Rveg User Manual

version 0.1.6

Přemysl Král and Jan Douda

Contents

1 INTRODUCTION

The Rveg package is database software for students and scientists, mainly botanists and ecologists, that allows transcription of your phytosociological relevés in the R environment. This is a comprehensive guide to understanding how the Rveg package works and its capabilities.

You are reading the manual for the first reviewed version (0.1.6), which provides a working solution for digitizing relevés. In future updates, we aim to improve the quality and functionality and explore new features like utilizing Ellenberg's indicator values, exporting to standardized exchange formats, or allowing for habitat classification. There are a few sources regarding Rveg development you might visit:

1. GitHub at https://github.com/sesitcsl2/Rveg - for the latest development version
2. Team website at https://plant-ecology-lab-czu.com/rveg/ - for news and video guides
3. CRAN repository - for the latest stable version

The manual is divided into two parts. If you are using Rveg for the first time, you might be interested in the first part, which is a step-by-step example of basic digitizing suitable for testing on your first relevé. In the second part, all of the functions and their arguments are described precisely.

1.1 overview

Rveg is capable of fast transcription, editing, and viewing of your relevés. It creates a *database* composed of two CSV files (comma-separated values): one for header data (environmental variables, file ending with `HEAD.csv`) and a second for abundance data (file ending with `REL.csv`). Values in both files can be edited outside of Rveg in any spreadsheet software while keeping compatibility with Rveg, but don't edit row names or column names.

The **Header** file consists of preset variables in rows and relevés in columns. It is advised not to use commas in values, as commas are used as separators.

The **Relevés** file consists of species number codes as row names and relevés in columns. The first column represents the species shortcut code combined with an underscore and the vegetation layer. It is based on the external checklist of species (Danihelka et al. 2012), which assigns a shortcut code (e.g., *TRIFPRA* for *Trifolium pratense*) to each species to prevent typos and speed up the process of writing. A REL table might look like this:

```
      X ShortName X1 X2
1   291 ALNUGLU_3 88  0
2 11787 ALNUGLU_2 10  0
3 29777 POA ANN_1 50  0
4 29789 POA RIP_1 23  0
5 32387 TRIFMON_1  0 40
6 32391 TRIFPRA_1  0  2
```

The Rveg package is based entirely around working with these two files, which together are referred to as the Rveg database. This is achieved primarily by the `addReleve` function, but Rveg also contains some functions with extra features. Rveg includes the following functions:

1. `addReleve()`
2. `RvegCombine()`
3. `RvegCheck()`
4. `RvegToJuice()`
5. `RvegMerge()`
6. `tvToRveg()`
7. `RvegToTv()`
8. `CreateChecklist()`
9. `RvegLoad()`

To understand each function, view the next chapter or run `help("function_name")`.

2 START

2.1 relevé digitizing

Install the latest version of the Rveg:

```
# Official CRAN installation
install.packages("Rveg")
require(Rveg)

# Alternative GitHub installation
# Be aware that devtools package use number of dependencies
install.packages("devtools")
devtools::install_github("sesitcsl2/Rveg")
require(Rveg)
```

Assuming you have successfully installed Rveg, we can start digitizing right away. We start with the core function for digitizing, `addReleve()`. For our first database to be created, we need the argument `SAVE`, which specifies where the database will be stored on our disk. Let's say we want to have our database stored in the home directory and named "MyFirstDatabase".

```
setwd("~")
addReleve(SAVE = "MyFirstDatabase")
```

By forming the function this way, we will immediately start by filling the first relevé. The process starts by filling header fields. Enter a value for each field; formats are not mandatory to follow. You can leave the fields empty as well.

```
DATE?(YYYY/MM/DD) 2023/1/5
SPRINGDATE?(YYYY/MM/DD)
LOCALITY? Prague-Suchdol
...
```

Here I show you the first three fields, where I have filled the `DATE` and `LOCALITY` but left `SPRINGDATE` empty, as we have not sampled spring aspect vegetation. The last field is `NOTE` for any additional remarks. After you complete the first header, Rveg starts to ask you if you want to add species. Let's add our first species, *Trifolium pratense*, with a 30% cover.

```
...
Note?
AddNewLayer? (Y/N) Y
Select Layer (3,2,1,J,0) 1
P - percentage, BB - Braun B. scale, CS - custom scale p
AddSpecies?(GenuSpe/N) trifpra
Abundance?(%) 30
                FullName
32391 Trifolium pratense
CorrectName?(Y/F(search for name)) y
      ShortName  0
32391 TRIFPRA_1 30
AddSpecies?(GenuSpe/N)
```

At this moment we are editing first reléve, herb layer in percentage and Rveg awaits our response if we want to add another specie (our first). Species are inserted by the shortname species codes, that is usually created by 4 letters of Genus and 3 letters of species, therefore for *Trifolium pratense* we type TrifPra. After we type the abundance, we are aske to doublecheck if the species is correct. This is also useful for exceptions of shortnames, as some codes could be duplicite or for other reasons, the species code might except the 4 genus 3 species letters rule. We confirm *Trifolium pratense* and Rveg list our relevé, which consist of one species now and already awaits for another species. The id row names are not important for us. Lets add few more species, *Arrhenatherum elatius*, *Rumex acetosa* and *Bellis perennis*.

In the first option, we confirmed we want to add/edit a new vegetation layer; we selected one which represents the herb layer and then we selected percentage as cover values. Note that Rveg suggests capital letters as answers, and while R is case-sensitive, Rveg capitalizes the text in the background, so you can write in lowercase as well. At this moment, we are editing the first relevé, herb layer in percentage, and Rveg awaits our response if we want to add another species (our first). Species are inserted by the shortname species codes, which are usually created by 4 letters of the genus and 3 letters of the species; therefore, for *Trifolium pratense* we type `trifpra`. After we type the abundance, we are asked to double-check if the species is correct. This is also useful for exceptions of shortnames, as some codes could be duplicates or for other reasons the species code might not follow the 4 genus and 3 species letters rule. We confirm *Trifolium pratense*, and Rveg lists our relevé, which consists of one species now and already awaits another species. The `ID` row names are not important for us. Let's add a few more species: *Arrhenatherum elatius*, *Rumex acetosa*, and *Bellis perennis*.

```
AddSpecies?(GenuSpe/N) arrhela
Abundance?(%) 25
                   FullName
23586 Arrhenatherum elatius
CorrectName?(Y/F(search for name)) y
      ShortName  0
23586 ARRHELA_1  25
32391 TRIFPRA_1  30
AddSpecies?(GenuSpe/N) rumeace
Abundance?(%) 5
           FullName
30902 Rumex acetosa
CorrectName?(Y/F(search for name)) y
      ShortName  0
23586 ARRHELA_1  25
30902 RUMEACE_1  5
32391 TRIFPRA_1  30
AddSpecies?(GenuSpe/N) bellper
Abundance?(%) 4
[1] FullName
<0 rows> (or 0-length row.names)
CorrectName?(Y/F(search for name))
```

