Package: camtrapmonitoring (via r-universe)

October 23, 2024

Title Camera Trap Monitoring For Estimating Wildlife Density

Version 0.12.1

Description Evaluating potential camera trap locations. Potential locations are evaluated using relevant spatial layers producing a dataset of selected locations with covariates that can be used to quantify sampling bias. Soon - density estimation methods.

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.0

Roxygen list(markdown = TRUE)

Suggests testthat, knitr, rmarkdown, ggplot2, usethis, osmdata, elevatr

VignetteBuilder knitr

Imports sf (>= 0.7-3), terra, distanceto

Depends R (>= 4.0)

URL https://robitalec.github.io/camtrapmonitoring/

BugReports https://github.com/robitalec/camtrapmonitoring/issues

Repository https://robitalec.r-universe.dev

RemoteUrl https://github.com/robitalec/camtrapmonitoring

RemoteRef HEAD

RemoteSha 6c0aefac5f7b727f981338b7b20622ed38d02ae4

Contents

`binary_layer` *Binary layer*

Description

Helper function to make a binary raster layer from input target.

Usage

```
binary_layer(target, fun = "equals", value, layer = 1)
```

Arguments

`target`	SpatRaster target (see `terra::rast()`)
`fun`	character indicating which function to use to compare layer to value. One of 'equals', 'gt', 'gte', 'lt', 'lte' or 'in'. Default: 'equals'.
`value`	numeric value in target. see Details.
`layer`	default 1, see `terra::extract()`

Details

Find all pixels given the function ('fun') matching the value ('value') and return a binary raster.

value' may only be length 1 if 'fun' is one of: 'equals', 'gt', 'gte', 'lt', 'lte'.

'value' may be greater than length 1 only if 'fun' is: 'in'.

Value

A binary raster `SpatRaster` (see `terra::rast()`) with two values:

- `TRUE` if pixel matches 'value' provided given the provided function
- `FALSE` if pixel does not match 'value' provided given the provided function

Examples

```
library(terra)
clearwater_lc_path <- system.file("extdata",
  "clearwater_lake_land_cover.tif", package = "camtrapmonitoring")
clearwater_lake_land_cover <- rast(clearwater_lc_path)

bin <- binary_layer(target = clearwater_lake_land_cover,
  fun = 'equals', value = 18)

plot(bin)

# fun = 'in'
bin <- binary_layer(target = clearwater_lake_land_cover,
  fun = 'in', value = c(1, 2))

plot(bin)
```

`camtrapmonitoring` *camtrapmonitoring*

Description

Evaluating potential camera trap locations. Potential locations are evaluated using relevant spatial layers producing a dataset of selected locations with covariates that can be used to quantify sampling bias.

Details

Soon - density estimation methods.

Author(s)

Maintainer: Alec L. Robitaille `<robit.alec@gmail.com>` (ORCID)

See Also

Useful links:

- `https://robitalec.github.io/camtrapmonitoring/`
- Report bugs at `https://github.com/robitalec/camtrapmonitoring/issues`

`clearwater_lake_density` *Example species density*

Description

A dataset containing simulated species density near Clearwater Lake, Manitoba.

Usage

```
clearwater_lake_density
```

Format

An `sf` (see `sf::st_sf()`) object with 272 features and 1 variable "density".

Details

Simulated species density with three levels "High", "Medium", Low" with probabilities 0.1, 0.3, 0.6. Grid size is 2 km x 2 km.

Examples

```
library(sf)
data(clearwater_lake_density)
plot(clearwater_lake_density)
```

`clearwater_lake_elevation` *Example elevation*

Description

A dataset containing elevation near Clearwater Lake, Manitoba.

Format

An external tif file to be read in with terra as a `SpatRaster` object.

Details

Elevation data are from the AWS using the elevatr package (`elevatr::get_elev_raster()`).

The units are meters.

Source

Hollister, J.W. (2022). elevatr: Access Elevation Data from Various APIs. R package version 0.4.2. https://CRAN.R-project.org/package=elevatr/

Examples

```
library(terra)
clearwater_elev_path <- system.file("extdata",
  "clearwater_lake_elevation.tif", package = "camtrapmonitoring")
clearwater_lake_elevation <- rast(clearwater_elev_path)
plot(clearwater_lake_elevation)
```

clearwater_lake_extent *Example extent*

Description

A dataset containing the extent near Clearwater Lake, Manitoba.

