

Package ‘moveVis’

November 3, 2017

Type Package

Title Movement Data Visualization

Version 0.9.6

Depends R (>= 2.10)

Date 2017-11-01

Description Tools to visualize movement data (e.g. from GPS tracking) and temporal changes of environmental data (e.g. from remote sensing) by creating video animations.

License GPL-3

RoxygenNote 6.0.1

Imports maptools, ggplot2, grid, rasterVis, raster, sp, geosphere, dismo, utils, grDevices, methods, stats, move, RCurl, gridExtra, graphics, simecol, plyr, zoo, lubridate, parallel, pbapply, RStoolbox

NeedsCompilation no

Author Jakob Schwalb-Willmann [aut, cre]

Maintainer Jakob Schwalb-Willmann <movevis@schwalb-willmann.de>

Repository CRAN

Date/Publication 2017-11-03 15:00:01 UTC

R topics documented:

moveVis-package	2
animate_move	3
animate_raster	9
animate_stats	13
basemap_data	17
get_formats	17
get_libraries	18
move_data	20

Index	25
--------------	-----------

Description

moveVis provides tools to visualize movement data of any kind, e. g by creating path animations from GPS point data. The package is under ongoing development, new functionalities are added constantly. The moveVis package is closely connected to the move package and mainly builds up on ggplot2.

Details

At the moment, the package includes the following functions:

[animate_move](#), which can create spatial movement data animations as GIF file. Among other functionalities, the function is able to

- visualize move class point data as (multiple) movement paths,
- display static basemap layers downloaded from Google Maps,
- display static basemap layers provided by the user,
- display dynamic, time-referenced raster data, e. g. to visualize land cover changes etc.,
- compute temporal interpolations from time-referenced raster data,
- create statistic plots, displaying the interaction of the individual movement paths with environmental data. ...

[animate_stats](#), which can create animated statistic plots from movement and basemap data as GIF file.

[animate_raster](#), which can create animated spatial plots of basemap data as GIF file.

[get_libraries](#), a helper function to locate/download/install the extern libraries ImageMagick, FFmpeg and libav and their tools, which are needed for different output file format support.

[get_formats](#), a helper function which returns all available output file formats (system-dependent).

Author(s)

Jakob Schwalb-Willmann. Maintainer: Jakob Schwalb-Willmann jakob@schwalb-willmann.de

See Also

[animate_move](#)

animate_move

*Animate movement data***Description**

animate_move animates movement data provided as move class objects or a list of them. The function creates an animated GIF or video file and saves it into the output directory. animate_move can be operated in different timing modes (see paths_mode) and with different background layer types (see layer, layer_type and map_type).

Usage

```
animate_move(m, out_dir, conv_dir = "", paths_mode = "true_data",
  paths_na.hold = TRUE, paths_col = "auto", paths_alpha = 1,
  indi_names = NA, layer = "basemap", layer_dt = "basemap",
  layer_int = FALSE, layer_type = "", layer_stretch = "none",
  layer_col = c("sandybrown", "white", "darkgreen"), layer_nacol = "white",
  map_type = "satellite", stats_create = FALSE, static_data = NA,
  static_gg = NA, extent_factor = 1e-04, tail_elements = 10,
  tail_size = 4, img_title = "title", img_sub = "subtitle",
  img_caption = "caption", img_labs = "labs", legend_title = "",
  legend_limits = NA, legend_labels = "auto", map_elements = TRUE,
  time_scale = TRUE, time_bar_col = "grey", scalebar_col = "white",
  scalebar_dist = "auto", north_col = "white", frames_layout = 0,
  frames_nmax = 0, frames_fps = 25, frames_nres = 1, frames_tres = 0,
  frames_width = NA, frames_height = NA, frames_pixres = 80,
  out_name = "moveVis", out_format = "gif", overwrite = FALSE,
  log_level = 1, log_logical = FALSE, ..., conv_cmd = "",
  conv_frames = 100)
```

Arguments

m	list or moveStack class object. Needs to contain one or several move class objects (one for each individual path to be displayed) containing point coordinates, timestamps, projection and individual ID.
out_dir	character. Output directory of the output file.
conv_dir	character. Command of or directory to required image/video converter library. Depends on, what is specified for out_format. If out_format = "gif", animate_move() works with the ImageMagick convert tool. In this case, specify command of or path to the convert tool. You can use get_libraries to find or download/install convert. If out_format is a video format (e.g. "mp4", "mov" ...), animate_move() works with either the FFmpeg ffmpeg tool or the libav avconv tool. In this case, specify command of or path to the ffmpeg or avconv tool. See also get_libraries . If not specified, animate_move() tries to find libraries automatically.

paths_mode	character vector. Mode to be used for dealing with time information when displaying multiple individual paths. If set to "true_data", paths are displayed based on true coverage times, showing only time periods that are covered. Time gaps will be skipped. Each frame is linked to a specific true time. If set to "true_time", paths are displayed based on true coverage times. Time gaps will be filled with non-movement frames. This mode is only recommended, if the dataset has no time gaps. Each frame is linked to a specific, true time. If set to "simple", all movement paths are displayed individually with no regard to the true coverage times. Time gaps will be skipped. Each frame displays several times at once, since each individual path has its own time. Default is "true_data".
paths_na_hold	logical. If TRUE, last path location is being hold on frame for NA path locations. If FALSE, path disappears until next path non-NA location. Default is TRUE.
paths_col	character vector. Colours of the individual animation paths. If set to "auto", a predefined colour set will be used. If single colour, all paths will be displayed by the same colour. If more individuals then colours, the colours are repeated.
paths_alpha	numeric. Set transparency of pathes. If set to 0, path is invisible. Default is 1.
indi_names	character. Optional vector of individual names. Length has to be equal to number of individuals in m. If NA, individual names are tried to be extracted from m using <code>idData</code> . Default is NA.
layer	raster, list or character "basemap". Single raster object or list of raster objects to be used as (dynamically changing) basemap layer. Default is "basemap" to download a static basemap layer. Use a <code>rasterBrick</code> class object and set <code>layer_type</code> to "RGB" to compute a RGB basemap.
layer_dt	POSIXct vector or list. Single POSIXct date/time stamp or list of POSIXct date/time stamps representing the acquisition dates of the layer raster objects.
layer_int	logical. Whether to interpolate the basemap layer objects over time, if several are provided (TRUE), or to display them one after another depending on the animation time frame that is displayed (FALSE). Default is FALSE.
layer_type	character. Layer type. Either "RGB" (if layer is a <code>rasterBrick</code> class onejct), "gradient" or "discrete". Default is "RGB". Ignored, if <code>layer = "basemap"</code> .
layer_stretch	character. Ignored, if <code>layer_type</code> is not "RGB". Either "none", "lin", "hist", "sqrt" or "log" for no stretch, linear, histogram, square-root or logarithmic stretch. Default is "none".
layer_col	character vector. Two or more colours to be used for displaying the background layer. If <code>layer_type = "gradient"</code> , a colour ramp between the colour is calculated. If <code>layer_type = "discrete"</code> , the colours will be used per value range. Ignored, if <code>layer_type = "RGB"</code> .
layer_nacol	character. Colour to be displayed for NA values. Default is "white".
map_type	character. Static basemap type. Chosse from "roadmap", "satellite", "hybrid", "terrain".
stats_create	logical. TRUE to create statistic plots side by side with the spatial plot. Use the arguments explained for <code>animate_stats</code> to adjust the plotting behaviour. Default is FALSE.

