vignette of package 'traitdataform'

Florian D. Schneider

25 Juli 2017

Contents

This vignette contains step-by step instructions for transferring own data into a standardized trait-dataset for upload to public databases. The output object uses the trait data standard put forward in the Whitepaper (refer to pub).

step 1 reading in data

First, load your own data into R, preferrably in a species–trait matrix, occurence table or measurement longtable format (See notes on different data formats of trait data).

You may rename the column names of the original file to match the column names described in the trait data standard, but this howto also describes the mapping of the column names along the process of data handling.

R knows many ways of getting your original data into an R object. In most cases you would read an object from a csv or txt file while maintaining the column headers.

```r
library(traitdataform)
```

```r
carabids <- read.table("../../data/carabid traits final.txt", header = TRUE)
```

step 2 transfer into measurement longtable format

As explained in the whitepaper, most trait data are stored in a species–trait matrix format or a table with replicated trait measurements on multiple individuals (i.e. occurences of a species).

- **species-trait matrix data** : a single account of a trait value for each species (in rows) for a couple of different traits (in columns). No replicates of species are reported. This is the most likely format for literature data, where aggregate measurements or facts for entire species have been collated into a single lookup table.
- **occurence table** : In case of measured data, authors may report multiple raw measurements of different traits (in columns) taken from a single occurence of a species, i.e. an individual specimen (in rows). Repeated measures of the same trait might also be included as columns or pooled into average values. This is valuable for investigations of intra-specific variation, and also leaves space for filtering by cofactors or analysing trait response along environmental gradients.

- **measurement longtable** : For a standardisation of trait data for use in online databases, we propose a measurement longtable format, where each row comprises the reporting of a single measurement or fact, linked to a trait definition as well as a valid taxon name, and optionally to other layers of information. This data format is more predictable in terms of columns and thus easiert to merge with other datasets.

In all cases, additional information on the reported value may be stored in further colums (e.g. the unit in which a value is reported or the literature source for this measurement or fact), or in a separate data sheet linked via identifiers for trait, taxon, occurence or sampling/measurement event. Examples below will explain how these information can be added to the data sheet.

The function `as.traitdata()` provided in the package assist in transferring any data format into the measurement longtable format. For this function to work, it needs to know about the columns of the original data that contain trait values (parameter `traits`), and the column which contains the taxonomic specification (parameter `taxa`).

```
dataset1 <- as.traitdata(carabids,
                         taxa = "name_correct",
                         traits = c("body_length",
                                    "antenna_length",
                                    "metafemur_length",
                                    "eyewidth_corr")
                         )
```

```
## it seems you are providing data in a species -- trait matrix. If this is not the case, please ch
```

```
head(dataset1)
```

```
##          scientificName   traitName traitValue
## 1 Abax_parallelepipedus body_length  15.846561
## 2  Acupalpus_meridianus body_length   2.670000
## 3        Agonum_ericeti body_length   5.873016
## 4    Agonum_fuliginosum body_length   5.090000
## 5        Agonum_gracile body_length   4.880000
## 6     Agonum_marginatum body_length   8.250000
```

Note that in the output table the columns have been named after the traitdata standard proposed in the whitepaper (ref). The essential columns are `traitName`, `traitValue` for the reported measurement or fact as well as `ScientificName` for the taxon assignment. The function auomaticall interprets data as species- trait matrix if the taxa column contains only unique entries and no duplicates.

