Package ‘malariaAtlas’

October 13, 2022

Title An R Interface to Open-Access Malaria Data, Hosted by the 'Malaria Atlas Project'

Version 1.0.1

Description A suite of tools to allow you to download all publicly available parasite rate survey points, mosquito occurrence points and raster surfaces from the 'Malaria Atlas Project' <`https://malariaatlas.org/`> servers as well as utility functions for plotting the downloaded data.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports curl, rgdal, raster, sp, xml2, grid, gridExtra, httr, dplyr, stringi, tidyr, methods, stats, utils, rlang

Depends ggplot2

RoxygenNote 7.0.2

Suggests testthat, knitr, rmarkdown, palettetown, magrittr, tibble, rdhs

URL `https://github.com/malaria-atlas-project/malariaAtlas`

BugReports `https://github.com/malaria-atlas-project/malariaAtlas/issues`

VignetteBuilder knitr

NeedsCompilation no

Author Daniel Pfeffer [aut] (<`https://orcid.org/0000-0002-2204-3488`>),
Tim Lucas [aut, cre] (<`https://orcid.org/0000-0003-4694-8107`>),
Daniel May [aut] (<`https://orcid.org/0000-0003-0005-2452`>),
Suzanne Keddie [aut] (<`https://orcid.org/0000-0003-1254-7794`>),
Jen Rozier [aut] (<`https://orcid.org/0000-0002-2610-7557`>),
Oliver Watson [aut] (<`https://orcid.org/0000-0003-2374-0741`>),
Harry Gibson [aut] (<`https://orcid.org/0000-0001-6779-3250`>),
Nick Golding [ctb],
David Smith [ctb]

Maintainer Tim Lucas <`timcdlucas@gmail.com`>

Repository CRAN

Date/Publication 2020-06-01 20:30:11 UTC

R topics documented:

`as.MAPraster` *Convert Raster objects into MAPraster objects*

Description

`as.MAPraster` converts a RasterLayer or RasterStack object into a 'MAPraster' object (data.frame) for easy plotting with ggplot.

Usage

```
as.MAPraster(raster_object)
```

Arguments

`raster_object`	RasterLayer or Rasterstack object to convert into a MAPraster.

Value

`as.MAPraster` returns a MAPraster object (data.frame) containing the below columns.

1. `x` - x coordinates of raster pixels
2. `y` - y coordinates of raster pixels
3. `z` - value of raster pixels
4. `raster_name` - name of raster for which values are stored in z

See Also

`getRaster`:

to download rasters directly from MAP.

`as.MAPraster`:

to convert RasterLayer/RasterStack objects into a 'MAPraster' object (data.frame) for easy plotting with ggplot.

`autoplot.MAPraster`:

to quickly visualise MAPraster objects created using `as.MAPraster`.

Examples

```
# Download PfPR2-10 Raster for Madagascar in 2015 and visualise this on a map.

MDG_shp <- getShp(ISO = "MDG", admin_level = "admin0")
MDG_PfPR2_10 <- getRaster(surface = "Plasmodium falciparum PR2-10", shp = MDG_shp, year = 2015)
MDG_PfPR2_10 <- as.MAPraster(MDG_PfPR2_10)
autoplot(MDG_PfPR2_10)

#Download global raster of G6PD deficiency from Howes et al 2012 and visualise this on a map.

G6PDd_global <- getRaster(surface = "G6PD Deficiency Allele Frequency")
G6PDd_global <- as.MAPraster(G6PDd_global)
autoplot(G6PDd_global)
```

`as.MAPshp` *Convert SpatialPolygon objects into MAPshp objects*

Description

`as.MAPshp` converts a SpatialPolygon or SpatialPolygonsDataframe object downloaded using getShp into a 'MAPshp object (data.frame) for easy plotting with ggplot.

Usage

```
as.MAPshp(object)
```

Arguments

`object`	SpatialPolygon or SpatialPolygonsDataframe object to convert into a 'MAPshp'.

Value

`as.MAPshp` returns a MAPshp object (data.frame) containing the below columns.

1. `country_id` ISO-3 code of given administrative unit (or the ISO code of parent unit for administrative-level 1 units).
2. `gaul_code` GAUL code of given administrative unit.
3. `admn_level` administrative level of the given administrative unit - either 0 (national) or 1 (first-level division)
4. `parent_id` GAUL code of parent administrative unit of a given polygon (for admin0 polygons, PARENT_ID = 0).
5. `country_level` composite `country_id_admn_level` field.

See Also

`autoplot.MAPshp`

to download rasters directly from MAP.

Examples

```
#Download shapefiles for Madagascar and visualise these on a map.


MDG_shp <- getShp(ISO = "MDG", admin_level = "admin0")
MDG_shp <- as.MAPshp(MDG_shp)
autoplot(MDG_shp)
```

`as.pr.points` *Convert data.frames to pr.points objects.*

Description

Will create empty columns for any missing columns expected in a pr.points data.frame. This function is particularly useful for use with packages like dplyr that strip objects of their classes.

Usage

```
as.pr.points(x)
```

Arguments

`x`	A data.frame

Examples

```
#Download PfPR data for Nigeria and Cameroon and map the locations of these points using autoplot

library(dplyr)
NGA_CMR_PR <- getPR(country = c("Nigeria", "Cameroon"), species = "Pf")

# Filter the data frame then readd pr.points class so that autoplot can be used.
NGA_CMR_PR %>%
  filter(year_start > 2010) %>%
  as.pr.points %>%
  autoplot
```

`as.vectorpoints` *Convert data.frames to vector.points objects.*

Description

Will create empty columns for any missing columns expected in a vector.points data.frame. This function is particularly useful for use with packages like dplyr that strip objects of their classes.

