# Rveg User Manual

version 0.1.0

Přemysl Král and Jan Douda

## Contents

**1 INTRODUCTION** **2**
1.1 overview . . . . . 2

**2 FUNCTIONS** **3**
2.1 addReleve . . . . . 3
2.2 RvegCombine . . . . . 6
2.3 RvegMerge . . . . . 7
2.4 RvegCheck . . . . . 7
2.5 RvegToJuice . . . . . 8
2.6 TvToRveg . . . . . 8

**3 DEVELOPER NOTES** **9**

**4 REFERENCES** **10**

# 1 INTRODUCTION

Rveg package is database software for students and scientists, mainly botanists and ecologists, that allows transcription of your phytosociological relevés in R environment. This is a full-depth guide to understand how Rveg package work and its capabilities.

You are reading the manual for the first published version (0.1.0), but it is expected to keep developing the package in the future, as well as debugging. For the latest versions of Rveg, check one of the different localizations.

1. GitHub at https://github.com/sesitcsl2/Rveg - for the latest development version
2. team websites at https://plant-ecology-lab-czu.com/rveg/ - for the news
3. or CRAN repository - for the latest stable version

If you don't want to dig deeply into the manual, you can try the quick setup guide available on the GitHub page.

## 1.1 overview

Rveg is capable of fast transcription, editing, and viewing of your relevés. It creates a *database* composed of two csv files (separated by comma), one for header data (environmental variables, file ending with HEAD.csv) and 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.

**Header** file consist of preset variables in rows, and relevés in columns. I would advise not to use commas in values, as it is used as separator.

**Relevés** file consists of species number codes as row names, and relevés in columns. The first column represent species shortcut code combined by underscore with vegetation layer. It is based on the external checklist of species (Danihelka 2012), which assign shortcut code (E.g. *TRIFPRA* for *Trifolium pratense*) to each species to prevent typo mistakes and to speed up the process of writing. 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 all around working with these two files, which is referred to as the Rveg database. That is achieved primarily by addReleve function, but Rveg also contains some functions with extra features. Rveg contains these functions:

1. addReleve()
2. RvegCombine()
3. RvegCheck()
4. RvegToJuice()
5. RvegMerge()
6. tvToRveg()

To understand each function, view next chapter or run help("fun").

# 2 FUNCTIONS

## 2.1 addReleve

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

Most important function, which is base of the whole package. It will allow you to create and edit databases. You can create a database for your project with preset header data, but it also allows you to add your own characteristics. Thanks to the species checklist (Danihelka et al. 2012) for Czech Republic, provided within the package, Rveg will connect species names to the 7 letter shortcuts. If you ever find the checklist insufficient, you can create your own checklist, or modify current checklist (preferable). The structure of the checklist is a dataframe with 3 columns. First column *Number* represents 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. Second column *ShortName* is already mentioned species shortcut. It is written in uppercase and consists of 3 letters of genus and 4 letters of species. It is important to keep uppercase alphabet and keep every shortname unique present only once. To get to the checklist for modifying, you can run:

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

Last column *FullName* contains full species names. Columns are separated by tabulator. I recomend to edit the file in a text editor rather than in R, but be careful as some text editors can alter tabulator with spaces.

**ARGUMENTS**

*DATABASE* is set by default to "NEW", which indicates that user is creating new database. You will need to change this value only in case, 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 (not the REL.csv files).

*SAVE* set the name of your database. It is therefore only required argument when creating a new database. When you are editing an already existing database, you can select to keep *SAVE* same as *DATABASE*, which will rewrite loaded database, or you can choose different *SAVE* value to create new database as well as keep the old one (E.g. backup or versioning).

*checklist* is set by default to "default", which makes Rveg to use 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.