static_data	data.frame. Data (e.g. static points) to be displayed within the spatial plot of the output animation. At least, "x", "y" columns for the coordinates and "names" for the naming of the point have to be included. If "static_gg" remains unspecified, "static_data" is plotted as points to the output map, annotated with their namings. Points outside the frame extent are not displayed. See "static_gg" for further options.
static_gg	character. One or several ggplot2 functions, concatenated by "+" specifying how "static_data" should be displayed, e.g. using geom_point and geom_text for displaying points annotated with text. ggplot2 data and aes, aes_ arguments etc. need to refer to the columns specified in "static_data". As default, "static_data" is plotted as geom_point and geom_label.
extent_factor	numeric. Defines the distance between the spatial extents of the movement data set and the basemap as proportion of the axis distance. Default is 0.0001. The higher the value, the larger the basemap extent. Ignored, if layer = "basemap".
tail_elements	numeric. Number of points to be displayed as path tail of the animation paths. Default is 10.
tail_size	numeric. Size of the first tail element. Default is 4.
img_title	character. Titel to be displayed above the animated plot. If not specified, no title will be displayed.
img_sub	character. Subtitel to be displayed underneath the title. If not specified, no subtitle will be displayed.
img_caption	character. Caption to be displayed underneath the plot. If not specified, no caption will be displayed.
img_labs	character. Axis titles to be displayed at the x and y axis of the plot. If not specified, labs will be computed depending on the projection or will be "x" and "y".
legend_title	character. Title to be displayed above the basemap layer legend (if layer_type is not "RGB"). Ignored, if layer = "basemap".
legend_limits	numeric vector. Fixed minimum and maximum limit values of the legend (gradient layer type). Default is NA for data-dependent minimum and maximum values. Ignored, if layer_type is "discrete" or "RGB".
legend_labels	character vectors. Label for each legend break class. If set to "auto", values are displayed. Default is "auto".
map_elements	logical. If FALSE, map elements (north arrow and scale bar) are hidden. Default is TRUE.
time_scale	logical. If FALSE, time scale is hidden. Default is TRUE.
time_bar_col	character. Colour of the time progress bar on the top edge of the map. Default is "grey".
scalebar_col	character. Colour of the scalebar text. Default is "white".
scalebar_dist	numeric. Distance represented by the scalebar in kilometers.
north_col	character. Colour of the north arrow. Default is "white".

frames_layout	matrix. Optional layout. Define, which plots should be placed where using a matrix representing the GIF/video frame. Matrix elements can be the following plot identifiers: "map" for the spatial plot, "st_all", "st_per" for the overall and periodic stats plot or "st_allR", "st_perR", "st_allG", "st_perG", "st_allB", "st_perB" for the overall and periodic plots per band, when using layer_type = "RGB", and 'st_leg' for a stats legend. Alternatively, integers from 1 to 8 corresponding to the described order can be used. Plots not mentioned using frames_layout identifiers are not displayed. If set to 0, layout is generated automatically. Default is 0.
frames_nmax	numeric. Number of maximum frames. If set, the animation will be stopped, after the specified number of frames is reached. Default is 0 (displaying all frames).
frames_fps	numeric. Frames to display per second (FPS). Note that the gif format only can handle FPS out of 100, e.g. 25. In that case, frames_fps input is rounded. Default is 25.
frames_nres	numeric. Interval of which frames of all frames should be used (nth elements). Default is 1 (every frame is used). If set to 2, only every second frame is used.
frames_tres	numeric. Defines temporal output resolution in seconds, 'm' should be interpolated to (linear interpolation). If 0, temporal resolution is detected automatically. Default is 0.
frames_width	numeric. Number of pixels of frame width. Default is 600 (with stats plots 1000).
frames_height	numeric. Number of pixels of frame height. Default is 600.
frames_pixres	numeric. Resolution of output file in pixel in ppi. The higher the ppi, the higher frames_height and frames_width should be to avoid large fonts and overlaps. Default is 80.
out_name	character. Name of the output file. Default is "moveVis_ani".
out_format	character. Output format, e.g. "gif", "avi", "3gp", "mov", "mpeg", "mp4". Use get_formats to get all supported file formats on your system. "gif" is recommended for short animations only (recommended max. frame number around 200 frames; GIF frames are unlimited, but computation time will be very long). Use a video format for long animations. Format "gif" requires ImageMagick, all other video formats require FFmpeg ('ffmpeg') or libav ('avconv') to be installed. For that, also see get_libraries .
overwrite	logical. If TRUE, files with equal file names to out_name will be overwritten. Default is FALSE.
log_level	numeric. Level of console output given by the function. There are three log levels. If set to 3, no messages will be displayed except erros that caused an abortion of the process. If set to 2, warnings and errors will be displayed. If set to 1, a log showing the process activity, warnings and errors will be displayed.
log_logical	logical. For large processing schemes. If TRUE, the function returns TRUE when finished processing successfully.
...	optional arguments. All arguments taken by animate_stats can be handed over to animate_move as well to create side-by-side spatial and statistic plot animations (see animate_stats).

conv_cmd	character. Recommended for expert use only. Passes additional command line options to the conversion command, e.g. with a <code>convert</code> call adding <code>'-limit'</code> for memory resource handling. For details, see check the documentations of ImageMagick <code>convert</code> , FFmpeg <code>ffmpeg</code> and libav <code>avconv</code> .
conv_frames	numeric. Recommended for expert use only. Only used, if <code>out_format = "gif"</code> . Number of frames to be used for creating GIF segments that will be assembled to a final GIF file. Correct number depends on system performance and total frames number. Default is 100. Ignored, if <code>out_format</code> is not "gif".

