Package: birdnetprocess (via r-universe)

November 13, 2026

Title Process and Visualize 'BirdNET' Audio Classification Results

Version 0.0.0.9000

Description Toolbox for processing and visualizing audio classification output from 'BirdNET'. Functionality includes reading and standardizing data from 'Raven' selection tables and CSV files, summarizing species detections, and creating time-series visualizations of vocal activity. It offers a lightweight workflow for ecoacoustic analysis.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

LazyData true

Depends R (>= 4.1.0),

Imports lubridate, dplyr, readr, tibble, ggplot2, scales, rlang, tidyr, hms, yaml

Remotes davidsjoberg/ggstream

Suggests covr, knitr, nanoparquet, rmarkdown, testthat (>= 3.0.0), ggrepel, ggspatial, ggstream, jsonlite, patchwork, gt, quarto, ragg, sf, suncalc, terra, withr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev libx11-dev

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

Date/Publication 2026-10-02 06:39:08 UTC

RemoteUrl https://github.com/traitecoevo/birdnetprocess

RemoteRef HEAD

RemoteSha 868a6df3934845b03fe9b522de2449ae5c98d01c

Contents

apply_thresholds . . . 3
assign_groups . . . 3
bird_net_data . . . 4
birdnet_palette . . . 5
birdnet_report . . . 6
birdnet_theme . . . 7
density_by_session . . . 8
detection_density . . . 8
diel_density . . . 10
export_top_clips . . . 11
fetch_openmeteo_weather . . . 12
filter_by_range . . . 13
group_colours . . . 15
new_deployment . . . 16
parse_birdnet_filename_datetime . . . 16
plot_activity_vs_covariate . . . 17
plot_activity_vs_weather . . . 18
plot_confidence . . . 19
plot_daily_activity . . . 19
plot_density . . . 22
plot_density_actogram . . . 24
plot_density_weather . . . 24
plot_diel_density . . . 25
plot_site_map . . . 26
plot_species . . . 27
plot_species_counts . . . 28
plot_species_stream . . . 29
plot_timeline . . . 30
plot_top_species . . . 31
read_birdnet_file . . . 32
read_birdnet_folder . . . 33
read_birdnet_sites . . . 34
read_bom_dwo . . . 35
read_deployment . . . 35
read_deployment_detections . . . 37
read_detector_labels . . . 38
session_anomaly . . . 39
site_report . . . 39
species_thresholds . . . 41
summarise_detections . . . 42
threshold_source . . . 43
validate_deployment . . . 44
weather_by_session . . . 44

Index **46**

apply_thresholds *Apply per-species thresholds to a detection table*

Description

Filters detections using each species' own threshold, and records which threshold was applied and where it came from.

Usage

```
apply_thresholds(data, cfg)
```

Arguments

data	A tibble of detections with `Common Name` and `Confidence`.
cfg	A `birdnet_deployment`, from `read_deployment`.

Value

`data`, filtered, with added `threshold` and `threshold_source` columns.

assign_groups *Sort detections into taxon groups*

Description

Adds a `group` column to a detection table - `"Birds"`, `"Insects"`, `"Frogs"` and so on - by matching each species' genus, taken from the detector's labels file, against the rules in a deployment config. Use it to colour or facet a plot by broad taxon, as `plot_daily_activity`'s `group_col` argument expects.

Usage

```
assign_groups(
  data,
  labels = NULL,
  rules = NULL,
  species_col = "Common Name",
  group_col = "group"
)
```

Arguments

data	A tibble of detections, e.g. from read_deployment_detections.
labels	Path to a labels file, or a tibble from read_detector_labels. If NULL, taken from the deployment config attached to data.
rules	A named list of grouping rules. If NULL, taken from analysis$groups in the attached config.
species_col	Column holding the common name. Default "Common Name".
group_col	Name of the column to add. Default "group".

Details

A custom classifier covering more than birds gives no direct signal about what a class *is*; the genus in the labels file is the signal. Rules are written per deployment because the relevant split differs by project - cicadas versus birds in one, native versus introduced in another.

Each rule may list genera (matched against the genus), species (matched against the common name, for one-off cases the genus can't express), or default: true. Exactly one group should be the default; everything unmatched lands there. Rules are applied in the order written, so an earlier group wins a tie.

Value

data with an added group_col column, a factor whose levels follow the order the groups were declared in.

See Also

read_detector_labels, group_colours

Examples

```
## Not run:
cfg <- read_deployment("deployments/smiths_lake/deployment.yml")
d <- read_deployment_detections(cfg) |> assign_groups()
table(d$group)

## End(Not run)
```

---

bird_net_data *Example birdnet dataset*

Description

An example dataset for using functions within the birdnetprocess package

Usage

```
bird_net_data
```

Format

A data frame with 15 columns and 1000 rows

Selection Unique ID for each record (integer)

View Spectrogram value (character)

Channel Channel number (numeric)

Begin Time (s) Number of seconds after start_time that recording began (numeric)

End Time (s) Number of seconds after start_time that recording ended (numeric)

Low Freq (Hz) Minimum frequency range (numeric)

High Freq (Hz) Maximum frequency range (numeric)

Common Name Common name of species (character)

Species Code Unique species code (character)

Confidence Confidence level of identification (numeric)

Begin Path File path to .wav file used for identification (character)

File Offset (s) Seconds from .wav file start to recording time (numeric)

file_name File path to .txt file used for df creation (character)

start_time Start time of file (POSIXct)

recording_window_time Time of recording start (POSIXct)

Source

Example birdnet data

Examples

```
head(bird_net_data)
```

---

birdnet_palette *Categorical colours for plotting*

Description

Returns `n` colours from the package's categorical palette, in a fixed order. The order is the point: it was chosen so that neighbouring series stay distinguishable to readers with colour-vision deficiency, so take the colours from the front rather than picking favourites.

Usage

```
birdnet_palette(n = NULL, all_pairs = FALSE)
```

Arguments

`n`	Number of colours. Defaults to the whole palette.
`all_pairs`	Use the stricter all-pairs-safe set? Default `FALSE`.

Details

Two orders are available. The default suits series that are read against their neighbours - bars, stacked areas, lines. `all_pairs = TRUE` returns a smaller set that holds up when *every* pair may be compared at once, such as coloured species labels scattered down a figure; past four such colours, no ordering of this palette separates reliably, so fold the rest into a residual group instead of asking for more.

Value

A character vector of hex colours, named by hue.

Examples

```
birdnet_palette(3)
birdnet_palette(4, all_pairs = TRUE)
```

`birdnet_report` *Render a deployment report*

Description

Turns a deployment config into a single self-contained HTML report: what was recorded, where, with which detector, and what it heard. Every figure is also written as a standalone PNG for reuse in talks and papers.

Usage

```
birdnet_report(
  cfg,
  output_dir = NULL,
  data = NULL,
  filename = NULL,
  quiet = TRUE
)
```

Arguments

`cfg`	A `birdnet_deployment`, from `read_deployment`, or a path to a `deployment.yml`.
`output_dir`	Directory for the report. Defaults to a `report/` folder beside the config file.
`data`	Optional pre-read detections. If `NULL`, read from the config.
`filename`	Output file name. Defaults to the deployment name, slugified.
`quiet`	Suppress Quarto's render output? Default `TRUE`.

Details

Everything the report needs comes from the config, so the same command regenerates the report after more recordings arrive or the threshold changes. Detections are read via `read_deployment_detections` unless you pass them in - useful if you want to modify them first.

Figures are rendered on a light background regardless of the reader's theme, since a PNG cannot restyle itself for a dark page.

Requires the `quarto` R package and a Quarto installation.

Value

The path to the rendered HTML, invisibly.

