Package: birdnetR (via r-universe)

May 22, 2026

Type Package

Title Deep Learning for Automated (Bird) Sound Identification

Version 0.3.2.9000

Description Use 'BirdNET', a state-of-the-art deep learning classifier, to automatically identify (bird) sounds. Analyze bioacoustic datasets without any computer science background using a pre-trained model or a custom trained classifier. Predict bird species occurrence based on location and week of the year. Kahl, S., Wood, C. M., Eibl, M., & Klinck, H. (2021) <doi:10.1016/j.ecoinf.2021.101236>.

License MIT + file LICENSE

URL https://birdnet-team.github.io/birdnetR/, https://github.com/birdnet-team/birdnetR

Depends R (>= 4.1)

Imports reticulate (>= 1.46)

Suggests arrow, curl, devtools, knitr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

BugReports https://github.com/birdnet-team/birdnetR/issues

Config/pak/sysreqs libpng-dev python3

Repository https://birdnet-team.r-universe.dev

Date/Publication 2026-05-22 10:11:12 UTC

RemoteUrl https://github.com/birdnet-team/birdnetr

RemoteRef HEAD

RemoteSha 0dc11e81175d6646a0d1558486a2e2f5e543fd05

Contents

`as.data.frame.birdnet_encoding` *Convert BirdNET encoding results to a data frame*

Description

Convert encoding results from an acoustic model to a data frame. Each row represents one audio segment. The `embedding` column is returned as a list of numeric vectors (one per segment).

Usage

```
## S3 method for class 'birdnet_encoding'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	A BirdNET encoding object (as returned by `encode()`).
`row.names`	`NULL` or a character vector giving the row names for the data frame. Not used.
`optional`	logical. Not used.
`...`	Additional arguments (ignored).

Value

A data frame containing the encoding results. Columns include `input`, `start_time`, `end_time`, and `embedding` (a list column of numeric vectors).

Examples

```
## Not run:
model <- load_birdnet(type = "acoustic")
audio_file <- system.file("extdata", "soundscape.mp3",
  package = "birdnetR"
)
enc <- encode(model, files = audio_file)
df <- as.data.frame(enc)
# Access the first embedding vector
df$embedding[[1]]

## End(Not run)
```

---

as.data.frame.birdnet_prediction

Convert BirdNET prediction results to a data frame

Description

Convert predictions from a geo or acoustic model to a data frame.

Usage

```
## S3 method for class 'birdnet_prediction'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'birdnet_prediction_geo'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	A BirdNET prediction object (as returned by `predict()`).
row.names	NULL or a character vector giving the row names for the data frame. Not used.
optional	logical. Not used.
...	Additional arguments (ignored).

Value

A data frame containing the prediction results.

Examples

```
## Not run:
# Load a BirdNET acoustic model
model <- load_birdnet(type = "acoustic")
# Predict species from audio files
audio_file <- system.file("extdata", "soundscape.mp3", package = "birdnetR")
```

```
predictions <- predict(model, files = audio_file)
# Convert predictions to a data frame
as.data.frame(predictions)

## End(Not run)
```

`birdnet_version` *Report Python and birdnet Version Information*

Description

Returns the Python executable path, Python version, and installed `birdnet` Python package version as a named list. Useful for debugging environment issues.

Usage

```
birdnet_version()
```

Value

A named list with elements:

python_version Character string with the Python version (e.g. `"3.12.3"`), or `NA` if Python is not available.

python_executable Character string with the path to the Python executable, or `NA` if Python is not available.

birdnet_version Character string with the installed `birdnet` package version (e.g. `"0.2.16"`), or `NA` if the package is not installed.

Examples

```
## Not run:
birdnet_version()

## End(Not run)
```

`encode` *Encode audio files using a BirdNET acoustic model*

Description

Run the encoder stage of a BirdNET acoustic model to obtain embedding vectors for each segment of the supplied audio files. This is useful for downstream tasks such as clustering, dimensionality reduction, or transfer-learning workflows that build on BirdNET embeddings.