Details

Make sure you have run [get_libraries](#) before you use `moveVis` for the first time: Depending on the selected output format (`out_format`, `animate_move` either needs the `convert` tool of the ImageMagick software package (.gif format) or either `ffmpeg` from FFmpeg or `avconv` from libav (video formats). The command or directory to the `convert` tool needs to be provided with `conv_dir`. Please use [get_libraries](#) to search for the needed libraries and command/tool directories on your system or to automatically download and install the required software. See [get_libraries](#) and `out_format` and `conv_dir` for details.

`animate_move` preprocesses your `move` data depending on the state of the data (see `paths_mode` and `frames_tres`). `animate_move` is based on `ggplot2`.

Value

None or logical (see `log_logical`). The output file is written to the `ouput` directory.

Author(s)

Jakob Schwalb-Willmann

See Also

[get_libraries](#), [animate_stats](#), [animate_raster](#)

Examples

```
## Not run:
#Load move and moveVis packages
library(move)
library(moveVis)

#Get the sample data from the moveVis package (a data.frame)
data("move_data")
move_data$dt <- as.POSIXct(strptime(move_data$dt, "%Y-%m-%d %H:%M:%S", tz = "UTC"))

#Create moveStack object including multiple individuals from the data.frame
#alternatively, use the move package to download data directly from movebank.org
m <- move(move_data$lon, move_data$lat, proj=CRS("+proj=longlat +ellps=WGS84"),
          time = move_data$dt, animal=move_data$individual, data=move_data)
```

```

#Find the command or directory to convert tool of ImageMagick
conv_dir <- get_libraries()

#Specify the output directory, e.g.
out_dir <- "/out/test"
#or to a temporary directory:
out_dir <- paste0(tempdir(),"/test")
dir.create(out_dir)

#Specify some optional appearance variables
img_title <- "Movement of the white stork population at Lake Constance, Germany"
img_sub <- paste0("including individuals ",paste(rownames(idData(m)), collapse=', '))
img_caption <- "Projection: Geographical, WGS84; Sources: Movebank 2013; Google Maps"

#Call animate_move() with an automatic basemap from Google, maximum frames at 50
#output format "gif"
animate_move(m, out_dir, conv_dir, tail_elements = 10,
             paths_mode = "true_data", frames_nmax = 50,
             img_caption = img_caption, img_title = img_title,
             img_sub = img_sub, log_level = 1, extent_factor = 0.0002,
             out_format = "gif")

#Improve your animation by adding a static points layer
static_data <- data.frame(x = c(8.94,8.943), y = c(47.75,47.753), names = c("Site 1","Site 2"))

#Call animate_move() with "static_data" added
#use another output format, e.g. "mov"
animate_move(m, out_dir, conv_dir, tail_elements = 10,
             paths_mode = "true_data", frames_nmax = 50,
             img_caption = img_caption, img_title = img_title,
             img_sub = img_sub, log_level = 1, extent_factor = 0.0002,
             static_data=static_data, out_format = "mov")

#Try a different paths_mode: Instead of "true_data" use "simple"
#output format "mp4". Longer videos then 100-200 frames should not be GIFs
animate_move(m, out_dir, conv_dir, tail_elements = 10,
             paths_mode = "simple", frames_nmax = 50,
             img_caption = img_caption, img_title = img_title,
             img_sub = img_sub, log_level = 1, extent_factor = 0.0002,
             static_data=static_data, out_format = "mp4")

#Use your own basemap by adding lists of rasters and of timestamps
data("basemap_data")
layer = basemap_data[[1]] #this is a example MODIS NDVI dataset
layer_dt = basemap_data[[2]] #this is a corresponding date/time list

#Call animate_move with NDVI data as basemap
#layer_type is "gradient", since NDVI values are continuous
animate_move(m, out_dir, conv_dir, tail_elements = 10, layer_type = "gradient",
             paths_mode = "true_data", frames_nmax = 50, layer =layer, layer_dt = layer_dt,
             img_caption = img_caption, img_title = img_title,
             img_sub = img_sub, log_level = 1, extent_factor = 0.0002)

```



```

#How do your moving individuals interact with their environments?
#Use "stats_create" to create statistics plots
animate_move(m, out_dir, conv_dir, tail_elements = 10, layer_type = "gradient",
             paths_mode = "true_data", frames_nmax = 50, layer = layer, layer_dt = layer_dt,
             img_caption = img_caption, img_title = img_title,
             img_sub = img_sub, log_level = 1, extent_factor = 0.0002,
             stats_create = TRUE)

#If you just want those stats plots, use animate_stats()

#Use "frames_layout" to change the layout of your animation
#e.g. change the position of st_all and st_per
frames_layout <- rbind(c("map", "map", "map", "st_all", "st_leg"),
                      c("map", "map", "map", "st_per", "st_leg"))

#or equalize the sizes of spatial map and stats plots
frames_layout <- rbind(c("map", "st_all", "st_per", "st_leg"))

animate_move(m, out_dir, conv_dir, tail_elements = 10, layer_type = "gradient",
             paths_mode = "true_data", frames_nmax = 50, layer = layer, layer_dt = layer_dt,
             img_caption = img_caption, img_title = img_title,
             img_sub = img_sub, log_level = 1, extent_factor = 0.0002,
             stats_create = TRUE, frames_layout=frames_layout)

## End(Not run)

```

animate_raster

Animate raster data

Description

animate_raster animates raster data provided as list of raster class objects. The function creates an animated GIF or video file and saves it into the output directory.