See Also

`read_deployment`, `site_report` for the simpler single-site PNG output.

Examples

```
## Not run:
cfg <- read_deployment("deployments/smiths_lake/deployment.yml")
birdnet_report(cfg)

## End(Not run)
```

`birdnet_theme` *A shared ggplot theme for report figures*

Description

Recessive chart furniture - hairline grid, muted axis text, no panel border - so that every figure in a report looks like it came from the same place.

Usage

```
birdnet_theme(base_size = 12)
```

Arguments

`base_size`	Base font size. Default `12`.

Value

A ggplot2 theme object.

Examples

```
## Not run:
plot_species_counts(d) + birdnet_theme()

## End(Not run)
```

`density_by_session` *Detection density per dawn and dusk session*

Description

Collapses an hourly `detection_density()` table into one row per site, date and session, where hours before noon are `"dawn"` and the rest `"dusk"`. Suits recorders scheduled around sunrise and sunset; it is not meaningful for continuous recording.

Usage

```
density_by_session(dens)
```

Arguments

`dens`	Output of `detection_density(..., unit = "hour")`.

Value

A tibble: `Site`, `date`, `session`, `hours`, `detections`, `rate`.

`detection_density` *Detection density: detections per recorded hour, with zeros kept*

Description

Counts detections at or above a confidence threshold per site and time bin, and divides by how much audio was actually recorded in that bin. Bins that were recorded but had no detections are kept as zeros, which is what separates "quiet" from "not recording" in any time-series plot.

Usage

```
detection_density(
  df,
  confidence = 0.5,
  unit = c("day", "hour"),
  species = NULL,
  effort = NULL,
  window_s = NULL
)
```

Arguments

df	Detections from read_birdnet_file(), read_birdnet_folder() or read_birdnet_sites().
confidence	Minimum confidence (0-1) for a window to count as a detection. Default 0.5.
unit	Time bin: "day" (default) or "hour".
species	Optional Common Name values to keep. NULL keeps all; each species gets its own rows.
effort	Optional data frame of recorded effort with columns Site, time (matching unit) and hours. Needed when df is not an all-window table.
window_s	Analysis window length in seconds, used to turn window counts into hours. Default: taken from the data (end - begin), else 3.

Details

Effort. Recording effort can only be measured from an *all-window* table - one where every analysis window has a row, whatever its score, as written by an inference run with no confidence floor (for example the Parquet score stores read by read_birdnet_file()). Effort is then the number of distinct windows per site and bin times the window length.

An ordinary thresholded selection table only lists windows that scored highly, so effort cannot be derived from it. Pass an effort table instead (columns Site, time and hours), for example from a recording manifest. If no effort is given and the lowest score in df is above 0.05, the table looks thresholded and the function warns.

Sites come from a Site column, or else from the filename prefix (everything before the _YYYYMMDD_HHMMSS stamp).

Time bins. unit = "day" gives one row per site-day (time is a Date); unit = "hour" gives one row per site-hour (time is a POSIXct at the start of the hour), which suits time-of-day views. Times are the clock times in the recording filenames.

Value

A tibble with one row per site, time bin and species: Site, time, Common Name, hours (recorded), detections and rate (detections per recorded hour).

Examples

```
## Not run:
d <- read_birdnet_folder("scores", pattern = "\\.parquet$")
dens <- detection_density(d, confidence = 0.9)

## End(Not run)
```

`diel_density` *Average diel pattern around sunrise and sunset*

Description

Folds every recording day onto time relative to sunrise and sunset, and measures detections per recorded hour in short bins. Pools all days, so it shows the *average* shape of calling around dawn and dusk, with the drift of sunrise through a season removed. Suits deployments that record windows set relative to the sun (e.g. one hour either side of sunrise and sunset).

Usage

```
diel_density(
  df,
  sites,
  tz,
  confidence = 0.5,
  bin_min = 5,
  species = NULL,
  window_s = NULL,
  by_date = FALSE
)
```

Arguments

`df`	Detections from `read_birdnet_folder()` etc., all-window.
`sites`	Data frame with columns `Site`, `latitude` and `longitude`. Every site in `df` needs a row.
`tz`	Time zone of the recorder clocks, e.g. `"Australia/Sydney"`.
`confidence`	Minimum confidence (0-1) for a detection. Default 0.5.
`bin_min`	Bin width in minutes. Default 5.
`species`	Optional `Common Name` values to keep.
`window_s`	Window length in seconds. Default: taken from the data.
`by_date`	Keep each date separate instead of pooling days? Default `FALSE`. With `TRUE` the result has a `date` column, for relating activity in a sun-relative window to that day's conditions.

Details

Each window is assigned to whichever of that day's sunrise or sunset is nearer, and its offset from that event is binned. Sun times come from the `suncalc` package using each site's own coordinates.

Clock time zone. `tz` must be the zone the recorder clocks were set to, which is not always local civil time: a recorder in one zone can be programmed in another. Check it against the recording schedule before trusting the offsets.

Effort. As in `detection_density()`, effort is counted from the windows themselves, so `df` must be an all-window table (every window has a row, whatever its score).

Value

A tibble: `Site`, (`date` if `by_date`), `anchor` (`"sunrise"` or `"sunset"`), `offset_min` (bin start, minutes relative to the event), `days` (days contributing), `hours` (recorded), `detections` and `rate` (detections per recorded hour).

Examples

```
## Not run:
coords <- data.frame(Site = "SiteA", latitude = -33.5, longitude = 151.2)
diel <- diel_density(scores, coords, tz = "Australia/Sydney", confidence = 0.9)
plot_diel_density(diel, confidence = 0.9)

## End(Not run)
```

`export_top_clips` *Export the highest-confidence audio clips for each species*

Description

Cuts the top-scoring detections of every species out of the source recordings and writes them to a folder, one subfolder per species. This is the fastest way to check whether a detector is right: a species table says a bird was heard 200 times, and five clips say whether it was that bird.

Usage

```
export_top_clips(
  data,
  cfg,
  output_dir,
  n_per_species = 5,
  pad = 1,
  species = NULL,
  n_species = NULL,
  flag_species = character(0)
)
```

Arguments

`data`	Detections, e.g. from `read_deployment_detections`. Needs `Common Name`, `Confidence`, and either `Begin Path` or `file_name` to identify the source recording.
`cfg`	A `birdnet_deployment`, used to find the recordings.
`output_dir`	Folder to write clips into. Created if needed.
`n_per_species`	Number of clips per species. Default `5`.
`pad`	Seconds of context before and after the detection. Default `1`.
`species`	Optional character vector limiting which species to export.

n_species	Export only the n_species most-detected species. NULL (the default) exports every species. Ignored if species is given.
flag_species	Species to mark as unvalidated on the listening sheet and sort to the top. Default none.

Details

Verification is the step a detection report cannot do for you, and the friction of finding the audio by hand is usually what stops it happening. A species' best clips are also its most flattering, so treat this as a first-pass sanity check — if the top five are wrong, the species is wrong; if they are right, the marginal ones still need listening to.

Clip filenames carry the confidence, site and timestamp, so a folder can be skimmed without opening the manifest.

Source recordings are located by file name against each site's recordings folder in the config, not by the path recorded in the detections. Detector runs analyse a temporary folder of symlinks that is deleted afterwards, so the path stored with a detection is generally dead by the time anyone wants the audio.

Requires ffmpeg on the PATH.

Value

A tibble describing every clip written, invisibly. Also written to clips.csv in output_dir. The number of species in the report it was drawn from is attached as attr(x, "n_species_total").