Usage

```
as.vectorpoints(x)
```

Arguments

`x`	A data.frame

Examples

```

library(dplyr)
Brazil_vec <- getVecOcc(country = "Brazil")

# Filter data.frame then readd vector points class so autoplot can be used.
Brazil_vec %>%
  filter(sample_method1 == 'larval collection') %>%
  as.vectorpoints %>%
  autoplot
```

`autoplot.MAPraster` *Quickly visualise Rasters downloaded from MAP*

Description

`autoplot.MAPraster` creates a map of all rasters in a MAPraster object and displays these in a grid.

Usage

```
## S3 method for class 'MAPraster'
autoplot(
  object,
  ...,
  shp_df = NULL,
  legend_title = "",
  plot_titles = TRUE,
  fill_scale_transform = "identity",
  fill_colour_palette = "RdYlBu",
  printed = TRUE
)
```

Arguments

`object`	MAPraster object to be visualised.
`...`	Other arguments passed to specific methods
`shp_df`	Shapefile(s) (data.frame) to plot with downloaded raster.
`legend_title`	String used as title for all colour scale legends.
`plot_titles`	Plot name of raster object as header for each individual raster plot?
`fill_scale_transform`	String givning a transformation for the fill aesthetic. See the trans argument in `continuous_scale` for possible values.
`fill_colour_palette`	String referring to a colorbrewer palette to be used for raster colour scale.
`printed`	Logical vector indicating whether to print maps of supplied rasters.

Value

`autoplot.MAPraster` returns a list of plots (gg objects) for each supplied raster.

See Also

`getRaster`:

to download rasters directly from MAP.

`as.MAPraster`:

to convert RasterLayer/RasterStack objects into a ’MAPraster’ object (data.frame) for easy plotting with ggplot.

`autoplot.MAPraster`:

to quickly visualise MAPraster objects created using /codeas.MAPraster.

Examples

```
# Download PfPR2-10 Raster (Bhatt et al 2015) and raw survey points
#   for Madagascar in 2013 and visualise these together on a map.

# Download madagascar shapefile to use for raster download.
MDG_shp <- getShp(ISO = "MDG", admin_level = "admin0")

# Download PfPR2-10 Raster for 2013 & plot this
MDG_PfPR2_10 <- getRaster(surface = "Plasmodium falciparum PR2-10",
                          shp = MDG_shp, year = 2013)
p <- autoplot_MAPraster(MDG_PfPR2_10, shp_df = MDG_shp)

# Download raw PfPR survey points & plot these over the top of the raster
pr <- getPR(country = c("Madagascar"), species = "Pf")
p[[1]] +
geom_point(data = pr[pr$year_start==2013,],
           aes(longitude, latitude, fill = positive / examined,
               size = examined), shape = 21) +
  scale_size_continuous(name = "Survey Size") +
  scale_fill_distiller(name = "PfPR", palette = "RdYlBu") +
  ggtitle("Raw PfPR Survey points\n +
          Modelled PfPR 2-10 in Madagascar in 2013")


# Download global raster of G6PD deficiency (Howes et al 2012) and visualise this on a map.
G6PDd_global <- getRaster(surface = "G6PD Deficiency Allele Frequency")
autoplot_MAPraster(G6PDd_global)
```

`autoplot.MAPshp` *Create a basic plot to visualise downloaded shapefiles*

Description

`autoplot.MAPshp` creates a map of shapefiles downloaded using getShp.

Usage

```
## S3 method for class 'MAPshp'
autoplot(object, ..., map_title = NULL, facet = FALSE, printed = TRUE)
```

Arguments

`object`	A MAPshp object downloaded using /code/linkgetShp with format = "df" specified.
`...`	Other arguments passed to specific methods
`map_title`	Custom title used for the plot.
`facet`	If TRUE, splits map into a separate facet for each administrative level.
`printed`	Should the plot print to graphics device.

Value

`autoplot.MAPshp` returns a map (gg object) of the supplied MAPShp dataframe.

Examples

```
MDG_shp <- getShp(ISO = "MDG", admin_level = "admin0")
autoplot(as.MAPshp(MDG_shp))
```

`autoplot.pr.points` *Create a basic plot showing locations of downloaded PR points*

Description

`autoplot.pr.points` creates a map of PR points downloaded from MAP.

Usage

```
## S3 method for class 'pr.points'
autoplot(
  object,
  ...,
  shp_df = NULL,
  admin_level = "admin0",
  map_title = "PR Survey Locations",
  fill_legend_title = "Raw PR",
  fill_scale_transform = "identity",
  facet = NULL,
  hide_confidential = FALSE,
  printed = TRUE
)
```

Arguments

`object`	a pr.points object downloaded using /code/linkgetPR
`...`	Other arguments passed to specific methods
`shp_df`	Shapefile(s) (data.frame) to plot with downloaded points. (If not specified automatically uses getShp() for all countries included in pr.points object).
`admin_level`	the administrative level used for plotting administrative boundaries; either /code"admin0"; /code"admin1" OR /code"both"
`map_title`	custom title used for the plot
`fill_legend_title`	Add a title to the legend.
`fill_scale_transform`	String givning a transformation for the fill aesthetic. See the trans argument in `continuous_scale` for possible values.
`facet`	if TRUE, splits map into a separate facet for each malaria species; by default FALSE if only one species is present in object, TRUE if both P. falciparum and P. vivax data are present in object.
`hide_confidential`	if TRUE, removes confidential points from the map
`printed`	Should the plot be printed to the graphics device.

Value

`autoplot.pr.points` returns a plots (gg object) for the supplied pr.points dataframe.

