Vegetation data access and taxonomic harmonization version 0.9.11.4

Florian Jansen

December 17, 2022

Abstract

An example session to show functionality and usage of R library `vegdata`.
After installation of `vegdata` you can invoke this PDF with
vignette('vegdata')

1 Preliminary notes

Some `vegdata` functions expect an installation, or more precisely the main directory structure, of the vegetation database program Turboveg for Windows (see `'http://www.synbiosys.alterra.nl/turboveg/'` and Hennekens and Schaminee [2001]. If the package can not find a Turboveg installation it will use the directory within the package installation path. If you want to use function `taxval` for taxonomic harmonization you will need to have GermanSL or an equally structured reference list. If you do not specify any, the most recent version of GermanSL will be used and if it can not be found within the specified path, it will be downloaded from `https://germansl.infinitenature.org/GermanSL/latest/GermanSL.zip`.

Turboveg uses dBase database format for storage. The package tries to deal with the limitations of that format but it is essential, that you use "Database -¿ Reindex" in Turboveg every time you delete something in your Turboveg database. Otherwise it will not be deleted immediately in the dBase file, instead it is only marked for deletion, i.e. it is still there when you access this file with R and will not be recognized as deleted until you reindex your Turboveg database.

2 Provided functionality

2.1 Taxonomic harmonisation

One of the most important steps in using vegetation data (from different sources) for statistical analysis is to take care about the taxonomic content of the names existing in the database. That is, to make sure, that exactly one (correct and valid) name defines one biological entity. Most researchers remember to convert synonyms to valid names but in many cases the care about e.g. monotypic subspecies or ambiguous taxonomic levels is lacking [Jansen and Dengler, 2010]. The package offers the function `taxval` with different options for the adjustment of synonyms, monotypic taxa, taxonomic levels, members of aggregates and undetermined species.

2.2 Database access

vegdata (will) provide direct access to different vegetation database formats:

Turboveg is a desktop program, written in VisualBasic. It provides basic functions to enter, import, maintain and export vegetation data. From the 2 000 000 vegetation plots registered in `https://www.GIVD.info` approximately 1.5 million are stored in Turboveg databases format.

vegetweb is the German national vegetation database. vegetweb is accessible at `https://www.vegetweb.de`. Data can be selected, open access data can be downloaded directly, other data after clearance from the owners.

VegX is an international exchange standard. An R package with a S4 implementation of the standard is in development

2.3 Cover standardization

Turboveg provides different abundance codes and all kinds of user defined cover codes can easily be added. For vegetation analysis a unique species performance platform is needed which will in most cases be the percentage cover of the observed plot area. Therefore, for every abundance code class the mean cover percentage is defined in Turboveg. Since different scales can occur in a database and the storage format of the code table in Turboveg is somewhat strange, the function `tv.coverperc` provides automatic conversion for convenience.

2.4 Layer aggregation

The most frequently used sample unit in vegetation science is a plot based vegetation relevé [Dengler et al., 2011]. A Braun-Blanquet relevé is a sample of names and coverage (abundance) of species in a specified area (usually between 1 and 1000 m^2) at a specific time. It contains (at least is intended to contain) a *complete* list of photo-autotrophic plants (or a defined subset) in that plot. This information can be stored in a three-column list of relevé ID, Taxon ID and performance measure (e.g. cover code).

Often additional information about the kind of occurrence is wanted. In Turboveg one additional column for the most widespread attribute is included by default: growth height classes. E.g. in a forest it is of interest, if a woody species reaches full height (tree layer) or occurs only as a small individual (herb layer). Other attributes like micro location (hummock or depression, rock or dead wood), development stage (juvenile or not, flowering status etc.) or the month of survey in a multi-seasonal survey could also be of interest and can be added in Turboveg. For analysis you may want to differentiate species with different species-plot attributes (e.g. growing in different layers). Function `tv.veg` provides possibilities for species-plot attribute handling.

2.5 Vegetation matrix

Turboveg stores relevés as a dataframe of occurrences (s. below) but almost all functions and programs for vegetation analyses use plot-species cross-tables with a 0 value for non-occurrence = observed absence. Function `tv.veg` inflates the Turboveg list to matrix format with plots in rows and species in columns. Column names can be either species numbers, species letter-codes (default) or full names (with underscores instead of blanks to match the R naming conventions).

3 Preparations

The best way to introduce the functionalities of the package is a session with example code.

We load the library as usual into our R environment.

```
library(vegdata)
```

Several functions of this package use the directory structure of Turboveg. The first time such a function is called, the internal function `tv.home` tries to find your Turboveg installation path. Depending on whether you have Turboveg installed on your computer or not, it will give you a message (and an invisible return) about the Turboveg installation path or the path to the Turboveg directory structure of package vegdata.

```
tv_home <- tv.home()
```

If you want to change this, declare manually by setting option "tv_home":

```
options(tv_home="path_to_your_Turboveg_root_directory")
options(tv_home="/home/jansen/aGitRepos/vegdata/vegdata/inst/tvdata/")
```

4 Service functions

```
tv.db()

[1] "./elbaue"   "./taxatest"
```

will give you a list of available Turboveg database names (directories within the Turboveg *Data directory*).

```
tv.refl()

[1] "GermanSL 1.5"
```

GermanSL is the default Taxonomic reference list in package `vegdata`. However, whenever you use a Turboveg database name in a function, the Reference list will be read from the database configuration file "tvwin.set" if possible. If you want to change the default reference list:

```
tv.refl('your_preferred_list')
```

will change option *tv.refl* which will be used whenever db or refl is not given.