Examples

```
## Not run:
cfg <- read_deployment("deployments/wild_deserts/deployment.yml")
d <- read_deployment_detections(cfg)
export_top_clips(d, cfg, "report/clips", n_per_species = 5)

## End(Not run)
```

---

fetch_openmeteo_weather

Fetch hourly historical weather from Open-Meteo

Description

Downloads hourly weather for one location from the Open-Meteo historical archive (ERA5 reanalysis and its higher-resolution land product). No account or key is needed. Reanalysis is gridded at roughly 10-25 km, so it suits wind, temperature and pressure better than patchy convective rain; prefer a local gauge for rainfall where one exists.

Usage

```
fetch_openmeteo_weather(
  latitude,
  longitude,
  start,
  end,
  tz,
 variables = c("temperature_2m", "precipitation", "wind_speed_10m", "wind_gusts_10m",
    "surface_pressure", "cloud_cover")
)
```

Arguments

latitude, longitude	Location in decimal degrees.
start, end	First and last date (Date or "YYYY-MM-DD").
tz	Time zone for the returned clock times. Use the zone the recorder clocks were set to, so weather lines up with recording times.
variables	Open-Meteo hourly variable names.

Value

A tibble with time (clock time in tz, carried as UTC like the recording times elsewhere in this package) and one column per variable, plus attributes grid_latitude and grid_longitude for the grid cell actually used.

Examples

```
## Not run:
w <- fetch_openmeteo_weather(-33.5, 151.2, "2024-01-01", "2024-01-31",
                             tz = "Australia/Sydney")

## End(Not run)
```

---

filter_by_range *Filter BirdNET detections to species likely present at a location*

Description

Removes detections of species whose modelled range does **not** cover the recorder's location, using a species-distribution-model (SDM) raster stack. This is aimed at the geographic false positives that BirdNET commonly produces (e.g. eastern/coastal species proposed at an arid inland site).

Usage

```
filter_by_range(
  data,
  abundance_raster,
  latitude,
  longitude,
  min_abundance = 0,
  name_overrides = NULL,
  keep_species = NULL,
  species_col = "Common Name"
)
```

Arguments

`data`	A tibble of BirdNET detections (e.g. from `read_birdnet_folder`).
`abundance_raster`	A `terra` `SpatRaster` of per-species abundance layers, or a file path to one. Requires the `terra` package.
`latitude, longitude`	Numeric. The recorder location, in decimal degrees (WGS84 / EPSG:4326).
`min_abundance`	Numeric. Species with a modelled abundance at or below this value are also removed. Default `0` (range-only behaviour: keep anything within the modelled range).
`name_overrides`	Optional named character vector mapping common names to layer names, for species the automatic matcher misses, e.g. `c("Some Bird" = "some_bird_layer")`.
`keep_species`	Optional character vector of common names to always keep, regardless of their modelled range. Useful for nomadic or vagrant species (e.g. waterbirds that irrupt into arid sites after rain) that the range-masked SDM wrongly excludes. Reported with status `"kept: manual override"`.
`species_col`	Name of the column holding common names. Default `"Common Name"`.

Details

The abundance raster is expected to be a multi-layer stack with one range-masked layer per species, named in lower-case underscore style (e.g. `zebra_finch`, `noisy_miner`). A cell value of `NA` is interpreted as "outside the modelled range" — the convention used by range-masked SDMs.

Each detected species is classified as:

- **kept: within modelled range** — a matching layer exists and the site has a (non-`NA`) abundance value above `min_abundance`.
- **removed: outside modelled range** — a matching layer exists but the site value is `NA`.
- **removed: below threshold** — a matching layer exists, the value is non-`NA` but `<= min_abundance` (only possible when `min_abundance > 0`).
- **unassessed: no range map** — no matching layer (non-birds such as insects/mammals/anthropogenic labels, and birds absent from the stack, e.g. arid specialists). These are always **kept**.

Only species in the "removed" categories are dropped, so the function never removes a species it cannot assess.

Value

The filtered data tibble. A per-species report is attached as attr(result, "range_report"), a tibble with columns species, n_detections, max_confidence (the highest detection confidence for that species), matched_layer, abundance, and status.

Examples

```
## Not run:
r <- terra::rast("nsw_abundance_stack_3km.tif")
d <- read_birdnet_folder("wilddeserts")
filtered <- filter_by_range(d, r, latitude = -29.23, longitude = 141.29)
attr(filtered, "range_report")

## End(Not run)
```

---

group_colours *Colours for taxon groups*

Description

Pulls the per-group colour entries out of a set of grouping rules into a named vector suitable for plot_daily_activity's group_colours argument. Groups without a declared colour fall back to the package palette, so a config need only name the colours it cares about.

Usage

```
group_colours(rules)
```

Arguments

rules	A named list of grouping rules, or a birdnet_deployment.

Details

Group colours are drawn from the stricter all-pairs-safe palette (see birdnet_palette), because groups usually appear as species labels scattered down a figure where any two may be compared, not as neighbouring bars.

A group named "Other" gets the muted residual grey unless the config says otherwise: it is a leftover bin rather than a category anyone is meant to track, and giving it a hue implies more than it means.

Value

A named character vector of colours, one per group.

See Also

assign_groups, birdnet_palette

new_deployment *Create a starter deployment configuration*

Description

Copies the annotated `deployment.yml` template shipped with the package into `dir`, as the starting point for a new deployment. The template documents every field, so it is easier to edit down than to write from scratch.

Usage

```
new_deployment(dir, name = NULL, overwrite = FALSE)
```

Arguments

`dir`	Directory to write `deployment.yml` into. Created if needed.
`name`	Optional deployment name to substitute into the template.
`overwrite`	Overwrite an existing `deployment.yml`? Default `FALSE`.

Value

The path to the written file, invisibly.

Examples

```
## Not run:
new_deployment("deployments/smiths_lake", name = "Smiths Lake phenology")

## End(Not run)
```

parse_birdnet_filename_datetime *Extract a start datetime from a BirdNET selection table filename*

Description

This function assumes a filename of the form: SOMETHING_YYYYMMDD_HHMMSS.BirdNET.selection.table.txt

Usage

```
parse_birdnet_filename_datetime(file_name)
```

Arguments

`file_name`	The full name or path of the BirdNET selection file.

Value

A POSIXct object representing the start datetime.

Examples

```
parse_birdnet_filename_datetime("1STSMM2_20241105_050000.BirdNET.selection.table.txt")
```

`plot_activity_vs_covariate` *Activity against one covariate, one panel per site*

Description

Each point is one site's session: detections per recorded hour against a covariate such as a temperature anomaly. Coloured by session, with a fitted curve per site and session from an overdispersed (quasi-Poisson) log-linear model of counts with recorded hours as the offset, i.e. the same shape of relationship a count model assumes. The y-axis is on a `log(1 + x)` scale so sessions with no detections stay on the plot.

Usage

```
plot_activity_vs_covariate(
  data,
  x,
  x_label = x,
  sites = NULL,
  free_y = TRUE,
  xref = 0,
  curves = NULL,
  confidence = NULL,
  title = NULL,
  subtitle = NULL
)
```

Arguments

`data`	Sessions joined to the covariate: needs `Site`, `session`, `detections`, `hours`, `rate` and the column named by `x`. Typically `density_by_session()` joined to `session_anomaly()` output.
`x`	Name of the covariate column.
`x_label`	Axis label for the covariate.
`sites`	Sites to show, in panel order. Default: all in `data`.
`free_y`	Own y-scale per site? Default `TRUE`.
`xref`	Draw a reference line at this x (e.g. 0 for an anomaly), or `NULL` for none. Default 0.

curves	Optional data frame of fitted lines to draw instead of the per-site fits, with columns Site, session, x and rate (e.g. predictions from a mixed model), so the figure shows the model reported.
confidence	Threshold used in detection_density(), for the axis label.
title, subtitle	Plot title and subtitle.