Examples

```
PfPR_surveys_NGA <- getPR(country = c("Nigeria"), species = "Pf")
autoplot(PfPR_surveys_NGA)

# Download PfPR2-10 Raster (Bhatt et al. 2015) and raw survey points for Madagascar in
#   2013 and visualise these together on a map.

# Download madagascar shapefile to use for raster download.
MDG_shp <- getShp(ISO = "MDG", admin_level = "admin0")

# Download PfPR2-10 Raster for 2013 & plot this
MDG_PfPR2_10 <- getRaster(surface = "Plasmodium falciparum PR2-10", shp = MDG_shp, year = 2013)
p <- autoplot_MAPraster(MDG_PfPR2_10)

# Download raw PfPR survey points & plot these over the top of the raster
pr <- getPR(country = c("Madagascar"), species = "Pf")
p[[1]] +
geom_point(data = pr[pr$year_start==2013,],
           aes(longitude, latitude, fill = positive / examined,
               size = examined), shape = 21) +
  scale_size_continuous(name = "Survey Size") +
  scale_fill_distiller(name = "PfPR", palette = "RdYlBu") +
```

```
  ggtitle("Raw PfPR Survey points\n + Modelled PfPR 2-10 in Madagascar in 2013")
```

---

autoplot.vector.points

Create a basic plot showing locations of downloaded Vector points

---

Description

`autoplot.vector.points` creates a map of Vector points downloaded from MAP.

Usage

```
## S3 method for class 'vector.points'
autoplot(
  object,
  ...,
  shp_df = NULL,
  admin_level = "admin0",
  map_title = "Vector Survey Locations",
  fill_legend_title = "Raw Vetor Occurrences",
  fill_scale_transform = "identity",
  facet = NULL,
  printed = TRUE
)
```

Arguments

`object`	a vector.points object downloaded using /code/linkgetVecOcc
`...`	Other arguments passed to specific methods
`shp_df`	Shapefile(s) (data.frame) to plot with downloaded points. (If not specified automatically uses getShp() for all countries included in vector.points object).
`admin_level`	the administrative level used for plotting administrative boundaries; either /code"admin0"; /code"admin1" OR /code"both"
`map_title`	custom title used for the plot
`fill_legend_title`	Add a title to the legend.
`fill_scale_transform`	String givning a transformation for the fill aesthetic. See the trans argument in `continuous_scale` for possible values.
`facet`	if TRUE, splits map into a separate facet for each malaria species; by default FALSE.
`printed`	Should the plot be printed to the graphics device.

Value

`autoplot.vector.points` returns a plots (gg object) for the supplied vector.points dataframe.

Examples

```
Vector_surveys_NGA_NG <- getVecOcc(country = c("Nigeria", "Niger"))
autoplot(Vector_surveys_NGA_NG)

# Download the predicted distribution of An. dirus species complex Raster and
#  vector points for Myanmar and visualise these together on a map.

# Download Myanmar shapefile to use for raster download.
MMR_shp <- getShp(ISO = "MMR", admin_level = "admin0")
MMR_shp_df <- as.MAPshp(MMR_shp)

# Download An. dirus predicted distribution Raster & plot this
MMR_An_dirus <- getRaster(surface = "Anopheles dirus species complex", shp = MMR_shp)
MMR_An_dirus_df <- as.MAPraster(MMR_An_dirus)
p <- autoplot_MAPraster(MMR_An_dirus, shp_df = MMR_shp_df, printed = FALSE)

# Download raw occurrence points & plot these over the top of the raster
species <- getVecOcc(country = "Myanmar", species = "Anopheles dirus")
p[[1]] +
geom_point(data = species,
 aes(longitude,
  latitude,
  colour = species))+
  scale_colour_manual(values = "black", name = "Vector suvery locations")+
scale_fill_distiller(name = "Predicted distribution of An. dirus complex",
 palette = "PuBuGn",
  direction = 1)+
  ggtitle("Vector Survey points\n + The predicted distribution of An. dirus complex")
```

`autoplot_MAPraster` *Quickly visualise Rasters downloaded from MAP*

Description

`autoplot_MAPraster` is a wrapper for /code/linkautoplot.MAPraster that calls `as.MAPraster` to allow automatic map creation for RasterLayer/RasterStack objects downloaded from MAP.

Usage

```
autoplot_MAPraster(object, ...)
```

Arguments

`object`	RasterLayer/RasterStack object to be visualised.
`...`	other optional arguments to autoplot.MAPraster (e.g. shp_df, legend_title, page_title...)

Value

`autoplot_MAPraster` returns a list of plots (gg objects) for each supplied raster.

See Also

`getRaster`:

to download rasters directly from MAP.

`as.MAPraster`:

to convert RasterLayer/RasterStack objects into a 'MAPraster' object (data.frame) for easy plotting with ggplot.

`autoplot.MAPraster`:

to quickly visualise MAPraster objects created using /codeas.MAPraster.

Examples

```
#Download PfPR2-10 Raster (Bhatt et al 2015) and raw survey points for Madagascar in
#  2013 and visualise these together on a map.

# Download madagascar shapefile to use for raster download.
MDG_shp <- getShp(ISO = "MDG", admin_level = "admin0")

# Download PfPR2-10 Raster for 2013 & plot this
MDG_PfPR2_10 <- getRaster(surface = "Plasmodium falciparum PR2-10", shp = MDG_shp, year = 2013)
p <- autoplot_MAPraster(MDG_PfPR2_10)

# Download raw PfPR survey points & plot these over the top of the raster
pr <- getPR(country = c("Madagascar"), species = "Pf")
p[[1]] +
geom_point(data = pr[pr$year_start==2013,],
        aes(longitude, latitude, fill = positive / examined, size = examined), shape = 21) +
  scale_size_continuous(name = "Survey Size") +
  scale_fill_distiller(name = "PfPR", palette = "RdYlBu") +
  ggtitle("Raw PfPR Survey points\n + Modelled PfPR 2-10 in Madagascar in 2013")


# Download global raster of G6PD deficiency (Howes et al 2012) and visualise this on a map.

G6PDd_global <- getRaster(surface = "G6PD Deficiency Allele Frequency")
autoplot_MAPraster(G6PDd_global)
```

