Package: wpeR (via r-universe)

June 11, 2026

Type Package

Title Streamlined Analysis of Wild Pedigree Data

Version 0.2.0

Description Analyzing pedigree data of wild populations. While primarily designed to process outputs from the 'COLONY' (Jones & Wang (2010) <doi:10.1111/j.1755-0998.2009.02787.x>) pedigree reconstruction software, it can also accommodate data from other sources. By linking reconstructed pedigrees with genetic sample metadata, 'wpeR' produces spatial and temporal visualizations as well as tabular summaries that support interpretation of family structures and dynamics. The main goal of the package is to provide a solution for the analysis of complex wild pedigree data and to help the user to gain insights into genetic relationships within wild animal populations.

License GPL (>= 3)

URL https://gr3602.github.io/wpeR/

BugReports https://github.com/GR3602/wpeR/issues

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, sf, stats, utils

Suggests basemaps, ggforce, ggrepel, gridExtra, kinship2, knitr, leaflet, leaflet.providers, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Tomaz Skrbinsek [aut], Gregor Simcic [aut, cre]

Maintainer Gregor Simcic <grega0simcic@gmail.com>

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

Date/Publication 2026-06-11 16:34:57 UTC

RemoteUrl https://github.com/cran/wpeR

RemoteRef HEAD

RemoteSha 65d3be2215ebf36bd033f9d77e34525679f9731a

Contents

`anim_timespan` *Get dates of individuals first and last sample*

Description

Takes data frame of all samples and returns the dates of individuals first and last sample. Besides that the function determines if an individual is dead based on the mortality_sample column.

Usage

```
anim_timespan(individual_id, sample_date, mortality_sample)
```

Arguments

`individual_id`	Column in the dataframe of all samples containing individual animal identifier code. Defined as `dataframe$column`.
`sample_date`	Column in the dataframe of all samples containing the date of sample collection. Must be in `Date` format. Defined as `dataframe$column`.
`mortality_sample`	Logical vector or column in the dataframe of all samples that identifies samples that represent a mortality event (e.g. from a dead animal). `TRUE` values mark mortality samples. Defined as `dataframe$column` or a standalone logical vector the same length as `individual_id`.

Value

A data frame with four columns and one row for each individual_id. Returned data frame columns correspond to individual identification key (ID), date of first (FirstSeen) and last (LastSeen) sample of individual and logical (TRUE/FALSE) value that identifies if the individual is dead (IsDead).

Examples

```
anim_timespan(
  individual_id = wolf_samples$AnimalRef,
  sample_date = wolf_samples$Date,
  mortality_sample = wolf_samples$IsMortality
)
```

---

check_sampledata *Check and prepare genetic sample metadata*

Description

Verifies the consistency of columns in the genetic sample metadata and prepares it for use with other functions in the wpeR package. The function ensures that the provided data is properly formatted and conforms to the standards of functions that make up the wpeR package.

Usage

```
check_sampledata(
  Sample,
  Date,
  AnimalRef,
  GeneticSex,
  lat,
  lng,
  SType,
  IsMortality = NULL,
  extraCols = NULL
)
```

Arguments

Sample	A vector of sample unique identifier codes.
Date	A vector of sample collection dates in 'YYYY-MM-DD' format.
AnimalRef	A vector of identifier codes of the particular individual that the sample belongs to.
GeneticSex	A vector of genetic sex information ('F' for female, 'M' for male, NA for unknown).
lat	A vector of latitude coordinates in the WGS84 coordinate system (EPSG: 4326).

lng	A vector of longitude coordinates in the WGS84 coordinate system (EPSG: 4326).
SType	A vector of sample types eg.: scat, hair, tissue.
IsMortality	Optional logical vector of the same length as the input vectors. TRUE marks samples that represent a mortality event (e.g. from a dead animal). Defaults to NULL.
extraCols	A vector of extra column names that the user wants to include in sampledata data frame (see Details).

Details

By specifying the extraCols parameter additional information can be included in the sampledata dataframe. Such additional information is not required for the functioning of the wpeR package functions, but can be useful to the user when interpreting results. When including additional columns the function inputs (Sample, Date, AnimalRef...) have to be defined as a vector extracted from data frame column (eg. Sample = dataframe$column) and the extraCols parameter is defined, as a vector of column names form the same data frame (eg. extraCols = c(column1, column2, column3)).

