TR8: Extract traits data for plant species

Gionata Bocci
Pisa (ITALY)
boccigionata@gmail.com

October 14, 2014

1 Rationale

The `TR8` package has been built in order to provide the user with the possibility of easily retrieving traits data for plant species from the following publicly available databases:

Biolflor `http://www2.ufz.de/biolflor/index.jsp` [5]

Ecological Flora of the British Isles `http://www.ecoflora.co.uk/` [3]

LEDA traitbase `http://www.leda-traitbase.org/LEDAportal/` [4]

Ellenberg values for Italian Flora [6]

Mycorrhizal intensity database [2]

Please note that not all the traits available on the listed databases are downloaded by the package: this may change in future versions of the package (ie. some functionalities may be added and more traits will be made available).

2 Installation

The `TR8` package is available on CRAN, thus it can be easily installed through:

```
> install.packages("TR8",dependencies = TRUE)
```

The option `dependencies = TRUE` takes care of installing the following packages (if they are not already installed) which are needed by `TR8` to work properly:

- plyr[12]
- reshape[11]
- RCurl[8]

- XML[9]
- taxize[1]
- gWidgets[10]
- gWidgetstcltk
- rappdirs

Once the package is installed, you can load it with:

```
> library(TR8)
```

Please note that:

The user is asked to always cite the data sources: the development of traits databases is a long and costly process, thus all the users of the `TR8` package are asked (and reminded **every time** they load the package) to always cite the original sources of the data (see paragraph 5).

The devel version of the package is hosted on github at the following address `https://github.com/GioBo/TR8`: to use this version (insted of the stable one, released from CRAN), you'll need the `devtools` package (`https://github.com/hadley/devtools`):

```
> ## install the package
> install.packages("devtools")
> ## load it
> library(devtools)
> ## activate dev_mode
> dev_mode(on=T)
> ## install TR8
> install_github("GioBo/TR8")
> ## you can now work with TR8 functions
>
> ## if you want to go back and use the CRAN version
> ## already installed, simply deactivate dev_mode
> dev_mode(on=F)
```

3 Simple usage

Using the `TR8` package is fairly simple: users just need to call the `tr8` function passing, as arguments, a vector of plant species names and a vector containing the codes corresponding to the traits which are to be downloaded:

```
> ## a vector containing a list of plant species names
> my_species<-c("Apium graveolens","Holcus mollis","Lathyrus sylvestris")
> ## a vector of traits
> to_be_downloaded<-c("reprod_B","strategy")
> ## now run tr8 and store the results in the my_traits object
> my_traits<-tr8(species_list = my_species,download_list = to_be_downloaded)
```

The codes to be used are contained in the `available_tr8` database:

```
> ## see the firs lines of available_traits database
> head(available_tr8)
  short_code            description       db
1      h_max         Maximum height Ecoflora
2      h_min         Minimum height Ecoflora
3    le_area              Leaf area Ecoflora
4    le_long         Leaf longevity Ecoflora
5  phot_path Photosynthetic pathway Ecoflora
6    li_form              Life form Ecoflora
```

The database is composed of three columns:

short_code contains the codes corresponding to the traits which can be downloaded through the `tr8` function. These are the codes that should be passed to the `download_list` argument of the `tr8` function.

description contains short description of each trait (please refer to the original sources for detalied descriptions).

db referso to the databases containing the traits

Suppose the user is interested in dowloading the *maximum height*, the *leaf area* and the *life form* (which are available through the *Ecolfora* database) for *Salix alba* and *Populus nigra* and store the resulting data in the `my_Data` object; the command should be:

```
> my_species<-c("Salix alba","Populus nigra")
> my_traits<-c("h_max","le_area","li_form")
> my_Data<-tr8(species_list = my_species, download_list = my_traits)
```

The `tr8` function will take care of downloading the data and store them in the `my_Data` object; you can see the results by printing them:

```
> ## see the downloaded data
> print(my_Data)
```

Or you can convert them to a data frame using the `extract_traits` function:

```
> traits_dataframe<-extract_traits(my_Data)
```

All the traits are now contained in a data frame with species as rows and columns as traits; where no trait data were available, you will see a `NA`.

In order to make the dataframe more readable, traits' names (ie. columns' names) are converted to shorter codes: to see a brief explanation of the codes used to identify the traits, use the `lookup` function:

```
> lookup(my_Data)
```

The object returned by the `lookup` function can also be stored in order to be available for further elaborations:

```
> my_lookup<-lookup(my_Data)
> head(my_lookup)
```

Up to now we've been using the `tr8` function in a non-interactive way. In order to help those user which are more familiar with a GUI approach, the function can also be run setting the `gui_config` parameter to `TRUE` (without providing any trait in the `download_list` parameter) and a multi-panel window will appear: the user is asked to choose those traits which are to be downloaded from the various databases.