Usage

```
clearwater_lake_extent
```

Format

An `sf` (see `sf::st_sf()`) object with 2 points.

Examples

```
library(sf)
data(clearwater_lake_extent)
plot(clearwater_lake_extent)
```

clearwater_lake_hydro *Example hydrology features*

Description

A dataset containing hydrology features (in this case, major lakes) near Clearwater Lake, Manitoba.

Usage

```
clearwater_lake_hydro
```

Format

An `sf` (see `sf::st_sf()`) object with 5 features and 8 fields. See the source below for details.

Details

Hydrology features are from Open Street Map, downloaded using the osmdata package.

Source

Mark Padgham, Bob Rudis, Robin Lovelace, Maëlle Salmon (2017). “osmdata.” *Journal of Open Source Software*, *2*(14), 305. doi:10.21105/joss.00305 https://doi.org/10.21105/joss.00305, https://joss.theoj.org/papers/10.21105/joss.00305.

Examples

```
library(sf)
data(clearwater_lake_hydro)
plot(clearwater_lake_hydro)
```

---

clearwater_lake_land_cover

Example land cover

Description

A dataset containing land cover classes near Clearwater Lake, Manitoba.

Format

An external tif file to be read in with terra as a `SpatRaster` object (see `terra::rast()`). See the source below for details.

Details

Land cover data are from 2020 Land Cover of Canada. Class index defined here https://drive.google.com/file/d/1TvOZdLO_N86HfsiJQtnE3BdfAFL1s0ux/view?usp=sharing

- 1 Temperate or sub-polar needleleaf forest/Forêt de conifères sempervirente tempérée ou subpolaire
- 2 Sub-polar taiga needleleaf forest/ Forêt de conifères (taïga) subpolaire
- 5 Temperate or sub-polar broadleaf deciduous forest/ Forêt feuillue tempérée ou subpolaire
- 6 Mixed forest/ Forêt mixte
- 8 Temperate or sub-polar shrubland/Arbustaie tempérée ou subpolaire
- 10 Temperate or sub-polar grassland/Prairie tempérée ou subpolaire
- 11 Sub-polar or polar shrubland-lichen-moss/Arbustaie à lichens et à mousses polaire ou subpolaire

- 12 Sub-polar or polar grassland-lichen-moss/Prairie à lichens et à mousses polaire ou subpolaire
- 13 Sub-polar or polar barren-lichen-moss/Lande à lichens et à mousses polaire ou subpolaire
- 14 Wetland/Milieu humide
- 15 Cropland/Terre cultivée
- 16 Barren Lands/Lande
- 17 Urban and built-up/Milieu urbain
- 18 Water/Eau
- 19 Snow and Ice/Neige et glace

Source

https://open.canada.ca/data/en/dataset/ee1580ab-a23d-4f86-a09b-79763677eb47

Examples

```
library(terra)
clearwater_lc_path <- system.file("extdata",
  "clearwater_lake_land_cover.tif", package = "camtrapmonitoring")
clearwater_lake_land_cover <- rast(clearwater_lc_path)
plot(clearwater_lake_land_cover)
```

`clearwater_lake_roads` *Example roads*

Description

A dataset containing roads near Clearwater Lake, Manitoba.

Usage

```
clearwater_lake_roads
```

Format

An `sf` (see `sf::st_sf()`) object with 172 roads and 48 variables. See the source below for details.

Details

Roads are from the Canadian National Road Network (NRN) with all corresponding fields.

Source

https://open.canada.ca/data/en/dataset/3d282116-e556-400c-9306-ca1a3cada77f

Examples

```
library(sf)
data(clearwater_lake_roads)
plot(clearwater_lake_roads)
```

clearwater_lake_wetlands *Example wetland features*

Description

A dataset containing wetland features near Clearwater Lake, Manitoba.

Usage

```
clearwater_lake_wetlands
```

Format

An `sf` (see `sf::st_sf()`) object with 5 features and 8 fields. See the source below for details.

Details

Hydrology features are from Open Street Map, downloaded using the osmdata package.