Now we hit a wall: Rveg does not know any species with the shortname `bellper`. Therefore, we must decline that the name is correct, which is followed by a question about the first three letters of the species full name. Rveg then lists all species with corresponding name starts, also with the correct ShortName codes.

```
<0 rows> (or 0-length row.names)
CorrectName?(Y/F(search for name)) f
SpeciesFirst3letters?(eg.Che) bel
     Number ShortName                   FullName
776    1058   ASTEBEL      Bellidiastrum michelii
1005   1332   BELIPER             Bellis perennis
1006   1333   BELI-SP              Bellis species
1007   1334   BELOHER           Belonia herculana
...
SpeciesName?(GenuSpe)
```

Now we see that according to the checklist, the correct shortname is BELIPER.

```
SpeciesName?(GenuSpe) beliper
            FullName
23997 Bellis perennis
CorrectSpecies?(Y/N) y
      ShortName  0
23586 ARRHELA_1 25
23997 BELIPER_1  4
30902 RUMEACE_1  5
32391 TRIFPRA_1 30
AddSpecies?(GenuSpe/N)
```

Now we have successfully added *Bellis perennis* to the herb layer; however, at our plot, we also found a shrub of *Sambucus nigra*. Let's add it to our relevé, and we can try the Braun-Blanquet scale. We will select layer two (shrub layer) and abundance +.

```
AddSpecies?(GenuSpe/N) f
AddNewLayer?(Y/N) y
Select Layer (3,2,1,J,0) 2
P - percentage, BB - Braun B. scale, CS - custom scale bb
AddSpecies?(GenuSpe/N) sambnig
Abundance?(0,R,+,1,2,M,A,B,3,4,5) +
           FullName
19520 Sambucus nigra
CorrectName?(Y/F(search for name)) y
      ShortName  0
19520 SAMBNIG_2  2
23586 ARRHELA_1 25
23997 BELIPER_1  4
30902 RUMEACE_1  5
32391 TRIFPRA_1 30
AddSpecies?(GenuSpe/N)
```

You can see *Sambucus nigra* is now in our list within the second layer, and the cover code was converted from + to 2%. In case you want to adjust the abundance or remove the species, we can just add the species again with a different cover; zero to remove. Let's try it and change the abundance of *A. elatius* to 35%. Therefore, we need to edit layer one again.

```
AddSpecies?(GenuSpe/N) N
AddNewLayer?(Y/N) Y
Select Layer (3,2,1,J,0) 1
P - percentage, BB - Braun B. scale, CS - custom scale P
AddSpecies?(GenuSpe/N) Arrhela
Abundance?(%) 35
                  FullName
23586 Arrhenatherum elatius
CorrectName?(Y/F(search for name)) Y
      ShortName  0
19520 SAMBNIG_2  2
23586 ARRHELA_1 35
23997 BELIPER_1  4
30902 RUMEACE_1  5
32391 TRIFPRA_1 30
```

Now we assume our first relevé is finished. Congratulations! Therefore, we decline adding new species and adding new layers.

```
AddSpecies?(GenuSpe/N) n
AddNewLayer?(Y/N) n
Species_richness
[1] 5
number of relevés:  1, number of headers:  1
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP)
```

Now we end up in a menu where we can select different prompts. For a detailed explanation, check the `addReleve` section in the FUNCTIONS chapter. We will now be satisfied with `PRINTREL` and `PRINTHEAD`. If we want to see our database state, we prompt `PRINTREL` for relevés and `PRINTHEAD` for headers. Prompting `Y` will start a new relevé with header as before.

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) PRINTREL
      ShortName X1
19520 SAMBNIG_2  2
23586 ARRHELA_1 35
23997 BELIPER_1  4
30902 RUMEACE_1  5
32391 TRIFPRA_1 30
number of relevés:  1, number of headers:  1
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) PRINTHEAD
    ShortName             X1
1          ID              1
2        DATE       2023/1/5
3  SpringDATE
4    LOCALITY Prague-Suchdol
5   FieldCODE              1
6     Authors             PK
7    PlotSize              4
8    Latitude       50.12981
9   Longitude       14.37081
10   Accuracy              5
11        CRS          WGS84
12      Slope
13   Exposure
14         E3              0
15         E2              5
16         E1             75
17       Ejuv              0
18         E0              0
19       Note
number of relevés:  1, number of headers:  1
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP)
```

3 FUNCTIONS

3.1 `addReleve`

```
addReleve(DATABASE = "NEW", SAVE = "default", checklist = "default", extrahead = NULL,start = TRUE)
```

ARGUMENTS

- **DATABASE** is set by default to `"NEW"`, which indicates that the user is creating a new database. You will need to change this value only if you want to edit or add new records to your database created before. In that case, you will enter the name of the database (without the `HEAD.csv`, `REL.csv` suffix).
- **SAVE** sets the path and name of your database. It is therefore the only required argument when creating a new database. When you are editing an already existing database, you can select to keep **SAVE** the same as **DATABASE**, which will overwrite the loaded database, or you can choose a different **SAVE** value to create a new database as well as keep the old one (e.g., backup or versioning). If kept on **default**, the temporary files will be used in the `tmp` directory.
- **checklist** is set by default to `"default"`, which makes Rveg use the provided checklist by Danihelka et al. (2012). At the moment, there are no more checklists, but you are able to edit or create your own checklists. Insert the full path to the custom checklist. A custom checklist can be structured to the required format with the function `CreateChecklist()`.
- **extrahead** is set by default to `NULL`, meaning there are no extra header characteristics. To add other fields, insert the character vector of the characteristics.
- **start** Is logical value if the user wants to start digitizing right away by default method. Works only when creating a new database.

DESCRIPTION

The most important function, which is the core of the whole package. It allows you to digitize relevés, therefore creating and editing Rveg databases which consist of two CSV files: one for evironmental (header) data and another for abundance (relevé). The whole process is based on communication with the user. After starting the function, the digitizing process begins and runs until closed; until then, the R session cannot be used in other ways. You can create a database for your project with preset header data, but it also allows you to add your own characteristics. There are several ways of digitizing using this function, each explained below. This function includes several mechanisms to make digitizing comfortable, efficient, and error-free. First, the database is continuously saved during the digitizing process after every step (e.g., after every relevé digitized); therefore, in an unexpected closure of the software, most of the data are not lost. Be aware that R needs rights to write the files onto the disk; therefore, ensure the files are not open in other software (e.g., Microsoft Excel), as it might cause errors. To prevent typographical mistakes and speed up the digitizing process, the species are inserted using 7-letter codes and require double-checking. Thanks to the species checklist (Danihelka et al. 2012) for the Czech Republic, provided within the package, Rveg will connect species names to the 7-letter codes. If you ever find the checklist insufficient, you can use custom checklists. In future updates, more checklists might be included. The structure of the checklist is a TXT file, a dataframe with 3 columns. The first column, *Number*, represents the ID of the species. Rveg will recalculate IDs by itself, so there is no real use for this column; you can use it for your own sorting and orientation. If you don't want to think of a number, you can simply enter 99999, or any other number. The second column, *ShortName*, is the already mentioned species shortcut. It is written in uppercase and consists of 4 letters of the genus and 3 letters of the species. It is important to keep uppercase letters and ensure every shortname is unique and present only once. To access the checklist for modifying, you can run:

```
checklist <- system.file("extdata", "DANIHELKA2012rko.txt", package="Rveg")
checklist <- read.delim(checklist,sep = "\t")
"ABCDEFG" %in% checklist$ShortName # check if the code is available
[1] FALSE # it is available / not used
```

The last column, *FullName*, contains full species names. Columns are separated by a tabulator. I recommend editing the file in a text editor rather than in R, but be careful as some text editors can alter tabulators with spaces.

Default header includes following fields:

1. **ID** (YYYY/MM/DD) - generated automatically, unique value for each relevé
2. **DATE** (YYYY/MM/DD) - date of data sampling
3. **SpringDATE** - date of sampling spring aspect
4. **LOCALITY** - place where relevé was taken
5. **FieldCODE** - internal code for distinguishing relevés from the same locality
6. **Authors** - name of the sampler
7. **PlotSize** (m^2) - size of the sampled area
8. **Latitude** & **Longitude** - XY coordinates
9. **Accuracy** - accuracy of coordinate measurement
10. **CRS** - used coordinate reference system
11. **Slope** (degrees) - slope of the plot
12. **Exposure** - exposure direction
13. **E3**, **E2**, **E1**, **Ejuv**, **E0** (%) - percentage cover of each layer (tree, shrub, herb, juvenile, and moss, respectively)
14. **Note** - any extra note you might need to record

Example of a relevé header:

```
1          ID               1
2        DATE        2023/1/5
3  SpringDATE                
4    LOCALITY Prague-Suchdol
5   FieldCODE               1
6     Authors              PK
7    PlotSize               4
8    Latitude        50.12981
9   Longitude        14.37081
10   Accuracy               5
11        CRS           WGS84
12      Slope                
13   Exposure                
14         E3               0
15         E2               5
16         E1              75
17       Ejuv               0
18         E0               0
19       Note                
```

Some characteristics have a recommended format; however, it is completely up to you and your preferences how you fill them. Everything is converted to string (text) format. You can notice that some of the fields remain empty. Every database contains this set of header fields, but it is up to the user if they want to fill them.

3.1.1 Creation of the database

Run function

```
addReleve(SAVE = "First_database")
```

That will start a dialogue communication with the user. Be careful not to stop the dialogue in the process by pressing the escape key, for example, as it would stop the digitizing process. When creating a new database with default settings (`start = TRUE`), it will start immediately recording your first relevé, starting with the header:

```
DATE?(YYYY/MM/DD): 2023/01/05
SPRINGDATE?(Y/M/D):
LOCALITY?: Yellowstone
...
E0?(%) 12
Note?
AddNewLayer?(Y/N)
```

You don't have to fill in all the characteristics; if you were not recording something or you will not have any use for it, you can leave the field blank (as we did with **SPRINGDATE** and **Note**). All values are converted to character. After that, the loop dialogue will allow you to start recording species.

```
AddNewLayer?(Y/N) Y
Select Layer (3,2,1,J,0) 1
P - percentage, BB - Braun B. scale, CS - custom scale bb
AddSpecies?(GenuSpe/N)
```

We started to write species in the first (herb) layer with covers noted in the modified Braun-Blanquet scale (BB). The other possible options are using percentage covers or indicating that a custom scale that is not pre-coded in Rveg will be used. In the case of a custom scale, the values will stay in the scale and will have to be converted later. That is possible with the function `RvegLoad`. In the case of used pre-coded scales (e.g., Braun-Blanquet), the values are automatically transferred to percentage, as you will see in this example.

$$r = 1\%;\ + = 2\%;\ 1 = 3\%;\ 2m = 4\%;\ 2a = 8\%;\ 2b = 18\%\ [2 = 15\%];\ 3 = 38\%;\ 4 = 63\%;\ 5 = 88\%$$

Now we have to start filling species using 7-letter codes, consisting of the first 4 letters of the genus and the first 3 letters of the species name. There are a few exceptions; for example, genus *Poa* is written with 3 letters of genus, a space, and 3 letters of the species name. Or you might want to record an aggregate; that is achieved with a hashtag symbol, e.g., *ARTE#VU* stands for *Artemisia vulgaris agg.* For a full list of exceptions see chapter `CreateChecklist`. There is no need to remember the codes, as Rveg can show them to you, but after some time, you might start memorizing them automatically.

