Package: urbioconnect (via r-universe)

June 5, 2026

Title Urban Habitat Connectivity Analysis

Version 0.1.0.9000

Description Analyse and visualise habitat connectivity in urban landscapes, accounting for barriers and buffer distances. Includes Shiny app for interactive analysis and report generation. Based on methods developed by Kirk et al (2023) <doi:10.1016/j.mex.2022.101989>.

License GPL (>= 3)

URL https://urbio-ecology.r-universe.dev/urbioconnect, https://github.com/urbio-ecology/urbioconnect, https://urbio-ecology.github.io/urbioconnect/

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.2.0)

Imports colorspace, dplyr, ggplot2, glue, here, knitr, purrr, quarto, scico, sf, shiny, stringr, terra, tibble, tidyr, tidyterra, stats, grDevices, cli, rlang, scales

Suggests bslib, conflicted, crew, DT, fasterize, gridExtra, igraph, marquee, magick, prettyunits, raster, rmarkdown, shinyjs, stars, tarchetypes, targets, testthat (>= 3.0.0), vctrs, vdiffr, withr, tictoc, ggspatial

Config/testthat/edition 3

VignetteBuilder knitr

BugReports https://github.com/urbio-ecology/urbioconnect/issues

LazyData true

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libicu-dev libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

Repository https://urbio-ecology.r-universe.dev

Date/Publication 2026-06-05 04:47:23 UTC

RemoteUrl https://github.com/urbio-ecology/urbioconnect

RemoteRef HEAD

RemoteSha 86bfae3010b970e936124e8ee415dcec163e9b81

Contents

add_patch_area *Add patch area layer*

Description

Adds an area layer to the raster of the area of each patch.

Usage

```
add_patch_area(raster)
```

Arguments

raster	terra SpatRaster. In the workflow, this is the patch ID raster.

Value

terra SpatRaster with two layers: patch_id and area.

Examples

```
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
buffered_habitat <- habitat_buffer(lizard_habitat, 5)
barrier_mask <- create_barrier_mask(lizard_barrier)
fragmented <- fragment_habitat(buffered_habitat, barrier_mask)
remaining_habitat <- drop_habitat_under_barrier(
  habitat = lizard_habitat,
  barrier = lizard_barrier
  )
fragment_patches <- assign_patches_to_fragments(
  remaining_habitat = remaining_habitat,
  fragment = fragmented
  )
library(terra)
add_patch_area(fragment_patches)
```

aggregate_connected_patches *Aggregate connected patch areas*

Description

Aggregate a raster with connected patch areas into a data frame where each row is a unique patch, and its area and area squared.

Usage

```
aggregate_connected_patches(raster)
```

Arguments

raster	terra SpatRaster. Raster with patch_id and area layers.

Value

Data frame with patch areas and areas squared.

Examples

```
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
buffered_habitat <- habitat_buffer(lizard_habitat, 5)
barrier_mask <- create_barrier_mask(lizard_barrier)
fragmented <- fragment_habitat(buffered_habitat, barrier_mask)
remaining_habitat <- drop_habitat_under_barrier(
  habitat = lizard_habitat,
  barrier = lizard_barrier
  )
fragment_patches <- assign_patches_to_fragments(
  remaining_habitat = remaining_habitat,
  fragment = fragmented
  )
library(terra)
patch_areas <- add_patch_area(fragment_patches)
aggregate_connected_patches(patch_areas)
```

---

assign_patches_to_fragments

Assign patches to fragments

Description

Assign patches to fragments

Usage

```
assign_patches_to_fragments(remaining_habitat, fragment)
```

Arguments

remaining_habitat	Terra SpatRaster. Remaining habitat.
fragment	Terra SpatRaster. Fragment geometry.

Value

Terra SpatRaster with patch IDs.

Examples

```
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
buffered_habitat <- habitat_buffer(lizard_habitat, 5)
barrier_mask <- create_barrier_mask(lizard_barrier)
fragmented <- fragment_habitat(buffered_habitat, barrier_mask)
remaining_habitat <- drop_habitat_under_barrier(
  habitat = lizard_habitat,
  barrier = lizard_barrier
  )
fragment_patches <- assign_patches_to_fragments(
  remaining_habitat = remaining_habitat,
  fragment = fragmented
  )
library(terra)
plot(fragment_patches)
```

---

clean *Clean any spatial data layer (shape file)*

Description

Many shape files contain errors, places where the edges of a polygon cross over or polygons which overlap. This helps remove some of those errors by smoothing the edges of polygons, removing corners, and dissolving edges where polygons overlap. This reduces the complexity of the shape file, making future steps faster.

Usage

```
clean(spatial_data, ...)
```

Arguments

spatial_data	spatial data frame from sf.
...	extra options (currently not used)

Value

An sf object with simplified and validated geometry.

Examples

```
lizard_barrier_shp <- example_barrier_shp()
clean(lizard_barrier_shp)
```

col2hex *Convert color name to hexadecimal*

Description

Convert color name to hexadecimal

Usage

```
col2hex(color_name)
```

Arguments

color_name	Character. Color name recognized by R.

Value

Character. Hexadecimal color code.