In case of occurence table data, an occurenceID is provided automatically, or can be provided by the author using the parameter `occurences` (as a column name or a vector of occurence IDs).

```
dataset2 <- as.traitdata(heteropteraRaw,
              traits = c("Body_length", "Body_width", "Body_height", "Thorax_length",
                         "Thorax_width", "Head_width", "Eye_width", "Antenna_Seg1",
                         "Antenna_Seg2", "Antenna_Seg3", "Antenna_Seg4", "Antenna_Seg5",
                         "Front.Tibia_length", "Mid.Tibia_length", "Hind.Tibia_length",
                         "Front.Femur_length", "Hind.Femur_length", "Front.Femur_width",
                         "Hind.Femur_width", "Rostrum_length", "Rostrum_width",
                         "Wing_length", "Wing_width"),
              taxa = "SpeciesID",
              occurences = "ID"
              )

# show different trait measurements for same occurence/individual
subset(dataset2, occurenceID == "5" )
```

```
##        scientificName          traitName traitValue occurenceID
## 5    Acalypta parvula        Body_length       1.84           5
## 430  Acalypta parvula         Body_width       0.82           5
## 855  Acalypta parvula        Body_height       0.56           5
## 1280 Acalypta parvula      Thorax_length       0.17           5
## 1705 Acalypta parvula       Thorax_width       0.84           5
## 2130 Acalypta parvula         Head_width       0.36           5
## 2555 Acalypta parvula          Eye_width       0.10           5
## 2980 Acalypta parvula       Antenna_Seg1       0.07           5
## 3405 Acalypta parvula       Antenna_Seg2       0.06           5
## 3830 Acalypta parvula       Antenna_Seg3       0.40           5
## 4255 Acalypta parvula       Antenna_Seg4       0.15           5
## 4716 Acalypta parvula Front.Tibia_length       0.39           5
## 5141 Acalypta parvula   Mid.Tibia_length       0.40           5
## 5566 Acalypta parvula  Hind.Tibia_length       0.50           5
## 5991 Acalypta parvula Front.Femur_length       0.42           5
## 6416 Acalypta parvula  Hind.Femur_length       0.43           5
## 6841 Acalypta parvula  Front.Femur_width       0.08           5
## 7266 Acalypta parvula   Hind.Femur_width       0.07           5
## 7691 Acalypta parvula     Rostrum_length       0.80           5
## 8116 Acalypta parvula      Rostrum_width       0.08           5
## 8541 Acalypta parvula        Wing_length       1.78           5
## 8966 Acalypta parvula         Wing_width       0.65           5
```

case example: provide measurement unit

For a standardisation of quantitative trait data, the unit of measurement is essential. Often, this information is kept in the metadata descriptions. But for a standardised table containing measurements from different sources, this information should always accompany the measurement value. A common way to provide the unit is adding another column to your original data table containing the unit in an unambiguous format. The function `as.traittable()` assists in adding the units via its parameter `units`.

This can be done for all traits in a single stroke (if all reported values refer to the same unit) or to each trait specifically (if they used different measuremnt units or if the table comprises a mixture of quantitative and qualitative traits).

The syntax for this uses the parameter `units`, which takes a single character string, or a vector of character strings, containing valid entries as expected by the package 'units' (Pebesma et al. 2016, https://github.com/edzer/units/, v0.4-5, Examples are 'mm', 'm2' or 'm^2', 'm/s').

```
dataset1 <- as.traitdata(carabids,
                         taxa = "name_correct",
                         traits = c("body_length",
                                    "antenna_length",
                                   "metafemur_length",
                                   "eyewidth_corr"),
                         units = "mm"
                         )
```

```
## it seems you are providing data in a species -- trait matrix. If this is not the case, please ch
```

```
head(dataset1)
```

```
##          scientificName   traitName traitValue traitUnit
## 1 Abax_parallelepipedus body_length  15.846561        mm
## 2  Acupalpus_meridianus body_length   2.670000        mm
## 3        Agonum_ericeti body_length   5.873016        mm
## 4    Agonum_fuliginosum body_length   5.090000        mm
## 5        Agonum_gracile body_length   4.880000        mm
## 6     Agonum_marginatum body_length   8.250000        mm
```