```
AddSpecies?(GenuSpe/N) TrifPra
Abundance?(0,R,+,1,2,M,A,B,3,4,5) b
                FullName
32391 Trifolium pratense
CorrectName?(Y/F(search for name)) y
      ShortName  0
32391 TRIFPRA_1 18
AddSpecies?(GenuSpe/N)
```

We added *Trifolium pratense* with a cover of 2b (18%). This process should be repeated for every species in the herb layer, and then again for all other layers you have sampled. What might happen is that Rveg, for some reason, does not recognize the ShortName code correctly; for example, if the code is reserved by any other species, then the code has to be different from the standard formula. This situation can look like this while we try to add *Galium aparine*:

```
AddSpecies?(GenuSpe/N): Galiapa
Abundance?(%): a
[1] FullName
<0 rows> (or  0-length row.names)
CorrectName?(Y/F(search for name)) F
SpeciesFirst3letters?(eg.Che) Gal
...
 Galium album ssp. album
3708    4575   GALUPYC                   Galium album ssp. pycnotrichum
3694    4552   GALUAPA                                  Galium aparine
3693    4551   GALU#AP                             Galium aparine agg.
3695    4554   GALUAUS                               Galium austriacum
3696    4555   GALU#BO                                  Galium boreale
...
SpeciesName?(GenuSpe) galuapa
            FullName
26686 Galium aparine
CorrectSpecies?(Y/N)  y
      ShortName  0
26686 GALUAPA_1  8
32391 TRIFPRA_1 18
AddSpecies?(GenuSpe/N)
```

When Rveg asks if the species name is correct, but the software gives you zero response or the wrong species name, it is then necessary to not confirm by entering “F”. Write 3 starting letters of the genus to get a full list of all species starting with those letters, with their corresponding ShortName codes. You just have to scroll through the alphabetically sorted list to find the desired species. After entering the correct code, Rveg recognizes the species correctly.

So far, we have inserted two species in the herb layer of our first relevé Let’s add one shrub species and one tree species. Notice that even though R is case-sensitive, Rveg transforms all responses to uppercase, and therefore all responses are valid. Tree species will be added with percentage instead of Braun-Blanquet.

```
AddSpecies?(GenuSpe/N) n
AddNewLayer?(Y/N) y
Select Layer (3,2,1,J,0) 2
P - percentage, BB - Braun B. scale, CS - custom scale bb
AddSpecies?(GenuSpe/N) sambnig
Abundance?(0,R,+,1,2,M,A,B,3,4,5) 3
            FullName
19520 Sambucus nigra
CorrectName?(Y/F(search for name)) y
      ShortName  0
19520 SAMBNIG_2 38
26686 GALUAPA_1  8
32391 TRIFPRA_1 18
AddSpecies?(GenuSpe/N) n
AddNewLayer?(Y/N) y
Select Layer (3,2,1,J,0) 3
P - percentage, BB - Braun B. scale, CS - custom scale p
AddSpecies?(GenuSpe/N) poputre
Abundance?(%) 45
            FullName
6986 Populus tremula
CorrectName?(Y/F(search for name)) y
      ShortName  0
6986  POPUTRE_3 45
19520 SAMBNIG_2 38
```

Very nice! Notice how layers are distinguished in one list. Now we realize that we have mistakenly inserted *Populus tremula*, but our intention was to insert *Pinus sylvestris*. How can we correct that mistake? The species can be removed by assigning it a zero cover. Cover can also be changed by inserting the species again with the correct cover value. Let's first remove *P. tremula* by assigning it a cover of zero, and then add the intended species, *P. sylvestris*.

```
AddSpecies?(GenuSpe/N): n
AddNewLayer?(Y/N) y
Select Layer (3,2,1,J,0) 3
P - percentage, BB - Braun B. scale, CS - custom scale p
AddSpecies?(GenuSpe/N) poputre
Abundance?(%) 0
            FullName
6986 Populus tremula
CorrectName?(Y/F(search for name)) y
      ShortName  0
19520 SAMBNIG_2 38
26686 GALUAPA_1  8
32391 TRIFPRA_1 18
AddSpecies?(GenuSpe/N) pinusyl
Abundance?(%) 45
             FullName
6656 Pinus sylvestris
CorrectName?(Y/F(search for name)) y
      ShortName  0
6656  PINUSYL_3 45
19520 SAMBNIG_2 38
26686 GALUAPA_1  8
32391 TRIFPRA_1 18
AddSpecies?(GenuSpe/N)
```

Now the mistake has been repaired. Let's assume that our first relevé is completed. We therefore decline adding more species or more layers.

```
AddSpecies?(GenuSpe/N) N
AddNewLayer?(Y/N) N
Species_richness
[1] 4
number of relevés:  1, number of headers:  1
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP)
```

You receive the number of species recorded in the relevé, which is good for checking if you recorded all the species. Notice that species present in more layers count as more species! You are now in the main menu, which offers some more functions. If we would run `addReleve(start = FALSE)`, we would also be in this menu, with the difference that there would be 0 relevés and headers. Let's explain each one of them and provide examples.

3.1.2 Menu options (prompts)

If you had started editing an existing database earlier, you would find yourself here. To add more relevés, enter `Y` and repeat the same process as before. The commands `PRINTHEAD` and `PRINTREL` will show you the current database header and relevés, respectively.

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) printrel
      ShortName X1
6656  PINUSYL_3 45
19520 SAMBNIG_2 38
26686 GALUAPA_1  8
32391 TRIFPRA_1 18
number of relevés:  1, number of headers:  1
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) printhead
    ShortName          X1
1          ID           1
2        DATE  2023/01/05
3  SpringDATE
4    LOCALITY Yellowstone
5   FieldCODE           1
6     Authors          PK
7    PlotSize         200
8    Latitude     4980364
9   Longitude   -110.5885
10   Accuracy          10
11        CRS       WGS84
12      Slope
13   Exposure          sw
14         E3          50
15         E2          30
16         E1          70
17       Ejuv
18         E0          12
19       Note
number of relevés:  1, number of headers:  1
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP)
```

It is possible to edit relevés with the `RREL` command:

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) RREL
ReleveNumber? 1
 [1] "1"           "2023/01/05"  ""            "Yellowstone" "1"           "PK"          "200"
"4980364"
 [9] "-110.5885"   "10"          "WGS84"       ""            "sw"          "50"          "30"
"70"
[17] ""            "12"          ""
CorrectNumber?(Y/N) y
      ShortNames Cover
26686  GALUAPA_1     8
6656   PINUSYL_3    45
19520  SAMBNIG_2    38
32391  TRIFPRA_1    18
AddNewLayer?(Y/N)
```

Similarly, RHEAD allows editing of header characteristics. Let's change shrub cover to 38%:

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) rhead
ReleveNumber? 1
 [1] "1"           "2023/01/05"  ""            "Yellowstone" "1"           "PK"          "200"
"4980364"
 [9] "-110.5885"   "10"          "WGS84"       ""            "sw"          "50"          "30"
"70"
[17] ""            "12"          ""
CorrectColumn?(Y/F) y
RepairHeader?(Y/N) y
 [1] "ID"         "DATE"       "SpringDATE" "LOCALITY"   "FieldCODE"  "Authors"    "PlotSize"
"Latitude"   "Longitude"
[10] "Accuracy"   "CRS"        "Slope"      "Exposure"   "E3"         "E2"         "E1"
"Ejuv"       "E0"
[19] "Note"
HeaderCharacteristic? E2
NewValue? 38
RepairHeader?(Y/N) n
number of relevés:  1, number of headers:  1
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) printhead
    ShortName          X1
1          ID           1
2        DATE  2023/01/05
3  SpringDATE
4    LOCALITY Yellowstone
5   FieldCODE           1
6     Authors          PK
7    PlotSize         200
8    Latitude     4980364
9   Longitude   -110.5885
10   Accuracy          10
11        CRS       WGS84
12      Slope
13   Exposure          sw
14         E3          50
15         E2          38
16         E1          70
17       Ejuv
18         E0          12
19       Note
number of relevés:  1, number of headers:  1
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP)
```

A rarely used command is REMOVEREL. It removes the chosen relevé with its header from the database.

