIS/nga_country_boundaries.shp")
```

```
## End(Not run)
```

---

spdf_points *Generate Point Spatial DataFrame*

Description

Generate Point Spatial DataFrame

Usage

```
spdf_points(.data, lat = "latitude", long = "longitude", crs = 4326)
```

Arguments

.data	Location data as a data frame, use extract_facilities
lat	Column name for latitude, default value is latitude
long	Column name for longitude, default value is longitude
crs	Coordinate Reference System, default value is EPSG Code for WGS 1984

Value

list of spatial data frames

Examples

```
## Not run:

cntry <- "Ethiopia"
level_fac <- grabr::get_ouorglevel(operatingunit = cntry, org_type = "facility")
df_facs <- extract_locations(country = cntry, level = level_fac)
df_facs <- df_facs %>% extract_facilities()
df_locs <- df_facs %>% select(-c(geom_type:nested))

spdf <- spdf_points(.data = df_locs)

## End(Not run)
```

terrain_map *Generate a terrain map*

Description

Generate a terrain map

Usage

```
terrain_map(
  countries,
  adm0 = NULL,
  adm1 = NULL,
  add_neighbors = FALSE,
  add_labels = FALSE,
  mask = FALSE,
  terr = NULL
)
```

Arguments

countries	List of countries to map
adm0	Admin 0 boundaries, optional sf geodata
adm1	Admin 1 boundaries, optional sf geodata
add_neighbors	Should the map include the neighbor countries
add_labels	Add neighbors countries's names (works only when add_neighbors is TRUE)
mask	Should the extracted data match the exact boundary limits?
terr	RasterLayer dataset or Path for terrain raster file

Value

ggplot basemap

Examples

```
## Not run:
library(gisr)
gisr::terrain_map(countries = list("Zambia"))
gisr::terrain_map(countries = list("Zambia"), add_neighbors = TRUE)

## End(Not run)
```

`zip_shapefiles` *Compress all shapefile components into a zipped file*

Description

Compress all shapefile components into a zipped file

Usage

```
zip_shapefiles(filename, folderpath = NULL)
```

Arguments

`filename`	Shapefile full path and name
`folderpath`	Where to place the zipped files

Value

`Boolean`

Index