Examples

```
col2hex("forestgreen")
col2hex("blue")
```

compare_connectivity *Compare measurements the connectivity of different scenarios*

Description

We can measure the connectivity of a given habitat and barrier with habitat_connectivity(). We can also compare the connectivity, say for example if you have the same area habitat and barrier, but you want to understand what the change in connectedness is when you remove, or add some habitat, or some barrier(s), or both. This function help you do that.

Usage

```
compare_connectivity(area_new, area_baseline, interpatch_distance, species)
```

Arguments

area_new	Numeric vector. Area of a connected patch.
area_baseline	Numeric vector. Baseline area of a connected patch.

`interpatch_distance`
: Numeric. The distance (in meters) where habitat patches are considered connected. E.g., if set to 500, patches 498m apart are connected, those 501m apart are not connected. This is passed internally to a spatial operation known as "buffering", where this distance is used as a radius from the edge of the habitat zone. This means the specified `interpatch_distance` is halved exactly. So an interpatch distance of 500 will be converted to 250.

`species`
: name of species

Value

tibble with "scenario", "interpatch_distance", "species", "n_patches", "effective_mesh_ha", and "prob_connectedness".

Examples

```
# for demonstration purposes - let's imagine the area decreases by 20%
baseline_areas <- round(lizard_areas_connected$area)
new_areas <- baseline_areas[-1] * 0.8
compare_connectivity(
  area_new = new_areas,
  area_baseline = baseline_areas,
  interpatch_distance = 10,
  species = "blue-tongued lizard"
)
```

`connectivity_probability` *Calculate connectivity probability*

Description

Computes the probability two randomly chosen points within habitat are connected, accounting for fragmentation. This requires the effective mesh size (via `effective_mesh_size()`), and the area of patches. This means that you can calculate the change in connectivity if you calculate the effective mesh size of a new habitat/barrier plan, and then use the baseline

Usage

```
connectivity_probability(effective_mesh_size, area_baseline)
```

Arguments

`effective_mesh_size`
: As calculated by `effective_mesh_size()`

`area_baseline`
: Numeric vector. Area of a connected patch. This argument is called "baseline" as when you are doing design scenarios you must refer to the baseline area when calculating connectivity probability. See vignette, TODO.

Value

Numeric. Probability of connectedness (0-1).

Examples

```
effective_mesh <- effective_mesh_size(
  area = lizard_areas_connected$area
  )
connectivity_probability(
  effective_mesh_size = effective_mesh,
  area_baseline = lizard_areas_connected$area
  )
# if you wanted to compare to a scenario, you would consider the effective
# mesh size to be the new scenario level, and the baseline as so:
connectivity_probability(
# scenario 1
  effective_mesh_size = effective_mesh,
  area_baseline = lizard_areas_connected$area
  )
```

---

create_barrier_mask *Create barrier mask*

Description

Converts a barrier layer into a multiplier mask for connectivity analysis. Takes a raster where barriers are coded as 1 (and non-barriers as NA), and inverts it to produce a mask with NA values where barriers exist and 1 elsewhere. This format allows barriers to be applied by multiplying the mask with connectivity surfaces, effectively blocking movement through barrier cells.

Usage

```
create_barrier_mask(barrier)
```

Arguments

barrier	Terra SpatRaster. Barrier layer with 1 = barrier, NA = no barrier.

Value

Terra SpatRaster. Mask with 1 where movement is allowed, NA where barriers exist.

Examples

```
lizard_barrier <- example_barrier()
create_barrier_mask(lizard_barrier)
```

drop_habitat_under_barrier *Remove habitat under barriers*

Description

Essentially just performs a `terra::mask()` operation, to remove the habitat parts that are under the mask.

Usage

```
drop_habitat_under_barrier(habitat, barrier_mask)
```

Arguments

habitat	Terra SpatRaster. Habitat layer.
barrier_mask	Terra SpatRaster. Barrier mask.

Value

Terra SpatRaster with habitat remaining after barrier removal.

Examples

```
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
barrier_mask <- create_barrier_mask(lizard_barrier)
remaining_habitat <- drop_habitat_under_barrier(
  habitat = lizard_habitat,
  barrier = lizard_barrier
  )
remaining_habitat
```

effective_mesh_size *Calculate effective mesh size*

Description

Computes the effective mesh size metric for habitat connectivity, This represents the probability that two randomly chosen points within habitat remain connected. Intended for usage from objects created by `habitat_connectivity()`. See examples below.

Usage

```
effective_mesh_size(area, area_baseline = area)
```

Arguments

area	Numeric vector. Area of connected patches.
area_baseline	Optional. Defaults to area if not specified. Numeric vector of connected patches of a baseline area. This is to allow for comparing the effective mesh size when comparing different scenarios. See future vignette on this topic (TODO).

Value

Numeric. Effective mesh size, in hectares.

Examples

```
effective_mesh_size(lizard_areas_connected$area)
```

---

empty_grid *Create Empty terra raster grid*

Description

Create Empty terra raster grid

Usage

```
empty_grid(habitat, resolution = 10)
```

Arguments

habitat	SF object or terra SpatRaster.
resolution	Numeric. Cell size in meters (default: 10).