For a detailed explanation of each level of a trait, please refer to the original websites (all the databases listed in the references provide the users with very precise and detailed descriptions).

Tipically users will have a their vegetation data in the form of a *sites*species* dataframe (or matrix), thus they may want to extract traits data for the whole dataset (this time using the GUI to select traits), ie. :

```
> ## suppose veg_data is our dataframe with
> ## plant species as columns and sites as rows
>
> ## extract species names
> specie_names<-names(veg_data)
> ## use the tr8() function
> ## and tick those traits of interest in the pop-up window
> my_traits<-tr8(species_names,gui_config=TRUE)
> ## print the results
> print(my_traits)
```

A NOTE OF CAUTION: searching the web is a time (and internet band) consuming activity, thus the higher the number of your plant species and the traits to be retrieved, the longer it will take to `tr8()` to complete its job. Moreover, in order not to overflow the remote databases with `http` requests, the `tr8` function will alway pause between one search and the following one.

A (SECOND) NOTE OF CAUTION: some users adopt the following workflow for analysing their vegetation data:

1. insert vegetation data into a *spreadsheet file* with species as columns' and sites' as rows

2. export the spreadsheet file as a `.csv` file

3. import the `.csv` file into a **R** dataframe.

When following these steps, a dot (".") will be inserted between Genus and Species of each plant species name (i.e. column names in the `R` dataframe will not be in the form `c("Abies alba", "Salix alba")` but in the form `c("Abies.alba", "Salix.alba")`). This may cause problems for further processing of plants' species names, thus, in order to avoid this problem, please use the `check.names=F` option in `read.csv`. Eg. suppose that `my_veg_data.csv` is the `csv` file: in the **R** console, one should use:

```
> My_data<-read.csv("my_veg_data.csv",
+                   header=T,row.names=1,check.names=F)
```

4 Interpreting retrieved data

Please note that for many traits there is more than one entry in the original databases: in those cases, in order to obtain a single value the following strategy was adopted:

Quantitative traits the mean of all the values was calculated (eg. when multiple values for "Seed weight mean" are available, the mean of these value is calculated)

Qualitative traits all the values are taken into account and "joined" together in a single string (the values are separated by a score "–")

Nota bene: in some cases some traits are stored as *string* in the original databases, even though they should be treated as numbers (e.g. the number *five* is stored as a string - i.e. "5", not as the numeric value `5`): in those case `tr8` function is not able to interpret that entry as a numeric, thus, applying the above mentioned criteria to merge multiple traits, strange outputs may result from `tr8`, e.g. if a species has two entries for the trait `height` - day 3 and 3.5 meters - the merged value will not be the numeric mean (3.25) but the union of the two strings ("3-3.5").

5 Citing sources of information

Users of the `TR8` package should always cite the sources of information which provided the traits data: the correct citations to be used for the retrieved data can be obtained through the `bib` method; just use:

```
> bib(my_traits)
```

6 Suggested usage

We strongly suggest to always check plant species names with the `tnrs` function (from the `taxize` package) before using the `tr8` function; thus a typical workflow would be the following:

1. Check plant species names (eg. with something like the following - please refere to the `taxize` package documentation[1] for further details)

   ```
   > species_names<-names(veg_data)
   > checked_names<-tnrs(species_names,source="iPlant_TNRS")
   > print(checked_names)
   ```

   Check which species (rows) in the table have a "score" value lower than 1 and check their names; if needed, correct them before using the tr8() function.

2. Run `tr8` (in this case using the GUI):

   ```
   > my_traits<-tr8(species_names)
   > print(my_traits)
   ```

3. You may want to have these traits available as a data frame: just use the `extract_traits` function which uses the results of `tr8` (in this case it's the `my_traits` objects) and returns a data frame.