Usage

```

animate_raster(layer, out_dir, conv_dir = "convert", layer_dt = 0,
              layer_type = "gradient", layer_stretch = "none",
              layer_col = c("sandybrown", "white", "darkgreen"), layer_nacol = "white",
              static_data = NA, static_gg = NA, img_title = "title",
              img_sub = "subtitle", img_caption = "caption", img_labs = "labs",
              legend_title = "", legend_limits = NA, legend_labels = "auto",
              map_elements = TRUE, time_bar_col = "grey", scalebar_col = "white",
              north_col = "white", frames_nmax = 0, frames_fps = 25,
              frames_nres = 1, frames_width = NA, frames_height = NA,
              frames_pixres = 80, out_name = "moveVis_ani", out_format = "gif",
              overwrite = FALSE, log_level = 1, log_logical = FALSE, conv_cmd = "",
              conv_frames = 100)

```

Arguments

layer	list. List of raster objects.
out_dir	character. Output directory of the output file.
conv_dir	character. Command of or directory to required image/video converter library. Depends on, what is specified for out_format. If out_format = "gif", animate_move() works with the ImageMagick convert tool. In this case, specify command of or path to the convert tool. You can use get_libraries to find or download/install convert. If out_format is a video format (e.g. "mp4", "mov" ...), animate_move() works with either the FFmpeg ffmpeg tool or the libav avconv tool. In this case, specify command of or path to the ffmpeg or avconv tool. See also get_libraries . If not specified, animate_move() tries to find libraries automatically.
layer_dt	POSIXct or vector. Optional vector of POSIXct date/time stamps corresponding to the acquisition dates of the layer raster objects to display a time scale.
layer_type	character. Layer type. Either "RGB" (if layer is a rasterBrick class object), "gradient" or "discrete". Default is "RGB". Ignored, if layer = "basemap".
layer_stretch	character. Ignored, if layer_type is not "RGB". Either "none", "lin", "hist", "sqrt" or "log" for no stretch, linear, histogram, square-root or logarithmic stretch. Default is "none".
layer_col	character vector. Two or more colours to be used for displaying the background layer. If layer_type = "gradient", a colour ramp between the colours is calculated. If layer_type = "discrete", the colours will be used per value range. Ignored, if layer_type = "RGB".
layer_nacol	character. Colour to be displayed for NA values. Default is "white".
static_data	data.frame. Data (e.g. static points) to be displayed within the spatial plot of the output animation. At least, "x", "y" columns for the coordinates and "names" for the naming of the point have to be included. If "static_gg" remains unspecified, "static_data" is plotted as points to the output map, annotated with their namings. Points outside the frame extent are not displayed. See "static_gg" for further options.
static_gg	character. One or several ggplot2 functions, concatenated by "+" specifying how "static_data" should be displayed, e.g. using geom_point and geom_text for displaying points annotated with text. ggplot2 data and aes, aes_ arguments etc. need to refer to the columns specified in "static_data". As default, "static_data" is plotted as geom_point and geom_label.
img_title	character. Title to be displayed above the animated plot. If not specified, no title will be displayed.
img_sub	character. Subtitle to be displayed underneath the title. If not specified, no subtitle will be displayed.
img_caption	character. Caption to be displayed underneath the plot. If not specified, no caption will be displayed.
img_labs	character. Axis titles to be displayed at the x and y axis of the plot. If not specified, labs will be computed depending on the projection or will be "x" and "y".

legend_title	character. Title to be displayed above the basemap layer legend (if layer_type is not "RGB"). Ignored, if layer = "basemap".
legend_limits	numeric vector. Fixed minimum and maximum limit values of the legend (gradient layer type). Default is NA for data-dependent minimum and maximum values. Ignored, if layer_type is "discrete" or "RGB".
legend_labels	character vectors. Label for each legend break class. If set to "auto", values are displayed. Default is "auto".
map_elements	logical. If FALSE, map elements (north arrow and scale bar) are hidden. Default is TRUE.
time_bar_col	character. Colour of the time progress bar on the top edge of the map. Default is "grey".
scalebar_col	character. Colour of the scalebar text. Default is "white".
north_col	character. Colour of the north arrow. Default is "white".
frames_nmax	numeric. Number of maximum frames. If set, the animation will be stopped, after the specified number of frames is reached. Default is 0 (displaying all frames).
frames_fps	numeric. Frames to display per second (FPS). Note that the gif format only can handle FPS out of 100, e.g. 25. In that case, frames_fps input is rounded. Default is 25.
frames_nres	numeric. Interval of which frames of all frames should be used (nth elements). Default is 1 (every frame is used). If set to 2, only every second frame is used.
frames_width	numeric. Number of pixels of frame width. Default is 600 (with stats plots 1000).
frames_height	numeric. Number of pixels of frame height. Default is 600.
frames_pixres	numeric. Resolution of output file in pixel in ppi. The higher the ppi, the higher frames_height and frames_width should be to avoid large fonts and overlaps. Default is 80.
out_name	character. Name of the output file. Default is "moveVis_ani".
out_format	character. Output format, e.g. "gif", "avi", "3gp", "mov", "mpeg", "mp4". Use get_formats to get all supported file formats on your system. "gif" is recommended for short animations only (recommended max. frame number around 200 frames; GIF frames are unlimited, but computation time will be very long). Use a video format for long animations. Format "gif" requires ImageMagick, all other video formats require FFmpeg ('ffmpeg') or libav ('avconv') to be installed. For that, also see get_libraries .
overwrite	logical. If TRUE, files with equal file names to out_name will be overwritten. Default is FALSE.
log_level	numeric. Level of console output given by the function. There are three log levels. If set to 3, no messages will be displayed except errors that caused an abortion of the process. If set to 2, warnings and errors will be displayed. If set to 1, a log showing the process activity, warnings and errors will be displayed.
log_logical	logical. For large processing schemes. If TRUE, the function returns TRUE when finished processing successfully.

conv_cmd	character. Recommended for expert use only. Passes additional command line options to the conversion command, e.g. with a <code>convert</code> call adding <code>'-limit'</code> for memory resource handling. For details, see check the documentations of ImageMagick <code>convert</code> , FFmpeg <code>ffmpeg</code> and libav <code>avconv</code> .
conv_frames	numeric. Recommended for expert use only. Only used, if <code>out_format = "gif"</code> . Number of frames to be used for creating GIF segments that will be assembled to a final GIF file. Correct number depends on system performance and total frames number. Default is 100. Ignored, if <code>out_format</code> is not "gif".

Details

`animate_raster` is based on `ggplot2`. Depending on the selected output format (`out_format`, it either needs the `convert` tool of the ImageMagick software package (.gif format) or either `ffmpeg` from FFmpeg or `avconv` from libav (video formats). The command or directory to the `convert` tool needs to be provided with `conv_dir`. Please use [get_libraries](#) to search for the needed libraries and command/tool directories on your system or to automatically download and install the required software. See [get_libraries](#) and `out_format` and `conv_dir` for details.

Value

None or logical (see `log_logical`). The output GIF or video file is written to the output directory.

