APPENDIX 1. Detailed explanation and example code for implementing LPIC, Likelihood-based photograph identification code, in Program R, Version 2.15.1. LPIC assigns animal ids to a database of photographs using phenotypic measurements extracted from each photograph.

Name of function

lpic (Likelihood-based photograph identification code)

Description

Assigns animal ids to a database of photographs using phenotypic measurements of each photograph.

Usage

lpic(base = base, pics = "pics", sd = sd, wt = wt, preface = "f")

Arguments

base	a dataframe with a column for id, image name, image information, and an unspecified number of measurement columns. Column 1 (name: "id") represents the known and unknown ids within the database of images in question. For known ids, ids must be either in integer or number format, so the function can continue to add ids to the database in consecutive order. For unknown ids a NA should be specified. Column 2 (name: "photo_name") represents the image name for each image in the database. Image names must be in character format without their suffix (i.e., without the .JPG). Column 3 (name: "info") represents user-specified extra information, which will be displayed for each photo during final match selection. This column must be in character format and no longer than 20 characters long. Columns 4 through un-specified (names: does not matter) represent each measurement corresponding to each image. These columns represent the data to be used during the likelihood calculations. If there is a missing measurement value, specify with NA (note: once a photograph with a NA for a measurement value has an id, its probability of being matched to a new unknown photo is low).
pics	a character string representing the name of the folder within the working drive where the images are located. The images must be in jpeg format, with the suffix, ".JPG". Images must also be smaller than 800kb in order for the software to run quickly. For efficiency purposes in the graphics window, we suggest cropping images to the area on the animal where the information is present, cropping each image the same way, and cropping to a rectangle where with height is greater than the width.
sd	a vector of standard deviations corresponding to each measurement. Length of this vector should equal the number of measurements in base.

wt — a vector of weights corresponding to each measurement. Length of this vector should equal the number of measurements in base. If weights are undesired, a vector of 1s should be specified.

preface — a character string representing a preface which will be added to the ids at the end of the analysis. This helps the user specify different groups of animals that are analyzed separately. For example, the default preface is "f", for the female segment of the population.

Details

The user must require the package "ReadImages" prior to running the function.

The function first provides the user with a summary of the data. The user must specify if everything seems ok by typing a "y" then pressing enter. The function then checks for known ids in base$id (i.e., non NAs in the column). It immediately creates a new dataframe of known individuals. Then using a step by step process, where the function goes through the NAs in base$id one row at a time, the function compares each unknown photo to the photos in the known database.

The software uses a likelihood approach based on a Gaussian distribution to determine the similarity score between two potential photos. The similarity score between two photos (the unknown photo, and a potential known photo) is estimated based on a density distribution calculated using the measurement value of the photo in question, the measurement value of the known photo, and the specified standard deviation for that measurement. The density value is then multiplied by the weight for the measurement in question. This is done for each measurement for a given pair of photos, then summed, then divided by the maximum similarity score possible (i.e., the similarity score if 2 photos had the exact same measurement values).

The function then ranks the photographs from most similar to least similar, and provides the user with the photograph of the unknown photo and up to 5 (depending on the number of available known photos) of the most similar potential matches to the unknown photograph. The user then specifies whether or not there is a correct match by typing a string of 1s and 0s, where 1 corresponds to a match, and 0 corresponds to a non-match. The user needs only to specify 1 correct match, even if two correct matches are present. If the user specifies no match, then the unknown photo gets a new id, and this photo is then moved to the known database. If the user specifies a match, then the unknown photo gets the id of the matched known individual, and the photo is moved to the known database. The software then moves on to the next photo in base.

Value

lpic() returns a list with 2 components and creates a folder within the user's working directory named "Results_Today'sDate." The first item in the list is a dataframe resembling the original dataframe named base specified by the user, but with all of the ids filled out. The second item is a dataframe, where every row represents a calculation between two photos. This dataframe represents every calculation between pairs of photos that was carried out during the course of lpic(). In addition, both of these items are saved as .csv files within the Results folder, named as unique_ids and score.matrix, respectively. Finally, within the Results folder, the

function creates another folder named “population” where each id is given its own folder and all the images corresponding to that image are copied into its respective folder.

Example using lpic()

```
#example code for lpic
#First thing, copy lpic.R into your working directory
setwd("C:/…")  #set your working directory
install.packages ("ReadImages")

#create pics folder within wd
dir.create(paste(getwd(), "images", sep = "/"))

#copy some example photos into your pics folder
file.copy(system.file("extdata", "Rlogo.jpg", package="ReadImages"),
paste(getwd(),"/images/", "Rlogo1.JPG", sep = ""))
for(i in 2:6){
  file.copy(paste(getwd(), "/images/Rlogo1.JPG", sep = ""),
          paste(getwd(), "/images/Rlogo", i, ".JPG", sep = ""))
}

#generate columns for the base dataframe
id <- c(1, rep(NA, times = 5))
photo_name <- substr(dir(paste(getwd(), "images", sep = "/")), start = 1, stop = 6)
info <- rep("userinfo", times = 6)
m1 <- rnorm(6, 200, 30)  # 3 random measurements for each image
m2 <- rnorm(6, 5, .2)
m3 <- rnorm(6, 20, 2)
base <- data.frame(id, photo_name, info, m1, m2, m3, stringsAsFactors = FALSE)

#provide vectors of sd and wt
std.dev <- c(30, .2, 2)
weights <- c(1, 1, 1)

#bring lpic() into R environment
source(paste(getwd(), "/lpic.R", sep = ""))

#run lpic() with the example dataset
example <- lpic(base = base, pics = "images", sd = std.dev, wt = weights, preface = "test")

#quickly look at main results:
head(example[[1]], 10)
#quickly look at entire matrix with scores
head(example[[2]], 10)
#How many unique ids do you have:
nrow(unique(example[[1]][1]))
#how many images do you have of each animal based on id:
```

```
table(example[[1]][1])
```