`convertPrevalence` *convert prevalences from one age range to another*

Description

convert prevalences from one age range to another

Usage

```
convertPrevalence(
  prevalence,
  age_min_in,
  age_max_in,
  age_min_out = rep(2, length(prevalence)),
  age_max_out = rep(9, length(prevalence)),
  parameters = "Pf_Smith2007",
  sample_weights = NULL
)
```

Arguments

`prevalence`	Vector of prevalence values
`age_min_in`	Vector of minimum ages sampled
`age_max_in`	Vector maximum ages sampled.
`age_min_out`	Length 1 numeric or vector of same length as prevalence given the required age range upper bound
`age_max_out`	Length 1 numeric or vector of same length as prevalence given the required age range lower bound
`parameters`	Specifies the set of parameters to use in the model. This can either be "Pf_Smith2007" to use the parameters for *Plasmodium falciparum* defined in that paper, "Pv_Gething2012" for the *P. vivax* parameters used in that paper, or a user-specified vector givng the values of parameters ‘b‘, ‘s‘, ‘c‘ and ‘alpha‘, in that order. If specified,
`sample_weights`	Must be a vector of length 85 giving the sample weights for each age category (the proportion of the population in that age category) . If ‘NULL‘, The sample weights used in Smith et al. 2007 are used.

References

Smith, D. L. et al. Standardizing estimates of the Plasmodium falciparum parasite rate. Malaria Journal 6, 131 (2007).

Gething, Peter W., et al. "A long neglected world malaria map: Plasmodium vivax endemicity in 2010." PLoS neglected tropical diseases 6.9 (2012): e1814.

Code written by Nick Golding and Dave Smith

Examples

```
# Convert from prevalence 2 to 5 to prevalence 2 to 10
pr2_10 <- convertPrevalence(0.1, 2, 5, 2, 10)

# Convert many surveys all to 2 to 10.
pr <- runif(20, 0.1, 0.15)
min_in <- sample(1:5, 20, replace = TRUE)
max_in <- rep(8, 20)
min_out <- rep(2, 20)
max_out <- rep(10, 20)
pr_standardised <- convertPrevalence(pr, min_in, max_in,
                                     min_out, max_out)

plot(pr_standardised, pr)
```

`extractRaster` *Extract pixel values from MAP rasters using point coordinates.*

Description

`extractRaster` extracts pixel values from MAP rasters at user-specified point locations (without downloading the entire raster).

Usage

```
extractRaster(
  df = NULL,
  csv_path = NULL,
  surface = "PfPR2-10",
  year = rep(NA, length(surface))
)
```

Arguments

`df`	data.frame containing coordinates of input point locations, must contain columns named 'latitude'/'lat'/'x' AND 'longitude'/'long'/'y')
`csv_path`	(optional) user-specified path to which extractRaster coordinates and results are stored. If not specified, tempdir() is used instead.
`surface`	string containing 'title' of desired raster(s), e.g. `c("raster1", "raster2")`. Defaults to "PfPR2-10" - the most recent global raster of PfPR 2-10. Check `listRaster` to find titles of available rasters.
`year`	default = `rep(NA, length(surface))`; for time-varying rasters: if downloading a single surface for one or more years, `year` should be a vector specifying the desired year(s). if downloading more than one surface, use a list the same length as `surface`, providing the desired year-range for each time-varying surface in `surface` or `NA` for static rasters.

Value

`getPR` returns the input dataframe (`df`), with the following columns apprended, providing raster values for each surface, location and year.

1. `layer` raster code corresponding to extracted raster values for a given row, check `listRaster` for raster metadata.
2. `year` the year for which raster values were extraced (time-varying rasters only; static rasters do not have this column).
3. `value` the raster value for the pixel in which a given point location falls.

See Also

`autoplot` method for quick mapping of PR point locations (`autoplot.pr.points`).

Examples

```
#Download PfPR data for Nigeria and Cameroon and map the locations of these points using autoplot

# Get some data and remove rows with NAs in location or examined or positive columns.
NGA_CMR_PR <- getPR(country = c("Nigeria", "Cameroon"), species = "Pf")
complete <- complete.cases(NGA_CMR_PR[, c(4, 5, 16, 17)])
NGA_CMR_PR <- NGA_CMR_PR[complete, ]

# Extract PfPR data at those locations.
data <- extractRaster(NGA_CMR_PR[, c('latitude', 'longitude')],
                      surface = 'Plasmodium falciparum PR2-10',
                      year = 2015)

# Data are returned in the same order.
all(data$longitude == NGA_CMR_PR$longitude)

# Some rasters are stored with NA encoded as -9999
data$value[data$value == -9999] <- NA

# We can quickly plot a summary
plot((NGA_CMR_PR$positive / NGA_CMR_PR$examined) ~ data$value)
```

`fillDHSCoordinates` *Add DHS locations to malaria data*

Description

We cannot directly share DHS data. We can share coordinates, but not the data values. This function attempts to fill the data gaps directly from the DHS server using the package rdhs. To use the function you will need to setup an account on the DHS website and request any datasets you wish to use (including requesting the GPS data). Confirmation can take a few days. Once this has been verified, you should be able to use this function.