Source

Mark Padgham, Bob Rudis, Robin Lovelace, Maëlle Salmon (2017). “osmdata.” *Journal of Open Source Software*, *2*(14), 305. doi:10.21105/joss.00305 https://doi.org/10.21105/joss.00305, https://joss.theoj.org/papers/10.21105/joss.00305.

Examples

```
library(sf)
data(clearwater_lake_wetlands)
plot(clearwater_lake_wetlands)
```

eval_buffer *Evaluate camera trap locations by buffered sampling of layers*

Description

Using buffered camera trap locations generated with camtrapmonitoring functions sample_ct() and grid_ct(), sample raster layers to characterize and select camera trap locations, and quantify potential sampling bias.

Usage

```
eval_buffer(features, target, buffer_size, buffer_fun = mean, layer = 1)
```

Arguments

features	sf features (see sf::st_sf())
target	SpatRaster target (see terra::rast())
buffer_size	radius of buffer around each point
buffer_fun	function for summarizing buffer region, default mean
layer	default 1, see terra::extract()

Value

vector of values from target matching buffered locations in features

See Also

Other eval: eval_dist(), eval_pt()

Examples

```
library(terra)

data("clearwater_lake_density")
clearwater_lake_elevation <- rast(system.file('extdata',
  'clearwater_lake_elevation.tif', package = 'camtrapmonitoring'))

# Sample points
pts <- sample_ct(region = clearwater_lake_density, 1, type = 'random')

# Make grid with queen's case
queen <- grid_ct(features = pts, case = 'queen', distance = 100)

# Evaluate each point with the land cover layer
queen$elev <- eval_buffer(
  features = queen, target = clearwater_lake_elevation, buffer_size = 150)

plot(queen["elev"])
```

eval_dist *Evaluate distance-to*

Description

Using camera trap locations generated with camtrapmonitoring functions sample_ct() and grid_ct(), evalaute the distance between features and camera trap locations to characterize and select locations, and quantify potential sampling bias.

Usage

```
eval_dist(features, target, measure = NULL)
```

Arguments

features	sf features (see sf::st_sf())
target	sf feature target (see sf::st_sf())
measure	measure type see geodist::geodist() for details

Details

To avoid the large overhead of creating distance to rasters for small/medium number of sample points, this function uses the vector-based distance approach from distanceto::distance_to(). It determines the nearest feature to each target then calculates the distance between each pair.

Value

vector of distances between target and features

See Also

Other eval: eval_buffer(), eval_pt()

Examples

```
data("clearwater_lake_density")
data("clearwater_lake_wetlands")

# Sample points
pts <- sample_ct(region = clearwater_lake_density, 1, type = 'random')

# Make grid with queen's case
queen <- grid_ct(features = pts, case = 'queen', distance = 100)

# Evaluate each point with the land cover layer
queen$dist_wetland <- eval_dist(features = queen, target = clearwater_lake_wetlands)

# Plot
plot(queen["dist_wetland"])
```

eval_pt *Evaluate camera trap locations by point sampling layers*

Description

Using camera trap locations generated with camtrapmonitoring functions sample_ct() and grid_ct(), sample raster layers to characterize and select camera trap locations, and quantify potential sampling bias.

Usage

```
eval_pt(features, target, layer = 1)
```

Arguments

features	sf features (see sf::st_sf())
target	SpatRaster target (see terra::rast())
layer	default 1, see terra::extract()

Value

vector of values from target matching locations in features

See Also

Other eval: eval_buffer(), eval_dist()

Examples

```
library(terra)

data("clearwater_lake_density")
clearwater_lake_land_cover <- rast(system.file('extdata',
  'clearwater_lake_land_cover.tif', package = 'camtrapmonitoring'))

# Sample points
pts <- sample_ct(region = clearwater_lake_density, n = 1, type = 'random')

# Make grid with queen's case
queen <- grid_ct(features = pts, case = 'queen', distance = 100)

# Evaluate each point with the land cover layer
queen$lc <- eval_pt(features = queen, target = clearwater_lake_land_cover)

plot(queen["lc"])
```

grid_ct *Make camera trap grids*

Description

Set up grids around focal points. For example, sample points in your study area with `sample_ct()` then use `grid_ct()` to establish a grid of camera traps around each.

