Table of Contents

Data types and import into R 1
Import from spreadsheet (Excel, Numbers) 1
Using *.txt (tab delimited) format 1
Using *.csv (comma separated values) format 2
Using clipboard 3
Import directly from Excel (*.xls or *.xlsx) file 3
Import *.RData file 4
Import from cc! (cep) format 4
Import from JUICE 5

Data types and import into R

Theory R functions **Examples** Exercise

Import from spreadsheet (Excel, Numbers)

This is a common situation - you have data in some spreadsheet software (eg Excel in Windows, or Numbers in Apple) and you want to upload the data into R. There are several ways how to do it. **I strongly recommend you to save the datasheet from Excel as a *.txt (preferably) or *.csv file** (choose *Save as* option, and in the Excel saving wizard in the field "Save as type" (below the name of the file) choose *Text (Tab delimited)(*.txt)* for plain text format where values are separated by tab character (recommended), or *CSV (Comma delimited)(*.csv)* for file where cells are separated by commas (or semicolon, depending on your language setting). Store the file in a chosen folder, **and load the data into R from this file**. Alternatively (but less optimally) you may copy the data spreadsheet from Excel to clipboard and load to R via the clipboard (however, this is not a reproducible way, since you cannot code it). Or, you may upload data directly from the Excel file stored on your computer (this may or may not work, depending on the version of Excel and R you are using).

When you **load the file from your computer** using some of the functions below, there are two ways how to do it:

- together with the name of the file, include also the absolute path to the file on your computer, using either forward slash (`/`), or double backslash (`\\`) to separate the folder names (e.g. "`c:/path/to/data/folder/data-for-import.txt`" or "`c:\\path\\to\\data\\folder\\data-for-import.txt`"), or
- change the R working directory to the folder in which the file is stored by `setwd("c:/path/to/data/folder")` function and then in the loading function you can use only the name of the file without absolute path (e.g. "`data-for-import.txt`").

As **example data**, we will use the species composition data from Forest vegetation data from Vltava river valley, namely data stored in the sheet *Vltava spe* in the excel file *vltava.xlsx*. The first row contains species names, the first column represents plot ID:

	Abiealb23	Acerpla23	Acerpse23	Alnuglu23	Alnuinc23	Berbvul23	...
1	0	0	0	0	0	0	...
2	0	0	0	0	0	0	...
3	0	0	0	0	0	0	...
4	0	0	0	0	0	0	...
5	0	0	0	40	0	0	...
...	...	...	...	...	...	...	...

Using *.txt (tab delimited) format

Use the file vltava-spe.txt, which is the plain text, with cells separated by tabulators:

```
    Abiealb23    Acerpla23    Acerpse23    Alnuglu23    Alnuinc23
```

```
Berbvul23    ...
1   0    0    0    0    0    0    ...
2   0    0    0    0    0    0    ...
3   0    0    0    0    0    0    ...
4   0    0    0    0    0    0    ...
5   0    0    0    40    0    0    ...
... ...    ...    ...    ...    ...    ...    ...
```

Save the file into some folder and specify the address to that folder in the `file` argument, for example (if the file is saved to root, e.g `c:`):

```
veg.data <- read.delim (file = 'c:/vltava-spe.txt', row.names = 1)
```

In address to the folder, use either `/`, or `\\`. Another option is to download the file into a working directory (you can find out which one is it by typing `getwd ()`). To read files from the current working directory, you don't need to specify the full address to the file.