Usage

```
encode(object, ...)

## S3 method for class 'birdnet_model_acoustic'
encode(
  object,
  files,
  overlap = 0,
  bandpass_fmin = 0L,
  bandpass_fmax = 15000L,
  progress = c("minimal", "progress", "benchmark"),
  n_producers = NULL,
  n_workers = NULL,
  batch_size = NULL,
  prefetch_ratio = NULL,
  speed = NULL,
  half_precision = NULL,
  max_audio_duration_min = NULL,
  ...
)
```

Arguments

`object`	A BirdNET model object of class `birdnet_model_acoustic` created with `load_birdnet()` or `load_perch()`.
`...`	Additional arguments passed to the generic (currently unused).
`files`	A character vector of one or more file paths to audio files. When multiple files are provided the resulting data frame (via `as.data.frame()`) includes an `input` column identifying the source file for each row.
`overlap`	A numeric value specifying the overlap duration in seconds between consecutive time intervals. Must be in the interval [0.0, 3.0].
`bandpass_fmin, bandpass_fmax`	An integer value to set minimum and maximum frequencies for the bandpass filter (in Hz).
`progress`	A character string specifying the type of progress reporting. Options are `"minimal"`, `"progress"`, or `"benchmark"`.
`n_producers`	An integer specifying the number of threads for producing audio batches. Must be >= 1. If `NULL` (default), the Python backend default is used.
`n_workers`	An integer specifying the number of backend workers for parallel inference. If `NULL` (default), the Python backend auto-detects based on available CPU cores.
`batch_size`	An integer specifying the number of audio segments evaluated per inference call. Must be >= 1. If `NULL` (default), the Python backend default is used.
`prefetch_ratio`	An integer specifying how many batches to decode ahead of processing. Must be >= 0. If `NULL` (default), the Python backend default is used.

speed — A numeric value for the resampling multiplier to accommodate different recording speeds. Must be in the interval [0.01, 100]. If NULL (default), the Python backend default is used.

half_precision — A logical value indicating whether to use float16 where supported for inference. If NULL (default), the Python backend default is used.

max_audio_duration_min — A numeric value specifying the maximum total audio duration per call in minutes. Must be > 0. If NULL (default), no limit is applied.

Value

An S3 object of class birdnet_encoding containing the encoding results. Use as.data.frame() to convert to a data frame.

Examples

```
## Not run:
model <- load_birdnet(type = "acoustic")
audio_file <- system.file("extdata", "soundscape.mp3",
  package = "birdnetR"
)
enc <- encode(model, files = audio_file)
as.data.frame(enc)

## End(Not run)
```

---

get_species_list *Get the species list from a loaded BirdNET model*

Description

Returns the full species list from a loaded model as a character vector. Works for both pretrained and custom models.

Usage

```
get_species_list(model)
```

Arguments

model — A BirdNET model object returned by load_birdnet() or load_custom().

Value

A character vector of species names in model order.

Examples

```
## Not run:
model <- load_birdnet(type = "acoustic")
species <- get_species_list(model)
head(species)

## End(Not run)
```

---

load_birdnet_model *Load a BirdNET Model*

Description

Functions to load BirdNET models for sound identification or species prediction from location and time. Models will be downloaded if not available locally.

- load_birdnet(): load a pre-trained BirdNET model or a geographic model.
- load_perch(): load the Perch v2 acoustic model.
- load_custom(): load a custom trained BirdNET model.

Usage

```
load_perch()

load_birdnet(
  type = "acoustic",
  version = "2.4",
  backend = "tf",
  library = NULL,
  precision = "fp32",
  language = "en_us"
)

load_custom(
  type = "acoustic",
  version = "2.4",
  backend = "tf",
  library = NULL,
  precision = "fp32",
  model = NULL,
  species_list = NULL,
  check_validity = TRUE,
  classifier_type = NULL,
  is_raven = NULL
)
```

Arguments

type	character. The type of model to load: "acoustic" or "geo".
version	character. The version of the model to load, e.g., "2.4".
backend	character. The backend to use: "tf" (TensorFlow) or "pb" (Protobuf).
library	character or NULL. The TensorFlow library to use: "litert" or "tflite". Only applies when backend = "tf".
precision	character. The precision of the model: "int8", "fp16", or "fp32".
language	character. Language code for the model to use e.g., "en_us". Common species names are returned in the specified language if available. Use supported_languages() to see all available languages.
model	If backend is "tf", path to custom model file. If backend is "pb", path to model directory.
species_list	Path to the species list file.
check_validity	Checks if the model is loadable by loading the model twice.
classifier_type	Advanced option for custom TensorFlow models. Controls how the custom classifier head is interpreted.
is_raven	Advanced option for custom protobuf models. Indicates whether the model uses the Raven protobuf layout.

Details

The argument type specifies the type of model to load:

- "acoustic": A model for species identification from audio recordings.
- "geo": A model for species prediction based on geographic location and time.

The argument version specifies the version of the model to load, e.g., "2.4". Use supported_model_configurations() to see all supported versions.

The argument backend specifies the backend to use:

- "tf": TensorFlow backend, which supports different precisions and libraries but is CPU only.
- "pb": Protobuf backend, which can be run on a GPU. GPU support requires additional system dependencies; see the upstream birdnet documentation for details.