When we were digitizing our first relevé, we digitized the header and then species data for each relevé If we prompt Y, we get the same process; however, there is also an option to digitize headers and relevés separately. Notice when we are creating the second header, we can speed up the process by prompting RE, which will result in repeating values from the previous relevé, saving time. Let's add one relevé and two headers, resulting in a database with two relevés and three headers.

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) addrel
AddNewLayer?(Y/N) y
Select Layer (3,2,1,J,0) 1
P - percentage, BB - Braun B. scale, CS - custom scale p
AddSpecies?(GenuSpe/N) planmaj
Abundance?(%) 10
[1] FullName
<0 rows> (or 0-length row.names)
CorrectName?(Y/F(search for name)) f
SpeciesFirst3letters?(eg.Che) pla

6739    8150   PLAALAN                              Plantago lanceolata
6740    8151   PLAAMAJ                                   Plantago major
10827  13792   PLAAMAA                                   Plantago major

SpeciesName?(GenuSpe) plaamaj
            FullName
29732 Plantago major
CorrectSpecies?(Y/N)  y
      ShortName  1
29732 PLAAMAJ_1 10
AddSpecies?(GenuSpe/N) n
AddNewLayer?(Y/N) n
Species_richness
[1] 1
number of relevés:  2, number of headers:  1
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) addhead
RE <- 2023/01/05
DATE?(YYYY/MM/DD) re
RE <-
SPRINGDATE?(YYYY/MM/DD)
RE <- Yellowstone
LOCALITY? re
RE <- 1
FieldCODE? 2
...

YY <-
Note?
number of relevés:  2, number of headers:  2
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) addhead
RE <- 2023/01/05
DATE?(YYYY/MM/DD) re
RE <-
SPRINGDATE?(YYYY/MM/DD) re
RE <- Yellowstone
LOCALITY? re
RE <- 2
FieldCODE? 3
...
```

```
number of relevés:  2, number of headers:  3
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) printhead
    ShortName          X1          X2          X3
1          ID           1           2           3
2        DATE  2023/01/05  2023/01/05  2023/01/05
3  SpringDATE
4    LOCALITY Yellowstone Yellowstone Yellowstone
5   FieldCODE           1           2           3
6     Authors          PK          PK          PK
7    PlotSize         200         200         200
8    Latitude     4980364     4980364     4980364
9   Longitude   -110.5885   -110.5885   -110.5885
10   Accuracy          10          10          10
11        CRS       WGS84       WGS84       WGS84
12      Slope
13   Exposure          sw          sw          sw
14         E3          50          50          50
15         E2          38          38          38
16         E1          70          70          70
17       Ejuv
18         E0          12          12          12
19       Note
number of relevés:  2, number of headers:  3
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) printrel
      ShortName X1 X2
6656  PINUSYL_3 45  0
19520 SAMBNIG_2 38  0
26686 GALUAPA_1  8  0
29732 PLAAMAJ_1  0 10
32391 TRIFPRA_1 18  0
```

3.1.3 Species wise digitizing

The last option is to digitize relevés by species. However, for that and other prompts and functions, you are required to have a complete database with an equal number of species and relevés Let's add a third relevé and then digitize three more relevés by species.

```
number of relevés:  2, number of headers:  3
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) ysp
Number of relevés and headers must match!!!
number of relevés:  2, number of headers:  3
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) addrel
AddNewLayer?(Y/N) y
Select Layer (3,2,1,J,0) 1
P - percentage, BB - Braun B. scale, CS - custom scale p
..

number of relevés:  3, number of headers:  3
```

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) ysp
How many relev&#233;s? 3
P - percentage, BB - Braun B. scale, CS - custom scale p
Add new species (GenuSpe)/N?  trifpra
Select Layer (3,2,1,J,0) 1
                FullName
32391 Trifolium pratense
CorrectName?(Y/F(search for name)) y
Relevé 1
Abundance?(%) 20
      ShortName r1 r2 r3
32391 TRIFPRA_1 20  0  0
Relevé 2
Abundance?(%) 10
      ShortName r1 r2 r3
32391 TRIFPRA_1 20 10  0
Relevé 3
Abundance?(%) 5
      ShortName r1 r2 r3
32391 TRIFPRA_1 20 10  5
      ShortName  1  2 3  4  5 6
6656  PINUSYL_3 45  0 0  0  0 0
19520 SAMBNIG_2 38  0 0  0  0 0
...
Add new species (GenuSpe)/N?  plaamaj
Select Layer (3,2,1,J,0) 1
            FullName
29732 Plantago major
CorrectName?(Y/F(search for name)) y
Relevé 1
Abundance?(%) 0
[1] ShortName r1        r2        r3
<0 rows> (or 0-length row.names)
Relevé 2
Abundance?(%) 10
      ShortName r1 r2 r3
29732 PLAAMAJ_1  0 10  0
Relevé 3
Abundance?(%) 10
      ShortName r1 r2 r3
29732 PLAAMAJ_1  0 10 10
      ShortName  1  2 3  4  5  6
6656  PINUSYL_3 45  0 0  0  0  0
19520 SAMBNIG_2 38  0 0  0  0  0
...
Add new species (GenuSpe)/N?  pinusyl
Select Layer (3,2,1,J,0) 3
             FullName
6656 Pinus sylvestris
CorrectName?(Y/F(search for name)) y
Relevé 1

...
```

```
Add new species (GenuSpe)/N?  n
Add headers? y
Relevé 1
RE <- 2023/01/05
DATE?(YYYY/MM/DD) RE
RE <-
SPRINGDATE?(YYYY/MM/DD)
RE <- Yellowstone
LOCALITY? RE
RE <- 3
FieldCODE? 4
RE <- PK
Authors? RE
RE <- 200
PlotSize?(m2) RE
...
```

Our database now has six relevés with headers. Let's assume we are finished and we can close the software by prompting `N` or any other method.