The argument `file` can contain also the link to the location of file on internet:

```
veg.data <- read.delim
('https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/vltava-
spe.txt', row.names = 1)
```

Using *.csv (comma separated values) format

Use the file vltava-spe.csv, in which cells are delimited by a comma (,) and decimals separated dot (.):

```
,Abiealb23,Acerpla23,Acerpse23,Alnuglu23,Alnuinc23,Berbvul23, ...
1,0,0,0,0,0,0, ...
2,0,0,0,0,0,0, ...
3,0,0,0,0,0,0, ...
4,0,0,0,0,0,0, ...
5,0,0,0,40,0,0, ...
...
```

Note that *.csv format is a tricky one, since it follows different standards in different countries and on different platforms - in many countries (US, Asia, Western Europe and elsewhere) the cells are delimited by commas (,) and decimals are separated by dots (.), while in some other countries (France, Germany, Central and Eastern part of Europe) the cells are separated by semicolon (;) and decimals by comma (,).

```
env.data <- read.csv (file = 'c:/vltava-env.csv', head = T, row.names = 1)
```

In case the *.csv file has cells separated by the semicolon and decimals by the comma, use the function `read.csv2` tailored for this situation. Alternatively, the general `read.table` function with arguments `sep` for cell separators and `dec` for decimal separators can be used.

Using clipboard

Use the file vltava.xlsx, select the sheet *Vltava spe*, and copy the table into clipboard (take care to copy really just cells with data, not empty cells beside or below). Then use:

```
veg.data <- read.delim (file = "clipboard", head = T, row.names = 1)
```

The function `read.delim` expects that input is plain text delimited by tabulators (being a specific variant of more general function `read.table`). The `file = "clipboard"` works in Windows; Mac OS does not have clipboard, and the workaround is to use `file = pipe("pbpaste")` (see e.g. here).

Import directly from Excel (*.xls or *.xlsx) file

Importing data directly from Excel used to be quite complicated (see e.g. this website), but the package `readxl` made it much easier. Still, I do not suggest you use this option, since it may not be replicable on every platform, and it may change with a newer version of Excel.

The use of the package `readxl` is pretty straightforward. There is a suite of functions for reading Excel file, like `read_excel`, `read_xls` and `read_xlsx`. Important arguments are `path` and `sheet`, first for the name of the file (optinally including the path to the folder) and the second the name of the sheet which should be imported. We may try it on the Excel file vltava.xlsx - save it somewhere on your computer, and use:

```
# install.packages ('readxl')
library (readxl)
veg.data.0 <- read_excel ('c:/path/to/data/folder/vltava.xlsx', sheet =
'Vltava spe')
```

The object `veg.data.0` created in R is not data.frame, but a `tibble` (alternative to data.frame in the `tidyverse` packages). If you don't like that (or you don't know how to use it), you can simply convert it into data.frame using `as.data.frame` function. If the first column of the data are in fact row names, they may need to be assigned as such:

```
veg.data <- as.data.frame (veg.data.0[,-1])
rownames (veg.data) <- as.matrix (veg.data.0[,1])
```

Note that the functions from package `readxl` cannot read Excel files directly from internet, unlike e.g. `read.table`. But there is a workaround - first, download the Excel file into R as a temporary file, and the read it using `readxl` function:

```
url <-
'https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/vltava.x
lsx'
destfile <- tempfile ()
download.file(url, destfile, mode = 'wb')
veg.data.0 <- readxl::read_excel(destfile, sheet = 'Vltava spe')
```

Note that in the function `download.file` it is important to specify the argument `mode = 'wb'` (on Windows, if the argument `mode` is not set up, the type of the file will be determined from the file extension; in case of *.xls and *.xlsx files R would attempt to download these files as plain text, but in fact these files need to be downloaded as binaries).

The library `readxl` is a part of the `tidyverse` packages, and as such it does not use standard `data.frame` format for data frames, but unique `tibble` (as discussed above). One feature of `tibble` is that **it does not have rownames, and rownames are therefore imported as the first column of the data frame**. This may or may not be handy for future analysis since most of the functions we will use for numerical analysis here use data in standard `data.frame` format with rownames indicating plot IDs. To convert `tibble` into standard `data.frame`, use the function `as.data.frame`, and move the first column into the rownames of the newly created data frame (see above). Alternatively, there are functions like `column_to_rownames` in the package `tibble` which can help you with that.

