

Package: camalienr (via r-universe)

October 1, 2026

Title Functions Used in the Plant Module of the Biodiversa+ Pilot on
Invasive Alien Species

Version 0.6.0

Description Tools used in the work with the vehicle mounted CamAlien
system for monitoring plants. The package provides functions to
interact with the CamAlien database.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

URL <https://gitlab.au.dk/ecos/biodiversa/camalien/camalienr>

BugReports <https://gitlab.au.dk/ecos/biodiversa/camalien/camalienr/-/issues>

Imports DBI, dbplyr, dplyr, fs, glue, jsonlite, magick, purrr,
RPostgres, sf, tibble, tidyr

Depends R (>= 3.5)

LazyData true

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libmagick++-
dev gsfonts libicu-dev libuv1-dev libssl-dev libpq-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository <https://ldalby.r-universe.dev>

Date/Publication 2026-09-01 11:17:06 UTC

RemoteUrl <https://gitlab.au.dk/ecos/biodiversa/camalien/camalienr>

RemoteRef HEAD

RemoteSha 99d412910f9cb46fe084c924bbc86dcb1f02b633

Contents

ca_as_sf	2
ca_connect	3
ca_disconnect	4

ca_get_image	4
ca_get_join_tbl	5
ca_get_polygon	6
ca_get_session_tbl	7
ca_get_tbl	7
ca_image_model	9
ca_img_detection	9
ca_img_draw_polygon	10
ca_img_grid	11
ca_img_plot	11
ca_img_read	12
ca_json_read	12
ca_parse_chunks	13
ca_parse_job	13
ca_stat_images	14
ca_stat_jobsize	14
ca_stat_tagged	15
ias	15
id_box	16
images	18
partners	18

Index	19
--------------	-----------

ca_as_sf	<i>Convert data.frame to Simple Feature collection</i>
----------	--

Description

Convert data.frame to Simple Feature collection by converting the position column (pg_geometry) to sfc.

Usage

ca_as_sf(tbl)

Arguments

tbl The data.frame to be converted

Value

tbl as Simple Feature collection

ca_connect	<i>Connect to the camalien database</i>
------------	---

Description

Establish a connection to a camalien database. This function is a wrapper around [DBI::dbConnect](#). The function will create a connection using [RPostgres::Postgres](#) as a driver. For the function to work you need to have the following environment variables set in your .REnviron file:

- CAMALIEN_HOST
- CAMALIEN_USER
- CAMALIEN_PASSWORD

Usage

```
ca_connect(host = NULL, dbname = NULL, port = NULL)
```

Arguments

host	character The host to connect to. Defaults to the CAMALIEN_HOST environment variable.
dbname	character The name of the database. Defaults to the CAMALIEN_DB environment variable, or camalien when that is not set.
port	The port to connect on. Defaults to the CAMALIEN_PORT environment variable, or 5432 when that is not set.

Details

All three are required and the function fails with an informative error if any of them are missing. There is deliberately no default host, so you never connect to a database you did not ask for. The database name and the port default to camalien and 5432 respectively, and can be changed with the optional CAMALIEN_DB and CAMALIEN_PORT variables. Set-but-empty variables count as unset. Arguments given to the function take precedence over the environment, which is handy for a one-off connection to another camalien database.

To modify or create an .REnviron file we suggest you use [usethis::edit_r_environ](#). Calling this function from RStudio will open the file for you to edit. Simply insert the variables, save and restart R.

Value

PqConnection

Examples

```
## Not run:
# Use the target given by the environment variables:
con <- ca_connect()

# Override it for a single connection:
con <- ca_connect(host = "ecos-postgis-snapshot.cjqc9gl8vdbe.eu-north-1.rds.amazonaws.com")

## End(Not run)
```

ca_disconnect	<i>Disconnect from the database</i>
---------------	-------------------------------------

Description

This is a simple wrapper around [DBI::dbDisconnect](#) to use when disconnecting from a database. It is good practice to disconnect when no longer using the connection.

