Dichotomous Key with R

Hao Li

3/19/2021

# Installation

install.packages("devtools")  
require(devtools)  
install\_github("HaoLi111/dichotomousKey")

require(dichotomousKey)#load the package

## Loading required package: dichotomousKey

# Dichotomous Key Used for classfication

Using a dk table, one can get the name of the species by answering a series of questions. For example, to classify myself as a human.

* First, I ask: am I animal-like or plant like? I am animal-like, so I go to the next question to see what kind of animal I am:
* Do I live on land or water? My answer is : on land. go to next question to classify land animals:
* Do I have feathers, or just skins and hair? Well certainly I am not an angel, so I do not have feathers. I belong to the section “mammal”

Now considering converting the series of questions to a table below:

knitr::kable(dk\_eg)

|  |  |  |  |
| --- | --- | --- | --- |
| id | P | G | ref |
| 1 | plants like | plants | 2 |
| 1 | animal like | animal | 3 |
| 2 | woody trunk | tree | 0 |
| 2 | soft stem | flower | 0 |
| 3 | live under water | fish | 0 |
| 3 | live on land | land\_animal | 4 |
| 4 | hair | mammal | 0 |
| 4 | feathers | bird | 0 |

Note that it follows the format of:

(words in the parenthesis () are notations, not to be included)

*id(index) | P(Phenotype) | G(Genotypical reference) | ref(reference) | … …*

*Note that phenotypes and genotypes are only analogies to the entries in the dataset.*

Clearly, one can start from the father node ID 1, starting from line 1,2

dk\_eg[dk\_eg$id==1,]

## id P G ref  
## 1 1 plants like plants 2  
## 2 1 animal like animal 3

since they both have father node ID 1. I am animal, so I look at the second row, 4th column. The number there points to the next father node ID I should look for, in this case, 3.

dk\_eg[dk\_eg$id==3,]

## id P G ref  
## 5 3 live under water fish 0  
## 6 3 live on land land\_animal 4

Since I live on land, I go to “land animal”. The reference point to node 4

dk\_eg[dk\_eg$id==4,]

## id P G ref  
## 7 4 hair mammal 0  
## 8 4 feathers bird 0

When the reference column(4th column) is 0. I know that this table cannot classify further.

## dk\_classify

The process above can be automated using *dk\_classify()*. Hence,

> me = dk\_classify(dk\_eg)  
classification start from dk\_eg   
 in interactive mode   
 to abort type'q'  
Phenotypical characteristic:  
Key :: 1 ::   
plants like :: || ::plants  
Key :: 2 ::   
animal like :: || ::animal  
2  
Phenotypical characteristic:  
Key :: 1 ::   
live under water :: || ::fish  
Key :: 2 ::   
live on land :: || ::land\_animal  
2  
Phenotypical characteristic:  
Key :: 1 ::   
hair :: || ::mammal  
Key :: 2 ::   
feathers :: || ::bird  
1  
> me  
[1] "mammal"

For each step of interaction, the function prints the options in the console. Press 1 or 2 to answer the questions to continue.

## Why use dk table?

Efficient: to classify among 2^n species, it needs n questions at its best. Deterministic: given a list of phenotypes, one cannot get 2 different answers

# Functionalities

So far we know how to classify a species using a naive example. Of course you can import your table as dataframes with StringsAsFactors==F. But that is uninteresting. The text below gives a thorough instruction on the capability of the data structure.

## Table-List convertions

You can convert a table form dk dataset to its list form, & vice versa. The corresponding functions are *dk\_as\_list()* and *list\_as\_dk*

dk\_as\_list(dk\_eg)

## [[1]]  
## [[1]][[1]]  
## [[1]][[1]]$pause  
## [1] TRUE  
##   
## [[1]][[1]]$prim  
## P G  
## 3 woody trunk tree  
##   
##   
## [[1]][[2]]  
## [[1]][[2]]$pause  
## [1] TRUE  
##   
## [[1]][[2]]$prim  
## P G  
## 4 soft stem flower  
##   
##   
## [[1]]$pause  
## [1] FALSE  
##   
## [[1]]$prim  
## P G  
## 1 plants like plants  
##   
##   
## [[2]]  
## [[2]][[1]]  
## [[2]][[1]]$pause  
## [1] TRUE  
##   
## [[2]][[1]]$prim  
## P G  
## 5 live under water fish  
##   
##   
## [[2]][[2]]  
## [[2]][[2]][[1]]  
## [[2]][[2]][[1]]$pause  
## [1] TRUE  
##   
## [[2]][[2]][[1]]$prim  
## P G  
## 7 hair mammal  
##   
##   
## [[2]][[2]][[2]]  
## [[2]][[2]][[2]]$pause  
## [1] TRUE  
##   
## [[2]][[2]][[2]]$prim  
## P G  
## 8 feathers bird  
##   
##   
## [[2]][[2]]$pause  
## [1] FALSE  
##   
## [[2]][[2]]$prim  
## P G  
## 6 live on land land\_animal  
##   
##   
## [[2]]$pause  
## [1] FALSE  
##   
## [[2]]$prim  
## P G  
## 2 animal like animal  
##   
##   
## $pause  
## [1] FALSE

Using other packages, put it in a better form:

library(Hmisc)

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':  
##   
## format.pval, units

list.tree(dk\_as\_list(dk\_eg),fill = " -")