*extrahead* is set by default to NULL, meaning there are no extra header characteristics. Preset commonly used values are:

1. ID - generated automatically, unique value for each relevé
2. DATE - date of data sampling
3. SpringDATE -
4. LOCALITY - place where relevé was taken
5. FieldCODE - internal code for distinguishing relevés from same locality
6. Authors - name of the sampler
7. PlotSize - size of the sampled area
8. Latitude & Longitude - XY coordinates
9. Accuracy - accuracy of coordinates measurement
10. Slope
11. Exposure
12. E3, E2, E1, Ejuv, E0 - Percentage cover of each layer (tree, shrub, herb, juvenile and moss, respectively)
13. Note - any extra note you might need to record

Some characteristics might have recommended format, but it is completely up to you and your preferences, how you fill them. Everything is converted to string (text) format.

## USAGE

Run function

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

That will start dialogue communication with the user. Be careful to don't stop the dialogue in the process (if you don't want that) by pressing escape key for example. When creating a new database, it will start immediately recording your first relevé, starting with the header

```
DATE?(Y/M/D): 2023/1/5
SPRINGDATE?(Y/M/D):
LOCALITY?: Yellowstone
.
.
.
```

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 left the field blank (as we did with *SPRINGDATE*). After that 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: P
AddSpecies?(GenuSpe/N): ...
```

We started to write species in the first (herb) layer with covers noted in percentage. The second possible option is using modified Braun Blanquet scale (BB). Now we have to start filling species using 7 letters code, consisting of first 4 letters of genus and first 3 letters of species name. There are a few exceptions, for example genus *Poa* is written with 3 letters of genus, space, and 3 letters of species name. Or you might want to record aggregate, that is achieved with hashtag symbol, e.g. *ARTE#VU* stands for *Artemisia vulgaris agg.* There is no need to remember the codes, as the Rveg can show them to you, but after some time, you might start memorizing them automatically.

```
AddSpecies?(GenuSpe/N): TrifPra
Abundance?(%): 25
                FullName
32391 Trifolium pratense
CorrectName?(Y/F(search for name)) Y
      ShortName  0
32391 TRIFPRA_1 25
AddSpecies?(GenuSpe/N): ...
```

We added *Trifolium pratense* with a cover of 25%. This process should be repeated for every species in the first layer, and then again for all other layers you have sampled. What might happen is, the 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 than in the standard formula. This situation can look like this, while we try to add *Galium aparine*:

```
AddSpecies?(GenuSpe/N): Galiapa
Abundance?(%): 15
[1] FullName
<0 rows> (or  0-length row.names)
CorrectName?(Y/F(search for name)) F
SpeciesFirst3letters?(eg.Che) Gal
..
.
```

```
3708    4575    GALUPYC           Galium album ssp. pycnotrichum
3694    4552    GALUAPA                        Gallium aparine
3693    4551    GALU#AP                     Galium aparine agg.
.
..
SpeciesName?(GenuSpe) Galuapa
            FullName
26686 Galium aparine
CorrectSpecies?(Y/N) Y
      ShortName  0
26686 GALUAPA_1 15
32391 TRIFPRA_1 25
AddSpecies?(GenuSpe/N): ...
```

When Rveg asks if the species name is correct, but the software gives you a nonsense response (or wrong species name). It is then needed to not confirm by entering "F". Write 3 starting letters of Genus to get a full list of all species, starting with those letters, with their corresponding shortname code. You just have to scroll through the alphabetically sorted list to find desired species. After entering the correct code, Rveg recognize the species correctly.

In the case you accidentally enter wrong abundance or wrong species, you can immediately repair the error by rewriting the same species with correct abundance or with 0 abundance if it is not present. After you successfully record all species in the first relevé, you refuse to add more species.