Package vegdata contains several service functions to query the taxonomic information contained in the reference list.

```
tax('Quercus robur')

Reference list used:  GermanSL 1.5
Taxonomic reference list file GermanSL 1.5 does not exist.  Search path was /tmp/RtmpAglGOW/Species/GermanSL 1.5/species.dbf
Will try to downlaod reflist GermanSL 1.5 (species.dbf) ...
Registered S3 method overwritten by 'httr':
method           from
print.cache_info hoardr
Data source broken.
No internet connection or server down.  Using example data instead.
Option "tv_home" is /tmp/RtmpAglGOW

    TaxonUsageID LETTERCODE                TaxonName NameAuthor SYNONYM
613         4685    QRCSROB             Quercus robur         L.   FALSE
614        29751    QRCSR-R Quercus robur subsp. robur       <NA>   FALSE
    TaxonConceptID              TaxonConcept          VernacularName
613           4685             Quercus robur             Stiel-Eiche
614          29751 Quercus robur subsp. robur Eiche (Unterart), Stiel-
    TaxonRank GRUPPE IsChildTaxonOfID IsChildTaxonOf     NACHWEIS  AccordingTo
613       SPE      S            60924        Quercus Buttler 2018 Buttler 2018
614       SSP      S             4685  Quercus robur Buttler 2018 Buttler 2018
    HYBRID BEGRUEND   EDITSTATUS                              EuroMed
613      0     <NA> Buttler 2018 7493f91e-cb63-4aeb-8fd4-1e660d35ee09
614      0     <NA> Buttler 2018 dfe76b32-b552-4d1d-88e9-a90dc08fde0f
```

The GermanSL is not included in vegdata to keep the R package small. Instead the reference list will be automatically downloaded into the tv_home directory (see `tv.home()`) or a temporary folder, if it is not installed but needed. If you want to use a different list, specify `refl=<Name of your list>` according to the directory name in the Turboveg directory *Species*. Function `tax` can use the given species name (with option strict=FALSE also name parts), or 7 letter abbreviation or the TaxonUsageID (called SPECIES_NR in Turboveg) to look for all (partially) matching species names within the reference list.

In GermanSL versions 1.1 to 1.3 additional information for every taxon is stored in an extra file (tax.dbf) which can be used with option *detailed = TRUE*. Since version 1.4 all information is included in the normal Turboveg file *species.dbf*.

tax will give you all matching names by default. If you set option *strict=TRUE*, only the species with exact match to the given character string will be returned.
syn will give you all taxon names within the swarm of synonyms. The valid name is marked in column SYNONYM with FALSE.

```
tax('Elytrigia repens')$TaxonName

Reference list used:  GermanSL 1.5

[1] "Elytrigia repens"                "Elytrigia repens var. caesia"
[3] "Elytrigia repens var. littoralis" "Elytrigia repens var. repens"

syn('Elytrigia repens')

Name swarm of Elytrigia repens :
    TaxonUsageID                              TaxonName SYNONYM   EDITSTATUS
62           115                       Agropyron repens    TRUE Buttler 2018
63          6541          Agropyron repens subsp. caesium    TRUE Buttler 2018
213        27778                    Elymus repens s. str.   FALSE Buttler 2018
214        10260 Elymus repens subsp. caesius nom. inval.    TRUE Buttler 2018
217        21639                       Elytrigia repens    TRUE Buttler 2018
982        24393                        Triticum repens    TRUE Buttler 2018
```

The reference list contains information about the taxonomic hierarchy which can be used with *child* or *parent*.

```
child(27, quiet=TRUE)$TaxonName
parent(32)
parent(32, rank = 'FAM')
```

If you want to learn more about the taxonomic reference list *GermanSL* for Germany, please look at Jansen and Dengler [2008].

4.1 Name manipulation

We often need to adapt spelling differences and changes of species names not belonging to species concepts. Package vegdata contains several service functions to help with that.

taxname.abbr is for standardisation of names, mostly for rank spellings.

taxname.simplify can be used to compare two name vectors of species names regarding "taxonomic fuzziness". Taxon names will differ through time and opportunity. But not all parts of the names are equally affected. Case endings change much more frequently than the beginning of names, ii and y are exchangeable, etc. Those name parts and diacritic marks, double consonants, "th", and other frequent differences in writing style will be eliminated.

parse.taxa: parse genus and epitheta from name strings.

taxname.removeAuthors Remove name authors from full scientific name strings.

5 Load data from Turboveg

Care about the taxonomic content of the datasets is crucial for every analysis. Some of these steps can be automated with an appropriate taxonomic reference. For background and details see [Jansen and Dengler, 2010].

```
db <- 'taxatest'
```

Defines the vegetation database name according to the name of the Turboveg database directory name

```
tv.metadata(db)
```

Metainformation, i.e. information about the kind of available information should always be given for every database. Since Turboveg does not ask and provide such information, write a simple text file called metainfo.txt and save it within the database folder. Turboveg does not provide any metadata handling. Database `taxatest` is an artificial dataset to show functionalities and necessary steps for taxonomic harmonization.