   ```
   > traits_df<-extract_traits(my_traits)
   ```

4. Observing a big data frame inside `R` could be difficult, thus users may want to save the `traits_df` data frame as a `.csv` file and open that with a spreadsheet software (eg. LibreOffice)

   ```
   >
   ```

7 Local storage of LEDA and Akhmetzhanova data

The LEDA Traitbase datafiles and Akhmetzhanova database are text files (either `.txt` or `.csv` files) which are available for download at the LEDA (`http://www.leda-traitbase.org/LEDAportal/data_files.jsp`) and at the *Ecological Archives* websites . These files are (quite) big in size, thus downloading them every time the `tr8()` function is used is a time consuming activity[1]. We thus suggest the users to run the `local_storage`[2] function once to store a local copy of these datafiles datafiles and use that local copy every time the `tr8()` function is run and LEDA or Akhmetzhanova are requested[3].

```
> ## run the function
> local_storage()
```

8 A 'real life' workflow

In this paragraph I will describe a typical workflow for a researcher interested in using the `TR8` package: this is meant to be a step-by-step guide involving most of the common problems that are faced in importing data, checking them and running `tr8`[4]. This section relies on the data set used by Sandau et al.(2014)[7] which is publicly available at `dryad`[7].

8.1 Retrieve original data

The dataset is available as a `xlsx` file (which is a common case); several alternatives are available to import this kind of files into `R` (eg. you can download the dataset and load it into an `R` session or you could save a `.csv` version of the file and then load it into `R` using `read.csv()`); in this case we will use `XLConnect` to download and load the dataset.

```
> ## the XLConnect package is needed
> install.packages("XLConnect",dependencies = T)
> library(XLConnect)
> ## store the  url of the dryad package
> url<-"http://datadryad.org/bitstream/handle//
+     10255/dryad.65646/MEE-13-11-651R2_data.xlsx?sequence=1"
> ## choose the extension for the temp file where
> ## data will be stored
> tmp = tempfile(fileext = ".xlsx")
> ## download the data
> download.file(url = url, destfile = tmp)
> ## we first read the "metadata" sheet from the xlsx file
```

[1]The text files are not distributed together with the `TR8` package - which would save time and memory when executing the `tr8()` function - in order to avoid possible licensing conflicts between the `TR8`' GPL license and these datasets.

[2]The name is quite self−explanatory...

[3]By default these files will be installed in the directories which are commonly used for storing applications' data (which depends on the underlying operating systems; see `https://github.com/hadley/rappdirs` for details).

[4]The process described here is rather lenghty in order to describe each single step in detail; users which are confident in the use of R could make most of the steps much shorter than what's presented here.

```
> metadata<-readWorksheetFromFile(file = tmp, sheet = "metadata",
+  header = FALSE, startRow = 15, startCol = 1, endCol = 3)
> ## then read the vegetation data
> veg_data <-readWorksheetFromFile(file = tmp, sheet = "data.txt",
+       header = TRUE, startRow = 1, startCol = 11, endCol = 123)
> ## round veg_data numbers to the second digit
> veg_data<-round(veg_data,digits = 2)
```

The dataframe `metadata` contains two columns, `Col1` contains short codes used by the authors as surrogates of the full scientific names of species for the species which are stored in the `Col2` column.

8.2 Check species names

The first suggested step is to check species names using the `taxize` package in order to see whether there are misspelled names; the `tnrs` function accepts a vector of plant species names and tries to match them with *accepted* scientific names; the function returns a dataframe with various columns: in the column `score` each entry is given a score according to the level of ”resemblance” with correct names; the score is ”1” if the name is correct, less than ”1” if some problems with the name are found. **NOTA BENE**: from the `tnrs` help page ”If there is no match in the Taxosaurus database, nothing is returned, so you will not get anything back for non matches” thus we should worry of both ”less than 1” scores **AND** missing entries in the dataframe returned by `tnrs`.

```
> library(taxize)
> check_names<-tnrs(metadata$Col2,source="iPlant_TNRS")
```

Check now if there are species which were discarded by `tnrs` output since they were not found in reference databases.

```
> setdiff(metadata$Col2,check_names$submittedname)
```

The results is 0, thus `tnrs` found at least a partial match for all the species names we provided. Next we should check which species got a `score` which is less than 1.

```
> issues<-with(check_names,check_names[score!="1",])
> issues[,c("score","submittedname")]
```

```
   score submittedname
34 0.9 Poaceae (undetermined)
36 0.9 Epilobium sp.
44 0.9 Cerastium sp.
53 0.96 Fallopia convolvulus (L.) A. Löwe
54 0.9 Festuca sp.
55 0.9 Chenopodium sp.
63 0.9 Phleum pratense agg.
```

```
69 0.9 Polygonum sp.
72 0.9 Polygonum mite (=Persicaria laxiflora)
76 0.9 Rubus sp.
79 0.9 Juncus sp.
83 0.9 Orobanche sp.
97 0.9 Triticum sp.
100 0.9 Taraxacum officinale!!!!!
107 0.96 Setaria pumila (Poir.)  Schult.
```

