Importing data into R

Importing data from spreadsheet (*.txt, *.csv, Excel file)

You have data in some spreadsheet-type software (eg Excel in Windows, or Numbers in macOS), and you want to upload the data into R. There are several ways to do it. The standard way is to save the datasheet from Excel into a *.txt file, store the file in a chosen folder, and load the data into R from this file. Saving a dataset from Excel into a *.csv file is also an option (but beware that the formatting of the file differs according to different language settings of your computer). You can also copy the data from an Excel spreadsheet onto the clipboard and load it to R via the clipboard, but this is not a reproducible way, since you cannot exactly code which data are to be used (and next time you may simply load a different chunkof data). 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). If data are available online (e.g. shared as a Dropbox/One Drive link, or stored on some website, like GitHub), you can import them directly from the online source. If data can be made public, you can create an online copy as a Gist on GitHub (check options here for how to make it).

When you **load the file from your computer** using some of the functions below, there are two ways 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 function used to load the data, use only the name of the file without an absolute path (e.g. "`data-for-import.txt`").

Example data. The dataset we will use here as an example (`data-for-import.*` files) looks like this:

rownames	x	y
P001	4.925152656	1.874884168
P002	4.915280266	1.874884168
P003	4.890599291	1.889368528
P004	4.836301144	1.913509128
P005	4.831364949	1.918337248
P006	4.796811584	1.952134088
...	...	...

Using *.txt (tab-delimited) format

Use the file data-for-import.txt, which is plain text, with cells separated by tabs (Tab keys, tabulators, coded as `\t`). Save the file into a folder, and then specify the address to the folder in the `file` argument, for example (if the file is saved to folder "`c:/path/to/data/folder`")):

```
imported_data_1 <- read.delim (file = "c:/path/to/data/folder/data-for-import.txt",
row.names = 1)
```

The object `imported_data_1` is a data frame. Note that the function `read.delim` is derived from more general function `read.table` by setting some of the `read.table` arguments to different default values (`sep = '\t', header =` TRUE, meaning that the function expects the data have values separated by tab characters (`\t`) and the first row of the data will become header = names of variables). If the first column of the data should become row names (as in this case), include the argument `row.names = 1`.

The argument `file` can also be a URL link to the location of the file on the Internet. The file "data-for-import.txt" can be found on the Internet here: https://raw.githubusercontent.com/zdealveindy/recol/main/data/data-for-import.txt

```
imported_data_1 <- read.delim
("https://raw.githubusercontent.com/zdealveindy/recol/main/data/data-for-import.txt
", row.names = 1)
```

Note: **different countries may have different specifications of what decimal separator is used**; for English and Chinese, the decimal separator is a dot (`.`), while for some other countries (Germany, France, Czech, among others) it is a comma (`,`). This behaviour can be changed in the Regional Settings of your computer. If your setting is using a comma as a decimal separator, then *.txt file saved from Excel will most likely also contain commas. However, the `read.delim` (and also `read.table` or `readr::read_delim`) function by default expects the separator to be a dot (default setting of argument `dec = "."`). A simple solution is to use `read.delim2` with argument `dec = ","`; in this way, the comma will be interpreted as a decimal separator. Values with decimal commas imported into R without changing `dec` into comma will be imported as factors (you can easily check that by applying `summary` function on the imported data frame: if the column is numeric, the summary will return the basic statistics (mean, median, min and max), while if the column is a factor, it will return a list of individual factor values)

Using *.csv (comma-separated values) format

Use the file data-for-import.csv, in which cells are separated by commas (,) and decimal signs are represented by dots (.). **Note that *.csv format is a tricky one, since it follows different standards in different countries and on different platforms** - so while in Taiwan, western Europe and elsewhere the cells are delimited by commas (,) and decimals are separated by dots (.), in France, Czech and elsewhere the *.csv format has cells delimited by semicolons (;) and decimals by commas (,).

```
imported_data_2 <- read.csv (file = 'data-for-import.csv', row.names = 1)
```

Alternatively, you can use the general function `read.table` with setting appropriate arguments: `head = TRUE` means that the first row of data is taken as header (column names), `sep = ","` sets the delimiter to be a semicolon, and `dec = "."` sets decimal separator to be a dot (this is default of `read.table` function, not necessary to change).