Let's have a look at the Turboveg data structure.

```
getOption('tv_home')

[1] "/tmp/RtmpAglGOW"

obs <- tv.obs(db)
# Adding species names
species <- tax('all', refl=tv.refl(db=db))
obs$TaxonName <-  species$TaxonName[match(obs$TaxonUsageID, species$TaxonUsageID)]
head(obs[,c('PlotObservationID','TaxonUsageID','COVER_CODE','LAYER','TaxonName')])

  PlotObservationID TaxonUsageID COVER_CODE LAYER
1                 2           27         2b     0
2                 2         4685          4     1
3                 2         4685          1     2
4                 2         4685          1     6
5                 1           31          3     6
6                 1        20096          +     6
                          TaxonName
1           Achillea millefolium agg.
2                       Quercus robur
3                       Quercus robur
4                       Quercus robur
5                 Achillea millefolium
6 Achillea millefolium subsp. collina
```

This condensed format shows only presences of species observations. Every species observation is stored in one row and the membership to a specific vegetation plot is given in column *RELEVE_NR*.

5.1 Taxonomic harmonisation - function taxval

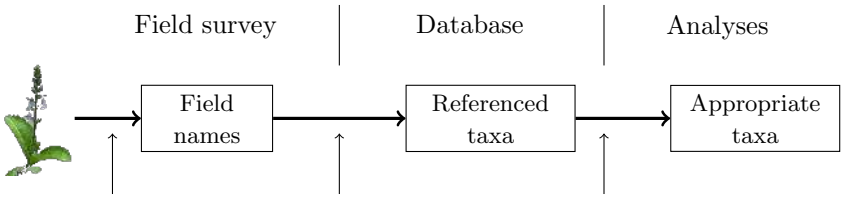
Prerequisits for taxonomic harmonisation in vegdata are

1. a taxonomic tree, i.e. a taxon list with parent-child relationships, given in a column called **IsChildTaxonOfID**
2. ordered taxonomic levels, given as a data.frame, see `taxlevels` with rank abbreviation and numerical values giving the lowest number for the highest level. The rank abbreviation must correlate to of column **TaxonRank** in the taxonomic reference list. Type *taxlevels* to see the implemented table for GermanSL.

We are using the taxonomic reference list GermanSL [Jansen and Dengler, 2008], which contains not only information about synonymy of species names, but also a taxonomic hierarchy. This enables several semi-automatic enhancements of the taxonomic information stored in your vegetation database. If your database is not referenced to GermanSL (and can not be converted), you have to care for a column IsChildTaxonOfID and an appropriate taxlevels data.frame to use function `taxval`. **Otherwise please use option tax=FALSE in `tv.veg`) and do the taxonomic harmonization by hand** (e.g. with function `comb.species`).

Dataset *taxatest* is used to evaluate package code and to show the functionality especially for the taxonomic harmonization. We have Achillea species with a subspecies sudetica, a synonym subspecies collina, two observations on species level (A. atrata and millefolium) and even one on genus level.

```
library(vegdata)
obs <- tv.obs('taxatest')
sort(tax(unique(obs$TaxonUsageID))$TaxonName)
```



1. Field interpretation
   - document your source(s) of taxonomic interpretation (Flora)
   - specify determination certainty
   - collect herbarium specimen
2. Database entry
   - document field records / original literature
   - reference as conservative as possible to a taxonomic reference list with all relevant taxa (synonyms, field aggregates, horticultural plants, ...)
   - document your interpretations
3. Preparation for analyses
   - standardize taxon name parts (rank abbreviations, genus sex, y versus i etc.)
   - convert synonyms
   - summarize monotypic taxa
   - clean up nested taxa
   - clean up taxonomic ranks
   - ...

Three steps of taxonomic interpretation

- need of appropriate tools (software, reference lists)
- standards
- threefold attention

Figure 1: Steps of taxonomic interpretation

```
Reference list used:  GermanSL 1.5

 [1] "Acer pseudoplatanus"
 [2] "Acer tataricum"
 [3] "Achillea"
 [4] "Achillea millefolium"
 [5] "Achillea millefolium agg."
 [6] "Achillea millefolium subsp. collina"
 [7] "Achillea millefolium subsp. sudetica"
 [8] "Acoraceae"
 [9] "Adonis aestivalis"
[10] "Agrostis stolonifera var. palustris"
[11] "Armeria bottendorfensis"
[12] "Armeria maritima subsp. bottendorfensis"
[13] "Armeria maritima subsp. elongata"
[14] "Armeria maritima subsp. halleri s. l."
[15] "Dactylis glomerata s. str."
[16] "Galium mollugo s. str."
[17] "Hieracium pilosella"
[18] "Hieracium subg. Pilosella"
[19] "Picea abies"
[20] "Quercus robur"
```

The database contains 20 different names in the beginning.

Synonyms First the number of species names which are synonyms are given. They are transferred to accepted taxon names, respectively numbers (see option syn='adapt'). If you want to preserve synonyms, choose option syn = 'conflict' or 'preserve'.