Author(s)

Jakob Schwalb-Willmann

See Also

[get_libraries](#), [animate_move](#),

Examples

```
## Not run:
#Create a list of several raster objects to be displayed one after another
#If layer_type = RGB, use a brick class object with RGB bands!
layer <- list(raster1, raster2, raster2)

#Get your convert directory/command
conv_dir <- get_libraries()

#Specify the output directory, e.g.
out_dir <- "/out/test"
#or to a temporary directory:
out_dir <- paste0(tempdir(),"/test")
dir.create(out_dir)

#Call animate_raster
animate_raster(layer,out_dir = out_dir, conv_dir = conv_dir, layer_type = "RGB",
              out_format = "gif")

#use another file format for longer videos
```

```

animate_raster(layer,out_dir = our_dir, conv_dir = conv_dir, layer_type = "RGB",
              out_format = "mov")

## End(Not run)

```

animate_stats

Animate movement data statistics

Description

animate_stats animates statistic plot from movement data provided as move class objects or a list of them and basemap data provided as raster. It extracts basemap values of pixels that are part of the movement paths and visualizes frequencies per value. The function creates an animated GIF or video file and saves it into the output directory. See also [animate_move](#).

Usage

```

animate_stats(m, out_dir, conv_dir = "convert", layer = "basemap",
             layer_dt = "basemap", layer_int = FALSE, layer_type = "",
             val_limits = NA, paths_col = "auto", paths_mode = "true_data",
             stats_type = "", stats_gg = "", stats_digits = 1, stats_tframe = 5,
             stats_title = "", frames_layout = 0, frames_nmax = 0, frames_fps = 25,
             frames_nres = 1, frames_tres = 0, frames_width = 800,
             frames_height = 300, frames_pixres = 80, out_name = "moveVis_ani",
             out_format = "gif", overwrite = FALSE, log_level = 1,
             log_logical = FALSE)

```

Arguments

m	list or moveStack class object. Needs to contain one or several move class objects (one for each individual path to be displayed) containing point coordinates, timestamps, projection and individual ID.
out_dir	character. Output directory of the output file.
conv_dir	character. Command of or directory to required image/video converter library. Depends on, what is specified for out_format. If out_format = "gif", animate_move() works with the ImageMagick convert tool. In this case, specify command of or path to the convert tool. You can use get_libraries to find or download/install convert. If out_format is a video format (e.g. "mp4", "mov" ...), animate_move() works with either the FFmpeg ffmpeg tool or the libav avconv tool. In this case, specify command of or path to the ffmpeg or avconv tool. See also get_libraries . If not specified, animate_move() tries to find libraries automatically.
layer	raster, list or character "basemap". Single raster object or list of raster objects to be used as (dynamically changing) basemap layer. Default is "basemap" to download a static basemap layer. Use a rasterBrick class object and set layer_type to "RGB" to compute a RGB basemap.

layer_dt	POSIXct vector or list. Single POSIXct date/time stamp or list of POSIXct date/time stamps representing the acquisition dates of the Layer raster objects.
layer_int	logical. Whether to interpolate the basemap layer objects over time, if several are provided (TRUE), or to display them one after another depending on the animation time frame that is displayed (FALSE). Default is FALSE.
layer_type	character. Layer type. Either "RGB" (if layer is a rasterBrick class object), "gradient" or "discrete". Default is "RGB". Ignored, if layer = "basemap".
val_limits	numeric vector. Fixed minimum and maximum limit values of the legend (gradient layer type). Default is NA for data-dependent minimum and maximum values. Ignored, if layer_type is "discrete" or "RGB".
paths_col	character vector. Colours of the individual animation paths. If set to "auto", a predefined colour set will be used. If single colour, all paths will be displayed by the same colour. If more individuals than colours, the colours are repeated.
paths_mode	character vector. Mode to be used for dealing with time information when displaying multiple individual paths. If set to "true_data", paths are displayed based on true coverage times, showing only time periods that are covered. Time gaps will be skipped. Each frame is linked to a specific true time. If set to "true_time", paths are displayed based on true coverage times. Time gaps will be filled with non-movement frames. This mode is only recommended, if the dataset has no time gaps. Each frame is linked to a specific, true time. If set to "simple", all movement paths are displayed individually with no regard to the true coverage times. Time gaps will be skipped. Each frame displays several times at once, since each individual path has its own time. Default is "true_data".
stats_type	character. Defines which standard plot design should be used. Select either "line" or "bar". Ignored, if stats_gg is used.
stats_gg	character. Enables usage of ggplot2 syntax for plot design. If set, stats_type is ignored. See details for information on the statistic data structure to be used by the user defined plot function.
stats_digits	numeric. Defines how detailed the statistic plot should be as number of decimals. Values with more decimals are rounded. Default is 1 for one decimal.
stats_tframe	numeric. Defines the temporal range of the periodic stats plot. Default is 5 meaning that five time frames back from the displayed frame are evaluated.
stats_title	character vector. Optional plot titles. Two character strings within a vector.
frames_layout	matrix. Optional layout. Define, which plots should be placed where using a matrix representing the GIF/video frame. Matrix elements can be the following plot identifiers: "map" for the spatial plot, "st_all", "st_per" for the overall and periodic stats plot or "st_allR", "st_perR", "st_allG", "st_perG", "st_allB", "st_perB" for the overall and periodic stats plots per band, when using layer_type = "RGB", and 'st_leg' for a stats legend. Alternatively, integers from 1 to 8 corresponding to the described order can be used. Plots not mentioned using frames_layout identifiers are not displayed. If set to 0, layout is generated automatically. Default is 0.
frames_nmax	numeric. Number of maximum frames. If set, the animation will be stopped, after the specified number of frames is reached. Default is 0 (displaying all frames).

