Package 'animl'

May 7, 2026

Title A Collection of ML Tools for Conservation Research

Version 3.2.0

Description Functions required to classify subjects within camera trap field data. The package can handle both images and videos. The authors recommend a two-step approach using Microsoft's 'MegaDector' model and then a second model trained on the classes of interest.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports methods, pbapply, reticulate, stats,

Depends R (>= 4.0.0)

NeedsCompilation no

Author Kyra Swanson [aut, cre] (ORCID:
<https://orcid.org/0000-0002-1496-3217>),
Mathias Tobler [aut]

Maintainer Kyra Swanson <tswanson@sdzwa.org>

Repository CRAN

Date/Publication 2026-02-04 01:00:02 UTC

Contents

`animl_install` *Load animl-py if available*

Description

Load animl-py if available

Usage

```
animl_install(envname = "animl_env", python_version = "3.12")
```

Arguments

envname	name of python environment
python_version	version of python to install

Value

animl-py module

Examples

```
## Not run: animl_py <- load_animl_py()
```

---

animl_install_instructions *Installation Instructions for animl-r Python dependencies*

Description

Installation Instructions for animl-r Python dependencies

Usage

```
animl_install_instructions()
```

---

build_file_manifest *File Management Module*

Description

This module provides functions and classes for managing files and directories.

Usage

```
build_file_manifest(
  image_dir,
  exif = TRUE,
  out_file = NULL,
  offset = 0,
  recursive = TRUE
)
```

Arguments

image_dir	folder to search through and find media files
exif	returns date and time information from exif data, defaults to true
out_file	.csv file to save manifest as
offset	add offset in hours for videos when using the File Modified date, defaults to 0
recursive	Should directories be scanned recursively? Default TRUE

Details

Kyra Swanson 2023 Find Image/Video Files and Gather exif Data

Value

files dataframe with or without file dates

Examples

```
## Not run:
files <- build_file_manifest("C:\\Users\\usr\\Pictures\\")

## End(Not run)
```

---

check_file *Check for files existence and prompt user if they want to load*

Description

Check for files existence and prompt user if they want to load

Usage

```
check_file(file, output_type)
```

Arguments

file	the full path of the file to check
output_type	str to specify file name in prompt description

Value

a boolean indicating wether a file was found and the user wants to load or not

Examples

```
## Not run:
  check_file("path/to/newfile.csv")

## End(Not run)
```

check_python *Check that the python version is compatible with the current version of animl-py*

Description

Check that the python version is compatible with the current version of animl-py

Usage

```
check_python(python_version = "3.12", initialize = TRUE)
```

Arguments

python_version	version of python to install
initialize	load reticulate library if true

Value

none

Examples

```
## Not run: check_python(initialize=FALSE)
```

classify *Infer Species for Given Detections*

Description

Infer Species for Given Detections

Usage

```
classify(
  model,
  detections,
  resize_width = 480,
  resize_height = 480,
  file_col = "filepath",
  crop = TRUE,
  normalize = TRUE,
  batch_size = 1,
  num_workers = 1,
  device = NULL,
  out_file = NULL
)
```

Arguments

model	loaded classifier model
detections	manifest of animal detections
resize_width	image width input size
resize_height	image height input size
file_col	column in manifest containing file paths
crop	use bbox to crop images before feeding into model
normalize	normalize the tensor before inference
batch_size	batch size for generator
num_workers	number of processes
device	send model to the specified device
out_file	path to csv to save results to

Value

detection manifest with added prediction and confidence columns

Examples

```
## Not run: animals <- classify(classifier, animals, file_col='filepath')
```

---

compute_batched_distance_matrix

Computes the distance matrix in a batched manner to save memory.

Description

Computes the distance matrix in a batched manner to save memory.

Usage

```
compute_batched_distance_matrix(
  input1,
  input2,
  metric = "cosine",
  batch_size = 10
)
```

Arguments

input1	2-D array of query features
input2	2-D array of database features
metric	The distance metric to use. Options include 'euclidean', 'cosine', etc
batch_size	The number of rows from input1 to process at a time

Value

distance matrix

Examples

```
## Not run:
dist_matrix <- compute_batched_distance_matrix(query_embeddings, database_embeddings,
                                                metric='cosine', batch_size=12)
## End(Not run)
```

---

compute_distance_matrix

A wrapper function for computing distance matrix.

Description

A wrapper function for computing distance matrix.