Monotypic species within the area Monotypic taxa are valid taxa which are the only child of their next higher taxonomic rank within the survey area. By default they will be converted by `taxval` to the higher rank. For instance *Poa trivialis* is in Germany only represented by *Poa trivialis subspecies trivialis*. Both taxa are valid, but for most analysis only one name for these identical entities must be used. By default a list of monotypic taxa within the GermanSL (whole Germany) is considered (see `tv.mono('GermanSL 1.5')`). The default is to set all monotypic species to the higher rank (because many monotypic subspecies can occur in vegetation databases).
If necessary, the procedure has to be repeated through the taxonomic

Critical Pseudonyms Taxon misapplication is maybe the greatest danger in using survey data. Known misapplications of names (.auct) are embedded within GermanSL. Please pay attention, if these might also be relevant for your dataset.
Completely independent from the questions of correct taxonomic naming of a specific specimen, the boundary of a taxon interpretation can differ much, see Jansen and Dengler [2010]. This should be adequately solved during data entry. Nevertheless the "Warning: Critical species" and "Warning: Potential pseudonyms" give you a last chance to rethink the correctness of your taxon assignments.

Coarsening to a specific taxonomic level For most analyses it will be necessary to analyse only one taxonomic level, e.g. not to confuse species numbers per plot, community means etc. For this you normally have to raise the lower observations to a coarser level. If you want the species level, you can specify ag='adapt' and rank='SPE'. All hierarchical levels below the species level (including the above specified monotypic subspecies) are set to species level in this case.

```
obs.tax <- taxval(obs, db='taxatest', ag='adapt', rank='SPE', check.critical = FALSE, taxlevels = taxlevels, mono = 'pre')

Original number of names: 20
5 synonyms found in dataset. Changed to valid names.
Monotypic taxa preserved!
For 2 taxa the taxonomic hierarchy will be adapted.
Number of taxa after harmonisation: 16
```

```
sort(tax(unique(obs.tax$TaxonUsageID), db=db)$TaxonName)

Reference list used:  GermanSL 1.5

 [1] "Acer pseudoplatanus"        "Acer tataricum"
 [3] "Achillea"                   "Achillea collina"
 [5] "Achillea millefolium"       "Achillea millefolium agg."
 [7] "Acoraceae"                  "Adonis aestivalis"
 [9] "Agrostis stolonifera"       "Armeria maritima s. l."
[11] "Dactylis glomerata s. str." "Galium mollugo s. str."
[13] "Hieracium pilosella"        "Hieracium subg. Pilosella"
[15] "Picea abies"                "Quercus robur"
```

The subspecies have been raised to the species level. However, there is still a conflict between the Achillea millefolium observations and the aggregate. The same for Hieracium pilosella and H. subg. Pilosella. To solve these issues we can us ag='conflict':

```
obs.tax <- taxval(obs, db='taxatest', ag='conflict', check.critical = FALSE)

Original number of names: 20
5 synonyms found in dataset. Changed to valid names.
1 monotypic taxa found in dataset.  Will be set to species rank if possible.
5 conflicting child taxa found in dataset.
[1] "Achillea collina"
[2] "Achillea millefolium"
[3] "Achillea millefolium agg."
[4] "Achillea millefolium subsp. sudetica"
[5] "Hieracium pilosella"
1 conflicting child taxa found in dataset.
[1] "Achillea millefolium"
1 conflicting child taxa found in dataset.
[1] "Achillea millefolium agg."
Number of taxa after harmonisation: 12

sort(tax(unique(obs.tax$TaxonUsageID), db=db)$TaxonName)

Reference list used:  GermanSL 1.5

 [1] "Acer pseudoplatanus"
 [2] "Acer tataricum"
 [3] "Achillea"
 [4] "Acorus"
 [5] "Adonis aestivalis"
 [6] "Agrostis stolonifera subsp. stolonifera"
 [7] "Armeria maritima subsp. elongata"
 [8] "Dactylis glomerata s. str."
 [9] "Galium mollugo s. str."
[10] "Hieracium subg. Pilosella"
[11] "Picea abies"
[12] "Quercus robur"
```

The yarrow taxa are all subsumed to the highest occurring taxon in this taxon tree, i.e. a genus observation. For the two Armeria species the subspecies level was kept because there is no conflicting higher taxon in the dataset.

The result might be disappointing, e.g. think of a single coarse determination only on the family level in a huge dataset. This will lead to raising all species of that family to the family level. This would impede many analyses and you might want to choose a maximum taxonomic level to remain in the dataset. This can be done with option "maxtaxlevel". If you use interactive=TRUE, you will be able to see the number of occurrences for the different levels in file "taxvalDecisionTable.csv" which might help to decide which way to go.

Combining "maxtaxlevel" and the two options of "ag" might come closest to what seems scientifically reasonable in many cases.

```
obs.tax <- taxval(obs, db='taxatest', ag='conflict', maxtaxlevel = 'AGG', check.critical = FALSE, interactive = TRUE)
obs.tax <- taxval(obs.tax, db='taxatest', ag='adapt', rank='SPE', check.critical = FALSE)
sort(tax(unique(obs.tax$TaxonUsageID), db=db)$TaxonName)
```

Check `?taxval` and `args(taxval)` for more options.