Usage

```
fillDHSCoordinates(
  data,
  email = NULL,
  project = NULL,
  cache_path = NULL,
  config_path = NULL,
  global = TRUE,
  verbose_download = FALSE,
  verbose_setup = TRUE,
  data_frame = NULL,
  timeout = 30,
  password_prompt = FALSE,
  prompt = TRUE
)
```

Arguments

`data`	Data to add DHS coordinates to
`email`	Character for email used to login to the DHS website.
`project`	Character for the name of the DHS project from which datasets should be downloaded.
`cache_path`	Character for directory path where datasets and API calls will be cached. If left bank, a suitable directory will be created within your user cache directory for your operating system (permission granting).
`config_path`	Character for where the config file should be saved. For a global configuration, ‘config_path‘ must be ’~/.rdhs.json’. For a local configuration, ‘config_path‘ must be ’rdhs.json’. If left bank, the config file will be stored within your user cache directory for your operating system (permission granting).
`global`	Logical for the config_path to be interpreted as a global config path or a local one. Default = TRUE.
`verbose_download`	Logical for dataset download progress bars to be shown. Default = FALSE.
`verbose_setup`	Logical for rdhs setup and messages to be printed. Default = TRUE.
`data_frame`	Function with which to convert API calls into. If left blank `data_frame` objects are returned. Must be passed as a character. Examples could be: `data.table::as.data.table` `tibble::as.tibble`
`timeout`	Numeric for how long in seconds to wait for the DHS API to respond. Default = 30.
`password_prompt`	Logical whether user is asked to type their password, even if they have previously set it. Default = FALSE. Set to TRUE if you have mistyped your password when using `set_rdhs_config`.
`prompt`	Logical for whether the user should be prompted for permission to write to files. This should not need be

Details

This function requires the package `rdhs` which is currently only suggested by the package (not a dependency). So you will need to install it.

Note that the `project` has to be the exact name in your DHS project.

Author(s)

OJ Watson

Examples

```
## Not run:
pf <- malariaAtlas::getPR("all",species = "pf")
pf <- fillDHSCoordinates(pf,
email = "pretend_email@emaildomain.com",
project = "pretend project name")

## End(Not run)
```

`getPR` *Download PR points from the MAP database*

Description

`getPR` downloads all publicly available PR points for a specified country (or countries) and returns this as a dataframe.

Usage

```
getPR(country = NULL, ISO = NULL, continent = NULL, species, extent = NULL)
```

Arguments

`country`	string containing name of desired country, e.g. `c("Country1", "Country2", ...)` OR = `"ALL"`. (Use one of `country` OR `ISO` OR `continent`, not combined)
`ISO`	string containing ISO3 code for desired country, e.g. `c("XXX", "YYY", ...)` OR = `"ALL"`. (Use one of `country` OR `ISO` OR `continent`, not combined)
`continent`	string containing continent (one of "Africa", "Americas", "Asia", "Oceania") for desired data, e.g. `c("continent1", "continent2", ...)`. (Use one of `country` OR `ISO` OR `continent`, not combined)
`species`	string specifying the Plasmodium species for which to find PR points, options include: `"Pf"` OR `"Pv"` OR `"BOTH"`
`extent`	2x2 matrix specifying the spatial extent within which PR data is desired, as returned by sp::bbox() - the first column has the minimum, the second the maximum values; rows 1 & 2 represent the x & y dimensions respectively (matrix(c("xmin", "ymin","xmax", "ymax"), nrow = 2, ncol = 2, dimnames = list(c("x", "y"), c("min", "max"))))

Details

`country` and `ISO` refer to countries and a lower-level administrative regions such as Mayotte and French Guiana. While we cannot direectly distribute DHS coordinates, we can distribute the number of examined and positive. If the coordinates are needed they can be downloaded from www.measuredhs.com, via the rdhs package or using malariaAtlas:fillDHSCoordinates().

Value

`getPR` returns a dataframe containing the below columns, in which each row represents a distinct data point/ study site.

1. `dhs_id` The dhs survey id if appropriate.
2. `site_id` Unique site identifier
3. `site_name` Name of site.

See Also

`autoplot` method for quick mapping of PR point locations (`autoplot.pr.points`).

Examples

```
#Download PfPR data for Nigeria and Cameroon and map the locations of these points using autoplot

NGA_CMR_PR <- getPR(country = c("Nigeria", "Cameroon"), species = "Pf")
autoplot(NGA_CMR_PR)

#Download PfPR data for Madagascar and map the locations of these points using autoplot
Madagascar_pr <- getPR(ISO = "MDG", species = "Pv")
autoplot(Madagascar_pr)

getPR(country = "ALL", species = "BOTH")
```

`getRaster` *Download Rasters produced by the Malaria Atlas Project*

Description

`getRaster` downloads publicly available MAP rasters for a specific surface & year, clipped to a provided bounding box or shapefile.

Usage

```
getRaster(
  surface = "Plasmodium falciparum PR2-10",
  shp = NULL,
  extent = NULL,
  file_path = tempdir(),
  year = as.list(rep(NA, length(surface))),
  vector_year = NULL
)
```

Arguments

`surface`	string containing 'title' of desired raster(s), e.g. `c("raster1", "raster2")`. Defaults to "PfPR2-10" - the most recent global raster of PfPR 2-10. Check `listRaster` to find titles of available rasters.
`shp`	SpatialPolygon(s) object of a shapefile to use when clipping downloaded rasters. (use either `shp` OR `extent`; if neither is specified global raster is returned).
`extent`	2x2 matrix specifying the spatial extent within which raster data is desired, as returned by sp::bbox() - the first column has the minimum, the second the maximum values; rows 1 & 2 represent the x & y dimensions respectively (matrix(c("xmin", "ymin","xmax", "ymax"), nrow = 2, ncol = 2, dimnames = list(c("x", "y"), c("min", "max")))) (use either `shp` OR `extent`; if neither is specified global raster is returned).
`file_path`	string specifying the directory to which working files will be downloaded. Defaults to tempdir().
`year`	default = `rep(NA, length(surface))` (use `NA` for static rasters); for time-varying rasters: if downloading a single surface for one or more years, `year` should be a string specifying the desired year(s). if downloading more than one surface, use a list the same length as `surface`, providing the desired year-range for each time-varying surface in `surface` or `NA` for static rasters.
`vector_year`	default = `NULL` for vector occurence rasters, the desired version as prefixed in raster_code returned using available_rasters. By default (`NULL`) returns the most recent raster version)

Value

`getRaster` returns a RasterLayer (if only a single raster is queried) or RasterStack (for multiple rasters) for the specified extent.