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) n
```

3.1.4 Editing existing database, custom headers, custom checklist

Now we can try loading the database. We end up in the menu and are able to continue as before. Changing the `SAVE` parameter would create another copy of our database under a new name. Let's remove one relevé

```
addReleve(DATABASE = "First_database", SAVE = "First_database")
number of relevés:  6, number of headers:  6
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) REMOVEREL
ReleveNumber? 4
 [1] "4"           "2023/01/05"  ""            "Yellowstone" "4"           "PK"          "200"
"4980364"
 [9] "-110.5885"   "10"          "WGS84"       ""            "sw"          "50"          "38"
"70"
[17] ""            "12"          ""
CorrectNumber?(Y/N) y
number of relevés:  5, number of headers:  5
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) printhead
    ShortName          X1          X2          X3          X4          X5
1          ID           1           2           3           5           6
2        DATE  2023/01/05  2023/01/05  2023/01/05  2023/01/05  2023/01/05
```

Let's try changing parameters in the function. We will now not start with the default settings, we will add two more fields into the header and use a custom checklist. Notice we start in the menu with zero relevés; the name for *Plantago major* is not `PLAAMAJ` as it is in the original checklist but `PLANMAJ`. Lastly, notice the extra fields in the final `PRINTREL` prompt.

```
addReleve(SAVE = "Second_database",checklist = "test.txt",extrahead = c("soil","weather"),
start = FALSE)
number of relevés:  0, number of headers:  0
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) ysp
How many relev&#233;s? 2
P - percentage, BB - Braun B. scale, CS - custom scale p
...
```

```
Add new species (GenuSpe)/N?  plaamaj
Select Layer (3,2,1,J,0) 1
[1] FullName
<0 rows> (or 0-length row.names)
CorrectName?(Y/F(search for name)) f
SpeciesFirst3letters?(eg.Che) pla
     Number ShortName                                FullName
3688   3688   PLANL;S Plantago lanceolata var. sphaerostachya
3689   3689   PLANMAJ                           Plantago major
3690   3690   PLAN#MA                      Plantago major agg.
3691   3691   PLANMAO              Plantago major subsp. major
3692   3692   PLANMAR            Plantago major subsp. winteri
3693   3693   PLANMAI                        Plantago maritima
...
SpeciesName?(GenuSpe) planmaj
            FullName
15191 Plantago major
CorrectSpecies?(Y/N)  y
CorrectName?(Y/F(search for name)) y
Relevé 1
Abundance?(%) 20
      ShortName r1 r2
15191 PLANMAJ_1 20  0
Relevé 2
Abundance?(%) 10
...


      ShortName  1  2
3670  PINUSYL_3 40 70
15191 PLANMAJ_1 20 10
16877 TRIFPRA_1 10 10
Add new species (GenuSpe)/N?  n
Add headers? y
Relevé 1
RE <- DATE
DATE?(YYYY/MM/DD) 2023/05/06
RE <- SpringDATE
SPRINGDATE?(YYYY/MM/DD)
RE <- LOCALITY
LOCALITY? Prague
...

RE <- E1
E1?(%) 20
RE <- Ejuv
Ejuv?(%) 0
RE <- E0
E0?(%) 0
RE <- Note
Note?
RE <- soil
soil? sandy
RE <- weather
weather? sunny
Relevé 2
...
```

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD/ADDREL/ADDHEAD/YSP) printhead
    ShortName         X1         X2
1          ID          1          2
2        DATE 2023/05/06 2023/05/06
3  SpringDATE
4    LOCALITY     Prague     Prague
5   FieldCODE          1          2
6     Authors         JJ         JJ
7    PlotSize        100        100
8    Latitude
9   Longitude
10   Accuracy
11        CRS
12      Slope
13   Exposure
14         E3         40         40
15         E2          0          0
16         E1         20         20
17       Ejuv          0          0
18         E0          0          0
19       Note
20       soil      sandy      sandy
21    weather      sunny      sunny
```

3.2 RvegCombine

```
RvegCombine(database, export = "export", checklist = "default")
```

This function allows us to combine all species from different layers into one or to merge two different species in all relevés.

ARGUMENTS

- **database**: Path to the database we want to edit.
- **export**: Name of our edited database (save). If kept on default **export**, the file will be saved in the `tmp` directory.
- **checklist**: Used checklist.

DESCRIPTION

Let's imagine the following example: we have *Alnus glutinosa* present in the second relevé, both in the tree and shrub layers, but we decided to distinguish only the herb layer and tree layer in our database.

```
Combine?(LAYER/SPEC/PRINTREL/N) PRINTREL
      ShortName X1 X2 X3
291   ALNUGLU_3  0 60  0
11787 ALNUGLU_2  0 23  0
26564 FRAGVES_1 23  0  0
29789 POA RIP_1 20  0  0

Combine?(LAYER/SPEC/PRINTREL/N) LAYER
Which layer?(3/2/1/0/J) 2
To which layer?(3/2/1/0/J) 3
Combine?(LAYER/SPEC/PRINTREL/N) printrel
      ShortName X1 X2 X3
291   ALNUGLU_3  0 69  0
26564 FRAGVES_1 23  0  0
29789 POA RIP_1 20  0  0

Combine?(LAYER/SPEC/PRINTREL/N) ...
```

We received a combination of *Alnus glutinosa* from the shrub layer and tree layer, calculated by the following formula (taken from Tichý 2011, p. 29):

$$A + (B * (1 - (A/100)))$$

Where A and B are covers in percentage. The second possible use of this function is to merge two species together using the same formula. Let's combine *Fragaria vesca* with *Poa riphaea.*

```
Combine?(LAYER/SPEC/PRINTREL/N) SPEC
Which specie?(GenuSpe_L) FRAGVES_1
To which layer?(GenuSpe_L) POA RIP_1
Combine?(LAYER/SPEC/PRINTREL/N) PRINTREL
      ShortName X1 X2 X3
291   ALNUGLU_3  0 69  0
26686 GALUAPA_1  0  0 45
26768 GERAPUS_1  0 23  0
29789 POA RIP_1 38  0  0
29791 POA TRI_1 40  0  0
31388 SILELAT_1  0  0  3
32391 TRIFPRA_1  0 34  0
Combine?(LAYER/SPEC/PRINTREL/N) ...
```

Do not confuse this function with `RvegMerge()`.