Value

Terra SpatRaster. Empty raster grid.

Examples

```
lizard_barrier_shp <- example_barrier_shp()
empty_grid(lizard_barrier_shp, resolution = 10)
```

example-lizard-data *Lizard Habitat and Barrier Data from Melbourne.*

Description

We provide Habitat and Barrier data on various lizard species (nominally, Blue-tongued Lizard). The data was collected from Darebin Creek in Melbourne, which runs between Preston and West Heidelberg. For analysis purposes, a interpatch distance of 200 metres is recommended for lizard connectivity assessments.

Usage

```
example_habitat()

example_barrier_shp()

example_barrier()
```

Details

We provide helper functions to load the raster and shapefile data. These are required due to how the raster and vector data are stored. These functions provide easy access to example raster and shapefile data included with the package:

- `example_habitat()` Returns a raster of lizard habitat data.
- `example_barrier_shp()` Returns a shapefile of lizard barrier data as an SF object.
- `example_barrier()` Returns a raster of lizard barrier data.

Value

A terra raster object or sf object depending on the function called

Examples

```
library(terra)

# Load habitat raster
lizard_habitat <- example_habitat()
plot(lizard_habitat, col = "darkgreen", legend = FALSE, main = "Lizard Habitat")

# Load barrier shapefile
lizard_barrier_shp <- example_barrier_shp()
plot(lizard_barrier_shp)

# Load barrier raster
lizard_barrier <- example_barrier()
plot(lizard_barrier, col = c("grey", "white"), legend = FALSE, main = "Lizard Barriers")
```

```
plot(lizard_barrier, col = c("grey", "white"), legend = FALSE, main = "Lizard Habitat and Barrier")
plot(lizard_habitat, col = "darkgreen", legend = FALSE, add = TRUE)
```

`fragment_habitat` *Fragment habitat*

Description

Takes a barrier mask (created with `create_barrier_mask()`) and fragments up the habitat where they intersect.

Usage

```
fragment_habitat(buffered_habitat, barrier_mask)
```

Arguments

`buffered_habitat`	Terra SpatRaster. Buffered habitat.
`barrier_mask`	Terra SpatRaster. Barrier mask.

Value

Terra SpatRaster with fragmented habitat.

Examples

```
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
buffered_habitat <- habitat_buffer(lizard_habitat, 5)
barrier_mask <- create_barrier_mask(lizard_barrier)
fragmented <- fragment_habitat(buffered_habitat, barrier_mask)
```

`generate_connectivity_report`
Generate Connectivity Report

Description

Creates a parameterised Quarto report from connectivity analysis results.

Usage

```
generate_connectivity_report(
  species_name,
  interpatch_distances,
  results_connect_habitat,
  areas_connected,
  habitat = NULL,
  barrier = NULL,
  habitat_raster = NULL,
  data_resolution = 10,
  target_resolution = 500,
  output_file = NULL,
  output_format = c("html", "pdf", "both"),
  output_dir = getwd()
)
```

Arguments

species_name	Character. Name of the species being analysed.
interpatch_distances	Numeric. The distances (in meters) where habitat patches are considered connected. E.g., if set to 500, patches 498m apart are connected, those 501m apart are not connected. This is passed internally to a spatial operation known as "buffering", where this distance is used as a radius from the edge of the habitat zone. This means the specified interpatch_distance is halved exactly. So an interpatch distance of 500 will be converted to 250.
results_connect_habitat	Data frame. Connectivity summary results.
areas_connected	List of data frames. Connected patch areas for each interpatch distance.
habitat	SF object. Habitat spatial data (optional, for mapping).
barrier	SF object. Barrier spatial data (optional, for mapping).
habitat_raster	Terra SpatRaster. Habitat raster (optional, for mapping).
data_resolution	Numeric. Data resolution in meters.
target_resolution	Numeric. Target resolution in meters.
output_file	Character. Output filename (without extension).
output_format	Character. Output format: "html" (default), "pdf", or "both".
output_dir	Character. Directory to save the report (default: current directory).

Value

Character vector of generated report file path(s).

Examples

```
## Not run:
report_path <- generate_connectivity_report(
  species_name = "Superb Fairy Wren",
  interpatch_distances = c(100, 250, 400),
  results_connect_habitat = results_df,
  areas_connected = patches_list,
  output_format = "html"
)

## End(Not run)
```

---

gg_barrier_habitat_interpatch_dist

Plot barrier, habitat, and interpatch distance layers

Description

Creates a visualisation of habitat, interpatch distance zone, and barriers using terra rasters.

Usage

```
gg_barrier_habitat_interpatch_dist(
  barrier,
  buffered,
  habitat,
  interpatch_distance,
  species,
  col_barrier,
  col_interpatch_dist,
  col_habitat,
  col_paper = NA
)
```

Arguments

barrier	Terra SpatRaster. Barrier layer (e.g., roads).
buffered	Terra SpatRaster. Buffered habitat layer.
habitat	Terra SpatRaster. Original habitat layer.
interpatch_distance	Numeric. The distance (in meters) where habitat patches are considered connected. E.g., if set to 500, patches 498m apart are connected, those 501m apart are not connected. This is passed internally to a spatial operation known as "buffering", where this distance is used as a radius from the edge of the habitat zone. This means the specified `interpatch_distance` is halved exactly. So an interpatch distance of 500 will be converted to 250.

species	Character. Species name for plot title.
col_barrier	Character. Color for barrier layer.
col_interpatch_dist	Character. Color for interpatch distance zone.
col_habitat	Character. Color for habitat patches.
col_paper	Character. Background color (default: "white").