6 Vegetation data

At the moment there exists no widely accepted formal class for vegetation data in R. But most functions in `vegan`, `ade4` or other packages expect vegetation data to be stored in a matrix with species in columns and plots in rows. Therefore, we need to inflate the Turboveg format (where zero occurrences are missing) to such a matrix.

`tv.veg` is a wrapper for the above mentioned functions and produces a vegetation matrix with releves as rows and species as columns. Additionally care about species-plot attribute differentiation and combination, and the handling of species codes is provided.

6.1 XML

6.2 Performance measures

At least in Europe most vegetation plots have information about the performance of a species within the survey area, often given in some kind of alphanumeric code for cover percentage within the survey plot. Different code systems are combined by using the mean cover percentage per cover code class. Function `tv.coverperc` will do this job according to the definitions in *Turboveg/Popup/tvscale.dbf* and the entries in the header data column `COVERSCALE`.

```
obs <- tv.obs(db)
# obs <- tv.coverperc(db, obs)
tail(obs)
```

```
   PlotObservationID TaxonUsageID COVER_CODE LAYER SEASON MICROREL FLOWER
19                 1           76          1     6      0     <NA>      0
20                 1        10024          1     6      0     <NA>      0
21                 2         2923          1     0      0     <NA>      0
22                 2        27309          3     6      0     <NA>      0
23                 3        12273          1     6      0     Bult      0
24                 2         4269          1     1      0     Bult      0
```

A few simple possibilities for percentage cover transformations are directly included in the `tv.veg` code, e.g. to use only presence-absence information you can choose option `cover.transform = 'pa'`.

6.3 Pseudospecies

How to account for different vegetation layers or other kinds of species differentiation?

The next step is the separation of pseudo-species. "Pseudo-species" are all kind of taxa split according to species-plot information beyond the performance measure which will be used within the matrix. At this point you have to decide which information should be preserved and which should be aggregated. For instance layer separation must be defined at this step. The default is to differentiate tree, shrub and herb layers but to combine finer layer specifications within them.

If you have more than one occurrence of the same species in a plot, e.g. because tree species growing as young stands and adult specimens were differentiated according to growth height classes, you have to create either pseudo-species which differentiate the occurrences in the resulting vegetation matrix or to combine species occurrences from different layers. For the latter you can use different calculations e.g. to sum up all cover percentages of different layers `lc='sum'` or the maximum value (`lc='max'`), mean value (`lc='mean'`). If you assume an independent occurrence of a species in different vertical layers, you can do the calculations with option lc = 'layer' (the default). This results in a probability sum: A species covering 50% in tree layer

1 and 50% in herb layer will get a combined cover of 75% because both layers will overlap 50% (1 - 0.5*0.5).

If you want to specify pseudo-species by other species-plot differentiation you can define a combination dataframe. Two example dataframes are included in the package (`lc.0` and `lc.1`). Option comb has to be given as a list with first element naming the column name holding the grouping variable and as second element the name of the combination dataframe. Try

```
data(lc.0)
obs <- tv.obs(db)
tv.veg(db, pseudo = list(lc.0, c("LAYER")), lc = "layer")
```

and check the column names:

```
tmp <- tv.veg(db, tax=FALSE, pseudo = list(lc.0, "LAYER"), lc = "layer", quiet=TRUE)

Reading tvabund.dbf
Taxonomic reference list:  GermanSL 1.5
Reading tvhabita.dbf

1 releves without date.  Not converted from factor to date format.

converting cover code ...
creating pseudo-species ...
combining occurrences using type LAYER and creating vegetation matrix ...
replacing species numbers with  shortletters  names ...

names(tmp)

 [1] "AGRES-S.6" "HRCM$P.6"  "ACERPSE.5" "ACERPSE.6" "ACERTAT.2" "ACERTAT.3" "DCTYGLO.6" "ACHLCOL.6"
 [9] "ARMRM-E"   "ARMRM-E.1" "ARMRM-E.2" "GALMMOL.6" "ACHLLAG"   "ARMRM-E.6" "HRCMPIL"   "ACHLMIL.6"
[17] "ACHLM-S.6" "PICEABI.1" "QRCSROB.1" "QRCSROB.2" "QRCSROB.6" "ACHL-SP.6" "#ACRNA.6"  "ADNEAES.6"
```

Separated by dots and layer numbers you can see the preserved layers. For meaning of layer numbers see Turboveg help.
Check `(data(lc.1))` for the default layer combination.

Beside layers you can use any kind of species-plot attributes to distinguish between occurrences, for instance in a multi-temporal survey.

```
comb <- list(data.frame(SEASON=0:4, COMB=c(0,'Spring','Summer','Autumn','Winter')),'SEASON')
names(tv.veg(db, tax=FALSE, pseudo=comb, quiet=TRUE))

Reading tvabund.dbf
Taxonomic reference list:  GermanSL 1.5
Reading tvhabita.dbf

1 releves without date.  Not converted from factor to date format.

converting cover code ...
creating pseudo-species ...
combining occurrences using type LAYER and creating vegetation matrix ...
replacing species numbers with  shortletters  names ...
 [1] "AGRES-S"        "HRCM$P"         "ACERPSE.Spring" "ACERPSE.Summer" "ACERTAT"        "DCTYGLO"
 [7] "ACHLCOL"        "ARMRM-E"        "ARMRM-E.1"      "ARMRM-E.2"      "GALMMOL"        "ACHLLAG"
[13] "ARMRM-E.3"      "HRCMPIL"        "ACHLMIL"        "ACHLM-S"        "PICEABI"        "QRCSROB"
[19] "ACHL-SP"        "#ACRNA"         "ADNEAES"
```

```
data(lc.1)
veg <- tv.veg(db, lc = "sum", pseudo = list(lc.1, 'LAYER'), dec = 1, check.critical = FALSE)

1 releves without date.  Not converted from factor to date format.

veg[,1:10]
```