If your *.csv file is following "semicolon/comma" standard instead of "comma/dot" one, you may use the function `read.csv2`, which expects separators to be a semicolon (;) and decimals to be a comma (,).

Using clipboard

Use the file data-for-import.xlsx, the sheet *pig*, and copy the table into the clipboard (take care to copy really just cells with data, not empty cells around - use of CTRL+A is thus not recommended). Then use:

```
imported-data_3 <- read.delim (file = 'clipboard', row.names = 1)
```

The function `read.delim` expects that the input (in this case, the one stored in the clipboard) is plain text delimited by tabulators.

Using the clipboard is a quick (but dirty) way of importing data into R; it is relatively fast, but a considerable drawback is that it is not reproducible - you cannot code it, instead, you need to always manually open the file, copy the table to the clipboard and upload into R (this is prone to errors if you want somebody to replicate the same thing).

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

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

```
# install.packages ('readxl')
library (readxl)
imported_data_4 <- read_excel ('c:/path/to/data/folder/data-for-import.xlsx', sheet
= 'pig')
```

The object `imported_data_4` created in R is not data.frame, but a `tibble` (alternative to data.frame in the `tidyverse` packages). If you're not satisfied with it (or unsure how to use it), you can easily convert it into `data.frame` using `as.data.frame` function. If the first column of the data is row names, they may need to be assigned as such:

```
imported_data_4a <- as.data.frame (imported_data_4[,-1], row.names =
imported_data_4[,1])
```

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

```
# install.packages ('readxl')
library (readxl)
url <-
'https://github.com/zdealveindy/recol/blob/main/data/data-for-import.xlsx?raw=true'
destfile <- tempfile ()
download.file (url, destfile, mode = 'wb')
imported_data_5 <- read_excel(destfile, sheet = 'pig')
```

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", it 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 the standard `data.frame` format for data frames, but a unique `tibble` (check `class (imported_data_5)`). 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. To convert `tibble` into a standard `data.frame`, use the function `as.data.frame`. Then, still, you need to move the first column into the rownames of the newly created data frame:

```
imported_data_5_df <- as.data.frame (imported_data_5)
rownames (imported_data_5_df) <- imported_data_5_df[,1]
imported_data_5_df <- imported_data_5_df[,-1]
```

or by one line:

```
imported_data_5_df <- as.data.frame (imported.data_5[,-1], row.names =
imported.data_5[,1])
```

Import *.RData file

Binary data storing the R object can be loaded into R using the function `load`. Download data data-for-import.rdata to your computer and use:

```
load ('data-for-import.RData')
```

This should create the variable `data_for_import` in your Global Environment (it will appear among variables). You should not assign imported data into a new variable (`imported_values <- load ('data-for-import.RData')` will not work).