3.3 RvegMerge

```
RvegMerge(x, y, save = "export_merge", head = TRUE)
```

ARGUMENTS

- **x** path to the first joining database
- **y** path to the second joining database
- **save** name of the exported database. If kept on default **export_merge**, the file will be saved in the `tmp` directory.
- **head** logical value, deciding if we want to join headers files as well

DESCRIPTION

This function allows you to join two databases together. It is useful when multiple people are working on the same project or when you have a large database and want to work with it in parts. When your database gets significantly bigger (e.g., 100 relevés), you might notice the software slows down a bit. If this reaches an unacceptable level, you can start a new database and merge them later with this function. It will automatically create new database files without any additional action needed. This function uses functions from the *dplyr* package (Wickham et al. 2022).

3.4 RvegCheck

```
RvegCheck(DATABASE, fullnames = FALSE, export = "export", checklist = "default")
```

ARGUMENTS

- **DATABASE** path to the database you want to check
- **fullnames** logical value if you want to include fullnames of species in the export
- **export** name of the exported table. If kept on default **export**, the file will be saved in the `tmp` directory.
- **checklist** checklist used to match shortnames with fullnames

DESCRIPTION

This function will be removed in later versions of the package as it will be efficiently replaced by `RvegLoad()`. It is able to check for duplicates, fix them, and will create and output full names for further analysis and presentations. As mentioned at the beginning of this manual, it is important to have a unique ShortName code for every species. However, some species might theoretically have more than one ShortName code in the checklist. If this happens and you find you used two different ShortName codes for one species, this function will detect the duplicate species and solve the issue for you by keeping the highest value in each relevé from both (in case you enter both ShortName codes in one relevé). Also, there is an option to include full names in the exported table; this might be useful when you want to browse data manually or present your data. However, be careful, as the table with full names will not be backward compatible with Rveg and thus uneditable in Rveg.

If it finds any duplicates, you can choose how to solve each one individually. Otherwise, it will just generate a new table without any additional action needed.

```
found duplicate codes for Stellaria nemorum
STEL#NE STELNEM
               fullName ShortName X1 X2 X3
10569 Stellaria nemorum STEL#NE_1 10 20  0
10570 Stellaria nemorum STELNEM_1  0  0 10
select merging method?(M - merge, N - none) M
This merging method will keep the higher value
select first row (with correct code) 10570
select second row 10569
```

Resulting in:

```
10570   Stellaria nemorum   STELNEM_1   10  20  10
```

3.5 RvegToJuice

```
RvegToJuice(Data, checklist = "default", export = "export")
```

ARGUMENTS

- **Data** path to the database you want to convert
- **checklist** checklist used to match ShortNames with FullNames
- **export** name of the exported table. If kept on default **export**, the file will be saved in the `tmp` directory.

DESCRIPTION

This function will convert the Rveg database format to the Juice-compatible format. Juice (Tichý 2002) is software for editing, classification, and analysis of large phytosociological tables and databases. After running the function, the table will be converted without any additional action needed. However, importing it might not be that intuitive. Let's go through a step-by-step guide on how to load the output of the function into Juice.

1. File -> Import -> Table -> from Spreadsheet File
2. Select the file
3. Press Next and select Comma as separator and use second column as layer information
4. Check if table size is correct; next, you will see a preview of the table
5. Select Percentage as Cover values
6. Finish

This way, the relevés will be imported into Juice; however, we might want to add headers to the relevés.

1. File -> Import -> Header Data -> From Comma Delimited File
2. Select the header file

3.6 TvToRveg

```
tvToRveg(tv, export = "export", checklist = "default",)
```

ARGUMENTS

- **tv** Path to the Turboveg csv export
- **checklist** Checklist used to match ShortNames with FullNames
- **export** Name of the exported table. If kept on default **export**, the files will be saved in the `tmp` directory.

DESCRIPTION

This function will convert CSV exports from Turboveg software into the Rveg-compatible format. Turboveg (Hennekens and Schaminée 2001) is software for the processing of phytosociological data. Since many members of our team used Turboveg, it influenced and inspired the creation of Rveg; you could say that Rveg is a simpler alternative to Turboveg. This function lets you edit or continue your work from Turboveg in Rveg. It will automatically create new database files without any additional action needed. The Turboveg CSV export has to include a header and relevés but should not include the Ellenberg indicator values. The possible cause of error might be wrong formating of CSV file, if you are having trouble check the example file at:

```
tvexport <- system.file("extdata", "tvexport.csv", package="Rveg")
read.csv(tvexport)
```

Guide for Turboveg export:

1. Select relevés
2. Export
3. Spreadsheet table
4. Select Format - Comma delimited
5. Select header data; don't select ecological data
6. Export

Then insert the path to the `tv` parameter of the function. The abundance can be either in percentage or Braun-Blanquet scale. Then you have to manually connect the species that were not recognized in the Rveg checklist; for example, *Salix fragilis* was not recognized in our data because the checklist uses the synonym *Salix euxina*. You can use a custom checklist as well. Also, you can check for each species what code is given to them according to the checklist, or you can skip this quickly by holding the enter button. When you know the correct ShortName code you can fill it right away.

```
Data in Bran blanquet or percentage? (B/%) B
[1] "Galium palustre"
[1] "GALUPAL_1"
correct name?(blank/N/GenuSpe)
[1] "Salix fragilis"
[1] NA
correct name?(blank/N/GenuSpe) N
AddSpeciesFirst3letters(eg.Che)? sal
Correct 3 letters? Y/N  y
7983    9633   SALXELE                                  Salix elaeagnos
7984    9636   SALXFRA                                     Salix euxina
7985    9639   SALXHAS                                    Salix hastata
SpeciesName?(7lettersGenuSpe) salxfra
          FullName
30976 Salix euxina
Correct 7lettersGenuSpe Y/N  Y
```

3.7 RvegToTv

```
RvegToTv(database, export = "export", ver = 3, checklist = "default")
```

ARGUMENTS

- **database** Path to the Rveg database to export
- **export** Name of the exported table / files. If kept on default **export**, the file will be saved in the `tmp` directory.
- **ver** Version of turvobeg to upload to, turboveg 3 can import one single file but turboveg 2 needs import of headers and relevés separately
- **checklist** Checklist used to match ShortNames with FullNames

DESCRIPTION

This function creates files importable to Turboveg 2 or Turboveg 3. The files will be created without any user input; however, uploading them to Turboveg might be challenging. Let's take a look at how to import such files into Turboveg 2. Two files with suffix R.csv (relevé) and H.csv (header) will be created.