Value

A ggplot2 object.

Examples

```
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
lizard_buffered <- habitat_buffer(lizard_habitat, buffer_radius =10)
gg_bar_hab_buf <- gg_barrier_habitat_interpatch_dist(
  barrier = lizard_barrier,
  buffered = lizard_buffered,
  habitat = lizard_habitat,
  interpatch_distance = 10,
  species = "Blue Tongue Lizard",
  col_barrier = "black",
  col_interpatch_dist = "lightgreen",
  col_habitat = "seagreen"
)
gg_bar_hab_buf

# add north arrow and scale bar with ggspatial
library(ggspatial)
library(tidyterra)
gg_bar_hab_buf +
 annotation_north_arrow(
   style = north_arrow_fancy_orienteering()
  ) +
  annotation_scale()
```

---

habitat_buffer *Buffer habitat raster*

Description

Buffer around the habitat a given distance in metres using `terra::focalMat(d = buffer_radius, type = "circle")`. This operation is used to identify connected patches of habitat. Two patches will connect when their edge-to-edge gap is <= 2 * `buffer radius`. So, we recommend you specify the `buffer_radius` value to be half the interpatch distance, which is the distance past which habitat patches are no longer considered connected. For example, if your interpatch distance is 500m, set `buffer_radius = 250`.

Usage

```
habitat_buffer(habitat, buffer_radius)
```

Arguments

habitat	Terra SpatRaster. Habitat raster.
buffer_radius	Numeric. The radius in metres around the habitat. Since patches of habitat will be connected when their edge-to-edge gap is <= 2 * buffer radius, we recommend you specify buffer_radius to be half the "interpatch distance". This is the distance past which habitat patches are no longer considered connected. For example, if your interpatch distance is 500m, set buffer_radius = 250. The buffer can only be represented if it is at least one raster cell, i.e. keep resolution <= interpatch_distance / 2. Below that the buffer is a no-op: habitat_buffer() warns and returns the habitat unchanged. See vignette("interpatch-distance-a

Value

Terra SpatRaster with buffered habitat.

See Also

vignette("interpatch-distance-and-resolution") for the relationship between interpatch distance, buffer radius, and resolution.

Examples

```
lizard_habitat <- example_habitat()
library(terra)
plot(lizard_habitat, col = "darkgreen", legend = FALSE)
# run with a small buffer radius
lizard_buff <- habitat_buffer(lizard_habitat, buffer_radius = 10)
plot(lizard_buff, col = "lightgreen", legend = FALSE)
plot(lizard_habitat, col = "darkgreen", legend = FALSE, add = TRUE)
```

---

habitat_connectivity *Calculate habitat connectivity using terra*

Description

This performs the entire connectivity workflow, returning a dataframe output. The steps are:

- create_barrier_mask(): Creating barrier mask.
- drop_habitat_under_barrier(): Removes Habitat underneath barrier.
- habitat_buffer(): Buffers the habitat layer by the interpatch distance (m).
- fragment_habitat(): Fragments habitat layer along barrier intersection.
- assign_patches_to_fragments(): Assign patch ID to fragments.
- aggregate_connected_patches(): Summarise area in each patch.

Usage

```
habitat_connectivity(
  habitat,
  barrier,
  interpatch_distance = NULL,
  buffer_radius = NULL,
  verbose = TRUE
)
```

Arguments

habitat	Terra SpatRaster. Habitat raster.
barrier	Terra SpatRaster. Barrier raster.
interpatch_distance	Numeric. The distance (in meters) where habitat patches are considered connected. E.g., if set to 500, patches 498m apart are connected, those 501m apart are not connected. This is passed internally to a spatial operation known as "buffering", where this distance is used as a radius from the edge of the habitat zone. This means the specified interpatch_distance is halved exactly. So an interpatch distance of 500 will be converted to 250. For the buffer to be representable on the raster, keep resolution <= interpatch_distance / 2; below that the buffer is a no-op and a warning is raised. See vignette("interpatch-distance-and-resoluti
buffer_radius	Numeric. The radius in metres around the habitat. Since patches of habitat will be connected when their edge-to-edge gap is <= 2 * buffer radius, we recommend you specify buffer_radius to be half the "interpatch distance". This is the distance past which habitat patches are no longer considered connected. For example, if your interpatch distance is 500m, set buffer_radius = 250. The buffer can only be represented if it is at least one raster cell, i.e. keep resolution <= interpatch_distance / 2. Below that the buffer is a no-op: habitat_buffer() warns and returns the habitat unchanged. See vignette("interpatch-distance-a
verbose	Logical. Display progress messages (default: TRUE).

Value

Data frame with connectivity metrics per patch.

See Also

vignette("interpatch-distance-and-resolution") for the relationship between interpatch distance, buffer radius, and resolution.