Usage

```
compute_distance_matrix(input1, input2, metric = "euclidean")
```

Arguments

input1	2-D feature matrix
input2	2-D feature matrix
metric	"euclidean" or "cosine", Default is "euclidean"

Value

distance matrix

Examples

```
## Not run: dist_matrix <- compute_distance_matrix(embeddings, embeddings, metric='cosine')
```

cosine_distance *Computes cosine distance of two sets of vectors*

Description

Computes cosine distance of two sets of vectors

Usage

```
cosine_distance(input1, input2)
```

Arguments

input1	2-D feature matrix
input2	2-D feature matrix

Value

distance matrix

Examples

```
## Not run: dist_matrix <- cosine_distance(embeddings, embeddings)
```

create_pyenv *Install python if necessary and create the environment animl_env*

Description

Install python if necessary and create the environment animl_env

Usage

```
create_pyenv(envname = "animl_env", python_version = "3.12")
```

Arguments

envname	name of the conda environment to create / use (default "animl-py")
python_version	python version to add to environment

Value

invisible TRUE on success, otherwise stops or returns FALSE invisibly on failure

delete_pyenv *Delete the animl_env environment*

Description

Delete the animl_env environment

Usage

```
delete_pyenv(envname = "animl_env")
```

Arguments

envname	python environment to remove

Value

none

Examples

```
## Not run: delete_pyenv('animl_env')
```

detect *Apply MegaDetector to a Given Batch of Images*

Description

Apply MegaDetector to a Given Batch of Images

Usage

```
detect(
  detector,
  image_file_names,
  resize_width,
  resize_height,
  letterbox = TRUE,
  confidence_threshold = 0.1,
  file_col = "filepath",
  batch_size = 1,
  num_workers = 1,
  device = NULL,
  checkpoint_path = NULL,
  checkpoint_frequency = -1
)
```

Arguments

detector	preloaded md model
image_file_names	list of image filenames, a single image filename, or folder
resize_width	width to resize images to
resize_height	height to resize images to
letterbox	if True, resize and pad image to keep aspect ratio, else resize without padding
confidence_threshold	only detections above this threshold are returned
file_col	select which column if image_file_names is a manifest
batch_size	size of each batch
num_workers	number of processes to handle the data
device	specify to run on cpu or gpu
checkpoint_path	path to checkpoint file
checkpoint_frequency	write results to checkpoint file every N images

Value

list of dictionaries of MegaDetector detections

Examples

```
## Not run: mdres <- detect(md_py, allframes$Frame, 1280, 960, device='cpu')
```

---

download_model *Download specified model to the given directory.*

Description

Download specified model to the given directory.

Usage

```
download_model(model_url, out_dir = "models")
```

Arguments

model_url	url of the model to download, obtained via the constants above
out_dir	Directory to save the model.

Value

None

Examples

```
## Not run:
  list_models()
  download_model("https://models.com/path/to/model.pt", out_dir='models')

## End(Not run)
```

`euclidean_squared_distance` *Computes euclidean squared distance of two sets of vectors*

Description

Computes euclidean squared distance of two sets of vectors

Usage

```
euclidean_squared_distance(input1, input2)
```

Arguments

`input1`	2-D feature matrix
`input2`	2-D feature matrix

Value

distance matrix

Examples

```
## Not run: dist_matrix <- euclidean_squared_distance(embeddings, embeddings)
```

`export_camtrapR` *Export data into sorted folders organized by station*

Description

Export data into sorted folders organized by station

Usage

```
export_camtrapR(
  manifest,
  out_dir,
  out_file = NULL,
  label_col = "prediction",
  file_col = "filepath",
  station_col = "station",
  unique_name = "uniquename",
  copy = FALSE
)
```

Arguments

manifest	dataframe containing images and associated predictions
out_dir	directory to export sorted images
out_file	if provided, save the manifest to this file
label_col	column containing species labels
file_col	column containing source paths
station_col	column containing station names
unique_name	column containing unique file name
copy	if true, hard copy

Value

manifest with link column

Examples

```
## Not run: manifest <- export_camtrapR(manifest, out_dir, out_file=NULL, label_col='prediction',
                                        file_col="filepath", station_col='station',
                                        unique_name='uniquename', copy=FALSE)
## End(Not run)
```

---

export_coco *Converts the .csv file to a COCO-formatted .json file.*

Description

Converts the .csv file to a COCO-formatted .json file.