6.4 Combine species manually

Beside semi-automatic taxon harmonization with function `taxval` there are two possibilities to change Taxonomy manually. If you decide to interpret a certain species name in your database different than stored in the standard view of the taxonomic reference you can replace species numbers within the observational dataframe and run `taxval` later on.

```
obs.tax$TaxonUsageID[obs.tax$TaxonUsageID == 27] <- 31
```

will replace all occurrences of *Achillea millefolium agg.* with *Achillea millefolium* which might be adequate for your survey and will prevent a too coarse taxon grouping. For a longer list of replacements you can use a dataframe.

```
taxon.repl <- data.frame(old=c(27), new=c(31))
obs.tax$TaxonUsageID <- replace(obs.tax$TaxonUsageID,
                                match(taxon.repl$old, obs.tax$TaxonUsageID), taxon.repl$new)
```

The second possibility is to use function `comb.species` on vegetation matrices.

```
comb.species(veg, sel=c('QUERROB','QUERROB.Tree'))
```

will use the first name ('QUERROB') for the replacement column with the cover sums of the selected columns.

7 Site conditions

Vegetation data should come with information about date, place, plot size as well as overarching properties of the plant community and site conditions, plot measurements etc. `tv.site` will load the Turboveg site (header) data and does some basic checks necessary because of the Turboveg dBase format.

```
site <- tv.site(db)

1 releves without date.  Not converted from factor to date format.
Some columns contain no data and are omitted.

 [1] TABLE_NR   NR_IN_TAB  PROJECT    NameAuthor SYNTAXON   UTM        ALTITUDE   EXPOSITION MOSS_IDENT
[10] LICH_IDENT

Some numeric columns contain only 0 values and are omitted.

 [1] COV_TOTAL  COV_TREES  COV_SHRUBS COV_HERBS  COV_MOSSES COV_LICHEN COV_ALGAE  COV_LITTER COV_WATER
[10] COV_ROCK   TREE_HIGH  TREE_LOW   SHRUB_HIGH SHRUB_LOW  HERB_HIGH  HERB_LOW   HERB_MAX   CRYPT_HIGH
[19] EPSG

Some numeric fields contain 0 values:

[1] PRECISION X_COORD   Y_COORD   L_MEA_B   T_MEA_B   K_MEA_B

Please check if these are really meant as 0 or if they are erroneously assigned because of DBase restrictions.
If so, use something like:
site$Column_name[site$Column_name==0] <- NA
```

The function is quite straightforward. After loading the file *tvhabita.dbf* from the specified database folder, warnings are given for plots without specified relevé area or date and fields are checked if they are empty (a lot of predefined header fields in Turboveg are often unused) or contain probably mistakable 0 values in numerical fields, due to dBase deficiencies (dBase can not handle NA = not available values reliably). It is stated in the output, if you have to check and possibly correct 0 values.

8 Additional functions

Use help(package='vegdata') for a complete list of available functions and data sets in vegdata.

8.1 Frequency tables

`syntab` produces a relative or absolute frequency table of a classified vegetation table with the possibility to filter according to threshold values. To exemplify the function we use the second dataset implemented in the package. It is the demonstration dataset from Leyer and Wesche [2007], a selection of grassland relevés from the floodplains of the river Elbe.

```
elbaue <- tv.veg('elbaue', check.critical = FALSE)
elbaue.env <- tv.site('elbaue')

Some columns contain no data and are omitted.
Some numeric columns contain only 0 values and are omitted.
Some numeric fields contain 0 values:
Please check if these are really meant as 0 or if they are erroneously assigned because of DBase restrictions.
If so, use something like:
site$Column_name[site$Column_name==0] <- NA
```

```
clust <- vector('integer', nrow(elbaue.env))
clust[elbaue.env$MGL < -50 & elbaue.env$SDGL < 50] <- 1          # dry sites, low deviation
clust[elbaue.env$MGL < -50 & elbaue.env$SDGL >= 50] <- 2         # dry sites, high deviation
clust[elbaue.env$MGL >= -50 & elbaue.env$SDGL >= 50] <- 3        # wet sites, high deviation
clust[elbaue.env$MGL >= -50 & elbaue.env$SDGL < 50] <- 4         # wet sites, low deviation
levels(clust) <- c('dry.ld','dry.hd', 'wet.hd','wet.ld')
```