The IsMortality column is a logical vector that flags samples representing a mortality event (e.g. from a dead animal). It is used by anim_timespan() to determine the death status of individuals.

Value

A data frame with a number of rows equal to the length of the input vectors. Each column corresponds to one of the input parameters. If IsMortality is provided, it is included as an additional column. If the function executes without warnings or errors, the result from check_sampledata() can be used as an input parameter for other functions within this package: get_colony(), get_ped(), org_fams() and plot_table().

Examples

```
sampledata <- check_sampledata(
  Sample = wolf_samples$Sample,
  Date = wolf_samples$Date,
  AnimalRef = wolf_samples$AnimalRef,
  GeneticSex = wolf_samples$GeneticSex,
  lat = wolf_samples$lat,
  lng = wolf_samples$lng,
  SType = wolf_samples$SType,
  IsMortality = wolf_samples$IsMortality
)
```

dyn_matrix *Get matrix of apparent survival*

Description

Creates a matrix that shows number of captured animals between multiple seasons.

Usage

```
dyn_matrix(animal_id, capture_date, start_dates, end_dates)
```

Arguments

animal_id	A column in the dataframe of all samples that stores individual animal identifier code.
capture_date	A column in the dataframe of all samples that stores the date of sample collection. Must be in Date format.
start_dates	Vector of dates in Date format that define the start of each season.
end_dates	Vector of dates in Date format that define the end of each season.

Value

A matrix with 1 + no. seasons rows and columns.

- diagonal: number of new captures in each session,
- above diagonal: number of recaptures from season x to season y,
- below diagonal: number of animals from season y that skipped season x.

Season x is defined in first row, season y in first column. Column Tot. Capts gives all detected individuals in season y. Row Tot. Skipped gives all individuals skipped in season x but detected later.

Examples

```
# Define start and end dates for sampling seasons.
seasons <- data.frame(
  start = c(
    as.Date("2017-01-01"),
    as.Date("2018-01-01"),
    as.Date("2019-01-01")
  ),
  end = c(
    as.Date("2017-12-31"),
    as.Date("2018-12-31"),
    as.Date("2019-12-31")
  )
)
```

```
# Create a dynamics matrix for animal captures.
dyn_matrix(
  animal_id = wolf_samples$AnimalRef,
  capture_date = wolf_samples$Date,
  start_dates = seasons$start,
  end_dates = seasons$end
)
```

`get_colony` *Organizes COLONY output*

Description

Extends `BestConfig_Ordered` output from COLONY pedigree reconstruction software with additional data about individuals included in pedigree. The function adds missing parents to `OffspringID`, assigns sex to each individual included in `OffspringID` and adds the computed probabilities of paternity and maternity assignments (probability of assignments is visible only if the `out` parameter is set to `"table"`). The function also prepares data so that the output of the function can be directly analyzed with `kinship2`, `pedtools` or `FamAgg` packages.

Usage

```
get_colony(
  colony_project_path,
  sampledata,
  rm_obsolete_parents = TRUE,
  out = "FamAgg"
)
```

Arguments

`colony_project_path`
: Character string. Path to the folder where COLONY output files are saved. Has to include file path and project name (see Details).

`sampledata`
: Data frame. Metadata for all genetic samples that belong to the individuals included in pedigree reconstruction analysis. This data frame should adhere to the formatting and naming conventions outlined in the `check_sampledata()` documentation.

`rm_obsolete_parents`
: Logical. Should unknown parents be removed from output. Applies just to offspring for which both parents are unknown. Defaults to `TRUE`.

`out`
: Character string. For use with which package should the output be formatted? `kinship2` (out = "kinship2"), `pedtools` (out = "pedtools"), `FamAgg` (out = "FamAgg") or the created data.frame can be outputted as is (out = "table"). Defaults to "FamAgg"

Details

COLONY output tables needed for this function (`.BestConfig_Ordered`, `.Maternity` and `.Paternity`) are read directly from the colony output folder and do not need to be imported into R session. The path to the outputs is defined with `colony_project_path` parameter. When defining `colony_project_path` the user needs to define a complete path to the directory where colony outputs are stored and also the file name (file name of COLONY outputs equals the project name
eg. /path/to/the/COLONY/output/folder/COLONY_project_name).

Value

A data frame describing a common pedigree structure. Each individual included in pedigree represents one row. Columns describe individual identifier code, identifier code for mother and father, sex and family of individual. Column names and arrangement depends on selected output (`out` parameter).