```
AddSpecies?(GenuSpe/N): N
AddNewLayer?(Y/N): N
[1] "Species_richness"
[1] 2
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD): ...
```

You receive 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 you would start editing existing database before, you would find yourself here. To add more relevés, enter "Y" and repeat the same process as before.

Commands PRINTHEAD & PRINTREL will show you current database header and relevés, respectively. Let's say we have a database with 3 relevés.

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD): PRINTREL
      ShortName X1 X2 X3
7354  QUERROB_3  0 70  0
26564 FRAGVES_1  0 10  0
26686 GALUAPA_1 15  0  0
29791 POA TRI_1  0  0 70
32391 TRIFPRA_1 25  0  0
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD): ...
```

It is possible to edit relevés with RREL command

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD): RREL
ReleveNumber? 2
 [1] "2" ... #all header characteristics to check if the correct rel is selected
CorrecNumber?(Y/N) Y
      ShortName Cover
7354  QUERROB_3    70
26564 FRAGVES_1    10
AddNewLayer?(Y/N) ...
```

Similarly RHEAD allows editing of header characteristics, let's change shrub cover to 70%

```
AddReleve?(Y/N/RREL/RHEAD/REMOVEREL/PRINTREL/PRINTHEAD): RHEAD
ReleveNumber? 2
 [1] "2" "x" ... "50" ... #all header values
CorrecColumn?(Y/F) Y
RepairHeader?(Y/N) Y
[1] "ID"    "DATE" ... "E2" ...
HeaderCharacteristic? E2
NewValue? 70
RepairHeader?(Y/N) N
```

The last one, rarely used command is REMOVEREL. It removes chosen relevé with its header from the database.

## 2.2 RvegCombine

```
RvegCombine(database, export)
```

This function allows us to combine all species from different layers to 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)

**USAGE**

```
      ShortName X1 X2 X3
291   ALNUGLU_3  0 60  0
11787 ALNUGLU_2  0 23  0
26564 FRAGVES_1 23  0  0
26686 GALUAPA_1  0  0 45
```

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

```
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 shrub layer and tree layer, calculated by 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 with 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) ...
```

Don't mismatch this function with RvegMerge().

## 2.3 RvegMerge

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

This function allows you to join two databases together. It is useful when more 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 slowed down a bit. If this reaches an unacceptable level, you can start a new database and join them later on with this function. It will automatically create a new database files without any additional action needed. This function uses functions from package *dplyr* (Wickham et al. 2022).

**ARGUMENTS**

*x* path to the first joining database

*y* path to the second joining database

*save* name of the exported database

*head* logical value, deciding if we want to join headers files as well

## 2.4 RvegCheck

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

As is said at the beginning of this manual, it is important to have unique shortname code for every species. On the other side, some species theoretically might have more shortname codes in the checklist. If this happens and you find you used 2 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 u enter both shortname codes in one relevé). Also, there is option to include fullnames in the exported table, this might be useful when you want to browse data manually, or present your data. But be careful, as table with fullnames will not be backward compatible with Rveg and thus uneditable (in Rveg).

**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

*checklist* checklist used to match shortnames with fullnames

**USAGE**

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
```

## 2.5 RvegToJuice

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

This function will convert Rveg database format to the Juice-compatible format. Juice (Tichý 2002) is software for editing, classification and analysis of large phytosociological tables and databases.

**ARGUMENTS**

*Data* path to the database you want to convert

*checklist* checklist used to match shortnames with fullnames

*export* name of the exported table

After running the function, the table will be converted without any additional action needed.

## 2.6 TvToRveg

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

This function will convert csv export from Turboveg software into the Rveg compatible format. Turboveg (HENNEKENS et al. 2001) is software for the processing of phytosociological data. As many members from our team used Turboveg, it probably influenced and inspired the creation of Rveg, and you can 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 a new database files without any additional action needed. Turboveg csv export has to include a header and relevés, but do not include the Ellenberg indicator values.

**ARGUMENTS**

*tv* path to the Turboveg csv export

*checklist* checklist used to match shortnames with fullnames

*export* name of the exported table

# 3 DEVELOPER NOTES

Our goal is to create a tool for quick processing of phytosociological relevés without need for additional software, when most of the statistical analysis and generations of plot happens in R anyway. Package is still in the early development phase. We use it on a regular basis without problems and we want to share it with the public, but it can be expected 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

# 4 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