We can e.g. look at the relative frequency of all species with more than 40% at least in one column, according to the height of the groundwater table (low or high) and the amplitude of the groundwater table fluctuations (high or low deviations from the mean). Additionally you can use the affiliation of species to abiotic clusters with the help of package `indicspecies`, which calculates species indicator values for one or several cluster [De Cáceres et al., 2010] to order the syntaxonical table. Together with Ellenberg indicator values with will get a comprehensive view into our data.

```
require(indicspecies)

Loading required package:  indicspecies
Loading required package:  permute

synt <- syntab(elbaue, clust, mupa=TRUE)

 Number of clusters:  4
 Cluster frequency 7 10 5 11

synt

$clust
 [1] 2 2 1 4 2 2 4 4 2 3 2 3 4 1 2 2 4 4 2 1 4 4 3 3 2 4 1 1 1 1 3 4 4
attr(,"levels")
[1] "dry.ld" "dry.hd" "wet.hd" "wet.ld"

$syntab
        dry.ld dry.hd wet.hd wet.ld index      stat p.value
CLTHPAL      0     30      0     27     4 0.6030227   0.090
STLLPAL      0      0      0      0     4 0.6289039   0.110
CARXIIS      0     18     14      0     4 0.4264014   0.275
AGRECAN     36      0      0      0     4 0.6030227   0.040
CARXPAG     14     20     20     55     5 0.7669650   0.005
AGRESTO     20      0     45     29     8 0.6282940   0.105
CARXVES      0     18     29     60     4 0.7385489   0.015
CARXVUL     64     14     20     40     2 0.5227402   0.345
ACERPSE      0      0      0     73     5 0.8744746   0.005
CRDMPRA      0     40      0      0     7 0.6869033   0.015
CRSMARV      0     82      0      0     1 0.6411939   0.015
CRSMVUL     18     43     20     60     4 0.3015113   1.000
DSCHCES      0      0     60      9     1 0.7229440   0.020
AGRECAP      0      0      9      0     5 0.6366941   0.100
```

```
ELCHUNI      0      0     43     40     9 0.4364358   0.490
CARXCCU     36      0      0     40     4 0.8742008   0.005
EPHRESU     43     40      0      9     1 0.6546537   0.015
PLNRULI     70     20      9     43     8 0.5773503   0.070
SLNMDUB      0      0     71      0     1 0.3779645   0.400
FSTC#S       0     14     90      0     7 0.4082483   0.435
GALMTAG      0      0     20     27    10 0.6976291   0.245
GALMRAG     40      0     36     43     1 0.8287419   0.010
POA TRI     40      0     14     60    14 0.5617343   0.630
GLYCFLU     18     14     10      0     3 0.4345586   0.565
GLYCMAX      0      0     20      0    10 0.7500000   0.010
HLCSLAN      0      0     18      0     7 0.6236096   0.060
ACHLMIL      0     18     14      0     5 0.5423261   0.130
JNCSEFF     55     29      0     40     4 0.5983999   0.120
DRABNAG     14     30     20     45     5 0.4850713   0.205
ALPCGEN     40     60     27      0     3 0.6474857   0.020
LTHYPRA      0     64     14      0     1 0.5946187   0.025
ALPCPRA     18     71     40     20     5 0.8830871   0.005
LOTSCOR     57     20     80      9    15 0.3892495      NA
LYCHFLO     90      0     64     14    13 0.4662524   0.415
PHLRARU      0      0     29     10    14 0.7556173   0.255
POA PAL      0     14     60      0     2 0.7245147   0.160
POA IAG     43     30     20     36     5 0.7066191   0.225
PTNTANS     10     20     45     43     7 0.5270463   0.135
PTNTREP      0     45     57     10     2 0.5477226   0.075
ANTSODO     55      0     60      0    12 0.5000000   0.725
RNNCFLA     43     10     20     18     4 0.7385489   0.005
RNNCPEP      0     20     27      0    14 0.7543762   0.340
RRPPAMP      0     18     29     10     3 0.7711677   0.005
RRPPSYL      9      0      0      0     8 0.6208071   0.075
RUMXACE      0      0      0      0     7 0.5773503   0.065
RUMXTHY      0     80     27     57     5 0.7276069   0.015
SIUMLAT      0     45      0     10    10 0.6614378   0.010
SYMPFAG      9     43     30      0    10 0.3535534   0.505
TRFLREP     57      0      0      0    12 0.4629100   0.735
VICICRA      0      0      0     57     5 0.4795544   0.715
VICILAT      0     36     43     60     2 0.3162278   0.695
VICITET     18     29     10      0     1 0.7079923   0.010
TRXCSET     57     30      0     18     5 0.7173843   0.020

attr(,"class")
[1] "syntab" "list"
```

9 Vegetation analyses

The package *vegdata* serves mostly as a helper for the analysis of vegetation data. Several powerful R packages like *vegan* and others exist, to provide a very broad range of possibilities.

9.1 Multivariate Ordinations

With the functions shown above we are now ready to do some example analyses in the wide area of vegetation analyses.