Examples

```
# Define the path to COLONY output
path <- paste0(system.file("extdata", package = "wpeR"), "/wpeR_samplePed")

# Get pedigree data in FamAgg format
get_colony(
    colony_project_path = path,
    sampledata = wolf_samples
    )
```

`get_ped` *Organizes pedigree data*

Description

Offers an alternative to `get_colony()` function in cases where the pedigree was not reconstructed with COLONY software. It takes a pedigree dataframe and assigns sex to each individual. The function also prepares data so that the output of the function can be directly analyzed with `kinship2`, `pedtools` or `FamAgg` packages.

Usage

```
get_ped(ped, sampledata, out = "FamAgg")
```

Arguments

`ped`	Data frame. Pedigree data frame with the most basic structure. Three columns corresponding to offspring, father and mother (see Details). Unknown parents should be represented by `NA` values.

sampledata	Data frame. Metadata for all genetic samples that belong to the individuals included in pedigree reconstruction analysis. This data frame should adhere to the formatting and naming conventions outlined in the check_sampledata() documentation.
out	Character string. For use with which package should the output be formatted? kinship2 (out = "kinship2"), pedtools (out = "pedtools") or FamAgg (out = "FamAgg") or the created data.frame can be outputted as is (out = "table"). Defaults to "FamAgg"

Details

The custom pedigree specified through the ped parameter should mirror the structure of a COLONY pedigree and share the same column names. It should consist of three columns for each offspring: OffspringID, FatherID, MotherID. When considering unknown parents they should be represented by NA values.

Value

A data frame describing a common pedigree structure. Each individual included in pedigree represents one row. Columns describe individual identifier code, identifier code for mother and father and sex of individual. Column names and arrangement depends on selected output (out parameter).

Examples

```
#example pedigree dataframe
ped <- data.frame(
  OffspringID = c(
    "M273P", "M20AM", "M2757", "M2ALK", "M2ETE", "M2EUJ", "MSV00E",
    "MSV018", "MSV05L", "MSV0M6", "MSV0T4", "MSV0T7", "MSV0TJ", "MSV0UL"
  ),
  FatherID = c(
    NA, NA, "M20AM", "M20AM", "M20AM", "M20AM", "M20AM",
    "M20AM", "M20AM", "M20AM", "M20AM", "M20AM", "M20AM", "M20AM"
  ),
  MotherID = c(
    NA, NA, "M273P", "M273P", "M273P", "M273P", "M273P",
    "M273P", "M273P", "M273P", "M273P", "M273P", "M273P", "M273P"
  )
)
#Get pedigree data in FamAgg format
get_ped(
    ped = ped,
    sampledata = wolf_samples
    )
```

nbtw_seasons *Number of detected animals between two sampling seasons*

Description

Gives an numeric overview of individuals captured within the second sampling season compared tho the first one.

Usage

```
nbtw_seasons(
  animal_id,
  capture_date,
  season1_start,
  season1_end,
  season2_start,
  season2_end
)
```

Arguments

animal_id	A column in the dataframe of all samples that stores individual animal identifier code.
capture_date	A column in the dataframe of all samples that stores the date of sample collection. Must be in Date format.
season1_start	String in Date format. Start of fist capture season. Start and end date are included in the capture season.
season1_end	String in Date format. End of fist capture season. Start and end date are included in the capture season.
season2_start	String in Date format. Start of second capture season. Start and end date are included in the capture season.
season2_end	String in Date format. End of second capture season. Start and end date are included in the capture season.

Value

A data frame with one row and six columns corresponding to season 1 and 2 start and end dates, number of detected animals in season 2 (total_cap), number of new detentions in season 2 (new_captures), umber of animals from season 1 detected within season 2 (recaptured) and number of individuals skipped in season 2 but detected after the end of that season (skipped).

Examples

```
# Calculate the number of animals detected between two sampling seasons.
nbtw_seasons(
 animal_id = wolf_samples$AnimalRef,
```

```
 capture_date = wolf_samples$Date,
 season1_start = as.Date("2017-01-01"),
 season1_end = as.Date("2017-12-31"),
 season2_start = as.Date("2018-01-01"),
 season2_end = as.Date("2018-12-31")
)
```

`org_fams` *Organize animals into families and expand pedigree data*

Description

Takes pedigree data from `get_colony()` or `get_ped()` function and groups animals into families. It also expands the pedigree data by adding information about the family that each individual was born in and the family in which the individual is the reproductive animal.