Examples

```
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
connectivity <- habitat_connectivity(
    habitat = lizard_habitat,
    barrier = lizard_barrier,
```

```
    interpatch_distance = 10
  )
connectivity
```

---

habitat_connectivity_full *Calculate habitat connectivity with visualization data*

Description

Like habitat_connectivity(), but also returns the intermediate rasters (buffered habitat, patch ID raster, barrier mask, remaining habitat) useful for mapping and reporting.

Usage

```
habitat_connectivity_full(
  habitat,
  barrier,
  interpatch_distance = NULL,
  buffer_radius = NULL,
  verbose = TRUE
)
```

Arguments

habitat	Terra SpatRaster. Habitat raster.
barrier	Terra SpatRaster. Barrier raster.
interpatch_distance	Numeric. The distance (in meters) where habitat patches are considered connected. E.g., if set to 500, patches 498m apart are connected, those 501m apart are not connected. This is passed internally to a spatial operation known as "buffering", where this distance is used as a radius from the edge of the habitat zone. This means the specified interpatch_distance is halved exactly. So an interpatch distance of 500 will be converted to 250. For the buffer to be representable on the raster, keep resolution <= interpatch_distance / 2; below that the buffer is a no-op and a warning is raised. See vignette("interpatch-distance-and-resoluti
buffer_radius	Numeric. The radius in metres around the habitat. Since patches of habitat will be connected when their edge-to-edge gap is <= 2 * buffer radius, we recommend you specify buffer_radius to be half the "interpatch distance". This is the distance past which habitat patches are no longer considered connected. For example, if your interpatch distance is 500m, set buffer_radius = 250. The buffer can only be represented if it is at least one raster cell, i.e. keep resolution <= interpatch_distance / 2. Below that the buffer is a no-op: habitat_buffer() warns and returns the habitat unchanged. See vignette("interpatch-distance-a
verbose	Logical. Display progress messages (default: TRUE).

Value

Named list with elements: `buffered_habitat`, `patch_id_raster`, `areas_connected`, `barrier_mask`, `remaining_habitat`.

Examples

```
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
result <- habitat_connectivity_full(
  lizard_habitat,
  lizard_barrier,
  interpatch_distance = 10,
  verbose = FALSE
)
names(result)
```

---

lizard_areas_connected

Connected habitat patch areas for Blue-tongued Lizard

Description

Pre-computed output of `habitat_connectivity()` run on the lizard example data at a 50 metre interpatch distance. Contains one row per connected habitat patch.

Usage

```
lizard_areas_connected

lizard_areas_connected
```

Format

A data frame with columns:

patch_id Integer. Connected fragment ID.

area Numeric. Total area of the connected patch in square metres.

area_squared Numeric. Squared area, used in connectivity metrics.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 59 rows and 3 columns.

Source

Generated from `example_habitat()` and `example_barrier()` at 50 metre interpatch distance.

See Also

`habitat_connectivity()`, `summarise_connectivity()`

Examples

```
# This was the code that was run to create this object. We don't run it
# as it takes some time to run
## Not run:
lizard_areas_connected <- habitat_connectivity(
    habitat = example_habitat(),
    barrier = example_barrier(),
    interpatch_distance = 50,
    verbose = FALSE
  )

## End(Not run)
lizard_areas_connected
```

`mean_patch_size` *Calculate mean patch size*

Description

This is just a wrapper around `mean()`, however it is written to clearly identify it's usage in the context of the area data. Intended for usage from objects created by `habitat_connectivity()`. See examples below.

Usage

```
mean_patch_size(area, ...)
```

Arguments

`area`	Numeric vector. Area of a connected patch.
`...`	extra arguments to pass to `mean()`.

Value

Numeric. Mean patch size.

Examples

```
mean_patch_size(lizard_areas_connected$area)
```

n_patches *Count number of habitat patches*

Description

Identify the number of habitat patches. A wrapper around `length()`, but named to establish its context. Intended for usage from objects created by `habitat_connectivity()`. See examples below.

Usage

```
n_patches(area)
```

Arguments

`area`	Numeric vector. Area of a connected patch.

Value

Integer. Number of patches.

Examples

```
n_patches(lizard_areas_connected$area)
```

plot_barrier_habitat_interpatch_dist *Save barrier habitat interpatch distance plot*

Description

Saved a plot created by `gg_barrier_habitat_interpatch_dist()` to file.

Usage

```
plot_barrier_habitat_interpatch_dist(
  barrier,
  buffered,
  habitat,
  interpatch_distance,
  species,
  col_barrier,
  col_interpatch_dist,
  col_habitat,
  col_paper
)
```

Arguments

barrier	barrier layer
buffered	buffered layer
habitat	habitat layer
interpatch_distance	Numeric. The distance (in meters) where habitat patches are considered connected. E.g., if set to 500, patches 498m apart are connected, those 501m apart are not connected. This is passed internally to a spatial operation known as "buffering", where this distance is used as a radius from the edge of the habitat zone. This means the specified interpatch_distance is halved exactly. So an interpatch distance of 500 will be converted to 250.
species	character, species name, e.g., "Superb Fairy Wren"
col_barrier	colour to colour the barrier layer
col_interpatch_dist	colour to colour the interpatch distance layer
col_habitat	colour to colour the habitat layer
col_paper	colour to colour the paper layer of ggplot

Value

Named character vector. The file path, named by the interpatch distance.