A character vector should have the same length as the provided vector of trait names (in parameter `traits`), or otherwise should be a named vector of the form `c(trait1 = "mm", trait2 = "mm2")`, where only the traits provided will receive units while the others will remain blank.

```
dataset3 <- as.traitdata(arthropodtraits,
                         taxa = "SpeciesID",
                         traits = c("Body_Size", "Dispersal_ability", "Feeding_guild",
                                    "Feeding_mode", "Feeding_specialization",
                                    "Feeding_tissue", "Feeding_plant_part",
                                    "Endophagous_lifestyle", "Stratum_use",
                                    "Stratum_use_short"),
                         units = c(Body_Size = "mm", Dispersal_ability = "unitless")
)
```

```
## it seems you are providing data in a species -- trait matrix. If this is not the case, please ch
```

```
head(dataset3)
```

```
##         scientificName traitName traitValue traitUnit
## 1 Anyphaena accentuata Body_Size       6.25        mm
## 2 Aculepeira ceropegia Body_Size         11        mm
## 3     Agalenatea redii Body_Size       6.93        mm
## 4   Araneus diadematus Body_Size      11.88        mm
## 5    Araneus marmoreus Body_Size      10.03        mm
## 6    Araneus quadratus Body_Size      12.25        mm
```

```
dataset4 <- as.traitdata(heteroptera,
              traits = c("Body_length", "Body_volume", "Rel_wing_length",
                         "Hind.Femur_shape", "Rel_Hind.Femur_length",
                         "Rel_Rostrum_length", "Front.Femur_shape",
                         "Body_shape", "Rel_eye_size", "Rel_Antenna_length"
                         ),
              taxa = "SpeciesID",
              units = c("mm", "mm3", rep("unitless", 8))
              )
```

```
## it seems you are providing data in a species -- trait matrix. If this is not the case, please ch
```

```
head(dataset4)
```

```
##                  scientificName   traitName traitValue traitUnit
## 1              Acalypta nigrina Body_length       2.19        mm
## 2              Acalypta parvula Body_length       1.73        mm
## 3          Acalypta platycheila Body_length       2.22        mm
## 4            Acetropis carinata Body_length       5.73        mm
## 5       Adelphocoris lineolatus Body_length       6.84        mm
## 6 Adelphocoris quadripunctatus Body_length       7.95        mm
```

Logical or factorial traits ususally don't come with a unit. In mixed data, the field should specify as empty, `""` or as`NA`.

case example: mutate original columns into derived values

Many traits comprise compound measures of multiple traits, such as length-mass ratios or morphometric indices. Other traits must be refined in terms of factor levels, or reduced to binary trait values. While many of these tasks can be achieved on the raw data using base functions like `transform()`, `factor()` or `match()` or the `mutate()` function provided by the package 'plyr'.

The function `mutate.traitdata()` performs these tasks (working as a wrapper to `plyr::mutate()`).

```
updated <- mutate.traitdata(dataset2,
                            Body_shape = Body_length/Body_width,
                            Body_volume = Body_length*Body_width*Body_height,
                            Wingload = Wing_length*Wing_width/Body_volume)

head(subset(updated, occurenceID == 23))
```

```
##                scientificName     traitName traitValue occurenceID
## 23   Adelphocoris seticornis   Body_length       5.95          23
## 448  Adelphocoris seticornis    Body_width       1.95          23
## 873  Adelphocoris seticornis   Body_height       1.87          23
## 1298 Adelphocoris seticornis Thorax_length       2.33          23
## 1723 Adelphocoris seticornis  Thorax_width       2.38          23
## 2148 Adelphocoris seticornis    Head_width       1.23          23
```

Note that all existing traits remain untouched and additional trait measures will be added to the dataset, unless a definition replaces an already existing trait (such as 'Stratum_use' in this example).

It is important to note that the mutate function works at the level of data resolution that is provided by the data, i.e. for occurence data with multiple measurements on a single individual, the data columns are mutated per occurenceID.

case example: raw data are coded as numeric factor levels

The data table should be human readable, thus you may consider translation into true factorial data via the function `mutate.traitdata()`.

This may not be useful if the numeric levels correspond to fine grained distinctions that cannot be translated into short factor levels.

A translation into factorials is even ill-adviced if factor levels are ordinal, i.e. they correspond to a sequence of logically ordered levels and the ordering would be lost by translating into factorials: The traitdata object will not keep ordinal level definitions of the original R data.frame. In this case, integer numerical values are best to describe the relational structure of the factor levels.