Usage

```
org_fams(ped, sampledata, output = "both")
```

Arguments

`ped`	Data frame. `FamAgg` output of `get_colony()` or `get_ped()` function. With `rm_obsolete_parents` parameter set to `TRUE`.
`sampledata`	Data frame. Metadata for all genetic samples that belong to the individuals included in pedigree reconstruction analysis. This data frame should adhere to the formatting and naming conventions outlined in the `check_sampledata()` documentation.
`output`	Character string. Determines the format of the output. Options are: "ped": returns an extended pedigree data frame. "fams": returns a table of all families present in the pedigree. "both": returns a list with two data frames: "ped" and "fams". (Default)

Details

Families and Half-sib Groups The result of `org_fams()` function introduces us to two important concepts within the context of this package: family and half-sib group. A family in the output of this function is defined as a group of animals where at least one parent and at least one offspring is known. A half-sib group refers to a group of half-siblings, either maternally or paternally related. In the function output the `DadHSgroup` groups paternal half-siblings and `MomHSgroup` maternal half-siblings.

Lineage Isolation for Visualization The `DadHSgroup` and `MomHSgroup` identifiers are useful for managing pedigree visualization in long-lived or polygamous species. By using `DadHSgroup` or `MomHSgroup`, users can identify all `FamIDs` or individuals associated with a specific parent. This

vector of IDs or individuals of interest can then be passed to the plot_fams or plot_indivs argument in plot_table() to isolate and visualize specific paternal or maternal lineages, preventing visual clutter in ped_satplot().

Temporal Estimation (famStart and famEnd) The fams output dataframe contains famStart and famEnd columns, which estimate a time window for the family based solely on sample collection dates provided in sampledata. famStart marks the date of the earliest sample collected from any offspring belonging to that family. famEnd indicates the date of the latest sample collected from either the mother or the father of that family. It is important to recognize that this method relies on observation (sampling) times. Consequently, famEnd (last parental sample date) can precede famStart (first offspring sample date), creating a biologically impossible sequence and a negative calculated family timespan. Users should interpret the interval between famStart and famEnd with this understanding.

Value

Depending on the output parameter, the function returns either a data frame (ped or fams) or a list containing both data frames (ped and fams).

- ped data frame. An extended version of the pedigree data from get_colony()/get_ped(). In addition to common pedigree information (individual, mother, father, sex, family), ped includes columns for:
  - parents: Identifier codes of both parents separated with _.
  - FamID: Numeric identifier for the family to which the individual belongs (see fams below).
  - FirstSeen: Date of first sample of individual.
  - LastSeen: Date of last sample of individual.
  - IsDead: Logical value (TRUE/FALSE) that identifies if the individual is dead.
  - DadHSgroup: Identifier of paternal half-sib group (see Details).
  - MomHSgroup: Identifier of maternal half-sib group (see Details).
  - hsGroup: Numeric value indicating if the individual is part of a half-sib group (see Details).
- fams data frame includes information on families that individuals in the pedigree belong to. The families are described by:
  - parents: Identifier codes of both parents separated with _.
  - father: Identifier code of the father.
  - mother: Identifier code of the mother.
  - FamID: Numeric identifier for the family.
  - famStart: Date when the first sample of one of the offspring from this family was collected (see Details).
  - famEnd: Date when the last sample of mother or father of this family was collected (see Details).
  - FamDead: Logical value (TRUE/FALSE) indicating if the family no longer exists.
  - DadHSgroup: Identifier connecting families that share the same father.
  - MomHSgroup: Identifier connecting families that share the same mother.
  - hsGroup: Numeric value connecting families that share one of the parents.

Examples

```
# Prepare the data for usage with org_fams() function.
# Get animal timespan data using the anim_timespan() function.
animal_ts <- anim_timespan(
  wolf_samples$AnimalRef,
  wolf_samples$Date,
  wolf_samples$IsMortality
)
# Add animal timespan to the sampledata
sampledata <- merge(wolf_samples, animal_ts, by.x = "AnimalRef", by.y = "ID", all.x = TRUE)
# Define the path to the pedigree data file.
path <- paste0(system.file("extdata", package = "wpeR"), "/wpeR_samplePed")
# Retrieve the pedigree data from the get_colony function.
ped_colony <- get_colony(path, sampledata, rm_obsolete_parents = TRUE, out = "FamAgg")

# Run the function
# Organize families and expand pedigree data using the org_fams function.
org_fams(
    ped = ped_colony,
    sampledata = sampledata
    )
```

`ped_satplot` *Temporal plot of pedigree*

Description

Creates "capture" history plot of individuals arranged by families included in data frame created by `plot_table()` function.