Usage

```
export_coco(manifest, class_list, out_file, info = NULL, licenses = NULL)
```

Arguments

manifest	dataframe containing images and associated detections
class_list	dataframe containing class names and their corresponding IDs
out_file	path to save the formatted file
info	info section of COCO file, named list
licenses	licenses section of COCO file, array

Value

coco formated json

Examples

```
## Not run: export_megadetector(manifest, output_file= 'results.json', detector='MDv6')
```

---

export_folders *Create SymLink Directories and Sort Classified Images*

Description

Create SymLink Directories and Sort Classified Images

Usage

```
export_folders(
  manifest,
  out_dir,
  out_file = NULL,
  label_col = "prediction",
  file_col = "filepath",
  unique_name = "uniquename",
  copy = FALSE
)
```

Arguments

manifest	DataFrame of classified images
out_dir	Destination directory for species folders
out_file	if provided, save the manifest to this file
label_col	specify 'prediction' for species or 'category' for megadetector class
file_col	Colun containing file paths
unique_name	Unique image name identifier
copy	Toggle to determine copy or hard link, defaults to link

Value

manifest with added link columns

Examples

```
## Not run:
manifest <- export_folders(manifest, out_dir)

## End(Not run)
```

`export_megadetector` *Converts the .csv file to the MD-formatted .json file.*

Description

Converts the .csv file to the MD-formatted .json file.

Usage

```
export_megadetector(
  manifest,
  out_file = NULL,
  detector = "MegaDetector v5a",
  prompt = TRUE
)
```

Arguments

`manifest`	dataframe containing images and associated detections
`out_file`	path to save the MD formatted file
`detector`	name of the detector model used
`prompt`	ask user to overwrite existing file

Value

None

Examples

```
## Not run: export_megadetector(manifest, output_file= 'results.json', detector='MDv6')
```

export_timelapse *Converts the Manifests to a csv file that contains columns needed for TimeLapse conversion in later step*

Description

Converts the Manifests to a csv file that contains columns needed for TimeLapse conversion in later step

Usage

```
export_timelapse(manifest, out_dir, only_animal = TRUE)
```

Arguments

manifest	a DataFrame that has entries of anuimal classification
out_dir	location of root directory where all images are stored (can contain subdirectories)
only_animal	A bool that confirms whether we want only animal detctions or all

Value

animals.csv, non-anim.csv, csv_loc

Examples

```
## Not run: export_timelapse(animals, empty, '/path/to/images/')
```

extract_frames *Extract frames from video for classification*

Description

Extract frames from video for classification

Usage

```
extract_frames(
  files,
  frames = 5,
  fps = NULL,
  out_file = NULL,
  out_dir = NULL,
  file_col = "filepath",
  parallel = TRUE,
  num_workers = 4
)
```

Arguments

files	dataframe of videos
frames	number of frames to sample
fps	frames per second, otherwise determine mathematically
out_file	csv file to which results will be saved
out_dir	directory to save frames to if not null
file_col	string value indexing which column contains file paths
parallel	Toggle for parallel processing, defaults to FALSE
num_workers	number of processors to use if parallel, defaults to 4

Value

dataframe of still frames for each video

Examples

```
## Not run:
frames <- extract_frames(manifest, out_dir = "C:\\Users\\usr\\Videos\\", frames = 5)

## End(Not run)
```

---

extract_miew_embeddings

Extract Embeddings from MiewID

Description

Extract Embeddings from MiewID

Usage

```
extract_miew_embeddings(
  miew_model,
  manifest,
  file_col = "filepath",
  batch_size = 1,
  num_workers = 1,
  device = NULL
)
```

Arguments

miew_model	loaded miewid model
manifest	list of files
file_col	column name containing file paths
batch_size	batch size for generator
num_workers	number of workers for generator
device	device to run model on

Value

matrix of embeddings

Examples

```
## Not run: embeddings = extract_embeddings(miew, manifest)
```

---

get_animals *Return a dataframe of only MD animals*

Description

Return a dataframe of only MD animals

Usage

```
get_animals(manifest)
```

Arguments

manifest	all megadetector frames

Value

animal frames classified by MD

Examples

```
## Not run:
animals <- get_animals(imagesall)

## End(Not run)
```

get_empty *Return MD empty, vehicle and human images in a dataframe*

Description

Return MD empty, vehicle and human images in a dataframe

Usage

```
get_empty(manifest)
```

Arguments

manifest	all megadetector frames

Value

list of empty/human/vehicle allframes with md classification

Examples

```
## Not run:
empty <- get_empty(imagesall)

## End(Not run)
```

get_frame_as_image *Given a video path, return a specific frame as an RGB image*

Description

Given a video path, return a specific frame as an RGB image

Usage

```
get_frame_as_image(video_path, frame = 0)
```

Arguments

video_path	path to video file
frame	frame number to extract (default is 0)

Value

rgb_frame: extracted frame as RGB image

Examples

```
## Not run: get_frame_as_image('/example/path/to/video.mp4', frame=213)
```

list_models *List available models for download.*

Description

List available models for download.