Usage

```
ca_disconnect(con)
```

Arguments

con	DBIConnection The connection object which in the case of camalien will be a PqConnection.
-----	---

Value

TRUE, invisibly

ca_get_image	<i>Get tables from the camalien database</i>
--------------	--

Description

Get tables from the data schema of the camalien database. These functions only return a lazy table and will not download any data to your session. Use [dplyr::collect](#) when you actually want to download the data. Normally you would want to do some selection and filtering before calling `collect()`

Usage

```
ca_get_image(con)

ca_get_job(con)

ca_get_plantnetcall(con)

ca_get_detection(con)

ca_get_detectionssummary(con)

ca_get_imagemeta(con)

ca_get_species(con)

ca_get_chunk(con)

ca_get_partner(con)

ca_get_imagefile(con)

ca_get_imagedata(con)

ca_get_additionalimagedata(con)

ca_get_session(con)
```

Arguments

con PqConnection The connection to the camalien database

Details

ca_get_image(), ca_get_imagemeta() and ca_get_detectionssummary() reach tables that only exist on hosts using the v1 image model, and ca_get_imagefile(), ca_get_imagedata(), ca_get_additionalimagedata and ca_get_session() reach the v2 tables that replaced them. See [ca_get_tbl\(\)](#) for the mapping between the two, and [ca_image_model\(\)](#) to ask a connection which one it carries.

Value

A lazy table

ca_get_join_tbl	Create and return a table for filtering rows to a partner
-----------------	---

Description

A common task is to filter e.g. detections made within one of the partner countries. This involves a number of joins to carry the partner name all the way through to the image or detection table. This function sets up the required joins and can filter the resulting table to a single partner by supplying the partner_name argument.

Usage

```
ca_get_join_tbl(con, partner_name = NULL)
```

Arguments

- con PqConnection The connection to the camalien database
- partner_name chr Name of the partner

Details

This is the ownership chain of the v1 image model, partner → job → chunk, and it is joined to [ca_get_image\(\)](#) on chunkid. On a host using the v2 image model use [ca_get_session_tbl\(\)](#) instead: data2.imagefile still has a chunkid, but it is not filled in for every image, so the chunk chain silently loses rows there.

Value

A lazy table

ca_get_polygon	<i>Extract polygon information from Plantnet response</i>
----------------	---

Description

This function will extract the relevant information from a JSON file returned from the Plantnet API.

Usage

```
ca_get_polygon(pn_json)
```

Arguments

- pn_json The JSON object (read by e.g. jsonlite::read_json)

Value

list of center_x, center_y and size

ca_get_session_tbl	Create and return a session table for filtering rows to a partner
--------------------	---

Description

The v2 counterpart of `ca_get_join_tbl()`. Images in the v2 model belong to a session rather than to a chunk, so ownership runs partner → session → imagedata. Join the result to `ca_get_imagedata()` on sessionid.

Usage

```
ca_get_session_tbl(con, partner_name = NULL)
```

Arguments

con	PqConnection The connection to the camalien database
partner_name	chr Name of the partner

Value

A lazy table with sessionid, session, year, partnerid and partner

Examples

```
## Not run:
con <- ca_connect()

con |>
  ca_get_imagedata() |>
  dplyr::inner_join(ca_get_session_tbl(con, "Denmark"), by = "sessionid")

## End(Not run)
```

ca_get_tbl	Get a table from the camalien database
------------	--

Description

Get any table from any schema of the camalien database. The `ca_get_*`() functions are thin wrappers around this one, and it is the way to reach the tables that have no wrapper of their own — the gbif, tasks and backup schemas, and the parts of data2 that are not about images.

Usage

```
ca_get_tbl(con, table, schema = "data")
```

Arguments

con	PqConnection	The connection to the camalien database
table	character	Name of the table
schema	character	Name of the schema holding the table. Defaults to data.

Details

This function only returns a lazy table and will not download any data to your session. Use [dplyr::collect](#) when you actually want to download the data. Normally you would want to do some selection and filtering before calling collect().

The table layout differs between camalien hosts. Hosts that still carry the original image model keep images in the data schema; hosts that have been migrated retired those tables and hold images in data2 in a different shape:

v1 (data)	v2 (data2)	accessor
image	imagefile	ca_get_imagefile()
imagemeta	imagedata	ca_get_imagedata()
	additionalimagedata	ca_get_additionalimagedata()
	session	ca_get_session()
detectionssummary	detectionssummary	ca_get_tbl(con, "detectionssummary", schema = "data2")

The two models are not interchangeable: imagefile carries the file (filename, path, type, chunkid) and imagedata the capture (position, timestamp, tagged), where v1 had image and imagemeta. Asking for a table the host does not carry therefore fails with an error naming the replacement rather than quietly returning a different table. Use [ca_image_model\(\)](#) to ask which model a connection carries.