Value

A ggplot object.

---

plot_activity_vs_weather

Activity against weather, session by session

Description

One panel per weather variable: each point is one site's dawn or dusk session, detections per recorded hour against the conditions in that session, with a smooth per session. A first look for weather-driven calling. It cannot separate a change in calling from a change in detectability: wind and rain also mask calls and alter false positives.

Usage

```
plot_activity_vs_weather(
  sessions,
  weather,
  sites = NULL,
  vars = c("wind_speed_10m", "temperature_2m", "rain_day", "pressure_change_24h",
    "moon_illumination", "cloud_cover"),
  date_range = NULL,
  log_rate = TRUE,
  confidence = NULL,
  title = NULL,
  subtitle = NULL
)
```

Arguments

sessions	Output of density_by_session().
weather	Output of weather_by_session().
sites	Sites to include. Default: all in sessions.
vars	Weather columns to plot against.
date_range	Optional two dates to restrict to (e.g. a breeding period).
log_rate	Plot log10(rate + 1)? Default TRUE, as activity is heavily skewed.
confidence	Threshold used in detection_density(), for the axis label.
title, subtitle	Plot title and subtitle.

Value

A ggplot object.

`plot_confidence` *Plot Confidence*

Description

Creates a streamgraph showing confidence of recorded calls over time.

Usage

```
plot_confidence(df, confidence = 0, bw = 0.75)
```

Arguments

df	The dataframe created by the read_birdnet_file or read_birdnet_folder function.
confidence	Optional - the minimum confidence level for the bird call identifications.
bw	Bandwidth for the density estimation (default: 0.75).

Value

A streamgraph indicating confidence of calls over time.

Examples

```
## Not run:
plot_confidence(df, confidence = 0)

## End(Not run)
```

`plot_daily_activity` *Plot daily (diel) activity heatmap*

Description

Folds every recording day onto a single 24-hour clock and draws a heatmap of calling activity by time of day, one row per species and one panel per site. Each species row is scaled to its own peak so the plot shows *when* a species calls rather than how often. This is useful for comparing diel (within-day) vocal patterns across species and sites.

Usage

```
plot_daily_activity(
  df,
  site = NULL,
  species = NULL,
  confidence = 0.5,
  date = NULL,
  min_detections = 1,
  exclude = NULL,
  group_col = NULL,
  group_colours = NULL,
  lat = NULL,
  lon = NULL,
  tz = "UTC",
  bin_min = 20,
  layout = c("rows", "columns"),
  title = NULL,
  subtitle = NULL,
  fill_palette = NULL
)
```

Arguments

df	A data frame of BirdNET results, ideally from `read_birdnet_folder` or `read_birdnet_sites`. Must contain `recording_window_time`, `Common Name` and `Confidence`. If a `Site` column is absent it is derived from the `file_name` prefix (everything before the `_YYYYMMDD_HHMMSS` stamp).
site	Optional character vector of site names to include. `NULL` (default) keeps every site; each site becomes its own facet panel.
species	Optional character vector of `Common Name` values to include. `NULL` (default) keeps every species.
confidence	Numeric. Minimum BirdNET confidence (0-1) to retain. Defaults to 0.5 (below this is mostly false positives).
date	Controls the time window. `NULL` (default) averages every day in the recording, folding them onto one 24-hour clock (each species row scaled to its own peak). Supply a single date (a `Date` or "YYYY-MM-DD" string) to instead plot just that one 24-hour period.
min_detections	Integer. Drop any species with fewer than this many detections (after the confidence/site/date filters) so the figure is not cluttered by one-off detections. Counted per site, so a species is shown wherever it clears the threshold. Defaults to 1 (keep everything).
exclude	Optional character vector of `Common Name` values to drop (e.g. environment/noise classes or unreliable species).
group_col	Optional name of a column that classifies each species into a group (e.g. `"group"` with values like Birds / Insects). When supplied, the y-axis species labels are coloured by group.

group_colours	Optional named vector mapping group -> colour (e.g. c(Birds = "#3b82c4", Insects = "#e9a33b")). Groups without an entry fall back to grey. Ignored unless group_col is set; defaults to a hue palette.
lat, lon	Optional latitude/longitude. When both are supplied (and the suncalc package is installed) the representative sunrise/sunset for the deployment is shaded as night bands, and species are ordered into a nocturnal block then a diurnal block.
tz	Time zone used to convert sunrise/sunset to local time of day. Defaults to "UTC".
bin_min	Numeric. Time-of-day bin width in minutes. Defaults to 20.
layout	Site panel arrangement: "rows" (default) stacks sites vertically, each panel repeating the species axis; "columns" places sites side-by-side sharing a single species axis (no repetition).
title, subtitle	Optional plot title / subtitle. When NULL (default) sensible text is generated from date and whether night shading is drawn.
fill_palette	Optional vector of colours for the activity fill, passed to ggplot2::scale_fill_gradientn(), lightest first. Defaults to NULL, meaning the viridis "magma" ramp. Pass birdnet_palette()'s sequential colours to match the rest of a deployment report.

Details

Times are folded by taking seconds since local midnight from recording_window_time and carrying them on an arbitrary reference date so the x-axis is a clean 24-hour time axis. Species are ordered by daylight fraction (when sunrise/sunset are known) and then by the circular mean of their call time, so the midnight wrap-around does not scatter night-active species.

Value

A ggplot object. Save it with ggplot2::ggsave(); using device = ragg::agg_png is recommended for crisp text.

Examples

```
## Not run:
d <- read_birdnet_sites(c("detections_SiteA", "detections_SiteB"))

# One site
plot_daily_activity(d, site = "SiteA", lat = -33.4, lon = 151.4,
                    tz = "Australia/Sydney")

# All sites at once, faceted
plot_daily_activity(d)

# A single 24h period from a longer deployment, selected species only
plot_daily_activity(d, site = "SiteA", date = "2024-01-15",
                    species = c("Laughing Kookaburra", "Grey Fantail"))

## End(Not run)
```

plot_density *Detection density over time: four alternative views*

Description

Draw the output of `detection_density()` over time, one row per site.

- `plot_density_bars()` - one small panel per site, a bar per day, shared y-axis. Reads exact values; good for a handful of sites.
- `plot_density_heatmap()` - sites as rows, days as columns, colour = density. The most compact; good for many sites and for seeing which days light up together.
- `plot_density_ridges()` - overlapping silhouettes, one per site, on a shared scale. Emphasises the shape and timing of bursts.
- `plot_density_cumulative()` - cumulative detections per site. Bursts show as steep steps; the final height ranks sites.

For time-of-day patterns use `plot_density_actogram()`.

Usage

```
plot_density_bars(
  dens,
  sites = NULL,
  date_range = NULL,
  marks = NULL,
  mark_label = "marked day",
  confidence = NULL,
  title = NULL,
  subtitle = NULL,
  ncol = 1
)

plot_density_heatmap(
  dens,
  sites = NULL,
  date_range = NULL,
  marks = NULL,
  mark_label = "marked day",
  confidence = NULL,
  title = NULL,
  subtitle = NULL
)

plot_density_ridges(
  dens,
  sites = NULL,
```

```
  date_range = NULL,
  confidence = NULL,
  title = NULL,
  subtitle = NULL,
  overlap = 2.5
)

plot_density_cumulative(
  dens,
  sites = NULL,
  date_range = NULL,
  highlight = NULL,
  confidence = NULL,
  title = NULL,
  subtitle = NULL
)
```

Arguments

dens	Output of detection_density() (unit = "day" or "hour"; hourly tables are summed to days). Should hold one species.
sites	Site order, top to bottom. Default: order of appearance. Sites listed with no rows are drawn as never recording.
date_range	Two dates bounding the x-axis. Days in range with no rows are drawn as "no recording". Default: the range of the data.
marks	Optional data frame with columns Site and time (dates) marking particular site-days.
mark_label	Legend text for marks.
confidence	Threshold used in detection_density(), for the axis label.
title, subtitle	Plot title and subtitle.
ncol	Panels per row for plot_density_bars(). Default 1, so every site shares one time axis.
overlap	Ridge height, as a multiple of the row spacing, for the densest day. Default 2.5.
highlight	For plot_density_cumulative(): sites to colour and label directly (at most four); the rest are drawn in grey. Default: the four with the most detections.