Usage

```
list_models()
```

Value

printout of models

Examples

```
## Not run:
  list_models()
  download_model("https://models.com/path/to/model.pt", out_dir='models')

## End(Not run)
```

load_animl *Load animl-py if available*

Description

Load animl-py if available

Usage

```
load_animl(envname = "animl_env", python_version = "3.12")
```

Arguments

envname	name of python environment
python_version	version of python to install

Value

none

Examples

```
## Not run: animl_install("animl_env", ANIML_VERSION, python_version="3.12")
```

load_classifier *Load a Classifier Model and Class_list*

Description

Load a Classifier Model and Class_list

Usage

```
load_classifier(model_path, classes, device = NULL, architecture = "CTL")
```

Arguments

model_path	path to model
classes	path to class list or loaded class list
device	send model to the specified device
architecture	model architecture

Value

classifier model, class list

Examples

```
## Not run:
classes <- load_class_list('sdzwa_andes_v1_classes.csv')
andes <- load_classifier('andes_v1.pt', nrow(classes))
## End(Not run)
```

load_class_list *Load class list .csv file*

Description

Load class list .csv file

Usage

```
load_class_list(classlist_file)
```

Arguments

classlist_file	path to class list

Value

dataframe version of csv

Examples

```
## Not run: classes <- load_class_list('andes_classes.csv')
```

---

load_data *Load .csv or .Rdata file*

Description

Load .csv or .Rdata file

Usage

```
load_data(file)
```

Arguments

file	the full path of the file to load

Value

data extracted from the file

Examples

```
## Not run:
  loadData("path/to/newfile.csv")

## End(Not run)
```

---

load_detector *Load an Object Detector*

Description

Load an Object Detector

Usage

```
load_detector(model_path, model_type, device = NULL)
```

Arguments

model_path	path to detector model file
model_type	type of model expected ie "MDV5", "MDV6", "YOLO", "ONNX"
device	specify to run on cpu or gpu

Value

detector object

Examples

```
## Not run: md_py <- megadetector("/mnt/machinelearning/megaDetector/md_v5a.0.0.pt",
                                  model_type='mdv5', device='cuda:0')
## End(Not run)
```

---

load_json *Load data from a JSON file.*

Description

Load data from a JSON file.

Usage

```
load_json(file)
```

Arguments

file	the full path of the file to load

Value

loaded json file

Examples

```
## Not run:
mdraw <- load_json('mdraw.json')

## End(Not run)
```

load_miew *Load MiewID model*

Description

Load MiewID model

Usage

```
load_miew(file_path, device = NULL)
```

Arguments

file_path	path to model weights
device	device to load model to

Value

meiwid model

Examples

```
## Not run: miew = load_miewid("miewid_v3.bin")
```

parse_detections *Parse MD results into a simple dataframe*

Description

Parse MD results into a simple dataframe

Usage

```
parse_detections(
  results,
  manifest = NULL,
  out_file = NULL,
  threshold = 0,
  file_col = "filepath"
)
```

Arguments

results	json output from megadetector
manifest	optional dataframe containing all frames
out_file	optional path to save dataframe
threshold	confidence threshold to include bbox
file_col	column in manifest that refers to file paths

Value

original dataframe including md results

Examples

```
## Not run:
mdresults <- parseMD(mdres)

## End(Not run)
```

---

plot_all_bounding_boxes

Plot all bounding boxes in a manifest

Description

Plot all bounding boxes in a manifest

Usage

```
plot_all_bounding_boxes(
  manifest,
  out_dir,
  file_col = "filepath",
  min_conf = 0.1,
  label_col = FALSE,
  show_confidence = FALSE,
  colors = NULL,
  detector_labels = NULL
)
```

Arguments

manifest	manifest of detections
out_dir	Name of the output directory
file_col	Column name containing file paths
min_conf	Confidence threshold to plot the box

label_col — Column name containing class to print above the box. If None, no label is printed.

show_confidence — If true, show confidence score above the box.

colors — Named list mapping class labels to BGR colors for the bounding boxes.

detector_labels — Named list mapping detector categories to human-readable labels.