Usage

```
ped_satplot(
  plottable,
  famSpacing = 2,
  hsGroupSpacing = 2,
  xWhiteSpace = 100,
  xlabel = "Date",
  ylabel = "Animal",
  title = "",
  subtitle = "",
  LegendLabel = "Sex",
  xlegend = 0.2,
  ylegend = 0.94,
  text_size = 2.5,
  fam_label_size = 2,
```

```
  polyHighlight = TRUE
)
```

Arguments

plottable	Data frame. Output of plot_table() function.
famSpacing	Y-axis spacing between families. Should be even number!
hsGroupSpacing	Y-axis spacing between half-sib groups. Should be even number!
xWhiteSpace	Spacing on the X-axis at the beginning and end of the plot.
xlabel	X-axis label.
ylabel	Y-axis label.
title	Plot title.
subtitle	Plot subtitle.
LegendLabel	Title of the legend.
xlegend	Horizontal position of the legend.
ylegend	Vertical position of the legend.
text_size	Plot text size.
fam_label_size	Family label text size.
polyHighlight	Logical. Show semi-transparent background for polygamous individuals. Default is TRUE.

Value

A graphical representation of detected family members trough time.

Examples

```
# Prepare the data for usage with plot_table() function.
# Get animal timespan data using the anim_timespan() function.
animal_ts <- anim_timespan(wolf_samples$AnimalRef,
  wolf_samples$Date,
  wolf_samples$IsMortality
)
# Add animal timespan to the sampledata
sampledata <- merge(wolf_samples, animal_ts, by.x = "AnimalRef", by.y = "ID", all.x = TRUE)
# Define the path to the pedigree data file.
path <- paste0(system.file("extdata", package = "wpeR"), "/wpeR_samplePed")
# Retrieve the pedigree data from the get_colony function.
ped_colony <- get_colony(path, sampledata, rm_obsolete_parents = TRUE, out = "FamAgg")
# Organize families and expand pedigree data using the org_fams function.
org_tables <- org_fams(ped_colony, sampledata, output = "both")

# Prepare data for plotting.
# Plot subset of families
pt <- plot_table(plot_fams = 1,
                 all_fams = org_tables$fams,
                 ped = org_tables$ped,
```

```
                 sampledata = sampledata
                )

# Run the function.
# Get a temporal pedigree plot.
ped_satplot(plottable = pt)



# Plot table for a subset of individuals
pt_subset <- plot_table(plot_indivs = "M200F",
                        all_fams = org_tables$fams,
                        ped = org_tables$ped,
                        sampledata = sampledata
                       )

# Run the function.
# Get a temporal pedigree plot.
ped_satplot(plottable = pt_subset)
```

`ped_spatial` *Get files for spatial representation of pedigree*

Description

Creates georeferenced data for spatial pedigree representation form the output of `plot_table()` function.

Usage

```
ped_spatial(
  plottable,
  na.rm = TRUE,
  output = "list",
  fullsibdata = NULL,
  sibthreshold = 0,
  path = "",
  filename = "",
  out.format = "geopackage",
  time.limits = c(as.Date("1900-01-01"), as.Date("2100-01-01")),
  time.limit.rep = FALSE,
  time.limit.offspring = FALSE,
  time.limit.moves = FALSE
)
```

Arguments

`plottable`	Data frame. Output of `plot_table()` function.

na.rm	Logical (TRUE/FALSE). Remove samples with missing coordinates and/or dates.
output	Character vector specifying the desired output type ('list' - default or 'gis'). Available outputs: list: all spatial data returned as list, gis: all spatial data returned as georeferenced files.
fullsibdata	Data frame with COLONY full-sibling data.
sibthreshold	Numeric. P-value threshold for sibship assignment.
path	System path for storing georeferenced files.
filename	Common name for all georeferenced files.
out.format	Character string. Type of georeferenced files to be generated. Can be ether "geopackage" or "shapefile". Default is "geopackage"
time.limits	Vector of two Date values as the time window.
time.limit.rep	Logical (TRUE/FALSE). Apply time limits to reference samples of reproductive animals.
time.limit.offspring	Logical (TRUE/FALSE). Apply time limits to reference samples of offspring.
time.limit.moves	Logical (TRUE/FALSE). Apply time limits to movement data.