Examples

```
## Not run:
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
buffered <- habitat_buffer(lizard_habitat, interpatch_distance = 10)
# Creates plot-barrier-interpatch-dist-habitat-*.png in the working directory
plot_barrier_habitat_interpatch_dist(
  barrier = lizard_barrier,
  buffered = buffered,
  habitat = lizard_habitat,
  interpatch_distance = 10,
  species = "Blue-tongued Lizard",
  col_barrier = "white",
  col_interpatch_dist = "lightgreen",
  col_habitat = "seagreen",
  col_paper = "grey50"
)

## End(Not run)
```

`plot_connectivity` *Plot connectivity metrics across interpatch distances*

Description

Creates faceted line plots showing how connectivity metrics change with different interpatch distances. This works best when you have multiple interpatch distances, otherwise it will just be a plot with one point.

Usage

```
plot_connectivity(results_connect_habitat)
```

Arguments

`results_connect_habitat`	Data frame. Connectivity summary results with columns for species, interpatch distance, and various metrics.

Value

A ggplot2 object with faceted plots of connectivity metrics.

Examples

```
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
results <- purrr::map(
  c(10, 20),
  function(d) {
    full <- habitat_connectivity_full(lizard_habitat, lizard_barrier,
      interpatch_distance = d, verbose = FALSE)
    summarise_connectivity(
      area = full$areas_connected$area,
      interpatch_distance = d,
      target_resolution = 500,
      data_resolution = 10,
      aggregation_factor = 50,
      species = "Blue-tongued Lizard"
    )
  }
) |> purrr::list_rbind()
plot_connectivity(results)
```

plot_patches *Plot connected habitat patches*

Description

Visualizes habitat patches colored by their connected fragment ID.

Usage

```
plot_patches(patch_id, interpatch_distance, species = "Species", n_cols = 7)
```

Arguments

patch_id	Terra SpatRaster. Raster with patch IDs.
interpatch_distance	Numeric. The distance (in meters) where habitat patches are considered connected. E.g., if set to 500, patches 498m apart are connected, those 501m apart are not connected. This is passed internally to a spatial operation known as "buffering", where this distance is used as a radius from the edge of the habitat zone. This means the specified interpatch_distance is halved exactly. So an interpatch distance of 500 will be converted to 250.
species	Character. Species name (default: "Species").
n_cols	Integer. Number of colors to cycle through (default: 7).

Value

A ggplot2 object showing patches with distinct colors.

Examples

```
lizard_habitat <- example_habitat()
lizard_barrier <- example_barrier()
interpatch_distance <- 20
buffer_radius <- interpatch_distance / 2
buffered_habitat <- habitat_buffer(lizard_habitat, buffer_radius)
barrier_mask <- create_barrier_mask(lizard_barrier)
fragmented <- fragment_habitat(buffered_habitat, barrier_mask)
remaining_habitat <- drop_habitat_under_barrier(
  habitat = lizard_habitat,
  barrier = lizard_barrier
  )
fragment_patches <- assign_patches_to_fragments(
  remaining_habitat = remaining_habitat,
  fragment = fragmented
  ) |> add_patch_area()

plot_patches(fragment_patches, interpatch_distance = interpatch_distance)
```

```
#' add north arrow and scale bar with ggspatial
library(ggspatial)
library(tidyterra)
plot_patches(fragment_patches, interpatch_distance = interpatch_distance) +
 annotation_north_arrow(
   style = north_arrow_fancy_orienteering()
  ) +
  annotation_scale()
```

`prepare_rasters` *Prepare habitat and barrier rasters*

Description

Convert vector (shapefile) SF habitat and barrier objects into rasters.

Usage

```
prepare_rasters(
  habitat,
  barrier,
  data_resolution = 10,
  target_resolution = 500
)
```

Arguments

`habitat`	SF object. Habitat spatial data.
`barrier`	SF object. Barrier spatial data.
`data_resolution`	Numeric. Fine resolution in meters. Default, 10.
`target_resolution`	Numeric. Coarse resolution in meters. Default, 500.

Value

List with `habitat_raster` and `barrier_raster` elements.

Examples

```
lizard_habitat_sf <- terra::as.polygons(example_habitat(), dissolve = TRUE) |>
  sf::st_as_sf()
lizard_barrier_shp <- example_barrier_shp()
prepare_rasters(lizard_habitat_sf, lizard_barrier_shp)
```

read_geometry *Read shapefile geometry*

Description

Reads a shapefile and extracts only the spatial geometry, discarding attribute data.

Usage

```
read_geometry(shapefile)

## S3 method for class 'sf'
read_geometry(shapefile)

## Default S3 method:
read_geometry(shapefile)
```

Arguments

shapefile	Character. File path to a shapefile, or an SF object.

Value

An sfc object containing only the spatial geometry.