Value

A lazy table

Examples

```
## Not run:
con <- ca_connect()

# The same as ca_get_partner(con):
ca_get_tbl(con, "partner")

# Reach a table with no wrapper of its own:
ca_get_tbl(con, "taxon_detection_summary", schema = "data2")

## End(Not run)
```

ca_image_model

Which image model does this connection carry?

Description

Camalien hosts do not all hold images the same way. The original (v1) model keeps them in data.image and data.imagemeta, owned through the partner → job → chunk → image chain. The v2 model retires both tables and splits them into data2.imagefile and data2.imagedata, owned through partner → session → imagedata. Use this to branch when you need code that runs against either.

Usage

```
ca_image_model(con)
```

Arguments

con PqConnection The connection to the camalien database

Value

"v1" or "v2"

Examples

```
## Not run:
con <- ca_connect()
ca_image_model(con)

## End(Not run)
```

ca_img_detection

Plot image and polygon with detection

Description

Draw an image and show the polygon where the detection was made as a polygon on top of the image.

Usage

```
ca_img_detection(data, height = 3000L, border = NA, col = NA, lwd = 4)
```

Arguments

data	data.frame Data.frame with imageUrl, centerx, centery and size.
height	Integer specifying the height of the desired plot. Width is scaled proportionally.
border	chr Polygon outline color
col	chr Color to use if plotting filled polygons. The format is #RRGGBBAA where AA is the alpha channel for transparency
lwd	int Polygon outline line width

Examples

```
## Not run:
images[1,] |>
  ca_img_detection(height = 500, border = "red")

images |>
  ca_img_detection(height = 500, col = "#FF000040")

## End(Not run)
```

ca_img_draw_polygon	<i>Draw polygons on top of images</i>
---------------------	---------------------------------------

Description

Draw a polygon and annotate it with ID

Usage

```
ca_img_draw_polygon(
  center_x,
  center_y,
  size,
  border = "red",
  col,
  lwd = 4,
  scale
)
```

Arguments

center_x, center_y	int coordinates of the centre of the polygon
size	int size of the polygon
border	chr Polygon outline color
col	chr Color to use if plotting filled polygons. The format is #RRGGBBAA where AA is the alpha channel for transparency

lwd	int Polygon outline line width
scale	double Scaling factor if the image is not rendered full size

ca_img_grid	<i>Plot grid of images</i>
-------------	----------------------------

Description

Plot a grid of multiple images.

Usage

```
ca_img_grid(img_paths, ncol = 3, nrow = 3, height = 300L)
```

Arguments

img_paths	Vector of paths (can be URLs). Should be of length nrow * ncol.
ncol, nrow	Int, the number of rows and columns
height	Integer specifying the height of the desired plot. Width is scaled proportionally.

Examples

```
## Not run:
images$imageurl |>
ca_img_grid(ncol = 5, nrow = 1)

## End(Not run)
```

ca_img_plot	<i>Plot image</i>
-------------	-------------------

Description

Plot and optionally scale image

Usage

```
ca_img_plot(img, height = 3000L)
```

Arguments

img	magick-image object read by ca_img_read()
height	Integer specifying the height of the desired plot. Width is scaled proportionally.

ca_img_read	<i>Read image from URL</i>
-------------	----------------------------

Description

Read image file for viewing or further manipulation

Usage

```
ca_img_read(url)
```

Arguments

url	URL to the image to be read.
-----	------------------------------

Value

magick-image object

ca_json_read	<i>Read state files from CamAlien uploader</i>
--------------	--

Description

A simple wrapper around `jsonlite::read_json()`.

Usage

```
ca_json_read(path)
```

Arguments

path	chr Path to the json file to read.
------	------------------------------------

Value

list with content of json file being read

ca_parse_chunks	<i>Parse json files with info on chunks</i>
-----------------	---

Description

Parse json files with information about the individual chunks of images from the CamAlien uploader.

Usage

```
ca_parse_chunks(path)
```

Arguments

path	chr Path to the json file to read.
------	------------------------------------

Details

The function will read all the chunk files in the directory at path.

Value

data.frame with content of json files being read

ca_parse_job	<i>Parse json files with info on the job</i>
--------------	--

Description

Parse json files with information about the job from the CamAlien uploader.