We can do, for instance, a "Nonmetric Multidimensional Scaling with Stable Solution from Random Starts Axis Scaling and Species Scores" which is a wrapper for Kruskal's Non-metric Multidimensional Scaling [Cox and Cox, 1994, 2001] from Jari Oksanen [Oksanen et al., 2008].

```
## Data analyses
if (requireNamespace('vegan', quietly = TRUE) ) {
  library(vegan)
veg.nmds <- metaMDS(elbaue, distance = "bray", trymax = 5, autotransform =FALSE,
                    noshare = 1, expand = TRUE, trace = 2)
#eco <- tv.traits()
```

```
#ecol$OEK_F <- as.numeric(ecol$OEK_F)
F <- cwm(veg = elbaue, trait.db = 'ecodbase.dbf', ivname = 'OEK_F', method = 'mean')
N <- cwm(veg = elbaue, trait.db = 'ecodbase.dbf', ivname = 'OEK_N', method = 'mean')
env <- envfit(veg.nmds, env = data.frame(F, N))
} else
  message("package vegan not available")

Loading required package:  lattice
This is vegan 2.6-4
```

To show the result in comparison with environmental measurements in a nice graphic we do some plotting magic.

The first axis of our NMDS plot show the influence of mean groundwater level on the patterns of the dataset. *Glyceria maxima* is marking the wet side of the gradient, whereas *Cnidium dubium Agrostis capillaris* or *Galium verum agg,* occur only at low mean groundwater level. The second axis can be assigned to the fluctuation of water levels measured as standard deviation of mean groundwater level. Species indicating high water fluctuation are *Agrostis stolonifera* or *Alopecurus geniculatus* whereas *Carex vesicaria* occurs only in more balanced situations.

References

Stephan M. Hennekens and Johannes Hendrikus Jacques Schaminee. Turboveg, a comprehensive data base management system for vegetation datasoftware package for input, processing, and presentation of phytosociological data. *Journal of Vegetation Science*, 12:589–591, 2001.

Florian Jansen and Juergen Dengler. Plant names in vegetation databases - a neglected source of bias. *Journal of Vegetation Science*, 21(6):1179–1186, Aug 2010. doi: 10.1111/j.1654-1103.2010.01209.x. URL `http://doi.wiley.com/10.1111/j.1654-1103.2010.01209.x`.

Jürgen Dengler, Florian Jansen, Falko Glöckler, R.K. Peet, Miquel De Cáceres, M. Chytrý, Jörg Ewald, Jens Oldeland, G. Lopez-Gonzalez, Manfred Finckh, and Others. The Global Index of Vegetation-Plot Databases (GIVD): a new resource for vegetation science. *Journal of Vegetation Science*, 22(4):582–597, 2011. doi: 10.1111/j.1654-1103.2011.01265.x. URL `http://onlinelibrary.wiley.com/doi/10.1111/j.1654-1103.2011.01265.x/full`.

Florian Jansen and Juergen Dengler. Germansl - eine universelle taxonomische referenzliste fuer vegetationsdatenbanken. *Tuexenia*, 28:239–253, 2008.

Ilona Leyer and Karsten Wesche. *Multivariate Statistik in der Oekologie*. Springer, Berlin, 2007.

Miquel De Cáceres, Pierre Legendre, and Marco Moretti. Improving indicator species analysis by combining groups of sites. *Oikos*, 119(10):1674–1684, October 2010. ISSN 00301299. doi: 10.1111/j.1600-0706.2010.18334.x. URL `http://doi.wiley.com/10.1111/j.1600-0706.2010.18334.x`.

T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman & Hall, 1994, 2001.

Jari Oksanen, Roeland Kindt, Pierre Legendre, Bob O'Hara, Gavin L. Simpson, and M. Henry H. Stevens. *vegan: Community Ecology Package*, 2008. URL `https://cran.r-project.org/,http://vegan.r-forge.r-project.org/`.

```
if (requireNamespace('interp', quietly = TRUE) & requireNamespace('labdsv', quietly = TRUE) & requireNamespace('vegan', quie
  library(labdsv)
  library(interp)
color = function(x)rev(topo.colors(x))
nmds.plot <- function(ordi, site, var1, var2, disp, plottitle =  'NMDS', env = NULL, ...) {
lplot <- nrow(ordi$points);  lspc <- nrow(ordi$species)
filled.contour(interp(ordi$points[, 1], ordi$points[, 2], site[, var1], duplicate = 'strip'),
           ylim = c(-1, 1.1), xlim = c(-1.4, 1.4),
           color.palette = color, xlab = var1, ylab = var2, main = plottitle,
           key.title = title(main = var1, cex.main = 0.8, line = 1, xpd = NA),
           plot.axes = { axis(1);  axis(2)
             points(ordi$points[, 1], ordi$points[, 2], xlab = "", ylab = "", cex= .5, col = 2, pch = '+')
             points(ordi$species[, 1], ordi$species[, 2], xlab = "", ylab = "", cex=.2, pch = 19)
             ordisurf(ordi, site[, var2], col = 'black', choices = c(1, 2), add = TRUE)
             orditorp(ordi, display = disp, pch = " ")
             legend("topright", paste("GAM of ", var2), col = 'black', lty = 1)
             if(!is.null(env)) plot(env, col='red')
           }
           ,...)
}

nmds.plot(veg.nmds, elbaue.env, disp='species', var1="MGL", var2="SDGL", env=env, plottitle = 'Elbaue floodplain dataset')
} else {
  message("packages interp and/or labdsv not available")
}
```

```
Loading required package:  mgcv
Loading required package:  nlme
This is mgcv 1.8-41.  For overview type 'help("mgcv-package")'.
This is labdsv 2.0-1
convert existing ordinations with as.dsvord()

Attaching package:  'labdsv'
The following object is masked from 'package:stats':
density
```