frames_fps	numeric. Frames to display per second (FPS). Note that the gif format only can handle FPS out of 100, e.g. 25. In that case, frames_fps input is rounded. Default is 25.
frames_nres	numeric. Interval of which frames of all frames should be used (nth elements). Default is 1 (every frame is used). If set to 2, only every second frame is used.
frames_tres	numeric. Defines temporal output resolution in seconds, 'm' should be interpolated to (linear interpolation). If 0, temporal resolution is detected automatically. Default is 0.
frames_width	numeric. Number of pixels of frame width. Default is 600 (with stats plots 1000).
frames_height	numeric. Number of pixels of frame height. Default is 600.
frames_pixres	numeric. Resolution of output file in pixel in ppi. The higher the ppi, the higher frames_height and frames_width should be to avoid large fonts and overlaps. Default is 80.
out_name	character. Name of the output file. Default is "moveVis_ani".
out_format	character. Output format, e.g. "gif", "avi", "3gp", "mov", "mpeg", "mp4". Use get_formats to get all supported file formats on your system. "gif" is recommended for short animations only (recommended max. frame number around 200 frames; GIF frames are unlimited, but computation time will be very long). Use a video format for long animations. Format "gif" requires ImageMagick, all other video formats require FFmpeg ('ffmpeg') or libav ('avconv') to be installed. For that, also see get_libraries .
overwrite	logical. If TRUE, files with equal file names to out_name will be overwritten. Default is FALSE.
log_level	numeric. Level of console output given by the function. There are three log levels. If set to 3, no messages will be displayed except erros that caused an abortion of the process. If set to 2, warnings and errors will be displayed. If set to 1, a log showing the process activity, warnings and errors will be displayed.
log_logical	logical. For large processing schemes. If TRUE, the function returns TRUE when finished processing successfully.

Details

animate_stats is a wrapper function of [animate_move](#) to create single statistic plots without spatial plotting. For statistic plot animations side-by-side with spatial plot animations, use [animate_move](#) (see stats_create argument). The function can handle all arguments taken by animate_stats as well.

Use stats_gg to provide an own ggplot2 plot design as shown in the examples. The statistics are stored for both plots (periodic and accumulated) with the variable stats_obj[[k]][[b]][[x]] (list of two, indexed by k ranging from 1 to 2 for each plot). Both stats_obj first-level lists contain one list per band (one list or three lists, if 'RGB', indexed by b). These second-level lists contain the stats elements framewise for each time step. For this, see the stats_gg example. The variable cols (list of two, one per plot) contains the defined colour values and namings.

Value

None or logical (see log_logical). The output GIF or video file is written to the output directory.

Author(s)

Jakob Schwalb-Willmann

See Also

[get_libraries](#)

Examples

```
## Not run:
#Load move and moveVis packages
library(move)
library(moveVis)

#Get the sample data from the moveVis package
data("move_data")
move_data$dt <- as.POSIXct(strptime(move_data$dt, "%Y-%m-%d %H:%M:%S", tz = "UTC"))

#Create moveStack object including multiple individuals
data_ani <- move(move_data$lon, move_data$lat, proj=CRS("+proj=longlat +ellps=WGS84"),
                time = move_data$dt, animal=move_data$individual, data=move_data)

#Load basemap MODIS NDVI data
data("basemap_data")
layer = basemap_data[[1]]
layer_dt = basemap_data[[2]]

#Find command or directory to convert tool of ImageMagick
conv_dir <- get_libraries()

#Specify the output directory, e.g.
out_dir <- "/out/test"
#or to a temporary directory:
out_dir <- paste0(tempdir(),"/test")
dir.create(out_dir)

#Call animate_stats()
animate_stats(data_ani, out_dir, conv_dir = conv_dir,
              layer=layer, layer_dt = layer_dt, layer_type = "gradient",
              stats_digits = 1, stats_type = "bar", out_name = "moveVis_ani",
              log_level = 1, frames_nmax = 60)

stats_gg <- 'ggplot(data = stats_obj[[k]][[b]][[x]],
                  aes_(x = ~val, y = ~value, colour = ~variable)) +
  geom_line() + geom_point() + theme_bw() + theme(aspect.ratio=1) +
  scale_y_continuous(expand = c(0,0), limits = c(0, stats_max[k])) +
  scale_x_continuous(expand = c(0,0)) +
  scale_color_manual(name= "", values = cols) +
  labs(x = "Basemap Value", y="Frequency",
       title=stats_title[[b]][[k]], label=c("123", "456"))+
  theme(plot.title = element_text(hjust = 0.5),
```



```
plot.subtitle = element_text(hjust = 0.5))'  
  
#Call animate_stats() with stats_gg  
animate_stats(data_ani, out_dir, conv_dir = conv_dir,  
              layer=layer, layer_dt = layer_dt, layer_type = "gradient",  
              stats_digits = 1, stats_gg = stats_gg, out_name = "moveVis_ani",  
              log_level = 1, frames_nmax = 60)  
  
## End(Not run)
```

basemap_data	<i>MODIS NDVI 2013 example data of Lake Constance area, Germany</i>
--------------	---

Description

Dataset containing five MODIS NDVI scenes of 2013 covering the Lake Constance area.

Usage

```
basemap_data
```

Format

List containing two lists of equal lengths: a list of rasters containing 10 MODIS NDVI raster class objects and a list of corresponding POSIXct class timestamp objects.

Details

The example data have been pre-processed and have equal extents and projections.

Source

MODIS 2013 freely available data, accessed using MODIS R package

get_formats	<i>Get all available output file formats</i>
-------------	--

Description

`get_formats` returns all available file formats that can be used with `animate_move` (`out_format`). The available output formats depend on the additional libraries that are installed on your system (see `get_libraries`).

Usage

```
get_formats(tool = "auto")
```

Arguments

`tool` character. Default is "auto" to return all available formats. If set to either 'convert', 'ffmpeg' or 'avconv', the output formats made available by one of these specific library tools are returned.

Value

A character vector containing available output formats. Each vector element can serve as input to the `out_format` argument of `animate_move`.

Author(s)

Jakob Schwalb-Willmann

See Also

[get_libraries](#)

Examples

```
## Not run:
#Get all formats available from the currently installed libraries
formats <- get_formats()
print(formats)

#For example, use the output as input to animate_move() out_format argument
out_format <- formats[1]

## End(Not run)

#Get formats provided by specific tool
formats <- get_formats(tool = "convert")
```

get_libraries	<i>Detect and/or download and install extern system libraries needed by moveVis</i>
---------------	---

Description

`get_libraries` tries to detect the libraries on your system that are needed by `moveVis` to be able to deal with different output file formats. `get_libraries()` searches for 'convert' from the ImageMagick library (needed for .gif support), 'ffmpeg' from the FFmpeg library or 'avconv' from the 'libav-tools' library (both needed for video support). You can execute `get_libraries()` to make sure, these libraries are correctly installed on your system. It is recommended to have ImageMagick including 'convert' and FFmpeg including 'ffmpeg' installed to gain support of all available output file formats. The function's return can serve as `conv_dir` input to the `animate_move()` function.

get_imconvert is an alias function of get_libraries included for compatibility reasons that does the same as get_libraries, but is only checking for the convert tool of ImageMagick. It is recommended to use get_libraries instead.