Details

The parameters path, filename and out.format, are used only when output parameter is set to "gis", since they control which georeferenced files should be created, where they will be saved and which common file name will they have.

Value

Depending on the output parameter the function can return a list of sf objects, a georeferenced vector data files or both.

Most of the objects are created separately for mothers, fathers and offspring, this include:

- Reference Points (motherRpoints, fatherRpoints, and offspringRpoints).
  - Each point corresponds to an animal included in the 'plot_table()' function output.
  - For reproductive animals (mothers and fathers), a reference point is the location of their last sample within the specified time window.
  - For offspring, the reference point is the location of their first sample within the time window.
- Movement Points (motherMovePoints, fatherMovePoints, and offspringMovePoints).
  - These points represent all the samples of the respective animals.
- Movement Lines (motherMoveLines, fatherMoveLines and offspringMoveLines).
  - Movement lines connect all '...MovePoints' of a specific animal in chronological order.
- Movement Polygons (motherMovePolygons, fatherMovePolygons and offspringMovePolygons):
  - Movement polygons represent a convex hull that encloses all the samples of an individual.
  - An individual must have more than two samples for this representation.

Besides that the function also produces lines that connect mothers and their offspring (maternityLines), fathers and their offspring (paternityLines), and if fullsibdata parameter is specified, full siblings (FullsibLines).

Examples

```
# Prepare the data for usage with ped_spatial() function.
# Get animal timespan data using the anim_timespan() function.
animal_ts <- anim_timespan(wolf_samples$AnimalRef,
  wolf_samples$Date,
  wolf_samples$IsMortality
)
# Add animal timespan to the sampledata
sampledata <- merge(wolf_samples, animal_ts, by.x = "AnimalRef", by.y = "ID", all.x = TRUE)
# Define the path to the pedigree data file.
path <- paste0(system.file("extdata", package = "wpeR"), "/wpeR_samplePed")
# Retrieve the pedigree data from the get_colony function.
ped_colony <- get_colony(path, sampledata, rm_obsolete_parents = TRUE, out = "FamAgg")
# Organize families and expand pedigree data using the org_fams function.
org_tables <- org_fams(ped_colony, sampledata, output = "both")
# Prepare data for plotting.
pt <- plot_table(plot_fams = 1,
  all_fams = org_tables$fams,
  ped = org_tables$ped,
  sampledata = sampledata,
)

# Run the function
# Get files for spatial pedigree representation in list format.
ped_spatial(plottable = pt)
```

---

plot_table *Prepares pedigree data for plotting and spatial representation*

Description

Combines extended pedigree (obtained by `org_fams()` function) and sample metadata data for visual (`ped_satplot()`) and spatial (`ped_spatial()`) representation of the pedigree.

Usage

```
plot_table(
  plot_fams = NULL,
  plot_indivs = NULL,
  all_fams,
  ped,
  sampledata,
 datacolumns = c("Sample", "AnimalRef", "GeneticSex", "Date", "SType", "lat", "lng",
    "FirstSeen", "LastSeen", "IsDead", "IsMortality")
)
```

Arguments

plot_fams	Character string or numeric vector. Defines which families to include. Use "all" (or NULL, default) to include all families, or a numeric vector of specific FamID numbers to plot a subset (see Details).
plot_indivs	Character string or vector of character strings. Individual identifiers to include. The families associated with these individuals (as father, mother, or offspring) will be included in the output (see Details).
all_fams	Data frame. Family (fams) data generated by org_fams() function.
ped	Data frame. Organized pedigree (ped) generated by org_fams() function.
sampledata	Data frame. Metadata for all genetic samples that belong to the individuals included in pedigree reconstruction analysis. For description of sampledata structure and sample information needed for plot_table() see Details.
datacolumns	Vector of column names included sampledata that are needed to produce this functions output (see Details).