Please don't forget to provide a definition of factor levels in the metadata description of variables or in an accompanying dataset containing trait definitions.

```
updated <- mutate.traitdata(dataset3,
                            predator = Feeding_guild == "c",
                            Feeding_specialization = factor(Feeding_specialization,
                                            levels = c("m","o", "p"),
                                            labels = c("monophagous", "oligophagous",
                                                       "polyphagous")
                                            ),
                            Stratum_use = factor(Stratum_use_short,
                                                 levels = c("s","g", "h", "t"),
                                                 ordered = TRUE,
                                                 labels = c("soil","ground","herb","shrub")
                                                 )
                            )

head(subset(updated, traitName == "Stratum_use"))
```

case example: keep additional data columns

The raw data might contain further information on the specimen or the trait measurement itself in further data columns that are valuable for later analysis. This can be for instance data about the sex or developmental stage of the individual, the sampling or preservation method of the specimen, or the conditions under which the measurement was taken.

The parameter `keep` allows you to specify which columns contain valuable information as a character vector. As a negative version of `keep`, specifying `drop` would allow you to name the columns that are not valueable, while all others will be kept. Not specifying `keep` or `drop` will result in dropping all columns except the core measurement and identifier columns.

```
dataset2 <- as.traitdata(heteropteraRaw,
              traits = c("Body_length", "Body_width", "Body_height", "Thorax_length",
                         "Thorax_width", "Head_width", "Eye_width", "Antenna_Seg1",
                         "Antenna_Seg2", "Antenna_Seg3", "Antenna_Seg4", "Antenna_Seg5",
                         "Front.Tibia_length", "Mid.Tibia_length", "Hind.Tibia_length",
                         "Front.Femur_length", "Hind.Femur_length", "Front.Femur_width",
                         "Hind.Femur_width", "Rostrum_length", "Rostrum_width",
                         "Wing_length", "Wing_width"),
              taxa = "SpeciesID",
              occurences = "ID",
              keep = c("Sex")
              )

head(dataset2)
```

```
##     scientificName   traitName traitValue occurenceID Sex
## 1 Acalypta nigrina Body_length       2.35           1   f
## 2 Acalypta nigrina Body_length       2.10           2   f
## 3 Acalypta nigrina Body_length       2.17           3   m
## 4 Acalypta nigrina Body_length       2.15           4   m
## 5 Acalypta parvula Body_length       1.84           5   f
## 6 Acalypta parvula Body_length       1.81           6   f
```

The traitdata standard (whitepaper) suggests standard names for many of these extra information, which might fall into the domain of the extensions for occurence or measurementOrFact (see below). We highly reccomend mapping the columns provided into these standard names by using the rename feature of the `as.traitdata()` function. This is simply acheived by providing a named vector for `keep` that uses the compatible column names as vector names.

```
dataset1 <- as.traitdata(carabids,
                         taxa = "name_correct",
                         traits = c("body_length", "antenna_length",
                                    "metafemur_length", "eyewidth_corr"),
                         units = "mm",
                         keep = c(measurementDeterminedBy = "source_measurement")
                         )
```

```
## it seems you are providing data in a species -- trait matrix. If this is not the case, please ch
```

```
head(dataset1)
```

```
##          scientificName   traitName traitValue traitUnit
## 1 Abax_parallelepipedus body_length  15.846561        mm
## 2  Acupalpus_meridianus body_length   2.670000        mm
## 3        Agonum_ericeti body_length   5.873016        mm
## 4    Agonum_fuliginosum body_length   5.090000        mm
## 5        Agonum_gracile body_length   4.880000        mm
## 6     Agonum_marginatum body_length   8.250000        mm
##   measurementDeterminedBy
## 1                   klink
## 2                WOODCOCK
## 3                   klink
## 4                  ribera
## 5                  ribera
## 6                  ribera
```

Note that a lack of a name in the named vector maintains the original name. Note also, that no checking for valid column names (as compared to the traitdata glossary) is performed at this stage. This is to ensure that the raw data table created by `as.traittable()` can contain any columns that the author considers relevant. The `keep` parameter can be used to rename columns into intuitive column names.

case example: adding further information on traits, species or single measurements

Beyond measurement units, further information might be available that are not recorded in the raw data table, but are related to the trait type, the taxon, the individual or specimen, or to the reported fact, measurement or sampling event.

In most cases those information are kept in seperate data sheets of your file, e.g. the place were a specimen has been sampled or the literature source from where a species value has been cited. In this case, a unique identifier might link to this other datasheet, such as a number for each individual occurence of a specimen (`occurenceID`) or a identifier for a single measurement or reported fact (`measurementID`).