Examples

```
# Read geometry from a file path
barrier_path <- system.file("ex/lizard_barrier.shp", package = "urbioconnect")
barrier_geom <- read_geometry(barrier_path)

# Can also pass an existing SF object
barrier_sf <- sf::st_read(barrier_path, quiet = TRUE)
barrier_geom <- read_geometry(barrier_sf)
```

run_connectivity_app *Launch the Connectivity Shiny App*

Description

Launches the connectivity analysis Shiny application.

Usage

```
run_connectivity_app()
```

Value

No return value, called for side effects (launches Shiny app)

Examples

```
## Not run:
run_connectivity_app()

## End(Not run)
```

`sf_add_patch_area` *Add patch area column*

Description

Add patch area column

Usage

```
sf_add_patch_area(patches)
```

Arguments

`patches`	SF object. Habitat patches.

Value

SF object with added `area` column in square meters.

Examples

```
lizard_habitat_sf <- terra::as.polygons(example_habitat(), dissolve = TRUE) |>
  sf::st_as_sf()
## Not run:
lizard_barrier_shp <- example_barrier_shp()
buffered <- sf_habitat_buffer(lizard_habitat_sf, buffer_radius = 10)
fragments <- sf_fragment_habitat(buffered, lizard_barrier_shp)
remaining <- sf_drop_habitat_under_barrier(lizard_habitat_sf, lizard_barrier_shp)
patches <- sf_assign_patches_to_fragments(remaining, fragments)
sf_add_patch_area(patches)

## End(Not run)
```

sf_aggregate_connected_patches

Aggregate connected patch areas

Description

Groups habitat patches by their connected fragment ID and calculates total and squared areas for connectivity metrics.

Usage

```
sf_aggregate_connected_patches(patch_areas)
```

Arguments

patch_areas	SF object. Habitat patches with area column.

Value

Data frame with `patch_id`, `area`, and `area_squared` columns.

Examples

```
lizard_habitat_sf <- terra::as.polygons(example_habitat(), dissolve = TRUE) |>
  sf::st_as_sf()
lizard_barrier_shp <- example_barrier_shp()
## Not run:
buffered <- sf_habitat_buffer(lizard_habitat_sf, buffer_radius = 10)
fragments <- sf_fragment_habitat(buffered, lizard_barrier_shp)
remaining <- sf_drop_habitat_under_barrier(lizard_habitat_sf, lizard_barrier_shp)
patches <- sf_assign_patches_to_fragments(remaining, fragments) |>
  sf_add_patch_area()
sf_aggregate_connected_patches(patches)

## End(Not run)
```

sf_assign_patches_to_fragments

Assign habitat patches to fragment IDs

Description

Determines which connected fragment each remaining habitat patch belongs to based on spatial intersection.

Usage

```
sf_assign_patches_to_fragments(remaining, fragment_id)
```

Arguments

remaining	SF object. Remaining habitat patches after barrier removal.
fragment_id	SF object. Fragment geometries with IDs.

Value

SF object with habitat patches labeled by their fragment ID.

Examples

```
lizard_habitat_sf <- terra::as.polygons(example_habitat(), dissolve = TRUE) |>
  sf::st_as_sf()
lizard_barrier_shp <- example_barrier_shp()
## Not run:
buffered <- sf_habitat_buffer(lizard_habitat_sf, buffer_radius = 10)
fragments <- sf_fragment_habitat(buffered, lizard_barrier_shp)
remaining <- sf_drop_habitat_under_barrier(lizard_habitat_sf, lizard_barrier_shp)
sf_assign_patches_to_fragments(remaining, fragments)

## End(Not run)
```

---

sf_drop_habitat_under_barrier

Remove habitat underneath barriers

Description

Removes all habitat areas that intersect with barriers and splits multipolygons into individual patches.

Usage

```
sf_drop_habitat_under_barrier(habitat, barrier)
```

Arguments

habitat	SF object. Original habitat geometry.
barrier	SF object. Barrier geometry.

Value

SF object with habitat patches that don't intersect barriers.

Examples

```
lizard_habitat_sf <- terra::as.polygons(example_habitat(), dissolve = TRUE) |>
  sf::st_as_sf()
lizard_barrier_shp <- example_barrier_shp()
sf_drop_habitat_under_barrier(lizard_habitat_sf, lizard_barrier_shp)
```

---

sf_fragment_habitat *Fragment habitat along barriers*

Description

Removes barrier areas from buffered habitat and splits the result into individual polygon fragments.

Usage

```
sf_fragment_habitat(habitat_buffered, barrier)
```

Arguments

habitat_buffered	SF object. Buffered habitat geometry.
barrier	SF object. Barrier geometry (e.g., roads).

Value

SF object with individual habitat fragments, each with a unique ID.

Examples

```
lizard_habitat_sf <- terra::as.polygons(example_habitat(), dissolve = TRUE) |>
  sf::st_as_sf()
lizard_barrier_shp <- example_barrier_shp()
## Not run:
buffered <- sf_habitat_buffer(lizard_habitat_sf, interpatch_distance = 10)
sf_fragment_habitat(buffered, lizard_barrier_shp)

## End(Not run)
```

sf_habitat_buffer *Buffer habitat by distance*

Description

Creates a buffer around habitat polygons and unions overlapping areas into a single polygon, using `sf::st_buffer()`. Unlike the raster `habitat_buffer()`, this works in continuous coordinate space, so there is **no resolution constraint**: any `buffer_radius` produces an exact buffer and the sub-cell "no buffer" problem of the raster path does not apply. The buffer arc is approximated by `nQuadSegs` straight segments per quarter circle (here 5, i.e. a 20-sided polygon), which affects the *smoothness* of the outline, not whether the buffer forms. See `vignette("interpatch-distance-and-resolution")`.