See Also

`autoplot_MAPraster`

to quickly visualise rasters downloded using `getRaster`.

`as.MAPraster`

to convert RasterLayer/RasterStack objects into a 'MAPraster' object (data.frame) for easy plotting with ggplot.

`autoplot.MAPraster`

to quickly visualise MAPraster objects created using /codeas.MAPraster.

Examples

```
# Download PfPR2-10 Raster for Madagascar in 2015 and visualise this immediately.

MDG_shp <- getShp(ISO = "MDG", admin_level = "admin0")
MDG_PfPR2_10 <- getRaster(surface = "Plasmodium falciparum PR2-10", shp = MDG_shp, year = 2015)
autoplot_MAPraster(MDG_PfPR2_10)

# Download global raster of G6PD deficiency from Howes et al 2012.
G6PDd_global <- getRaster(surface = "G6PD Deficiency Allele Frequency")
autoplot_MAPraster(G6PDd_global)

# Download a temporal raster by range
MDG_PfPR2_10_range <- getRaster(surface = "Plasmodium falciparum PR2-10",
                                shp = MDG_shp, year = 2012:2015)

# Download a mix of rasters
MDG_rasters <- getRaster(surface = c("Plasmodium falciparum PR2-10",
                                     'Plasmodium falciparum Incidence',
                                     'Plasmodium falciparum Support'),
                         shp = MDG_shp, year = list(2009:2012, 2005:2007, NA))
p <- autoplot_MAPraster(MDG_rasters)
```

---

getShp *Download MAPadmin2013 Administrative Boundary Shapefiles from the MAP geoserver*

Description

getShp downloads a shapefile for a specified country (or countries) and returns this as either a spatialPolygon or data.frame object.

Usage

```
getShp(
  country = NULL,
  ISO = NULL,
  extent = NULL,
  admin_level = c("admin0"),
  format = "spatialpolygon",
  long = NULL,
  lat = NULL
)
```

Arguments

`country`	string containing name of desired country, e.g. `c("Country1", "Country2", ...)` OR = `"ALL"` (use either `ISO` OR `country`)
`ISO`	string containing ISO3 code for desired country, e.g. `c("XXX", "YYY", ...)` OR = `"ALL"` (use either `ISO` OR `country`)
`extent`	2x2 matrix specifying the spatial extent within which polygons are desired, as returned by sp::bbox() - the first column has the minimum, the second the maximum values; rows 1 & 2 represent the x & y dimensions respectively (matrix(c("xmin", "ymin","xmax", "ymax"), nrow = 2, ncol = 2, dimnames = list(c("x", "y"), c("min", "max")))). Note: getShp downloads the entire polygon for any polygons falling within the extent.
`admin_level`	string specifying the administrative level for which shapefile are desired (only "admin0","admin1","admin2","admin3", or "all" accepted). N.B. Not all administrative levels are available for all countries. Use listShp to check which shapefiles are available. If an administrative level is requested that is not available, the closest available administrative level shapefiles will be returned.
`format`	string specifying the desired format for the downloaded shapefile: either "spatialpolygon" or "df"
`long`	longitude of a point location falling within the desired shapefile.
`lat`	latitude of a point location falling within the desired shapefile.

Value

`getShp` returns either a dataframe or spatialPolygon object for requested administrative unit polygons. The following attribute fields are included:

1. `id` unique identifier for any given polygon/administrative unit.
2. `iso` ISO-3 code of given administrative unit (or the ISO code of parent unit for administrative-level 1 units).
3. `admn_level` administrative level of the given administrative unit - either 0 (national), 1 (first-level division), 2 (second-level division), 3 (third-level division).
4. `name_0` name of admin0 parent of a given administrative unit (or just shapefile name for admin0 units)
5. `id_0` id code of admin0 parent of the current shapefile (or just shapefile id for admin0 units)
6. `type_0` if applicable, type of administrative unit or admin0 parent
7. `name_1` name of admin1 parent of a given administrative unit (or just shapefile name for admin1 units); NA for admin0 units
8. `id_1` id code of admin1 parent of the current shapefile (or just shapefile id for admin1 units); NA for admin0 units
9. `type_1` if applicable, type of administrative unit or admin1 parent
10. `name_2` name of admin2 parent of a given administrative unit (or just shapefile name for admin2 units); NA for admin0, admin1 units
11. `id_2` id code of admin2 parent of the current shapefile (or just shapefile id for admin2 units); NA for admin0, admin1 units

12. `type_2` if applicable, type of administrative unit or admin2 parent
13. `name_3` name of admin3 parent of a given administrative unit (or just shapefile name for admin3 units); NA for admin0, admin1, admin2 units
14. `id_3` id code of admin3 parent of the current shapefile (or just shapefile id for admin3 units); NA for admin0, admin1, admin2 units
15. `type_3` if applicable, type of administrative unit
16. `source` source of administrative boundaries
17. `country_level` composite `iso_admn_level` field.

See Also

`autoplot` method for quick mapping of PR point locations (`autoplot.pr.points`).