Usage

```
grid_ct(features, distance, case, n, id = "id_sample_ct")
```

Arguments

`features`	sf features (see `sf::st_sf()`)
`distance`	distance between adjacent camera traps. Don't worry about the hypotenuse.
`case`	"queen", "rook", "bishop", or "triplet". Ignored if `n` is provided.
`n`	number of points around each focal point. `n` overrides the `case` argument, do not provide both - see Details.
`id`	default: "id_sample_ct" generated automatically from `sample_ct()`

Value

Extended sf object either nine times the length of input features for 'queen' case or 5 times the length of input DT for 'rook' or 'bishop' case. Otherwise n * number the length of input features. See examples.

The logical 'focal' column indicates which point is the focal camera trap for each grid.

See Also

Other grid: `grid_design()`

Examples

```
data("clearwater_lake_density")
pts <- sample_ct(clearwater_lake_density, 1, type = 'random')

# Make grid with case, eg. 'queen'
queen <- grid_ct(features = pts, distance = 100, case = 'queen')

# Plot
plot(queen['focal'])

# Make grid with n
n_grid <- grid_ct(features = pts, distance = 100, n = 25)
plot(n_grid['id_grid_ct'])
```

grid_design *Grid design*

Description

Helper function used internally by `grid_ct()` to establish grids around focal locations. Provided to the user to explore grid design options before applying to their data.

Usage

```
grid_design(distance, case, n)
```

Arguments

`distance`	distance between adjacent camera traps. Don't worry about the hypotenuse.
`case`	"queen", "rook", "bishop", or "triplet". Ignored if `n` is provided.
`n`	number of points around each focal point. `n` overrides the `case` argument, do not provide both - see Details.

Value

grid design data.frame

See Also

Other grid: `grid_ct()`

Examples

```
plot(grid_design(distance = 100, case = 'queen'))
plot(grid_design(distance = 100, n = 13))
```

sample_ct *Camera trap sampling*

Description

Sample potential camera trap locations. For stratified sampling, provide a suitable column to stratify region by. Alternatively, `sf::st_sample()` is used directly to sample points across all features.

Usage

```
sample_ct(region, n, type, strata = NULL)
```

Arguments

`region`	spatial feature object across which points will be sampled
`n`	number of random points. If `strata` is provided, n represents the number of random points per strata
`type`	type of sampling, see `sf::st_sample()`
`strata`	column name in region indicating strata

Value

`sf` (see `sf::st_sf()`) object with POINT geometry

Examples

```
# Example grid with density levels (High, Medium, Low)
data(clearwater_lake_density)

# Stratified random points for each density level
pts_random <- sample_ct(
  region = clearwater_lake_density, n = 20,
  type = 'random', strata = 'density')

# Plot density grid and sampled points
plot(clearwater_lake_density, reset = FALSE)
plot(pts_random, add = TRUE, pch = 1, strata = 1)

# Regular sampled points across all features
pts_regular <- sample_ct(
  region = clearwater_lake_density, n = 20, type = 'regular')

# Plot density grid and sampled points
plot(clearwater_lake_density, reset = FALSE)
plot(pts_regular, add = TRUE, pch = 2, strata = 1)
```

`scale_layer` *Scale in region of interest*

Description

Helper function to scale a target layer in a region of interest. Can be used to compare locations with `eval_pt()` and `eval_buffer()`, and select locations based off of relative values instead of absolute values.

Usage

```
scale_layer(target, region, center = TRUE, scale = TRUE)
```

Arguments

`target` SpatRaster target (see `terra::rast()`)

`region` object which can be passed to `terra::ext()` including `sf`, `SpatRaster`, and 2x2 matrices.

`center` see `terra::scale()`

`scale` see `terra::scale()`

Value

`SpatRaster` layer, cropped to extent of provided region, and scaled.

See Also

`terra::scale()`

Examples

```
# Load packages
library(terra)

# Load data
data("clearwater_lake_hydro")
clearwater_elev_path <- system.file(
  "extdata", "clearwater_lake_elevation.tif", package = "camtrapmonitoring")
clearwater_lake_elevation <- rast(clearwater_elev_path)

# Region of interest around Clearwater lake
roi <- clearwater_lake_hydro[4,]

# Scale elevation in extent of density grid
elev_scaled <- scale_layer(target = clearwater_lake_elevation, region = roi)
```

Index