Import *.RData file

Binary data storing the R object can be loaded into R using function `load`. Download data vltava.RData to your computer and use:

```
load ('c:/path/to/data/folder/vltava.RData')
```

This should create the variable `vltava` in your Global environment (it will appear among variables), which is alist of variables (see details here). Do not assign the result to a new variable, it will not work (if you use `vltava.spe <- load ('vltava.RData')`, the object `vltava.spe` will be assigned the character string `vltava`, not the data).

Alternatively, `load` function can read directly data from website, if the URL link is wrapped by function `url`:

```
load (url
('https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/vltava.
RData'))
```

Import from cc! (cep) format

Format `cc!` (or `cep`, Cornell Ecology Program) was used by M.O.Hill to code large matrices of abundance-based data for import into his popular DECORANA and TWINSPAN programs (written in Fortran). Later it became also default import format for data into CANOCO, where it was used until the version 4.5 [1]. Also other programs, like PC-ORD, were or still are using this format. From spreadsheet format, `cc!` file was converted using e.g. *CanoImp* software, wrapped with *Canoco* installation. The format `cc!` is also one way how to export species × sample data from *Turboveg for Windows 2* (used by vegetation scientists).

In R, import of `cc!` file is done using function `read.cep`, available in `vegan` package, or alternatively

also `read.CEP` from package `rioja`[2]. As an example, download the file vltava.cep, place it into the working directory[3] and use the following script:

```
library (vegan)
veg.data <- read.cep (file = 'vltava.cep', force = T)
```

The argument `force = T` is necessary to indicate that you are aware of the risk that the procedure can suddenly shut down and that you have saved all your data in advance. For larger matrices (with more than 10.000 cells) it's necessary to set up also an argument `maxdata`, indicating the maximum possible number of zero values (if you know the size of the imported matrix, this value could be calculated as the **number of rows × number of columns**).

CEP format comes in three versions: the original strict *condensed* version, somewhat relaxed *full condensed* and *free* format (see `?read.cep` in `vegan` to learn details). Function `read.cep` from `vegan` can read all three formats, while `read.CEP` from `rioja` can read *condensed* and *full* (at least according to `?read.CEP` in `rioja`).

Species names in *.cep format have only up to 8 characters. This was originally a technical limitation imposed by Fortran, but still, these short names are used e.g. while projecting species onto ordination diagrams (to reduce names overlap). If you need to create such names from original long species names, consider using `make.cepnames` function from `vegan`. When exporting matrix data into *.cep format using `write.CEP` from `rioja`, this function applies `make.cepnames` on longer data to concatenate them.

Import from JUICE

JUICE is a program used by vegetation ecologists for editing vegetation data. To export data from *JUICE*, the simplest is to use the *JUICE-R function*. In JUICE, open the file with your data you want to export, and in JUICE menu go to *File* > *Export* > *Table* > *To R Project txt File* (or simply press **CTRL+W**). Sample × species matrix will be exported as a file `table.txt` separated by tabs, and you can find it in the folder of Rgui.exe file, which is currently associated with JUICE[4]. To read it into R, use e.g.

```
JUICE.table <- read.table ('table.txt', sep = '\t', check.names = F, head = T, row.names = 1)
```

The argument `check.names = F` has two reasons: 1) the function ignores duplicate entries of variables (i.e. species - the sample × species table can contain several species of the same name and even of the same vegetation layer, although this itself is probably not correct), and 2) the function doesn't attempt to modify the names of the species so as they are syntactically correct (otherwise, e.g., the space in the name would be replaced by a dot). For more details about the file format, consult JUICE-R website.

1)
Recently introduced *Canoco 5* can handle directly spreadsheet data from Excel, so there is no need for cc! files. However, for compatibility with older versions, cc! file is still supported.

2)
To *export* matrix of community data into cc! file, consider using function `write.CEP` from package `rioja`.

3)

To figure out current working directory, use the function `getwd()`.

4)

You may check out which folder is it; in JUICE, go to *File* > *Options* > *External Program Paths* and check the folder address of R-PROJECT. If none is specified, you need to specify it first before you start to export the data.