Value

None

Examples

```
## Not run: plot_all_bounding_boxes(manifest, 'Plots/', label_col='prediction',
                                    show_confidence=TRUE,
                                    colors=list("1" = c(0, 255, 0),
                                                "2" = c(0, 0, 255),
                                                "3" = c(255, 0, 0)))
## End(Not run)
```

---

plot_box *Plot bounding boxes on image from md results*

Description

Plot bounding boxes on image from md results

Usage

```
plot_box(
  rows,
  file_col = "filepath",
  min_conf = 0,
  label_col = NULL,
  show_confidence = FALSE,
  colors = NULL,
  detector_labels = NULL,
  return_img = FALSE
)
```

Arguments

rows — row or rows of images in which the bounding box will be plotted

file_col — Column name containing file paths

min_conf — minimum confidence to plot box

label_col	Column name containing class to print above the box. If None, no label is printed.
show_confidence	If true, show confidence score above the box.
colors	Named list mapping class labels to BGR color tuples for the bounding boxes.
detector_labels	Named list mapping detector categories to human-readable labels.
return_img	If true, return the image array with boxes overlaid, otherwise display it

Value

no return value, produces bounding box in plot panel

Examples

```
## Not run:
test_image <- classify(classifier_model, test_image, file_col='filepath')
plot_box(test_image, file_col='filepath', minconf = 0.5, prediction=TRUE)

## End(Not run)
```

---

remove_diagonal *Removes the diagonal elements from a square matrix.*

Description

Removes the diagonal elements from a square matrix.

Usage

```
remove_diagonal(A)
```

Arguments

A	square matrix

Value

martix A with diagonals removed

Examples

```
## Not run: cleaned_dist <- remove_diagonal(dist_matrix)
```

remove_link *Remove Sorted Links*

Description

Remove Sorted Links

Usage

```
remove_link(manifest, link_col = "link")
```

Arguments

manifest	DataFrame of classified images
link_col	column in manifest that contains link paths

Value

manifest without link column

Examples

```
## Not run:
remove_link(manifest)

## End(Not run)
```

save_classifier *Save model state weights*

Description

Save model state weights

Save model state weights

Usage

```
save_classifier(
  model,
  out_dir,
  epoch,
  stats,
  optimizer = NULL,
  scheduler = NULL
)
```

```
save_classifier(
  model,
  out_dir,
  epoch,
  stats,
  optimizer = NULL,
  scheduler = NULL
)
```

Arguments

model	pytorch model
out_dir	directory to save model to
epoch	current training epoch
stats	performance metrics of current epoch
optimizer	pytorch optimizer (optional)
scheduler	pytorch scheduler (optional)

Value

None

None

Examples

```
## Not run: save_classifier(model, 'models/', 10, list(acc = 0.85))
## Not run: save_classifier(model, 'models/', 10, list(acc = 0.85))
```

---

save_data *Save Data to Given File*

Description

Save Data to Given File

Usage

```
save_data(data, out_file, prompt = TRUE)
```

Arguments

data	the dataframe to be saved
out_file	the full path of the saved file
prompt	if true, prompts the user to confirm overwrite

Value

none

Examples

```
## Not run:
 saveData(files,"path/to/newfile.csv")

## End(Not run)
```

---

save_json *Save data to a JSON file.*

Description

Save data to a JSON file.

Usage

```
save_json(data, out_file, prompt = TRUE)
```

Arguments

data	the dictionary to be saved
out_file	full path to save file to
prompt	prompt user to confirm overwrite

Value

None

Examples

```
## Not run:
save_json(mdresults, 'mdraw.json')

## End(Not run)
```

`sequence_classification` *Leverage sequences to classify images*

Description

This function applies image classifications at a sequence level by leveraging information from multiple images. A sequence is defined as all images at the same camera/station where the time between consecutive images is <=maxdiff. This can improve classification accuracy, but assumes that only one species is present in each sequence. If you regularly expect multiple species to occur in an image or sequence don't use this function.