Value

A ggplot object.

Examples

```
## Not run:
dens <- detection_density(scores, confidence = 0.9)
plot_density_heatmap(dens, confidence = 0.9)

## End(Not run)
```

plot_density_actogram *Time-of-day activity across a season (actogram)*

Description

One panel per site: date along the x-axis, hour of day up the y-axis, colour = detections per recorded hour in that hour. Shows whether calling sits at dawn or dusk and how that shifts through a burst. Hours never recorded are left blank. Only hours that were ever recorded get a row, so a recording schedule such as dawn and dusk only stays compact.

Usage

```
plot_density_actogram(
  dens,
  sites = NULL,
  confidence = NULL,
  title = NULL,
  subtitle = NULL,
  ncol = 2
)
```

Arguments

dens	Output of detection_density() with unit = "hour".
sites	Sites to draw, in order. Default: every site with a detection.
confidence	Threshold used in detection_density(), for the axis label.
title, subtitle	Plot title and subtitle.
ncol	Panels per row. Default 2.

Value

A ggplot object.

plot_density_weather *Detection heatmap with weather strips beneath*

Description

The plot_density_heatmap() view on top, and one strip per weather variable underneath on the same date axis, so days of high activity can be read against the conditions on those days. Session variables are drawn as a dawn line and a dusk line; rain_day and moon_illumination once per day.

Usage

```
plot_density_weather(
  dens,
  weather,
  sites = NULL,
  date_range = NULL,
  vars = c("wind_speed_10m", "temperature_2m", "rain_day", "pressure_change_24h",
    "moon_illumination"),
  marks = NULL,
  mark_label = "marked day",
  confidence = NULL,
  title = NULL,
  subtitle = NULL,
  heights = 3
)
```

Arguments

dens	Output of detection_density().
weather	Output of weather_by_session().
sites	Site order, top to bottom. Default: order of appearance. Sites listed with no rows are drawn as never recording.
date_range	Two dates bounding the x-axis. Days in range with no rows are drawn as "no recording". Default: the range of the data.
vars	Weather columns to show, top to bottom.
marks	Optional data frame with columns Site and time (dates) marking particular site-days.
mark_label	Legend text for marks.
confidence	Threshold used in detection_density(), for the axis label.
title, subtitle	Plot title and subtitle.
heights	Relative height of the heatmap to each strip. Default 3.

Value

A patchwork object (print or ggsave() it like a ggplot).

---

plot_diel_density *Plot the average diel pattern around sunrise and sunset*

Description

One row per site, a sunrise panel and a sunset panel, bars of detections per recorded hour against minutes from the event. A vertical line marks the event itself.

Usage

```
plot_diel_density(
  diel,
  sites = NULL,
  min_days = 5,
  free_y = TRUE,
  shade_dark = TRUE,
  confidence = NULL,
  title = NULL,
  subtitle = NULL
)
```

Arguments

diel	Output of diel_density().
sites	Sites to draw, in order. Default: every site with a detection.
min_days	Drop bins recorded on fewer than this many days, which are noisy edges where the schedule drifted. Default 5.
free_y	Give each site its own y-scale? Default TRUE, because the point is the shape of each site's pattern; strip labels carry each site's total so a tall bar on a sparse site is not mistaken for a strong one.
shade_dark	Shade the side of each panel where the sun is down (before sunrise, after sunset)? Default TRUE.
confidence	Threshold used, for the axis label.
title, subtitle	Plot title and subtitle.

Value

A ggplot object.

---

plot_site_map *Map of sites sized by a log-transformed total*

Description

One point per site at its coordinates, sized by log10(total + 1) so that sites differing by orders of magnitude stay readable side by side. Sites with a total of zero and sites that did not record are drawn with their own symbols rather than as tiny points, so "none" and "no data" never look like "a little".

Usage

```
plot_site_map(
  totals,
  value = "detections",
  value_label = "detections",
  labels = TRUE,
  layers = NULL,
  scale_bar = TRUE,
  title = NULL,
  subtitle = NULL
)
```

Arguments

totals	Data frame with columns Site, latitude, longitude and the column named by value. An optional hours column marks sites with no recording (0 or NA).
value	Name of the column to size points by. Default "detections".
value_label	Legend title for point size.
labels	Label each site with its name and total? Default TRUE. Uses ggrepel when installed so labels avoid each other.
layers	Optional list of ggplot layers drawn beneath the points, e.g. geom_sf() layers of rivers or boundaries (these need the sf package).
scale_bar	Add a scale bar? Default TRUE; needs ggspatial.
title, subtitle	Plot title and subtitle.

Value

A ggplot object.

Examples

```
## Not run:
totals <- dplyr::tibble(Site = c("A", "B", "C"), latitude = c(-31, -31.01, -31.02),
                        longitude = c(141.8, 141.81, 141.82),
                        detections = c(1500, 12, 0))
plot_site_map(totals)

## End(Not run)
```

---

plot_species *Plot Specific BirdNET Species*

Description

Plot Specific BirdNET Species

Usage

```
plot_species(
  data,
  species_list,
  confidence_threshold = 0.5,
  unit = "hour",
  facet_by = NULL,
  latitude = NULL,
  longitude = NULL,
  tz = "UTC",
  log_scaling = FALSE,
  ...
)
```

Arguments

data	A data frame containing BirdNET detections.
species_list	Character vector. Exact names of species to plot.
confidence_threshold	Numeric. Minimum confidence. Default 0.5.
unit	Character. Time unit for aggregation, passed to lubridate::floor_date. Default "hour".
facet_by	Character. Optional output column name to facet by (e.g. "Site").
latitude	Numeric. Optional. Latitude for day/night shading (requires suncalc).
longitude	Numeric. Optional. Longitude for day/night shading (requires suncalc).
tz	Character. Timezone for day/night shading alignment. Default "UTC".
log_scaling	Logical. If TRUE, uses a pseudo-log scale for the y-axis (handles zeros elegantly). Default FALSE.
...	Additional arguments passed to internal plotting.

---

plot_species_counts *Plot Species Counts*

Description

Takes BirdNET data and creates a figure representing number of recordings of each species, over the given confidence interval.

Usage

```
plot_species_counts(
  df,
  confidence = 0,
  remove.dominants = FALSE,
  log.scale = TRUE
)
```

Arguments

df	The dataframe created by the read_birdnet_file or read_birdnet_folder function
confidence	The minimum confidence level for the species identifications
remove.dominants	Removes dominant species from plot for cleaner visualisation (deprecated if log.scale is TRUE)
log.scale	Logical. If TRUE (default), uses a log10 scale for the y-axis, which handles dominant species visualization automatically.

Value

A plot showing the number of calls per species + prints removed species if applicable

Examples

```
## Not run:
plot_species_counts(df, 0.5)

## End(Not run)
```

---

plot_species_stream *Plot Species Stream*

Description

Creates a streamgraph showing the number of calls over time for selected species. This visualization allows for the comparison of species prevalence across different times or seasons.