Usage

```
ca_parse_job(path)
```

Arguments

path	chr Path to the json file to read.
------	------------------------------------

Value

data.frame with content of json file being read

ca_stat_images	<i>Get statistics on images</i>
----------------	---------------------------------

Description

Calculate statistics on the images uploaded by each partner

Usage

```
ca_stat_images(con)
```

Arguments

con	PqConnection The connection to the camalien database
-----	--

Details

The query follows whichever image model the host carries (see [ca_image_model\(\)](#)). On a v1 host the count is of jpg images, reached through partner → job → chunk. On a v2 host it is of captures in data2.imagedata, reached through partner → session, which is the same quantity — one capture yields one jpg — but counted on the chain that is filled in there.

Value

lazy_tbl

ca_stat_jobsize	<i>Get statistics on upload jobs</i>
-----------------	--------------------------------------

Description

Calculate sizes on the upload jobs that each partner has started. There is some inaccuracies relating to this as we can't see if images in one job are also part of other jobs.

Usage

```
ca_stat_jobsize(con)
```

Arguments

con	PqConnection The connection to the camalien database
-----	--

Details

The Danish images were uploaded with an early version of the uploader which did not report the size of the job and therefore total_size is NA here.

Value

lazy_tbl

ca_stat_tagged	<i>Get statistics on tagged images</i>
----------------	--

Description

Calculate the number and ratio of tagged images.

Usage

ca_stat_tagged(con)

Arguments

con PqConnection The connection to the camalien database

Details

The query follows whichever image model the host carries (see [ca_image_model\(\)](#)). On a v1 host an image counts as tagged when its imagemeta row has a tag; on a v2 host data2.imagedata carries a tagged flag of its own.

Value

lazy_tbl with partner, n_tagged, n and prop

ias	<i>List of target species for all partners</i>
-----	--

Description

Each partner has their own list of invasive alien species which they targeted during the field campaign. The same species can be on the list of multiple partners. Target species can be an entire genus. The scientific and the canonical name are from a lookup on [GBIF](#).

Usage

ias

Format

Data.frame with gbifid and name

partner chr, The partner name

canonical_name chr, The canonical name from GBIF with the modification that for genera we add .sp

scientificname chr, The scientific taxon name

gbifid int, The gbifid used by Plantnet and in the camalien database

pnid int, The ID used by Plantnet (hence pnid) and in the camalien database

taxon_name_key chr, The GBIF key for the taxon name

taxon_rank chr, The taxon rank, either species or genus. Derived from GBIF rankMarker

id_box

Draw an ID box on an image

Description

Modified [graphics::legend](#) function to draw small boxes with ID information on top of images

Usage

```
id_box(
  x,
  y = NULL,
  legend,
  col = par("col"),
  border = "black",
  angle = 45,
  bty = "o",
  bg = "white",
  box.lwd = par("lwd"),
  box.lty = par("lty"),
  box.col = par("fg"),
  cex = 1,
  xjust = 0,
  yjust = 1,
  adj = c(0.25, 0),
  text.width = NULL,
  text.col = par("col"),
  text.font = NULL,
  ncol = 1,
  horiz = FALSE,
  inset = 0,
  xpd,
  seg.len = 1
)
```

Arguments

<code>x, y</code>	the x and y co-ordinates to be used to position the legend. They can be specified by keyword or in any way which is accepted by xy.coords : See 'Details'.
<code>legend</code>	a character or expression vector of length ≥ 1 to appear in the legend. Other objects will be coerced by as.graphicsAnnot .
<code>col</code>	the color of points or lines appearing in the legend.
<code>border</code>	the border color for the boxes (used only if <code>fill</code> is specified).
<code>angle</code>	angle of shading lines.
<code>bty</code>	the type of box to be drawn around the legend. The allowed values are "o" (the default) and "n".
<code>bg</code>	the background color for the legend box. (Note that this is only used if <code>bty != "n"</code> .)
<code>box.lty, box.lwd, box.col</code>	the line type, width and color for the legend box (if <code>bty = "o"</code>).
<code>cex</code>	character expansion factor relative to current <code>par("cex")</code> . Used for text, and provides the default for <code>pt.cex</code> .
<code>xjust</code>	how the legend is to be justified relative to the legend x location. A value of 0 means left justified, 0.5 means centered and 1 means right justified.
<code>yjust</code>	the same as <code>xjust</code> for the legend y location.
<code>adj</code>	numeric of length 1 or 2; the string adjustment for legend text. Useful for y-adjustment when labels are plotmath expressions.
<code>text.width</code>	the width of the legend text in x ("user") coordinates. (Should be positive even for a reversed x axis.) Can be a single positive numeric value (same width for each column of the legend), a vector (one element for each column of the legend), NULL (default) for computing a proper maximum value of strwidth (legend), or NA for computing a proper column wise maximum value of strwidth (legend).
<code>text.col</code>	the color used for the legend text.
<code>text.font</code>	the font used for the legend text, see text .
<code>ncol</code>	the number of columns in which to set the legend items (default is 1, a vertical legend).
<code>horiz</code>	logical; if TRUE, set the legend horizontally rather than vertically (specifying <code>horiz</code> overrides the <code>ncol</code> specification).
<code>inset</code>	inset distance(s) from the margins as a fraction of the plot region when legend is placed by keyword.
<code>xpd</code>	if supplied, a value of the graphical parameter <code>xpd</code> to be used while the legend is being drawn.
<code>seg.len</code>	the length of lines drawn to illustrate <code>lty</code> and/or <code>lwd</code> (in units of character widths).