The trait data standard described in the whitepaper provides two extensions of the namespace that should be used to describe these data:

- the `occurence` extension contains information on the level of individual specimens, such as date and location and method of sampling and preservation, or physiological specifications of the phenotype, such as sex, life stage or age.
- the `measurementOrFact` extension takes information at the level of single measurements or reported values, such as the original literature from where the value is cited, the method of measurement or statistical method of aggregation.

The extensions are compatible with Darwin Core Standard and EOL TraitBank.

You may decide to keep the information in a seperate data sheet. In that case, the traitdata table should at least contain a column with the respective identifier that directs to the covariate datasheet. The identifier might also take the format of a globally valid URI or API call.

It is however recommended to add these information directly as own columns within the data table to enable an analysis of cofactors and correlations further down the road. This way, if datasets of different source are merged, the information is readily available without the risk of breaking the reference to an external datasheet.

The function `as.traittable()` provides a set of parameters to add information at the different levels. The following three examples will illustrate how to add covariates to each occurence or measurement. The principle is always the same: A unique identifier for these levels of information can be associated with a vector or data table containing the additional information, which will then be merged into the data table.

Adding information on specimen level (occurence Extension)

- under construction -

Adding information on measurement or fact (measurementOrFact Extension)

- under construction -

step 3 standardise taxon names and trait values

Step 1 and 2 produced a tidy and correctly formatted version of your own trait data. We now turn to the challenging task of standardisation. Two aspects of trait data need thorough standardisation: the names of species and higher taxa need to be mapped to globally accepted definitions and the names of traits should be referenced to unambiguous definitions and, where possible, translated to standard units and accepted factor levels.

taxon name standardisation

For taxon name standardisation, the function `standardize.taxonomy()` makes use of fuzzy matching algorithms provided by the package 'taxize' to match the entries of column `scientificName` against the GBIF Backbone Taxonomy (taxize v). The result is written into a new column `scientificNameStd`. Additional columns comprise the order (for ambiguous names), the reported taxon rank, as well as a globally unique taxon ID which references the taxon to GBIF Backbone Taxonomy in a universal URI format.

If further layers of taxonomic information are desired as an output, the function takes the parameter `return`, which by default contains `c("taxonID", "scientificNameStd", "order", "taxonRank")`. Other specifications can be added here.

Note that for this to work, `scientificName` must contain a full account of the species name or higher taxon, no abbreviations (spaces or underscores are handled alright).

Note also, that taxon name mapping requires an internet connection and might take some time, depending on the length of your species list.

```r
dataset1std <- standardize.taxonomy(dataset1)
head(dataset1std)
```

```
##          scientificName                              taxonID taxonRank
## 1 Abax_parallelepipedus http://www.gbif.org/species/5754772   species
## 2 Abax_parallelepipedus http://www.gbif.org/species/5754772   species
## 3 Abax_parallelepipedus http://www.gbif.org/species/5754772   species
## 4 Abax_parallelepipedus http://www.gbif.org/species/5754772   species
## 5  Acupalpus_meridianus http://www.gbif.org/species/1037633   species
## 6  Acupalpus_meridianus http://www.gbif.org/species/1037633   species
##          traitName traitValue traitUnit     scientificNameStd warnings
## 1   antenna_length   8.518519        mm Abax parallelepipedus
## 2      body_length  15.846561        mm Abax parallelepipedus
## 3 metafemur_length   5.608466        mm Abax parallelepipedus
## 4    eyewidth_corr   0.481250        mm Abax parallelepipedus
## 5      body_length   2.670000        mm  Acupalpus meridianus
## 6    eyewidth_corr   0.090000        mm  Acupalpus meridianus
##   measurementDeterminedBy      order
## 1                   klink Coleoptera
## 2                   klink Coleoptera
## 3                   klink Coleoptera
## 4                   klink Coleoptera
## 5                WOODCOCK Coleoptera
## 6                WOODCOCK Coleoptera
```

trait name and value standardisation

Due to the heterogeneity of approaches and research questions related to trait-based research, a universal trait definition standard does not exist at the time of writing this. Therefore it is difficult to assign globally unique identifiers that provide a reference to an unambiguous definition. Some databases offer a list of traits in some way or another, e.g. as a datasheet of in-text table, but few offer a stable URI reference or an API. Exceptions are the Gramene Ontology, which offers trait definitions for crop plants, and the TOP thesaurus for plant traits (http://top-thesaurus.org), which is rather comprehensive, but does not provide easy means of referencing. Many such trait ontologies are currently under construction for different animal phyla.