Usage

```
sequence_classification(
  animals,
  empty,
  predictions_raw,
  classes,
  station_col = "station",
  empty_class = "",
  human_class = "",
  vehicle_class = "",
  sort_columns = NULL,
  file_col = "filepath",
  maxdiff = 60
)
```

Arguments

`animals`	sub-selection of all images that contain MD animals
`empty`	optional, data frame non-animal images (empty, human and vehicle) that will be merged back with animal imagages
`predictions_raw`	data frame of prediction probabilities from the classifySpecies function
`classes`	class list associated with classifier model
`station_col`	a column in the animals and empty data frame that indicates the camera or camera station
`empty_class`	a string indicating the class that should be considered 'Empty'
`human_class`	a string indicating the class that should be considered 'Human'
`vehicle_class`	a string indicating the class that should be considered 'Vehicle'
`sort_columns`	optional sort order. The default is 'station_column' and datetime.
`file_col`	a field indicating a single record. The default is FilePath for single images/videos.
`maxdiff`	maximum difference between images in seconds to be included in a sequence, defaults to 60

Details

This function retains "Empty" classification even if other images within the sequence are predicted to contain animals. Classification confidence is weighted by MD confidence.

Value

data frame with predictions and confidence values for animals and empty images

Examples

```
## Not run:
predictions_raw <-classify(classifier, images, resize_width=456, resize_height=456)
animals <- get_animals(images)
empty <- get_empty(images)
animals <- sequence_classification(animals, empty, predictions_raw, classes,
                                   station_column="StationID",
                                   empty_class = "Empty",
                                   sort_columns = c("StationID", "DateTime"),
                                   maxdiff=60)

## End(Not run)
```

---

single_classification *Get Maximum likelihood label for each Detection*

Description

Get Maximum likelihood label for each Detection

Usage

```
single_classification(
  animals,
  empty,
  predictions_raw,
  class_list,
  best = FALSE
)
```

Arguments

animals	manifest of animal detections
empty	manifest of md human, vehicle and empty images
predictions_raw	softmaxed likelihoods from predict_species
class_list	list of class labels
best	whether to return one prediction per file

Value

dataframe with prediction and confidence columns

Examples

```
## Not run: animals <- single_classification(animals, empty, pred_raw, class_list)
```

---

test_main *Test a model with a Config file*

Description

Test a model with a Config file

Usage

```
test_main(cfg)
```

Arguments

cfg	config .yml file containing test settings

Value

None

Examples

```
## Not run: test_main('training_cfg.yml')
```

---

train_main *Model Training*

Description

This module provides functions for training and testing classifier models

Usage

```
train_main(cfg)
```

Arguments

cfg	config .yml file containing training settings

Details

Kyra Swanson 2025 Train a model with a Config file

Value

None

Examples

```
## Not run: train_main('training_cfg.yml')
```

`train_val_test` *Splits the manifest into training validation and test datasets for training*

Description

Splits the manifest into training validation and test datasets for training

Usage

```
train_val_test(
  manifest,
  label_col = "class",
  file_col = "filepath",
  conf_col = "confidence",
  out_dir = NULL,
  val_size = 0.1,
  test_size = 0.1,
  seed = 42
)
```

Arguments

`manifest`	list of files to split for training
`label_col`	column name containing class labels
`file_col`	column containing file paths
`conf_col`	column containing prediction confidence
`out_dir`	location to save split lists to
`val_size`	fraction of data dedicated to validation
`test_size`	fraction of data dedicated to testing
`seed`	RNG seed for reproducibility

Value

train manifest, validate manifest, test manifest

Examples

```
## Not run:
  output <- train_val_test(manifest)

## End(Not run)
```

`update_animl_py` *Update animl-py version for the given environment*

Description

Update animl-py version for the given environment

Usage

```
update_animl_py(envname = "animl_env")
```

Arguments

`envname`	name of python environment

Value

None

Examples

```
## Not run: update_animl_py(py_env = "animl_env")
```

`update_labels_from_folders` *Udate Results from File Browser*

Description

Udate Results from File Browser

Usage

```
update_labels_from_folders(manifest, export_dir, unique_name = "uniquename")
```

Arguments

`manifest`	dataframe containing file data and predictions
`export_dir`	directory to sort files into
`unique_name`	column containing unique file names

Value

dataframe with new "Species" column that contains the verified species

Examples

```
## Not run:
results <- update_labels_from_folders(manifest, export_dir)

## End(Not run)
```

`WorkingDirectory` *Set Working Directory and Save File Global Variables*

Description

Set Working Directory and Save File Global Variables

Usage

```
WorkingDirectory(workingdir, pkg.env)
```

Arguments

`workingdir`	local directory that contains data to process
`pkg.env`	environment to create global variables in

Value

None

Examples

```
## Not run:
WorkingDirectory("/home/kyra/animl/examples",globalenv())

## End(Not run)
```

Index