images	<i>Extract of detections and image metadata</i>
--------	---

Description

Small example dataset with scores, boxes and urls.

Usage

images

Format

Data.frame with the following columns:

callid chr, ID of the Plantnet call
score float, The model confidence score from Plantnet
boxsize int, Size of the polygon where the detection was made
centerx int, The center x coordinate of the polygon
centery int, The center y coordinate of the polygon
speciesid int, The plantnet species ID
imageid int, The image ID
imageurl int, The public URL to the image file

partners	<i>List of partners and their acronyms</i>
----------	--

Description

List of partners and their acronyms

Usage

partners

Format

Data.frame with partner and partner_acronym

partner chr, Name of the partner
partner_acronym chr, The acronym used in the Biodiversa+ project

Index

* datasets

- ias, [15](#)
- images, [18](#)
- partners, [18](#)

as.graphicsAnnot, [17](#)

ca_as_sf, [2](#)

ca_connect, [3](#)

ca_disconnect, [4](#)

ca_get_additionalimagedata
(ca_get_image), [4](#)

ca_get_additionalimagedata(), [8](#)

ca_get_chunk (ca_get_image), [4](#)

ca_get_detection (ca_get_image), [4](#)

ca_get_detectionssummary (ca_get_image),
[4](#)

ca_get_image, [4](#)

ca_get_image(), [6](#)

ca_get_imagedata (ca_get_image), [4](#)

ca_get_imagedata(), [7](#), [8](#)

ca_get_imagefile (ca_get_image), [4](#)

ca_get_imagefile(), [8](#)

ca_get_imagemeta (ca_get_image), [4](#)

ca_get_job (ca_get_image), [4](#)

ca_get_join_tbl, [5](#)

ca_get_join_tbl(), [7](#)

ca_get_partner (ca_get_image), [4](#)

ca_get_plantnetcall (ca_get_image), [4](#)

ca_get_polygon, [6](#)

ca_get_session (ca_get_image), [4](#)

ca_get_session(), [8](#)

ca_get_session_tbl, [7](#)

ca_get_session_tbl(), [6](#)

ca_get_species (ca_get_image), [4](#)

ca_get_tbl, [7](#)

ca_get_tbl(), [5](#)

ca_image_model, [9](#)

ca_image_model(), [5](#), [8](#), [14](#), [15](#)

ca_img_detection, [9](#)

ca_img_draw_polygon, [10](#)

ca_img_grid, [11](#)

ca_img_plot, [11](#)

ca_img_read, [12](#)

ca_img_read(), [11](#)

ca_json_read, [12](#)

ca_parse_chunks, [13](#)

ca_parse_job, [13](#)

ca_stat_images, [14](#)

ca_stat_jobsize, [14](#)

ca_stat_tagged, [15](#)

DBI::dbConnect, [3](#)

DBI::dbDisconnect, [4](#)

dplyr::collect, [4](#), [8](#)

expression, [17](#)

graphical parameter, [17](#)

graphics::legend, [16](#)

ias, [15](#)

id_box, [16](#)

images, [18](#)

jsonlite::read_json(), [12](#)

partners, [18](#)

plotmath, [17](#)

RPostgres::Postgres, [3](#)

strwidth, [17](#)

text, [17](#)

usethis::edit_r_envirion, [3](#)

xy.coords, [17](#)