Usage

```
get_libraries(lib.tool = "all", dir = "none", ...)
```

```
get_imconvert(dir = "auto")
```

Arguments

lib.tool	character. Vector of libraries to look for. This can be either 'convert', 'ffmpeg', 'avconv' or a combination. Default is "all" to check for all possible libraries.
dir	character. Directory were to download, unzip and install ImageMagick. If set to "auto", a temporary directory is used. Default is "none", which will cause no automatic installation.
...	additional arguments. Currently not used.

Details

The following tools and libraries are needed by moveVis:

- the convert tool of ImageMagick to support the GIF format
- the ffmpeg tool of FFmpeg to support video formats
- alternatively to ffmpeg, the avconv tool of libav-tools to support video formats.

It is recommended to have both ImageMagick and one of the mentioned video libraries installed to be able to create all output formats with the animate_move() function (see argument out_format of [animate_move](#))

If you are running Windows, the function can download and either temporarily or permanently install ImageMagick on your Windows system in case that no existing installation can be found. For FFmpeg or libav, a download link will be provided.

If you are running Linux, the function provides a root permission requiring command to be executed once by the user in the terminal to install ImageMagick, FFmpeg or libav, if not installed. On standard Ubuntu distributions, ImageMagick and FFmpeg belong to the preinstalled packages by default. If you are running macOS (former OSX), manual installation is necessary or installation via a package manager such as 'brew'.

You can install the necessary libraries manually and then run get_libraries to just identify the needed commands:

- ImageMagick: <https://www.imagemagick.org/script/download.php>)
- FFmpeg from <https://www.ffmpeg.org/download.html>
- libav from <https://libav.org/download/>

Value

A character vector including all found commands or directories to the needed tools of the requested libraries. The return can serve as conv_dir input to the animate_move function.

Author(s)

Jakob Schwalb-Willmann

See Also

[animate_move](#)

Examples

```
#conv_dir of the animate_move() function
conv_dir <- get_libraries()
```

move_data	<i>Movement data of a White Stork population located nearby Lake Constance, Germany</i>
-----------	---

Description

Dataset containing longitude/latitude point coordinates and acquisition times of several white stork individuals of a population located nearby Lake Constance.

Usage

```
move_data
```

Format

Data frame containing 2408 rows and 5 variables

Details

- lon. Longitude
- lat. Latitude
- individual. Name of the individual
- population. Name of the population
- dt. Date and timestamps (to be used e. g. as POSIXct) ...

The example data have been pre-processed as explained in the examples. Please note that a very basic moving-/non-moving segmentation algorithm has been used in order to keep the code simple. The provided code and data cannot be applied for actual data analysis, only adequate for movement data visualization!

Source

Movebank (2013): URL <http://www.movebank.org/>

Examples

```

#DISCLAIMER: The provided code and data cannot be applied for actual data analysis
#and should only show the derivation of the example data provided with moveVis.
#This code is only adequate for movement data visualization.

## Not run:
#Calculate start/stop times, FUNCTION
st_times <- function(data){
  for(i in 1:length(data)){
    if(i == 1){
      start_dt <- data[[i]]$dt[1]
      stop_dt <- data[[i]]$dt[length(data[[i]]$dt)]
    }else{
      start_dt <- c(start_dt, data[[i]]$dt[1])
      stop_dt <- c(stop_dt, data[[i]]$dt[length(data[[i]]$dt)])
    }
  }
  return(list(start_dt,stop_dt))
}
#Distance recalculation, FUNCTION
calc_dist <- function(data){
  p1 <- data[1:(length(data$lon)-1),1:2]
  p2 <- data[2:length(data$lon),1:2]
  data$dist_m <- NA
  data$dist_m[2:length(data$dist_m)] <- distGeo(p1,p2)
  return(data)
}

#Moving/non-moving segmentator, FUNCTION
class_moving <- function(data,resting_rad){
  data$moving <- 0
  data$moving[which(data$dist_m >= resting_rad)] <- 1
  return(data)
}

#Read data
data_dir <- "/path/to/movedata.txt" #In this case, an ASCII file with multiple populations
dr <- read.table(data_dir,header=TRUE,sep=";")
pop_levels <- levels(dr$population)
pop_levels_n <- length(pop_levels)

#Differentiate data per population
for(i in 1:pop_levels_n){
  if(i == 1){
    pop_subset <- list(subset(dr, population == pop_levels[i]))
  }else{
    pop_subset <- c(pop_subset,list(subset(dr, population == pop_levels[i])))
  }
}

#Differentiate data per individual
indi_levels <- levels(dr$individual)

```

```

indi_levels_n <- length(indi_levels)
for(i in 1:indi_levels_n){
  if(i == 1){
    indi_subset <- list(subset(dr, individual == indi_levels[i]))
  }else{
    indi_subset <- c(indi_subset,list(subset(dr, individual == indi_levels[i])))
  }
}

#Compute animation input list with all individuals per population
pop_select <- 2 #Selectin a population within the data
match_indi_subs <- as.integer(na.omit(match(indi_levels,pop_subset[[pop_select]]$individual)))
for(i in 1:length(match_indi_subs)){
  if(i == 1){
    indi_subset_pop <- list(pop_subset[[pop_select]][match_indi_subs[i]:
      (match_indi_subs[i+1]-1),])
  }else{
    if(i != length(match_indi_subs)){
      indi_subset_pop[i] <- list(pop_subset[[pop_select]][match_indi_subs[i]:
        (match_indi_subs[i+1]-1),])
    }else{
      indi_subset_pop[i] <- list(pop_subset[[pop_select]][match_indi_subs[i]:
        length(pop_subset[[pop_select]][,1]),])
    }
  }
}

#Calculate dt stamps, extract start_dt and stop_dt for all arrays
for(i in 1:length(indi_subset_pop)){
  dt <- c(paste0(indi_subset_pop[[i]]$date, " ",indi_subset_pop[[i]]$time))
  dt_stamps <- as.POSIXct(strptime(dt, "%Y-%m-%d %H:%M:%S", tz = "UTC"))
  indi_subset_pop[[i]]$dt <- align.time(dt_stamps, n=60)
}
start_dt <- st_times(indi_subset_pop)[[1]]
stop_dt <- st_times(indi_subset_pop)[[2]]

#Animal specifications, here for the White Stork
max_speed <- 45
max_speed_min <- max_speed/60 #km/min
tolerance_m_min <- 50 #meter
temp_res <- 5 #for simplicity, we do not detect the temp. resolution automatically here

max_speed_spec <- max_speed_min*temp_res #km/temp_res
max_dist_m <- (max_speed_spec*1000)+tolerance_m_min #m/temp_res + tolerance range in m/temp_res

#Clean up individual data by defined props (eliminating peaks etc.), store them per individual
for(i in 1:length(indi_subset_pop)){
  data_clean <- indi_subset_pop[[i]]
  no_peaks <- FALSE
  while(no_peaks == FALSE){
    data_clean <- calc_dist(data_clean)
    dist_peaks <- which(is.na(data_clean$dist_m) == FALSE & data_clean$dist_m >= max_dist_m)
    if(length(dist_peaks) <= 1){