Examples

```
#Download PfPR data & associated shapefiles for Nigeria and Cameroon

NGA_CMR_PR <- getPR(country = c("Nigeria", "Cameroon"), species = "Pf")
NGA_CMR_shp <- getShp(country = c("Nigeria", "Cameroon"))

#Download PfPR data & associated shapefiles for Chad
Chad_PR <- getPR(ISO = "TCD", species = "both")
Chad_shp <- getShp(ISO = "TCD")

#' #Download PfPR data & associated shapefiles defined by extent for Madagascar
MDG_PR <- getPR(country = "Madagascar", species = "Pv")
```

`getVecOcc` *Download Vector Occurrence points from the MAP database* `getVecOcc` *downloads all publicly available vector occurrence points for a specified country (or countries) and returns this as a dataframe.* `country` *and* `ISO` *refer to countries and a lower-level administrative regions such as French Guiana.*

Description

Download Vector Occurrence points from the MAP database `getVecOcc` downloads all publicly available vector occurrence points for a specified country (or countries) and returns this as a dataframe. `country` and `ISO` refer to countries and a lower-level administrative regions such as French Guiana.

Usage

```
getVecOcc(
  country = NULL,
  ISO = NULL,
  continent = NULL,
  species = "all",
  extent = NULL
)
```

Arguments

`country`	string containing name of desired country, e.g. `c("Country1", "Country2", ...)` OR = `"ALL"`. (Use one of `country` OR `ISO` OR `continent`, not combined)
`ISO`	string containing ISO3 code for desired country, e.g. `c("XXX", "YYY", ...)` OR = `"ALL"`. (Use one of `country` OR `ISO` OR `continent`, not combined)
`continent`	string containing continent (one of "Africa", "Americas", "Asia", "Oceania") for desired data, e.g. `c("continent1", "continent2", ...)`. (Use one of `country` OR `ISO` OR `continent`, not combined)
`species`	string specifying the Anopheles species for which to find vector occurrence points, options include: `"Anopheles...."` OR `"ALL"`
`extent`	2x2 matrix specifying the spatial extent within which vector occurrence data is desired, as returned by sp::bbox() - the first column has the minimum, the second the maximum values; rows 1 & 2 represent the x & y dimensions respectively (matrix(c("xmin", "ymin","xmax", "ymax"), nrow = 2, ncol = 2, dimnames = list(c("x", "y"), c("min", "max"))))

Value

`getVecOcc` returns a dataframe containing the below columns, in which each row represents a distinct data point/ study site.

1. `COLUMNNAME` description of contents
2. `COLUMNNAME` description of contents
3. `COLUMNNAME` description of contents

See Also

`autoplot` method for quick mapping of Vector occurrence point locations (`autoplot.vector.points`).

Examples

```
# Download vector occurrence data for Brazil and map the locations using autoplot.vector.points

Brazil_vec <- getVecOcc(country = "Brazil")
autoplot(Brazil_vec)

# Download vector data for Madagascar and map the locations using autoplot
Madagascar_vec <- getVecOcc(ISO = "MDG", species = "All")
```

```
autoplot(Madagascar_vec)

# Subset by extent.
extent_vec <- getVecOcc(extent = matrix(c(100,13,110,18), nrow = 2), species = 'all')
```

`isAvailable` *Available data to download from the MAP geoserver.*

Description

`isAvaiable` is a wrapper for isAvailable_pr and isAvailable_vec, listing data (PR survey point location data and vector occurrence locations available to download from the MAP geoserver.

Usage

```
isAvailable(
  sourcedata = NULL,
  full_results = FALSE,
  country = NULL,
  ISO = NULL,
  continent = NULL
)
```

Arguments

`sourcedata`	One of ’pr points’ or ’vector points’
`full_results`	Should the list be printed to the console?
`country`	string containing name of desired country, e.g. `c("Country1", "Country2", ...)` OR = `"ALL"` (use one of `country` OR `ISO` OR `continent`, not combined)
`ISO`	string containing ISO3 code for desired country, e.g. `c("XXX", "YYY", ...)` OR = `"ALL"` (use one of `country` OR `ISO` OR `continent`, not combined)
`continent`	string containing continent for desired data, e.g. `c("continent1", "continent2", ...)` (use one of `country` OR `ISO` OR `continent`, not combined)

Value

`isAvailable` returns a data.frame detailing the administrative units for which shapefiles are stored on the MAP geoserver.

See Also

`link{isAvailable_pr}` `isAvailable_vec`

Examples

```
available_pr_locations <- isAvailable_pr(ISO = 'IDN')
available_vector_locations <- isAvailable_vec(ISO = 'IDN')
```

`isAvailable_pr` *Check whether PR points are available for a given location*

Description

`isAvailable_pr` checks whether the MAP database contains PR points for the specified country/location.

Usage

```
isAvailable_pr(
  sourcedata = "pr points",
  country = NULL,
  ISO = NULL,
  continent = NULL,
  full_results = FALSE
)
```

Arguments

`sourcedata`	by default this is "pr points", highlighting the dataset to be searched
`country`	string containing name of desired country, e.g. `c("Country1", "Country2", ...)` (use one of `country` OR `ISO` OR `continent`, not combined)
`ISO`	string containing ISO3 code for desired country, e.g. `c("XXX", "YYY", ...)` (use one of `country` OR `ISO` OR `continent`, not combined)
`continent`	string containing name of continent for desired data, e.g. `c("Continent1", "Continent2", ...)`(use one of `country` OR `ISO` OR `continent`, not combined)
`full_results`	By default this is FALSE meaning the function only gives a message outlining whether specified country is available, if `full_results` == `TRUE`, the function returns a named list outlining data availability.

Value

`isAvailable_pr` returns a named list of input locations with information regarding data availability.

if `full_results` == `TRUE`, a named list is returned with the following elements:

1. `location` - specified input locations
2. `is_available`- 1 or 0; indicating whether data is available for this location
3. `possible_match`- agrep-matched country names indicating potential mispellings of countries where is_available == 0; NA if data is available for this location.

Examples

```
isAvailable_pr(country = "Suriname")
x <- isAvailable_pr(ISO = "NGA", full_results = TRUE)
x <- isAvailable_pr(continent = "Oceania", full_results = TRUE)
```

`isAvailable_vec` *Check whether Vector Occurrence points are available for a given location*

Description

`isAvailable_vec` checks whether the MAP database contains Vector Occurrence points for the specified country/location.