The argument library is only used for the TensorFlow backend and specifies the library to use:

- "litert": The LiteRT library for running TensorFlow Lite models.
- "tflite": The standard TensorFlow Lite library.

The argument precision specifies the precision of the model:

- "int8": 8-bit integer precision, which is the most efficient in terms of speed and memory usage.
- "fp16": 16-bit floating point precision, which is a good balance between speed and accuracy.
- "fp32": 32-bit floating point precision, which is the most accurate but also the slowest and most memory-intensive. Protobuf and geographic models only support "fp32" precision.

To get an overview of all valid model configuration combinations use supported_model_configurations().

Value

A BirdNET model object (S3 class `birdnet_model` with type- and version-specific subclasses). The returned object supports:

- `predict()` — run inference on audio files or geographic coordinates
- `get_species_list()` — retrieve the species list used by the model

Internal list fields are implementation details and should not be relied upon.

Examples

```
# Load a pre-trained acoustic model
## Not run:
model <- load_birdnet(type = "acoustic", version = "2.4", backend = "tf", precision = "int8")

## End(Not run)
```

---

predict.birdnet_model_acoustic

Predict species from audio files using a BirdNET acoustic model

---

Description

This function predicts species from audio files using a BirdNET acoustic model.

Usage

```
## S3 method for class 'birdnet_model_acoustic'
predict(
  object,
  files,
  min_confidence = 0.1,
  min_confidence_custom = NULL,
  top_k = 5L,
  overlap = 0,
  apply_sigmoid = TRUE,
  sigmoid_sensitivity = 1,
  bandpass_fmin = 0L,
  bandpass_fmax = 15000L,
  species_list = NULL,
  progress = c("minimal", "progress", "benchmark"),
  n_producers = NULL,
  n_workers = NULL,
  batch_size = NULL,
  prefetch_ratio = NULL,
  speed = NULL,
  half_precision = NULL,
```

```
  max_audio_duration_min = NULL,
  ...
)
```

Arguments

object	A BirdNET model object of class `birdnet_model_acoustic` created with `load_birdnet()`.
files	A character vector of one or more file paths to audio files. When multiple files are provided, the returned prediction object will contain results for all files; the resulting data frame (via `as.data.frame()`) includes an `input` column identifying the source file for each prediction row.
min_confidence	A numeric value to set the minimum confidence threshold for predictions.
min_confidence_custom	A named list where each element is a single numeric value to set custom minimum confidence thresholds for specific species. A custom threshold will override the default one.
top_k	An integer specifying the number of top predictions to return for each time interval if above minimum confidence threshold.
overlap	A numeric value specifying the overlap duration in seconds between consecutive time intervals. Must be in the interval [0.0, 3.0].
apply_sigmoid	A logical value indicating whether to apply a sigmoid function to the confidence scores.
sigmoid_sensitivity	A numeric value that adjusts the sensitivity of the sigmoid function. Must be in the interval [0.5, 1.5].
bandpass_fmin, bandpass_fmax	A integer value to set minimum and maximum frequencies for the bandpass filter (in Hz).
species_list	A character vector or list of species names to filter the predictions. If `NULL`, all species are considered.
progress	A character string specifying the type of progress reporting. Options are "minimal", "progress", or "benchmark".
n_producers	An integer specifying the number of threads for producing audio batches. Must be >= 1. If `NULL` (default), the Python backend default is used.
n_workers	An integer specifying the number of backend workers for parallel inference. If `NULL` (default), the Python backend auto-detects based on available CPU cores.
batch_size	An integer specifying the number of audio segments evaluated per inference call. Must be >= 1. If `NULL` (default), the Python backend default is used.
prefetch_ratio	An integer specifying how many batches to decode ahead of processing. Must be >= 0. If `NULL` (default), the Python backend default is used.
speed	A numeric value for the resampling multiplier to accommodate different recording speeds. Must be in the interval [0.01, 100]. If `NULL` (default), the Python backend default is used.
half_precision	A logical value indicating whether to use float16 where supported for inference. If `NULL` (default), the Python backend default is used.

`max_audio_duration_min`
: A numeric value specifying the maximum total audio duration per call in minutes. Must be > 0. If `NULL` (default), no limit is applied.

`...`
: Additional arguments passed to the generic (currently unused).

Details

Sigmoid Activation:

When `apply_sigmoid = TRUE`, the raw logit scores from the linear classifier are passed through a sigmoid function, scaling them into the range [0, 1]. This unitless confidence score reflects BirdNET's certainty in its prediction (it is not a direct probability of species presence). Adjusting the `sigmoid_sensitivity` parameter modifies the score distribution:

- Values **< 1** tend to produce more extreme scores (closer to 0 or 1).
- Values **> 1** result in scores that are more moderate (centered around intermediate values).