Details

- sampledata has to include columns that contain information on:
  - unique identifier of each sample; character or numeric (default column name = Sample, see check_sampledata() function),
  - date of sample collection in Date format (default = Date),
  - assignment of sample to particular individual; character or numeric (default = AnimalRef, see check_sampledata() function),
  - sex of the animal coded as F, M or NA; character (default = GeneticSex, see check_sampledata() function),
  - longitude and latitude coordinates of sample collection location; numeric (default = lng and lat, see check_sampledata() function),
  - type of particular sample eg. scat, tissue, saliva; character (default = SType, see check_sampledata() function),
  - date of first and last sample of individual in Date format (default = FirstSeen and LastSeen, see anim_timespan() function),
  - value identifying if the individual is dead; logical (default = IsDead, see anim_timespan() function).
  - value identifying if this particular sample represents a mortality sample (default = IsMortality, see check_sampledata() function)

Filtering Options The plot_table function allows for flexible subsetting of families to be plotted using the plot_fams and plot_indivs arguments:

- plot_fams: A numeric vector of specific FamIDs to include.
- plot_indivs: A vector of individual identifiers. Families where these individuals appear as a father, mother, or offspring are automatically identified and included.

Behavior:

- If both plot_fams and plot_indivs are NULL (default), all families are included.
- If one or both are provided, the final set of families is the **union** of families identified by both filters.

Value

Extended sampledata data frame that includes all columns defined in datacolumns parameter and adds information needed for visual and spatial representation of pedigree:

- plottingID: Numeric. Identifier number for temporal pedigree plot ped_satplot(). In case of polygamous animals same individual can be included in more than one family.
- FamID: Numeric. Identifier number of family that individual belongs to.
- hsGroup: Numeric. Identifier number for the half-sib group of individual.
- rep: Logical. Is individual reproductive in current family, (current family defined with FamID for a particular entry).
- later_rep: Logical. Is individual reproductive in any other (later) families.
- isPolygamous: Logical. Does the individual have more than one mate.
- dead: Logical. Is individual dead.
- first_sample: Logical. Is this particular sample the first sample of the individual.
- last_sample: Logical. Is this particular sample the last sample of the individual.

Examples

```
# Prepare the data for usage with plot_table() function.
# Get animal timespan data using the anim_timespan() function.
animal_ts <- anim_timespan(wolf_samples$AnimalRef,
  wolf_samples$Date,
  wolf_samples$IsMortality
)
# Add animal timespan to the sampledata
sampledata <- merge(wolf_samples, animal_ts, by.x = "AnimalRef", by.y = "ID", all.x = TRUE)
# Define the path to the pedigree data file.
path <- paste0(system.file("extdata", package = "wpeR"), "/wpeR_samplePed")
# Retrieve the pedigree data from the get_colony function.
ped_colony <- get_colony(path, sampledata, rm_obsolete_parents = TRUE, out = "FamAgg")
# Organize families and expand pedigree data using the org_fams function.
org_tables <- org_fams(ped_colony, sampledata, output = "both")

# Run the function
# Standard usage: all families
pt <- plot_table(plot_fams = "all",
  all_fams = org_tables$fams,
  ped = org_tables$ped,
  sampledata = sampledata,
)


# Plot table for a subset of individuals
pt_subset <- plot_table(plot_indivs = "M200F",
                        all_fams = org_tables$fams,
                        ped = org_tables$ped,
                        sampledata = sampledata
                       )
```

`wolf_samples` *Wolf monitoring genetic samples metadata*

Description

Metadata of selected genetic samples of wolves collected between 2015 and 2021, in the scope of Slovenian National Wolf Monitoring

Usage

```
wolf_samples
```

Format

A data frame with 407 rows and 8 columns:

Sample Sample unique identifier code

Date Date of sample collection (format: `YYYY-MM-DD`)

AnimalRef Identification string for particular animal

GeneticSex Sex of animal to which the sample belong (format: `M` = male, `F` = female)

lat latitude (N-S) of the sample (CRS: WGS84; EPSG: 4326)

lng longitude (W-E) of the sample (CRS: WGS84; EPSG: 4326)

SType Type of the sample. (`Direct Saliva, Scat, Urine, Saliva, Tissue, Decomposing Tissue, Blood`)

IsMortality A logical column (TRUE/FALSE) that marks samples collected from dead animals. TRUE = mortality event (e.g., a tissue sample from a carcass), FALSE = non-mortality sample (e.g., scat, hair, saliva from a live animal).

Source

Slovenian National Wolf Monitoring

Index