We see here that for some entries, only the Genus is present, thus `tr8` function will not be able to return traits values for those species; entry 83 is `Taraxacum officinale!!!!!`: in this case we should remove the `"!"` from the species names.

```
> ## which entry in the metadata dataframe
> ## correspond to "Taraxacum officinale!!!!!"?
> mistake<-which(metadata$Col2=="Taraxacum officinale!!!!!")
> ## now correct the entry (both the
> ## name and the score)
> metadata[mistake,]<-c("1","Taraxacum officinale F.H. Wigg.")
> ##
```

My suggestion is to remove those entries for which only the Genus is provided (for which the score is "0.9"):

```
> ## we go back using the check_names data frame
> ## which contains both correct and problematic species
> ## and remove problematic ones from them
>
> #convert score from string to numeric
> check_names$score<-as.numeric(check_names$score)
> accepted_species<-check_names[check_names$score>0.9,
+                         c("submittedname","acceptedname")]
> ## and now merge it with the metadata dataset
> final_df<-merge(metadata,accepted_species,
+                 by.x="Col2",by.y="submittedname")
>
```

In this way we now have the `temp_df` data frame in which each entry has the name present in the original dataset and the correct scientific names, which should be passed to the `tr8` function.

8.3 Using the `tr8` function

References

[1] Chamberlain, S. and Szocs, E. taxize - taxonomic search and retrieval in r. *F1000Research*, 2013.

[2] A.A. Akhmetzhanova, N.~A. Soudzilovskaia, V.~G. Onipchenko, W.K. Cornwell, V.A. Agafonov, I.A. Selivanov, and J.~H.C. Cornelissen. A rediscovered treasure: mycorrhizal intensity database for 3000 vascular plant species across the former Soviet Union: Ecological archives e093-059. *Ecology*, 93(3):689–690, 2012.

[3] A.~H. Fitter and H.~J. Peat. The Ecological Flora Database. *J. Ecol*, 82:415–425, 1994.

[4] M.~Kleyer, R.~M. Bekker, I.~C. Knevel, J.~P. Bakker, K.~Thompson, M.~Sonnenschein, P.~Poschlod, J.~M. van Groenendael, L.~Klimes, J.~Klimesova, S.~Klotz, G.~M. Rusch, M.~Hermy, D.~Adriaens, G.~Boedeltje, B.~Bossuyt, A.~Dannemann, P.~Endels, L.~Götzenberger, J.~G. Hodgson, A-K. Jackel, I.~Kühn, D.~Kunzmann, W.~A. Ozinga, C.~Römermann, M.~Stadler, J.~Schlegelmilch, H.~J. Steendam, O.~Tackenberg, B.~Wilmann, J.~H.~C. Cornelissen, O.~Eriksson, E.~Garnier, and B.~Peco. The LEDA Traitbase: a database of life-history traits of the Northwest European flora. *JOURNAL OF ECOLOGY*, 96(6):1266–1274, 2008.

[5] S.~Klotz, I.~Kühn, and W.~Durka. BIOLFLOR - Eine Datenbank zu biologisch-ökologischen Merkmalen zur Flora von Deutschland. Schriftenreihe für Vegetationskunde 38: 1-333. (Bundesamt für. Bonn, Bundesamt für Naturschutz). *Schriftenreihe für Vegetationskunde*, 38:1–333, 2002.

[6] S.~Pignatti, P.~Menegoni, and S.~Pietrosanti. Biondicazione attraverso le piante vascolari. Valori di indicazione secondo Ellenberg (Zeigerwerte) per le specie della Flora d'Italia. *Braun-Blanquetia*, 39:97, 2005.

[7] N~Sandau, RP~Rohr, RE~Naisbit, Y~Fabian, OT~Bruggisser, P~Kehrli, A~Aebi, and L~Bersier. Data from: Including community composition in biodiversity-productivity models, 2014.

[8] D.~Temple~Lang. *RCurl: General network (HTTP/FTP/...) client interface for R*, 2013. R package version 1.95-4.1.

[9] D.~Temple~Lang. *XML: Tools for parsing and generating XML within R and S-Plus.*, 2013. R package version 3.98-1.1.

[10] J.. Based on the iwidgets code of Simon~Urbanek Verzani, suggestions~by Simon~Urbanek, Philippe Grosjean, and Michael Lawrence. *gWidgets: gWidgets API for building toolkit-independent, interactive GUIs*, 2012. R package version 0.0-52.

[11] H.~Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12), 2007.

[12] H.~Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1):1–29, 2011.