Usage

```
isAvailable_vec(
  sourcedata = "vector points",
  country = NULL,
  ISO = NULL,
  continent = NULL,
  full_results = FALSE
)
```

Arguments

`sourcedata`	by default this is "vector points", highlighting the dataset to be searched
`country`	string containing name of desired country, e.g. `c("Country1", "Country2", ...)` (use one of `country` OR `ISO` OR `continent`, not combined)
`ISO`	string containing ISO3 code for desired country, e.g. `c("XXX", "YYY", ...)` (use one of `country` OR `ISO` OR `continent`, not combined)
`continent`	string containing name of continent for desired data, e.g. `c("Continent1", "Continent2", ...)`(use one of `country` OR `ISO` OR `continent`, not combined)
`full_results`	By default this is FALSE meaning the function only gives a message outlining whether specified country is available, if `full_results` == `TRUE`, the function returns a named list outlining data availability.

Value

`isAvailable_Vec` returns a named list of input locations with information regarding data availability.

if `full_results` == `TRUE`, a named list is returned with the following elements:

1. `location` - specified input locations
2. `is_available`- 1 or 0; indicating whether data is available for this location
3. `possible_match`- agrep-matched country names indicating potential mispellings of countries where is_available == 0; NA if data is available for this location.

Examples

```
isAvailable_vec(country = "Suriname")
x <- isAvailable_vec(ISO = "NGA", full_results = TRUE)
x <- isAvailable_vec(continent = "Oceania", full_results = TRUE)
```

`listData` *List data available to download from the MAP geoserver.*

Description

`listData` is a wrapper for listPoints; listRaster and listShp, listing data (PR survey point data; raster data; shapefiles) available to download from the MAP geoserver.

Usage

```
listData(datatype, printed = TRUE, ...)
```

Arguments

`datatype`	One of 'pr points', 'vector points', 'raster' or 'shape'
`printed`	Should the list be printed to the console?
`...`	Other arguments to be passed to `list*` functions. (e.g. `admin_level` for `listShp`)

Value

`listData` returns a data.frame detailing the administrative units for which shapefiles are stored on the MAP geoserver.

See Also

`link{listPoints}` `listRaster` `listShp`

Examples

```
available_admin_units <- listShp()
available_pr_points <- listPoints(sourcedata = "pr points")
available_vector_points <- listPoints(sourcedata = "vector points")
available_rasters <- listRaster()
```

`listPoints` *List countries with available MAP Point data.*

Description

`listPoints` lists all countries for which there are publicly visible datapoints in the MAP database required.

Usage

```
listPoints(printed = TRUE, sourcedata)
```

Arguments

`printed`	Should the list be printed to the console?
`sourcedata`	String contining desired dataset within the Malaria Atlas database to be searched, e.g `"pr points"` OR `"vector points"`

Value

`listPoints` returns a data.frame detailing the countries for which PR points are publicly available.

Examples

```
listPoints(sourcedata = "pr points")
listPoints(sourcedata = "vector points")
```

`listRaster` *List all MAP Rasters available to download.*

Description

`listRaster` lists all rasters available to download from the Malaria Atlas Project database.

Usage

```
listRaster(printed = TRUE)
```

Arguments

`printed`	Should the list be printed to the console?

Value

`listRaster` returns a data.frame detailing the following information for each raster available to download from the Malaria Atlas Project database.

1. `raster_code` unique identifier for each raster
2. `title` abbreviated title for each raster, used as `surface` argument in getRaster()
3. `title_extended` extended title for each raster, detailing raster content
4. `abstract` full description of each raster, outlining raster creation methods, raster content and more.
5. `citation` citation of peer-reviewed article in which each raster has been published
6. `pub_year` year in which raster was published, used as `pub_year` argument in getRaster() to updated raster versions from their predecessor(s).
7. `min_raster_year` earliest year for which each raster is available
8. `max_raster_year` latest year for which each raster is available

Examples

```
available_rasters <- listRaster()
```

`listShp` *List administrative units for which shapefiles are stored on the MAP geoserver.*

Description

`listShp` lists all administrative units for which shapefiles are stored on the MAP geoserver.

Usage

```
listShp(printed = TRUE, admin_level = c("admin0", "admin1"))
```

Arguments

`printed`	Should the list be printed to the console?
`admin_level`	Specifies which administrative unit level for which to return available polygon shapefiles. A string vector including one or more of`"admin0"`, `"admin1"`, `"admin2"` OR `"admin3"`. Default: `c("admin0", "admin1")`

Value

`listShp` returns a data.frame detailing the administrative units for which shapefiles are stored on the MAP geoserver.

Examples

```
available_admin_units <- listShp()
```

`listSpecies` *list all species which have occurrence data within the MAP database.*

Description

`listSpecies` lists all species occurrence data available to download from the Malaria Atlas Project database.

Usage

```
listSpecies(printed = TRUE)
```

Arguments

`printed`	should the list be printed to the database.

Value

`listSpecies` returns a data.frame detailing the following information for each species available to download from the Malaria Atlas Project database.

1. `species` string detailing species

Examples

```
available_species <- listSpecies()
```

`malariaAtlas` *An R interface to open-access malaria data, hosted by the Malaria Atlas Project.*

Description

`malariaAtlas` provides a suite of tools to allow you to download all publicly available PR points for a specified country (or ALL countries) as a dataframe within R.

malariaAtlas functions

1. `listAll` - lists all countries for which there are publicly visible PR datapoints in the MAP database.
2. `is_available` - checks whether the MAP database contains PR points for the specified country/countries.
3. `getPR` - downloads all publicly available PR points for a specified country (or countries) and returns this as a dataframe.

Index