For most cases, you would instead refer to an own lookup table, a so called thesaurus of traits, using dataset specific identifiers. The thesaurus may also be part of your metadata accompanying the trait dataset.

To transfer the user provided traits and trait values into standardised values, the function `standardize.traits()` merges the data table with a reference table of trait definitions to produce values of a compliant format.

refer to an existing trait thesaurus

A couple of trait ontologies do exist, e.g. the TOP Thesaurus of plant traits (used by TRY) or Gramene.org offer definitions of plant traits via an API. For soil invertebrates, the T-SITA thesaurus offers a set of traits relevant for this organism group. To date, no script for a systematic access of these ontologies can be provided here. Thus, the key information must be provided manually as an own data object in R.

This procedure is only recommended if *all* of the traits reported in your dataset refer to a definition in an online thesaurus.

```
traits1 <- as.thesaurus(data.frame(traitName = c("body_length", "antenna_length",
                                                 "metafemur_length"),
                        traitID = c(
                          "http://t-sita.cesab.org/BETSI_vizInfo.jsp?trait=Body_length",
                          "http://t-sita.cesab.org/BETSI_vizInfo.jsp?trait=Antenna_length",
                          "http://t-sita.cesab.org/BETSI_vizInfo.jsp?trait=Femur_length"
                          ),
                        traitType = c("numeric"),
                        traitUnit = c("mm"),
                        traitUnitStd = "mm")
)
```

We highly encourage the implementation of open online resources for these glossaries (e.g. via APIs), which would allow a looking up existing trait definitions programmatically in the future.

refer to an own trait list

For an interim solution, until comprehensive trait ontologies are compiled, your dataset should therefore be referenced to a publicly available dataset or file that specifies the trait in detail. Ideally it is stored as an asset along with your trait dataset. This can be a csv or txt file published on a open access repository (figshare, researchgate or github, to name but a few), or a website providing direct links to the trait definition (URI).

The reference file should contain at least the following columns:

- `traitName` should be a short descriptive name. No spaces should be used. Rather use a scheme with underscore or capital letters to highlight multiple words (e.g. 'body_length' or 'bodyLenght').

- `traitID` should specify an alphanumeric ID for the specific use in your dataset or - better - a URI that reliably links to the definition of the trait measurement on an online repository. This could be achieved by providing a online version of your traitlist (TODO: provide instructions in wiki how to achieve this). We highly encourage to submit your own trait definitions to existing ontology servers to facilitate this process of trait standardisation (e.g. with GFBio).
- `traitDescription`: a detailled and unambiguous, human readable definition.

- `traitType` to specify the expected kind of entries. Set it to 'numeric' for quantitative traits, 'integer' for counts or ordinal traits, 'character' for trait values that are provided as free text, 'factor' for traits that take one of few non-ordinal levels, 'logical' for binary/boolean entries (yes/no).

- For *numeric traits*, the field `traitUnit` should provide the expected unit for the trait. The R script will then try to convert trait values into this unit.
- for *categorical traits* of kind 'factor' or 'integer', the field `factorLevels` should contain a list the valid factorial traits separated by semicolon. In case of ordinal traits, the order must be precisely corresponding to the number of possible integer values.
- `Comments` may contain examples and clarifications

Further Columns can be added to specify the trait definition in relation to other current or outdated definitions or to set a version number

- `Refines` would link the trait definition to an existing ontology on the web. Ideally this contains a URI.