Usage

```
plot_species_stream(
  df,
  confidence = 0,
  bird.names = unique(df$`Common Name`),
  bw = 0.75
)
```

Arguments

df	A data frame representing BirdNET results, ideally created by `read_birdnet_file` or `read_birdnet_folder`. Must contain columns: `start_time`, `Common Name`, and `Confidence`.
confidence	Numeric. The minimum confidence level (0-1) for species identifications to be included. Defaults to 0.

bird.names	Character vector. One or more species names ("Common Name") to visualize. Defaults to all unique species in the provided dataframe.
bw	Numeric. The bandwidth for the kernel density estimation performed by ggstream. Adjusts the smoothness of the stream. Defaults to 0.75.

Details

This function requires the ggstream package to be installed. It filters the data based on the provided confidence threshold and species names, then aggregates the count of calls per day. The resulting plot is a "ridge" type streamgraph.

Value

A ggplot object showing the streamgraph of species calls over time.

Examples

```
## Not run:
# Assuming 'results' is a dataframe from read_birdnet_folder()
# Plot stream for specific species with a confidence threshold
plot_species_stream(results, confidence = 0.5, bird.names = c("Galah", "Magpie-lark"))

## End(Not run)

# Runnable example with mock data
data <- data.frame(
  start_time = as.POSIXct(c(
    "2023-01-01 12:00:00", "2023-01-02 12:00:00",
    "2023-01-01 13:00:00", "2023-01-03 12:00:00"
  )),
  `Common Name` = c("Species A", "Species A", "Species B", "Species B"),
  Confidence = c(0.9, 0.8, 0.95, 0.7),
  check.names = FALSE
)
if (requireNamespace("ggstream", quietly = TRUE)) {
  plot_species_stream(data, confidence = 0.5)
}
```

---

plot_timeline *Plot Timeline*

Description

Creates a simple line graph showing number of calls over time.

Usage

```
plot_timeline(df, confidence = 0, bird_names = unique(df$`Common Name`))
```

Arguments

df	The dataframe created by the read_birdnet_file or read_birdnet_folder function.
confidence	The minimum confidence level for the bird call identifications.
bird_names	String of bird names for which recordings over time will be shown.

Value

A figure showing number of calls over time for the given confidence interval.

Examples

```
## Not run:
plot_timeline(df, 0.5)

## End(Not run)
```

---

plot_top_species *Plot Top N BirdNET Species*

Description

Plot Top N BirdNET Species

Usage

```
plot_top_species(
  data,
  n_top_species = 10,
  confidence_threshold = 0.5,
  unit = "hour",
  facet_by = NULL,
  latitude = NULL,
  longitude = NULL,
  tz = "UTC",
  log_scaling = FALSE,
  ...
)
```

Arguments

data	A data frame containing BirdNET detections.
n_top_species	Integer. Number of top species to show. Default 10.
confidence_threshold	Numeric. Minimum confidence. Default 0.5.

unit	Character. Time unit for aggregation, passed to lubridate::floor_date. Default "hour".
facet_by	Character. Optional output column name to facet by (e.g. "Site").
latitude	Numeric. Optional. Latitude for day/night shading (requires suncalc).
longitude	Numeric. Optional. Longitude for day/night shading (requires suncalc).
tz	Character. Timezone for day/night shading alignment. Default "UTC".
log_scaling	Logical. If TRUE, uses a pseudo-log scale for the y-axis (handles zeros elegantly). Default FALSE.
...	Additional arguments passed to internal plotting.

`read_birdnet_file` *Read a single BirdNET selection table file*

Description

Read a single BirdNET selection table file

Usage

```
read_birdnet_file(file_path, tz = "UTC")
```

Arguments

file_path	Path to a BirdNET file (either tab-delimited Raven selection table or CSV).
tz	Timezone to be used for the start time. Default "UTC".

Details

The function attempts to read the file based on its extension.

- Ends in .csv or .CSV: reads as comma-separated.
- Ends in .parquet: reads a BirdNET results table stored as Parquet (needs the nanoparquet package). These are typically *all-window* score stores, with a row for every analysis window and no confidence floor, which is what detection_density() needs to measure recording effort.
- Otherwise: reads as tab-separated (Raven selection table default).

It also standardizes the start time column:

- If Begin Time (s) exists (Raven), it is used.
- If Start (s) exists (CSV), it is renamed to Begin Time (s) internally (or standardizing to begin_time_s).

Filenames must ideally follow the pattern YYYYMMDD_HHMMSS to allow automatic start_time parsing.

Value

A tibble with all columns from the BirdNET file plus:

- `file_name` for reference
- `start_time` the parsed date-time from the filename
- `recording_window_time` the absolute time of each detection

Examples

```
## Not run:
# Read a Raven selection table
df_raven <- read_birdnet_file("data/SiteA_20240101_120000.BirdNET.selection.table.txt")

# Read a CSV
df_csv <- read_birdnet_file("data/SiteA_20240101_120000.BirdNET.results.csv")

## End(Not run)
```

`read_birdnet_folder` *Read all BirdNET selection files in a folder*

Description

Read all BirdNET selection files in a folder

Usage

```
read_birdnet_folder(
  folder = ".",
  pattern = "BirdNET.*\\.(txt|csv|parquet)$",
  recursive = FALSE
)
```

Arguments

`folder`	A folder path containing BirdNET files.
`pattern`	A regex pattern to match the files. Default matches `.txt`, `.csv` or `.parquet` files that look like BirdNET outputs, e.g. "BirdNET.*\.(txt|csv|parquet)$". If a folder holds the same recordings in more than one format, narrow the pattern so each detection is read once.
`recursive`	Whether to search recursively in subfolders. Default FALSE.

Value

A single tibble combining all data.

Examples

```
## Not run:
all_detections <- read_birdnet_folder("path/to/folder")

## End(Not run)
```

`read_birdnet_sites` *Read BirdNET selection files from multiple sites (folders)*

Description

Read BirdNET selection files from multiple sites (folders)

Usage

```
read_birdnet_sites(
  folder_paths,
  pattern = "BirdNET.*\\.(txt|csv|parquet)$",
  recursive = FALSE
)
```

Arguments

`folder_paths`	A character vector of folder paths, one for each site.
`pattern`	A regex pattern to match the files. Default "BirdNET.*\.(txt|csv|parquet)$".
`recursive`	Whether to search recursively in subfolders. Default FALSE.

Value

A single tibble combining all data. Includes a `Site` column derived from the folder name.

Examples

```
## Not run:
folders <- c("detections_SiteA", "detections_SiteB")
all_sites <- read_birdnet_sites(folders, pattern = "BirdNET.results.csv$")

## End(Not run)
```

read_bom_dwo *Read Bureau of Meteorology "Daily Weather Observations" files*

Description

Parses the monthly CSVs the Australian Bureau of Meteorology publishes for each automatic weather station (product `IDCJDW****`, downloadable for the last ~13 months from `bom.gov.au/climate/dwo/`). Station measurements, as opposed to the gridded reanalysis from `fetch_openmeteo_weather()`.

Usage

```
read_bom_dwo(paths)
```

Arguments

`paths`	One or more monthly CSV files.

Details

Rainfall in these files is the 24 hours **to 9 am** on the stated date, so most of it fell on the previous day. `rain_prev_day_to_9am` keeps BOM's value; `rain_day` re-attributes it to the calendar day it mostly fell on (the next row's value).