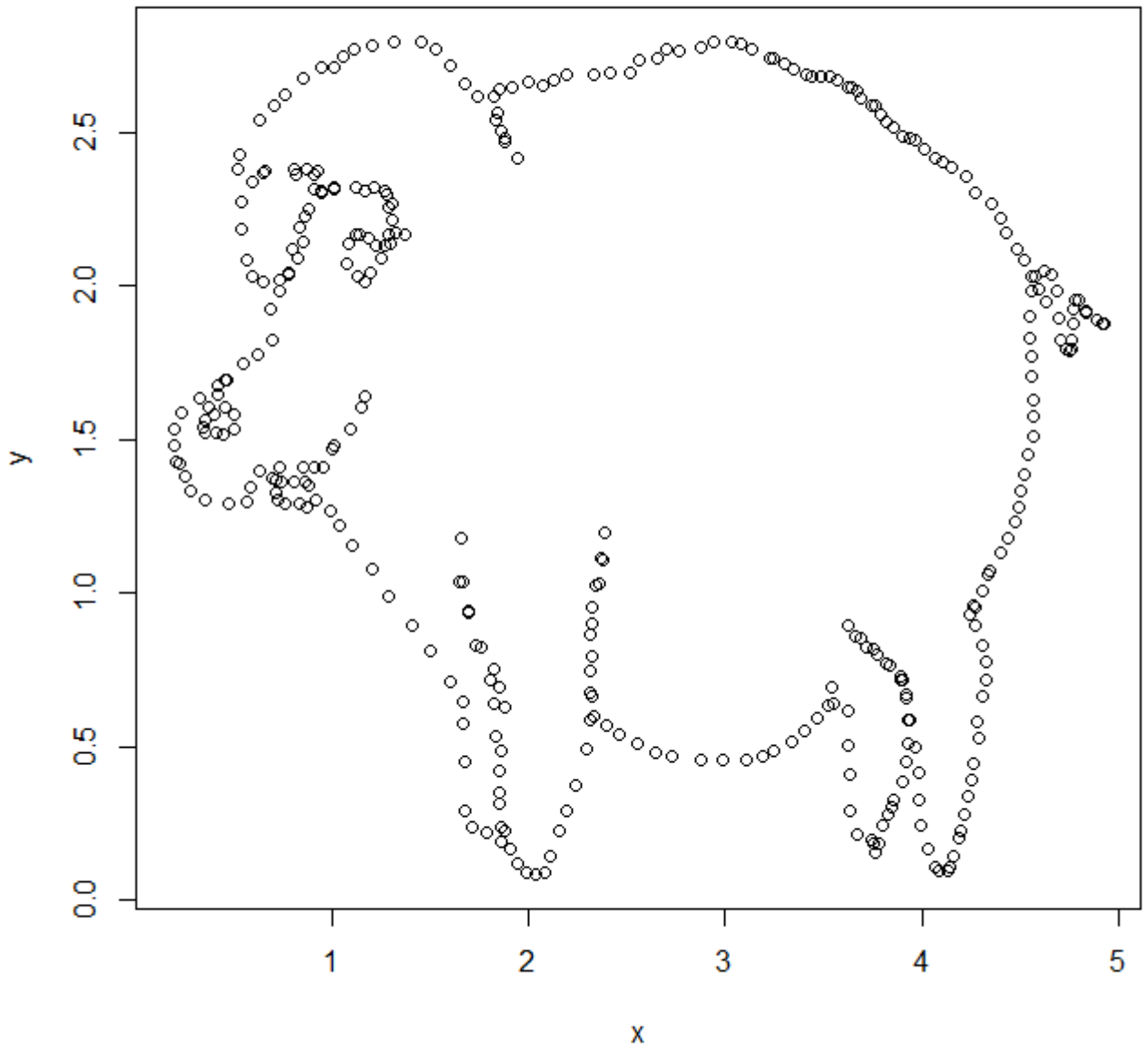
Alternatively, `load` function can read data directly from a website, if the URL link is wrapped by the function `url`:

```
load (url
('https://github.com/zdealveindy/recol/blob/main/data/data-for-import.rdata?raw=tru
e'))
```

And what are the data in the example data-for-import file?

It's actually really a pig!

```
load (url
('https://github.com/zdealveindy/recol/blob/main/data/data-for-import.rdata?raw=tru
e'))
plot (data_for_import)
```



The pig! (note that the colour I used to fill the polygon here is called Pig Pink :)

```
plot (data_for_import, type = 'o', pch = 16, col = 'red', frame.plot = F, axes = F)
polygon (data_for_import, pch = 16, col = '#FCD7DE99', border = NA)
```