Usage

```
sf_habitat_buffer(habitat, buffer_radius)
```

Arguments

`habitat`	SF object. Habitat spatial data.
`buffer_radius`	Numeric. The radius in metres around the habitat. Specify it as half the inter-patch distance (see `habitat_buffer()`). Because vector buffering is done in continuous space, there is no minimum representable radius — unlike the raster path, there is no resolution below which the buffer becomes a no-op.

Value

SF object with buffered and unioned habitat geometry.

Examples

```
lizard_habitat_sf <- terra::as.polygons(example_habitat(), dissolve = TRUE) |>
  sf::st_as_sf()
## Not run:
sf_habitat_buffer(lizard_habitat_sf, buffer_radius = 10)

## End(Not run)
```

sf_habitat_connectivity *Calculate habitat connectivity*

Description

Performs complete habitat connectivity analysis using vector-based spatial operations. Buffers habitat, fragments it along barriers, and calculates areas of connected patches.

Usage

```
sf_habitat_connectivity(
  habitat,
  barrier,
  interpatch_distance = NULL,
  buffer_radius = NULL
)
```

Arguments

habitat	SF object. Original habitat spatial data.
barrier	SF object. Barrier spatial data (e.g., roads, waterways).
interpatch_distance	Numeric. The distance (in meters) where habitat patches are considered connected. E.g., if set to 500, patches 498m apart are connected, those 501m apart are not connected. This is passed internally to a spatial operation known as "buffering", where this distance is used as a radius from the edge of the habitat zone. This means the specified interpatch_distance is halved exactly. So an interpatch distance of 500 will be converted to 250. Note that interpatch_distance is mutually exclusive to habitat_buffer, so you can only specify either interpatch_distance or habitat_buffer, and never both.
buffer_radius	Numeric. The radius in metres around the habitat. Since patches of habitat will be connected when their edge-to-edge gap is <= 2 * buffer radius, we recommend you specify buffer_radius to be half the "interpatch distance". This is the distance past which habitat patches are no longer considered connected. For example, if your interpatch distance is 500m, set buffer_radius = 250. Note that interpatch_distance is mutually exclusive to habitat_buffer, so you can only specify either interpatch_distance or habitat_buffer, and never both.

Value

Data frame with connectivity metrics for each connected patch, including patch_id, area, and area_squared.

Examples

```
lizard_barrier_shp <- example_barrier_shp()
lizard_habitat_sf <- terra::as.polygons(example_habitat(), dissolve = TRUE) |>
  sf::st_as_sf()
result <- sf_habitat_connectivity(lizard_habitat_sf, lizard_barrier_shp, interpatch_distance = 10)
result
```

summarise_connectivity *Summarise connectivity metrics*

Description

Calculates a comprehensive set of habitat connectivity metrics including effective mesh size, probability of connectedness, and patch statistics. Intended for usage from objects created by `habitat_connectivity()`. See examples below.

Usage

```
summarise_connectivity(
  area,
  area_baseline = NULL,
  interpatch_distance,
  target_resolution,
  data_resolution,
  aggregation_factor,
  species
)
```

Arguments

`area`	Numeric vector. Areas of connected patches.
`area_baseline`	Numeric vector. Areas of connected patch baseline.
`interpatch_distance`	Numeric. The distance (in meters) where habitat patches are considered connected. E.g., if set to 500, patches 498m apart are connected, those 501m apart are not connected. This is passed internally to a spatial operation known as "buffering", where this distance is used as a radius from the edge of the habitat zone. This means the specified `interpatch_distance` is halved exactly. So an interpatch distance of 500 will be converted to 250.
`target_resolution`	Numeric. Target resolution in meters.
`data_resolution`	Numeric. Data resolution in meters.
`aggregation_factor`	Numeric. Factor by which Data resolution was aggregated.
`species`	Character. Name of species analysed.

Value

A tibble with connectivity metrics including number of patches, probability of connectedness, effective mesh size, mean and total patch areas.

Examples

```
summarise_connectivity(
  area = lizard_areas_connected$area,
  interpatch_distance = 10,
  target_resolution = 500,
  data_resolution = 10,
  aggregation_factor = 50,
  species = "Blue-tongued Lizard"
)
```

`total_habitat_area` *Calculate total habitat area*

Description

Calculate the total habitat area in hectars. A wrapper around summing the area and multiplying by 0.0001 to give the units in hectares. Intended for usage from objects created by `habitat_connectivity()`. See examples below.

Usage

```
total_habitat_area(area)
```

Arguments

`area`	Numeric vector. Area of a connected patch.

Value

Numeric. Total habitat area in hectares.

Examples

```
total_habitat_area(lizard_areas_connected$area)
```

Index