## dk\_as\_list(dk\_eg) = list 3 (11856 bytes)  
## - [[1]] = list 4  
## - - [[1]] = list 2  
## - - - pause = logical 1= TRUE  
## - - - prim = list 2( data.frame )  
## - - - - P = character 1= woody trunk   
## - - - - G = character 1= tree   
## - - -A row.names = integer 1= 3  
## - - [[2]] = list 2  
## - - - pause = logical 1= TRUE  
## - - - prim = list 2( data.frame )  
## - - - - P = character 1= soft stem   
## - - - - G = character 1= flower   
## - - -A row.names = integer 1= 4  
## - - pause = logical 1= FALSE  
## - - prim = list 2( data.frame )  
## - - - P = character 1= plants like   
## - - - G = character 1= plants   
## - -A row.names = integer 1= 1  
## - [[2]] = list 4  
## - - [[1]] = list 2  
## - - - pause = logical 1= TRUE  
## - - - prim = list 2( data.frame )  
## - - - - P = character 1= live under water   
## - - - - G = character 1= fish   
## - - -A row.names = integer 1= 5  
## - - [[2]] = list 4  
## - - - [[1]] = list 2  
## - - - - pause = logical 1= TRUE  
## - - - - prim = list 2( data.frame )  
## - - - - - P = character 1= hair   
## - - - - - G = character 1= mammal   
## - - - -A row.names = integer 1= 7  
## - - - [[2]] = list 2  
## - - - - pause = logical 1= TRUE  
## - - - - prim = list 2( data.frame )  
## - - - - - P = character 1= feathers   
## - - - - - G = character 1= bird   
## - - - -A row.names = integer 1= 8  
## - - - pause = logical 1= FALSE  
## - - - prim = list 2( data.frame )  
## - - - - P = character 1= live on land   
## - - - - G = character 1= land\_animal   
## - - -A row.names = integer 1= 6  
## - - pause = logical 1= FALSE  
## - - prim = list 2( data.frame )  
## - - - P = character 1= animal like   
## - - - G = character 1= animal   
## - -A row.names = integer 1= 2  
## - pause = logical 1= FALSE

## dk\_extract: Given a specimen, extract expected phenotypes

Suppose I know I am a mammal, but I want to know what properties I have, I can extract my expected phenotypes from a dk table.

me = "mammal"  
dk\_extract(dk\_eg,me)

## P G  
## 2 animal like animal  
## 6 live on land land\_animal  
## 7 hair mammal

## dk\_append() what if I want to combine 2 tables together?

dk\_append(dk\_eg,dk\_bird,"bird")

## id P G ref  
## 1 1 plants like plants 2  
## 2 1 animal like animal 3  
## 3 2 woody trunk tree 0  
## 4 2 soft stem flower 0  
## 5 3 live under water fish 0  
## 6 3 live on land land\_animal 4  
## 7 4 hair mammal 0  
## 8 4 feathers bird 5  
## 9 5 Big Big Bird 0  
## 10 5 Small Small Bird 0

## dk\_partition() Given a large dk set, partition a smaller dk table only classifying some subspecies or sub subclasses

In this case, want the smaller dk starting from questions on animals only. But since it is best to start with father node index==1, we use dk\_reduce to correct the id and ref data.

dk\_partition(dk\_append(dk\_eg,dk\_bird,"bird"),"animal")

## id P G ref  
## 1 3 live under water fish 0  
## 2 3 live on land land\_animal 4  
## 3 4 hair mammal 0  
## 4 4 feathers bird 5  
## 5 5 Big Big Bird 0  
## 6 5 Small Small Bird 0

dk\_reduce(dk\_partition(dk\_append(dk\_eg,dk\_bird,"bird"),"animal"))

## id P G ref  
## 1 1 live under water fish 0  
## 2 1 live on land land\_animal 2  
## 3 2 hair mammal 0  
## 4 2 feathers bird 3  
## 5 3 Big Big Bird 0  
## 6 3 Small Small Bird 0

## dk\_is\_a(): check inheritance: does one specie belong to a given class?

dk\_is\_a("Small Bird","land\_animal",dk\_partition(dk\_append(dk\_eg,dk\_bird,"bird"),"animal"))

## [1] TRUE

dk\_is\_a("bird","animal",dk\_eg)

## [1] TRUE

dk\_is\_a("fish","land\_animal",dk\_eg)

## [1] FALSE

# Application on larger datasets?

Suppose I have a list of observations:

fieldwork\_findings = data.frame(n = 1:10,sps=c("bird","bird","Big Bird","Small Bird","fish","mammal","fish","bird","Small Bird","Bird"))

Let’s do bird count

is\_bird = sapply(fieldwork\_findings$sps, dk\_is\_a,b = "bird",dk = dk\_append(dk\_eg,dk\_bird,"bird"))  
is\_bird

## bird bird Big Bird Small Bird fish mammal fish   
## TRUE TRUE TRUE TRUE FALSE FALSE FALSE   
## bird Small Bird Bird   
## TRUE TRUE FALSE

length(is\_bird[is\_bird==TRUE])

## [1] 6

More importantly one might want to calculate diversity up to a certain level, for example, how many *species* does the dataset contain can differ from how many *subspecies* the dataset contains. However this is only sensible for large amount of data.

# Prospective: Abstraction by row index: for datasets more than 4 columns

What if one wants to add links as a 5th column of the table? or maybe pictures? A sensible way to do such manipulations is to specify the output row IDs.