```

```

        no_peaks <- TRUE
      } else {
        data_clean <- data_clean[-dist_peaks[1:length(dist_peaks)],]
      }
    }
    if(i == 1){indi_subset_clean <- list(data_clean)}
  }else{indi_subset_clean[i] <- list(data_clean)}
  }
  indi_subset_clean[[i]]$peak <- 0
  indi_subset_clean[[i]]$peak[
    which(is.na(indi_subset_clean[[i]]$dist_m) == FALSE &
      indi_subset_clean[[i]]$dist_m >= max_dist_m)] <- 1
}

#Recalculate start/stop times
start_dt <- st_times(indi_subset_clean)[[1]]
stop_dt <- st_times(indi_subset_clean)[[2]]

#Interpolate tracks
for(i in 1:length(indi_subset_clean)){
  out_n <- as.integer(difftime(stop_dt[i],start_dt[i],units="mins"))+1
  coords <- matrix(c(indi_subset_clean[[i]]$lon,indi_subset_clean[[i]]$lat),
    nrow=length(indi_subset_clean[[i]]$lon))
  lon_inter <- approx(coords[,1], n = out_n)
  lat_inter <- approx(coords[,2], n = out_n)
  dt_new <- seq(indi_subset_clean[[i]]$dt[1],
    indi_subset_clean[[i]]$dt[length(indi_subset_clean[[i]]$dt)], length.out = out_n)

  pop_name <- as.character(dt_new)
  pop_name[1:length(pop_name)] <- as.character(indi_subset_clean[[i]]$population[1])
  indi_name <- as.character(dt_new)
  indi_name[1:length(indi_name)] <- as.character(indi_subset_clean[[i]]$individual[1])
  if(i==1){
    indi_subset_int <- list(data.frame(lon_inter$y,lat_inter$y,pop_name,indi_name,dt_new))
    colnames(indi_subset_int[[i]]) <- c("lon","lat","population","individual","dt")
  }else{
    indi_subset_int[i] <- list(data.frame(lon_inter$y,lat_inter$y,pop_name,indi_name,dt_new))
    colnames(indi_subset_int[[i]]) <- c("lon","lat","population","individual","dt")
  }
  indi_subset_int[[i]] <- calc_dist(indi_subset_int[[i]])
}

#Segmentate moving/resting
resting_rad <- 20 #meters/min
for(i in 1:length(indi_subset_int)){
  if(i == 1){
    indi_subset_class <- list(class_moving(indi_subset_int[[i]],resting_rad))
  }else{
    indi_subset_class[i] <- list(class_moving(indi_subset_int[[i]],resting_rad))
  }
}

#Removing non-moving time periods...

```

```

for(i in 1:length(indi_subset_class)){
  if(i == 1){indi_subset_moving <- list(
    indi_subset_class[[i]][which(indi_subset_class[[i]]$moving == 1),]}
  else{indi_subset_moving[i] <- list(
    indi_subset_class[[i]][which(indi_subset_class[[i]]$moving == 1),]}
}

#Extract names of each individual
for(i in 1:length(indi_subset_moving)){
  if(length(indi_subset_moving[[i]][,1]) > 0){
    if(i==1){
      indi_names <- indi_subset_moving[[i]]$individual[1]
    }else{
      indi_names <- paste0(indi_names,", ",indi_subset_moving[[i]]$individual[1])
    }
  }
}

#Create move objects list
move_index <- 0
for(i in 1:length(indi_subset_moving)){
  maxi <- length(indi_subset_moving[[i]]$lon)
  #maxi <- 200 #If the maximum length should be defined by the user
  #if(length(indi_subset_moving[[i]]$lon) < maxi){
    maxi <- length(indi_subset_moving[[i]]$lon)
  }

  if(length(indi_subset_moving[[i]][,1]) > 0){
    if(i == 1){data_ani <-
      list(move(x=indi_subset_moving[[i]]$lon[1:maxi],y=indi_subset_moving[[i]]$lat[1:maxi],
        time=indi_subset_moving[[i]]$dt[1:maxi],proj=CRS("+proj=longlat +ellps=WGS84"),
        animal=unlist(strsplit(indi_names,", "))[i]))
    }else{data_ani[i-move_index] <-
      list(move(x=indi_subset_moving[[i]]$lon[1:maxi],y=indi_subset_moving[[i]]$lat[1:maxi],
        time=indi_subset_moving[[i]]$dt[1:maxi],proj=CRS("+proj=longlat +ellps=WGS84"),
        animal=unlist(strsplit(indi_names,", "))[i]))
    }else{move_index <- move_index+1}
  }
}

#write out
file.create("moveVis_sample.txt")
for(i in 1:length(indi_subset_moving)){
  print(i)
  if(i == 1){
    writethis <- indi_subset_moving[[i]]
    write.table(writethis,file="samples.txt",sep=";")
  }else{
    writethis <- rbind(writethis,indi_subset_moving[[i]])
    write.table(writethis,file="samples.txt",sep=";")
  }
}

## End(Not run)

```


Index

*Topic **datasets**

basemap_data, 17

move_data, 20

animate_move, 2, 3, 6, 12, 13, 15, 17–20

animate_raster, 2, 7, 9

animate_stats, 2, 4, 6, 7, 13

basemap_data, 17

get_formats, 2, 6, 11, 15, 17

get_imconvert (get_libraries), 18

get_libraries, 2, 3, 6, 7, 10–13, 15–18, 18

idData, 4

move_data, 20

moveVis (moveVis-package), 2

moveVis-package, 2