Value

A tibble, one row per date, with snake-case columns: `date`, `tmin`, `tmax`, `rain_prev_day_to_9am`, `rain_day`, `gust_max`, `gust_dir`, and `t_9am`, `rh_9am`, `wind_9am`, `wind_dir_9am`, `pressure_9am` with the same for `3pm`. "Calm" winds are 0.

read_deployment *Read a deployment configuration file*

Description

Reads a `deployment.yml` describing one field deployment - where the recorders were, which sites and dates they covered, which detector produced the detections, and how the results should be filtered and grouped. The resulting object is the single input to `read_deployment_detections` and `birdnet_report`.

Usage

```
read_deployment(path)
```

Arguments

`path`	Path to the YAML file, or to a directory containing a `deployment.yml`.

Details

The same file is intended to be read by the detector-running step (in Python) as well as by this package, so that the site list and target dates cannot drift apart between the two halves of the pipeline.

Relative paths in the file are resolved against the directory containing the file itself, not the working directory, so a deployment folder can be moved or checked out anywhere. A leading ~ is expanded.

See new_deployment to write a commented starter file.

The expected structure is:

```
name: Smiths Lake phenology
location:
  latitude: -32.38
  longitude: 152.51
  tz: Australia/Sydney
  place: Smiths Lake, NSW
detector:
  name: pelican0-15
  model: path/to/pelican0-15.tflite
  labels: path/to/pelican0-15_Labels.txt
sites:
  - id: Powerline Strip
    detections: path/to/detections/Powerline Strip
    date: 2026-02-03
analysis:
  confidence: 0.5
```

Value

An object of class `birdnet_deployment`: a list with elements `name`, `location`, `detector`, `sites`, and `analysis`, plus the resolved `path` and `dir` of the file it came from. Each element of `sites` gains `start_after` and `end_before` bounds derived from its `date` (or `start`/`end`).

See Also

new_deployment, read_deployment_detections

Examples

```
## Not run:
cfg <- read_deployment("deployments/smiths_lake/deployment.yml")
cfg

## End(Not run)
```

`read_deployment_detections` *Read all detections described by a deployment config*

Description

Reads the detection tables for every site in a deployment, trims each to that site's date window, and returns one tibble with a `Site` column taken from the config.

Usage

```
read_deployment_detections(
  cfg,
  pattern = detection_file_pattern(),
  recursive = FALSE
)
```

Arguments

`cfg`	A `birdnet_deployment`, from `read_deployment`.
`pattern`	Regex matching the detection files within each site folder. Defaults to both arms' output: BirdNET `.txt`/`.csv` tables and a perch-head `*predictions.csv`.
`recursive`	Search site folders recursively? Default `FALSE`.

Details

This is the config-driven counterpart to `read_birdnet_folder` and `read_birdnet_sites`, and it settles which site a detection belongs to. Elsewhere in the package `Site` is inferred two different ways - `read_birdnet_sites` takes it from the folder basename, while `plot_daily_activity` falls back to the recorder prefix in the file name - and the two disagree whenever the folder is named differently from the recorder. Here the config's site `id` wins, so the label in the report is the one you chose.

Each site is trimmed to its own window, which matters when sites were recorded on different days: a single global date range would either keep another site's day or drop your own.

Species listed in `analysis$exclude` are dropped - noise classes, and birds someone with local knowledge says are not there. That list may be written either as a plain sequence of common names or as a mapping of name to the reason for dropping it; the reasons are carried through to the report, since removing a species is a judgement a reader should be able to see and argue with. What was dropped, and how much of it, comes back as `attr(x, "excluded")`.

The returned tibble also carries the config as an attribute so the report can stamp its provenance. No confidence filtering happens here - the plotting functions each apply `analysis$confidence` themselves.

Value

A tibble of detections with a `Site` column, one row per detection, with the deployment config attached as `attr(x, "deployment")`.

See Also

read_deployment, assign_groups

Examples

```
## Not run:
cfg <- read_deployment("deployments/smiths_lake/deployment.yml")
d <- read_deployment_detections(cfg)
table(d$Site)

## End(Not run)
```

---

read_detector_labels *Read a BirdNET labels file*

Description

Parses the label list that ships alongside a BirdNET classifier - one line per class, in the form `Genus species_Common Name` - into a tibble. The scientific name is the only place a custom classifier records what kind of organism a class is, which is what makes it possible to sort detections into taxon groups (see assign_groups).

Usage

```
read_detector_labels(path)
```

Arguments

path	Path to the labels `.txt` file.

Details

Lines without an underscore are kept with an empty scientific name, so non-taxonomic classes such as noise categories survive rather than silently vanishing.

Value

A tibble with columns `scientific`, `common`, and `genus`.

Examples

```
## Not run:
labels <- read_detector_labels("recognizers/pelican0-15_Labels.txt")
head(labels)

## End(Not run)
```

session_anomaly *Weather anomalies relative to the seasonal norm for each session*

Description

For each session (e.g. dawn and dusk separately), fits a smooth seasonal trend to a weather variable across dates and returns each value's departure from it, in the variable's own units. "2 degrees warmer than a usual dawn at this time of year", rather than a raw temperature that mostly measures the time of day and the season.

Usage

```
session_anomaly(weather, var, df = 3, date_range = NULL)
```

Arguments

weather	Output of weather_by_session() (or any table with date, session and the variable).
var	Column to convert, e.g. "temperature_2m".
df	Degrees of freedom of the natural-spline seasonal trend. Default 3. Use 0 for a plain departure from the session mean.
date_range	Optional two dates: fit the trend over this period only.

Value

weather with an added column <var>_anomaly.

site_report *Build a per-site report and plot set from BirdNET detections*

Description

A convenience wrapper that produces a standard set of outputs for a single recording site: optionally range-filters the detections (removing geographic false positives), summarises them, and builds two plots — a species-counts bar chart and a top-species activity plot with day/night shading. When output_dir is supplied, the plots and a species-filter report are written to disk.

Usage

```
site_report(
  data,
  latitude,
  longitude,
  abundance_raster = NULL,
  confidence = 0.5,
```

```
  output_dir = NULL,
  site_name = NULL,
  n_top_species = 10,
  tz = "Australia/Sydney",
  start_after = NULL,
  end_before = NULL,
  min_abundance = 0,
  name_overrides = NULL,
  keep_species = NULL
)
```

Arguments

data	A tibble of BirdNET detections (e.g. from read_birdnet_folder).
latitude, longitude	Numeric. The recorder location (decimal degrees, WGS84). Used both for range filtering and for day/night shading.
abundance_raster	Optional terra SpatRaster (or path) of per-species abundance layers. When supplied, detections are range-filtered via filter_by_range.
confidence	Numeric. Minimum confidence for the summary and plots. Default 0.5.
output_dir	Optional directory to write the report CSV and plot PNGs to. Created if it does not exist. If NULL, nothing is written.
site_name	Optional label for the site (used in messages / file names).
n_top_species	Number of top species for the activity plots. Default 10.
tz	Time zone for day/night shading. Default "Australia/Sydney".
start_after, end_before	Optional deployment-window bounds. Detections with recording_window_time before start_after or after end_before are dropped. Useful when the recorder was running before it was placed at the site (e.g. test recordings made elsewhere). May be a POSIXct, a Date, or a string such as "2026-05-26" (parsed in UTC, matching the times produced by read_birdnet_file).
min_abundance	Passed to filter_by_range. Default 0.
name_overrides	Passed to filter_by_range.
keep_species	Passed to filter_by_range: common names to always keep regardless of modelled range (e.g. nomadic waterbirds).

Details

The work is delegated to existing package functions: filter_by_range, summarise_detections, plot_species_counts, and plot_top_species. If abundance_raster is NULL, no range filtering is performed and all detections are reported.