- `Reference` would contain a original literature reference where the trait measurement has been described.
- `Version`
- `DateIssued`
- `DateModified`
- `Replaces`

The trait thesaurus can be created from a data.frame using the function `thesaurus()`. The parameter `replace` can be used for fixing column names to the expected names outlined above (see function `rename()` of the reshape package).

```
#NOT WORKING!
traits1 <- as.thesaurus(read.csv("../docs/traitlist_arthropods.csv"),
                     replace = c(measurementType = "traitName",
                                 measurementTypeDescription = "traitDescription",
                                 measurementTypeID = "traitID",
                                 measurementValueType = "valueType")
                     )
```

Alternatively, the thesaurus can be created manually by providing objects of class 'trait' for the function `as.thesaurus()` which will be used to create a valid data frame. This is especially useful if your data comprise only a small number of traits. Using the `as.trait()` syntax may allow a more flexible trait definition and an ensures compliance with the terms of the traitdata standard outlined above. It also allows building a library of trait definitions where single traits can be reused in multiple projects.

```
traits1 <- as.thesaurus(
       body_length = as.trait("body_length", traitUnit = "mm", traitUnitStd = "mm"),
       antenna_length = as.trait("antenna_length", traitUnitStd = "mm"),
       metafemur_length = as.trait("metafemur_length", traitUnit = "mm", traitUnitStd = "mm"),
       eyewidth = as.trait("eyewidth_corr", traitUnit = "mm", traitUnitStd = "mm")
)
```

providing the thesaurus to standardise trait data

The function `standardize.traits()` now finally has all it needs to complete its job.

```
dataset1final <- standardize.traits(dataset1std, traits1)
head(dataset1final)
```

```
##            scientificName                                taxonID taxonRank
## 1  Abax_parallelepipedus http://www.gbif.org/species/5754772   species
## 2  Cymindis_vaporariorum http://www.gbif.org/species/5757155   species
## 3 Demetrias_atricapillus http://www.gbif.org/species/5757310   species
## 4   Paradromius_linearis http://www.gbif.org/species/4480302   species
## 5     Carabus_granulatus http://www.gbif.org/species/1036789   species
## 6    Harpalus_rufipalpis http://www.gbif.org/species/5873211   species
##        traitName traitID traitValue traitUnit      scientificNameStd
## 1 antenna_length       2   8.518519        mm  Abax parallelepipedus
## 2 antenna_length       2   4.390000        mm  Cymindis vaporariorum
## 3 antenna_length       2   1.990000        mm Demetrias atricapillus
## 4 antenna_length       2   1.998333        mm   Paradromius linearis
## 5 antenna_length       2  10.590000        mm     Carabus granulatus
## 6 antenna_length       2   3.505291        mm    Harpalus rufipalpis
##     traitNameStd traitValueStd traitUnitStd warnings
## 1 antenna_length            NA           mm
## 2 antenna_length            NA           mm
## 3 antenna_length            NA           mm
## 4 antenna_length            NA           mm
## 5 antenna_length            NA           mm
## 6 antenna_length            NA           mm
##   measurementDeterminedBy      order
## 1                   klink Coleoptera
## 2                  ribera Coleoptera
## 3                  ribera Coleoptera
## 4           woodklinkmean Coleoptera
## 5                  ribera Coleoptera
## 6                   klink Coleoptera
```

```
write.csv(dataset1final, file = "traitdataset_carabids.csv")
```

What does the function do in terms of standardisation.

- **Unit conversion**: on all numerical traits, unit conversion to the target unit will be attempted. Unit conversion can only be successfully performed if both columns `traitUnit` and `traitUnitStd` are provided with valid unit names for the numeric trait.
- **factor level checking** : if a controlled vocabulary is provided in the trait thesaurus, the function checks whether the provided factor levels are valid and asks for a mapping vector otherwise. (**buggy and incomplete!**)
- **logical value harmonization** : for logical traits, the function harmonizes the standardised output. By default it produces a vector of TRUE and FALSE entries. Missing values return NA. The parameters `output` and `categories` can be provided to function `standardize()`. See `?fixlogical` for further detail.

two-in-one standardization

The functions described here are applied sequentially. The output of the first step can be piped into the second step, etc.

To make things even simpler, the functions for format conversion and standardization are wrapped into one named `standardize()`. Therefore it is possible to run the functions in a single handed way, if all necessary parameters for the intermediate steps are provided. A single call will do, taking all the optional parameters described above.

```r
dataset1final <- standardize(carabids,
            thesaurus = traits1,
            taxa = "name_correct",
            units = "mm",
            keep = c(measurementDeterminedBy = "source_measurement")
            )
```