For additional details on BirdNET confidence scores and guidelines for converting them to probabilities, see Wood & Kahl (2024).

Value

An S3 object of class `birdnet_prediction_acoustic` and `birdnet_prediction` containing the prediction results.

References

Wood, C. M., & Kahl, S. (2024). Guidelines for appropriate use of BirdNET scores and other detector outputs. Journal of Ornithology. https://doi.org/10.1007/s10336-024-02144-5

Examples

```
## Not run:
# Load a BirdNET acoustic model
model <- load_birdnet(type = "acoustic")
# Predict species from audio files
audio_file <- system.file("extdata", "soundscape.mp3", package = "birdnetR")
predictions <- predict(model, files = audio_file)
# convert predictions to a data frame
as.data.frame(predictions)

## End(Not run)
```

`predict.birdnet_model_geo`
Predict species using a BirdNET geo model

Description

This function predicts species occurence for a location and week of the year using a BirdNET geo model.

Usage

```
## S3 method for class 'birdnet_model_geo'
predict(
  object,
  latitude,
  longitude,
  week = NULL,
  min_confidence = 0.03,
  half_precision = NULL,
  ...
)
```

Arguments

object	A BirdNET model object of class birdnet_model_geo created with load_birdnet().
latitude	A numeric value representing the latitude of the location.
longitude	A numeric value representing the longitude of the location.
week	An integer value representing the week of the year (1-52).
min_confidence	A numeric value to set the minimum confidence threshold for predictions.
half_precision	A logical value indicating whether to use float16 where supported for inference. If NULL (default), the Python backend default is used.
...	Additional arguments passed to the generic (currently unused).

Value

An S3 object of class birdnet_prediction_geo and birdnet_prediction containing the prediction results.

Examples

```
## Not run:
# Load a BirdNET geo model
model <- load_birdnet(type = "geo")
# Predict species for a specific location and week
predictions <- predict(model, latitude = 50.8334, longitude = 12.9231, week = 18L)
# Convert predictions to a data frame
as.data.frame(predictions)

## End(Not run)
```

supported_languages *List supported BirdNET model languages*

Description

Returns a character vector of supported language codes for BirdNET models, as provided by the Python backend.

Usage

```
supported_languages()
```

Value

Character vector of supported language codes (e.g., `"en_us"`, `"de"`, `"es"`).

Examples

```
supported_languages()
```

supported_model_configurations *List supported BirdNET model configurations*

Description

Returns a data frame of valid combinations of model types, versions, backends, libraries, and precisions supported by the underlying BirdNET Python backend. Use this to explore which model settings are available for use in `load_birdnet()` and related functions.

Usage

```
supported_model_configurations(compact = FALSE)
```

Arguments

`compact`	Logical. If `TRUE`, returns a compacted version of the configurations table.

Details

Not all combinations of arguments are valid. For example, the `library` argument is only relevant for TensorFlow backends, and some backends or model types only support specific precisions. Use the output to guide valid choices for loading models.

Perch v2 is not included in this table because it uses a separate loader. Use `load_perch()` to load the Perch v2 acoustic model.

Value

A data frame of valid model configuration combinations. Each row represents a supported configuration.

Examples

```
supported_model_configurations()                    # Show all supported combinations
supported_model_configurations(compact = TRUE)      # Show compacted version
```

`write_predictions` *Write BirdNET prediction results to file*

Description

Write prediction results to disk using the underlying Python result object. The format can be inferred from the file extension or specified explicitly.

Usage

```
write_predictions(x, file, format = NULL, ...)

## S3 method for class 'birdnet_prediction_geo'
write_predictions(x, file, format = NULL, ...)

## S3 method for class 'birdnet_prediction'
write_predictions(x, file, format = NULL, ...)
```

Arguments

`x`	A BirdNET prediction object (as returned by `predict()`).
`file`	Output file path.
`format`	Output file format. If `NULL` (default), inferred from the file extension. Explicit values: `"csv"`, `"parquet"`, `"npz"`.
`...`	Additional arguments passed to the underlying Python method.

Details

Supported formats by prediction type:

- **Acoustic predictions**: CSV (`.csv`), Parquet (`.parquet`), NumPy (`.npz`)
- **Geo predictions**: CSV (`.csv`), NumPy (`.npz`)

Value

Invisibly returns the output file path.

Examples

```
## Not run:
write_predictions(pred, "results.csv")
write_predictions(pred, "results.parquet")
write_predictions(pred, "results.npz")
write_predictions(pred, "output.csv", format = "csv")

## End(Not run)
```

Index