Value

Invisibly, a list with elements report (per-species filter report, or NULL if no raster), summary (the summarise_detections tibble), plots (named list of ggplot objects), data_filtered (the filtered detections), and output_dir.

Examples

```
## Not run:
r <- terra::rast("nsw_abundance_stack_3km.tif")
d <- read_birdnet_folder("wilddeserts")
res <- site_report(
  d,
  latitude = -29.229286, longitude = 141.286856,
  abundance_raster = r, tz = "Australia/Sydney",
  output_dir = "wilddeserts_plots/filtered"
)
res$report

## End(Not run)
```

`species_thresholds` *Per-species detection thresholds*

Description

Returns the confidence threshold that applies to each of `species`, taking a species' own calibrated cut where the config has one and the deployment-wide `analysis$confidence` otherwise.

Usage

```
species_thresholds(cfg, species)
```

Arguments

`cfg`	A `birdnet_deployment`, from `read_deployment`.
`species`	Character vector of common names.

Details

A single global threshold treats every species as equally detectable, which no evaluation ever supports: a distinctive call and a faint, easily-confused one need different cuts to reach the same precision. Where a labelled soundscape has been scored, `analysis$confidence_by_species` carries the per-species operating points; everything else falls back to `analysis$confidence`.

Thresholds must be in the same space as the detections' `Confidence` column. Evaluation harnesses often work in **logits** while detectors emit post-sigmoid **probabilities** — a threshold moved between the two without converting looks entirely plausible and silently passes almost everything. Convert before putting values in a config.

Value

A numeric vector of thresholds, parallel to `species`.

See Also

threshold_source

Examples

```
## Not run:
cfg <- read_deployment("deployments/wild_deserts/deployment.yml")
species_thresholds(cfg, c("Brown Songlark", "Budgerigar"))

## End(Not run)
```

---

summarise_detections *Summarise Detections*

Description

Provides a quick summary of BirdNET results, including species counts, total recordings, time range, and average detection rates.

Usage

```
summarise_detections(df, confidence = 0.5)
```

Arguments

df	A dataframe of BirdNET results, typically created by read_birdnet_file or read_birdnet_folder.
confidence	Numeric. The minimum confidence level (0 to 1) for a detection to be included in the summary. Default is 0.5.

Details

This function filters the input data based on the provided confidence threshold and then calculates several summary statistics:

- Number of unique species detected
- Total number of detections (recordings)
- Date range of the recording window
- The most frequently detected species
- The peak hour of activity
- Average number of detections per day
- Average number of detections per hour

It returns a tibble with two columns: statistic (the name of the metric) and value (the calculated value formatted as a string).

Value

A tibble with columns `statistic` and `value`.

Examples

```
## Not run:
# Summarise with default confidence (0.5)
summarise_detections(df)

# Summarise with higher confidence threshold
summarise_detections(df, confidence = 0.8)

## End(Not run)
```

---

threshold_source *Where each species' threshold came from*

Description

Labels each species `"calibrated"`, `"unvalidated"` or `"default"`, so a report can say which detections rest on a measured operating point and which rest on an assumption.

Usage

```
threshold_source(cfg, species)
```

Arguments

`cfg`	A `birdnet_deployment`, from `read_deployment`.
`species`	Character vector of common names.

Details

The distinction that matters is between a species that was *never tested* and one that was *tested and failed*. Both end up on the deployment's default threshold, but they mean opposite things: the first is simply unknown, while the second is a species the evaluation showed the detector could not identify reliably at any cut. Collapsing them into one bucket quietly launders a known-bad species into a plain result, so `analysis$unvalidated_species` names the second group explicitly and the report flags it.

Value

A character vector parallel to `species`.

See Also

`species_thresholds`

validate_deployment *Validate a deployment configuration*

Description

Checks a `birdnet_deployment` for the mistakes that would otherwise surface much later as a confusing plot or an empty report: missing required fields, an out-of-range or transposed latitude/longitude, an unknown time zone, or duplicate site ids.

Usage

```
validate_deployment(cfg)
```

Arguments

`cfg`	A `birdnet_deployment`, from `read_deployment`.

Details

Missing *files* are a warning rather than an error, because a config is normally written before the detector has run and so before the detection folders exist.

Called automatically by `read_deployment`.

Value

`cfg`, invisibly. Called for its errors and warnings.

weather_by_session *Summarise hourly weather over sun-relative recording sessions*

Description

For each date, takes the hours inside a window around sunrise ("dawn") and around sunset ("dusk") and summarises the weather there: what conditions were like while the recorder was listening. Adds moon illumination, and the 24-hour change in pressure (a common cue for weather-driven behaviour).

Usage

```
weather_by_session(
  weather,
  latitude,
  longitude,
  tz,
  before_min = 60,
  after_min = 60
)
```

Arguments

`weather`	Hourly weather, e.g. from `fetch_openmeteo_weather()`, with a `time` column of clock times carried as UTC.
`latitude, longitude`	Location used for sun and moon times.
`tz`	Time zone of the clock times in `weather`.
`before_min, after_min`	Window either side of sunrise and sunset, in minutes. Default 60 each.

Value

A tibble with one row per date and session (`"dawn"`, `"dusk"`): the mean of each numeric weather column over the window (sums for `precipitation`, maxima for `wind_gusts_10m`), plus `rain_day` (that day's total precipitation, when present), `pressure_change_24h` (when `surface_pressure` is present) and `moon_illumination` (0-1).

Index

∗ **datasets**
bird_net_data, 4

apply_thresholds, 3
assign_groups, 3, *15*, *38*

bird_net_data, 4
birdnet_palette, 5, *15*
birdnet_palette(), *21*
birdnet_report, 6, *35*
birdnet_theme, 7

density_by_session, 8
density_by_session(), *17, 18*
detection_density, 8
detection_density(), *8*, *10*, *18*, *22–25*, *32*
diel_density, 10
diel_density(), *26*

export_top_clips, 11

fetch_openmeteo_weather, 12
fetch_openmeteo_weather(), *35*, *45*
filter_by_range, 13, *40*

ggplot2::ggsave(), *21*
ggplot2::scale_fill_gradientn(), *21*
group_colours, *4*, 15

new_deployment, 16, *36*

parse_birdnet_filename_datetime, 16
plot_activity_vs_covariate, 17
plot_activity_vs_weather, 18
plot_confidence, 19
plot_daily_activity, *3*, *15*, 19, *37*
plot_density, 22
plot_density_actogram, 24
plot_density_actogram(), *22*
plot_density_bars *(*plot_density*)*, 22
plot_density_cumulative *(*plot_density*)*, 22
plot_density_heatmap *(*plot_density*)*, 22
plot_density_heatmap(), *24*
plot_density_ridges *(*plot_density*)*, 22
plot_density_weather, 24
plot_diel_density, 25
plot_site_map, 26
plot_species, 27
plot_species_counts, 28, *40*
plot_species_stream, 29
plot_timeline, 30
plot_top_species, 31, *40*

read_birdnet_file, 32, *40*, *42*
read_birdnet_file(), *9*
read_birdnet_folder, *14*, 33, *37*, *40*, *42*
read_birdnet_folder(), *9, 10*
read_birdnet_sites, 34, *37*
read_birdnet_sites(), *9*
read_bom_dwo, 35
read_deployment, *3*, *6, 7*, 35, *37, 38*, *41*, *43, 44*
read_deployment_detections, *4*, *7*, *11*, *35, 36*, 37
read_detector_labels, *4*, 38

session_anomaly, 39
session_anomaly(), *17*
site_report, *7*, 39
species_thresholds, 41, *43*
summarise_detections, *40*, 42

threshold_source, *42*, 43

validate_deployment, 44

weather_by_session, 44
weather_by_session(), *18*, *25*, *39*