1. In Turboveg, open or start a new database
2. Select Import
3. Select Free format species data table
4. Select the Relevé data
5. Now select Species column #1, Layer column #2, click Next
6. Species have been matched; now select percentage (%) as cover scale
7. Select all relevés, click Next, then Complete

This way, we have inserted the relevés into the Turboveg software; now we have to match header data.

1. Select Import
2. Select Free format header table
3. Select the header
4. Click Next; the header table should now appear visible
5. Select field names, field alias names in row 1; check 'Add new fields automatically' box
6. Press Next; ignore the warning messages. Most of the fields Turboveg cannot recognize and will be created as new fields
7. Select the first row 'ID' and change it to 'releve_nr' and mark it as key field (other fields are optional to match with Turboveg known fields)
8. Press Complete; headers should be matched

3.8 CreateChecklist

```
CreateChecklist(specieslist, export = "export")
```

ARGUMENTS

- **specieslist** Path to the list of species, which will be used to create checklist
- **export** Name and path to the export of the checklist. If kept on default **export**, the file will be saved in the `tmp` directory.

DESCRIPTION

This function allows you to form a checklist in Rveg-readable format from a character vector of species names. There are certain rules in the process, so we recommend editing full names according to these, which are in the original checklist. Therefore:

1. Hybrids name with 'x' (e.g., *Galium x pomeranicum* = GALU*PO); the shortname will contain '*' sign
2. Unrecognized species with suffix 'species' (e.g., *Corylus species* = CORL-SP)
3. Subspecies have a dash in their species name (e.g., *Primula veris ssp. veris* = PRIMV-V)
4. Aggregates of species have a hashtag in their species name (e.g., *Carex flava agg.* = CARE#FL)
5. Complicated genera might have sections starting with 'SE' (e.g., *Taraxacum sect. Taraxacum* = TARASEO)

We can show how the species list will be created from a few species.

```
Species <- c("Galium aparine","Galium x pomeranicum", "Galium species", "Primula veris",
"Primula veris ssp.  veris", "Carex flava agg.", "Taraxacum sect.  Taraxacum")
write(Species,"species.txt")
CreateChecklist(specieslist = "species", export = "species_checklist")
read.table("species_checklist.txt")
      V1        V2                         V3
1 Number ShortName                   FullName
2      1   GALIAPA             Galium aparine
3      2   GALI*PO       Galium x pomeranicum
4      3   GALI-SP             Galium species
5      4   PRIMVER              Primula veris
6      5   PRIMV-V  Primula veris ssp. veris
7      6   CARE#FL           Carex flava agg.
8      7   TARASET Taraxacum sect. Taraxacum
```

The checklist then will be usable in `addReleve()`.

3.9 RvegLoad

```
RvegLoad(DATABASE = "default",CustomScale = FALSE,checklist = "default")
```

ARGUMENTS

- **DATABASE** Path to your database. Default option **default** will load example database.
- **CustomScale** Boolean if custom not precoded scale was used during the digitizing
- **checklist** TXT checklist used during creation of the database

DESCRIPTION

This function loads the Rveg database into the R environment and is ready for analysis. Also, when custom scales were used during the digitizing, here it will allow you to transfer to percentage values. Let's use our previously created database, where we used percentages, but we will pretend it was a cover scale from 0-60. The output is a `data.frame` in R.

```
RvegLoad(CustomScale = TRUE)
Percentage value for (45) 90
Percentage value for (38) 76
Percentage value for (8) 16
Percentage value for (0) 0
Percentage value for (18) 36
Percentage value for (10) 20
Percentage value for (2) 4
Percentage value for (4) 8
Percentage value for (60) 100
Percentage value for (5) 10
       ShortName           fullName layer         X1         X2         X3         X4
1             ID                               1          2          3          5
2           DATE                      2023/01/05 2023/01/05 2023/01/05 2023/01/05
3     SpringDATE
4       LOCALITY                     Yellowstone Yellowstone Yellowstone Yellowstone
5      FieldCODE                               1          2          3          4
6        Authors                              PK         PK         PK         PK
7       PlotSize                             200        200        200        200
8       Latitude                         4980364    4980364    4980364    4980364
9      Longitude                       -110.5885  -110.5885  -110.5885  -110.5885
10      Accuracy                              10         10         10         10
11           CRS                           WGS84      WGS84      WGS84      WGS84
12         Slope
13      Exposure                              sw         sw         sw         sw
14            E3                              50         50         50         50
15            E2                              38         38         38         38
16            E1                              70         70         70         70
17          Ejuv
18            E0                              12         12         12         12
19          Note
6656     PINUSYL   Pinus sylvestris     3        90          0          0          0
19520    SAMBNIG     Sambucus nigra     2        76          0          0          0
26686    GALUAPA     Galium aparine     1        16          0          0          0
29732    PLAAMAJ     Plantago major     1         0         20          4         20
32391    TRIFPRA Trifolium pratense     1        36          0          8         20
```

4 DEVELOPER NOTES

Our goal is to create a tool for quick processing of phytosociological relevés without the need for additional software, especially when most of the statistical analysis and generation of plots happens in R anyway. The package is still in the first reviewed version, yet still in continuous development. We use it on a regular basis without problems and we want to share it with the public, but it can be expected that as the user base grows, so will the number of flaws and reported bugs. We will be more than happy for any improvement suggestions or bug reports. Contact us at:

- kralp@fzp.czu.cz
- or via GitHub system at https://github.com/sesitcsl2/Rveg

5 FAQ

6 REFERENCES

DANIHELKA J., J. CHRTEK a Z. KAPLAN, 2012. Checklist of vascular plants of the czech republic. Preslia. 84(3), 647–811. ISSN 00327786.

HENNEKENS S.M. a J.H.J. SCHAMINÉE, 2001. TURBOVEG, a comprehensive data base management system for vegetation data. Journal of Vegetation Science. 12(4), 589–591. ISSN 1100-9233. Available at: doi: 10.2307/3237010

TICHÝ L., 2002. JUICE, software for vegetation classification. Journal of Vegetation Science. 13(3), 451. ISSN 1100-9233. Available at: doi: 10.1658/1100-9233(2002)013[0451:jsfvc]2.0.co;2

TICHÝ L., J. HOLT a M. NEJEZCHLEBOVÁ, 2011. Juice Program Manual, 2nd edition. p. 29 Available at: https://www.sci.muni.cz/botany/juice/?idm=9

WICKHAM H., R. FRANCOIS, L HENRY, K MULLER, 2022. dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr