

R Introduction to data frames and packages

Working with data frames

Key Points:

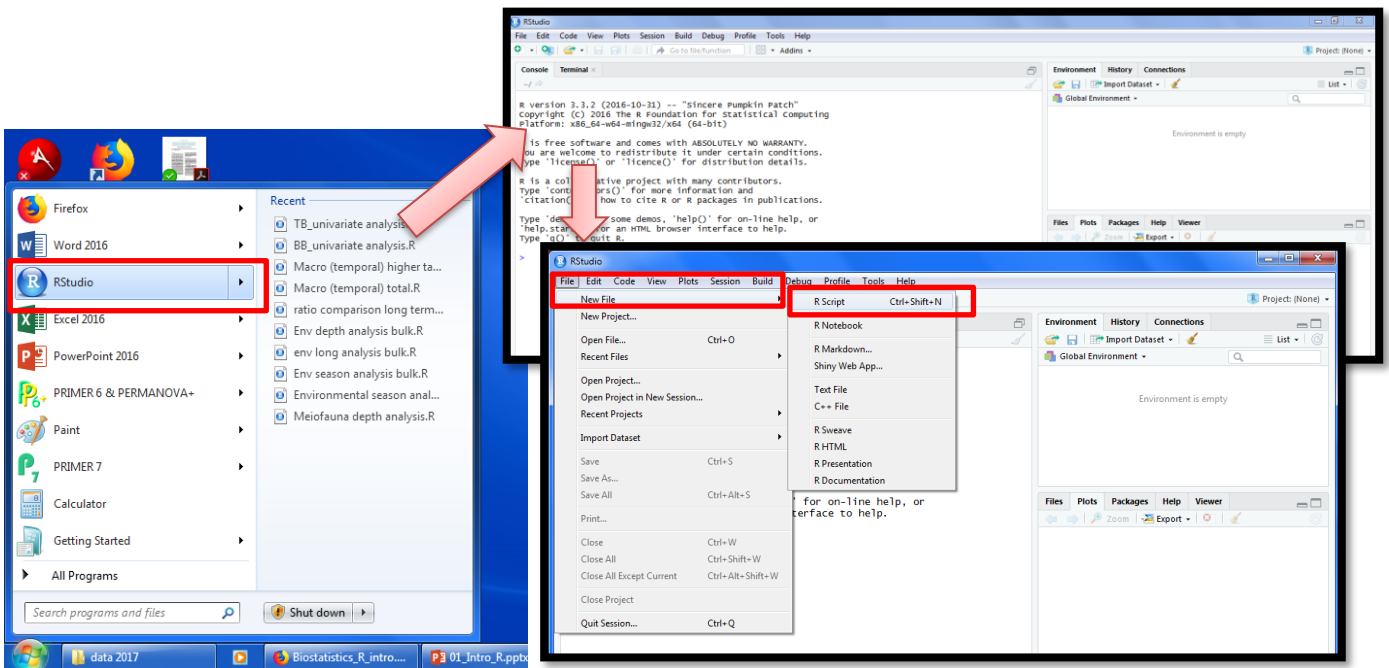
- R can only open datasets as “.txt” or “.csv” files
- R only recognizes “.” As a decimal sign
- The first row usually contains the column names/ names of the variables that you are investigating and testing
- Your column names cannot contain white spaces eg. “glucose_level” instead of “glucose level”

1. If you are working in excel, you can save your data into a text file as shown here:

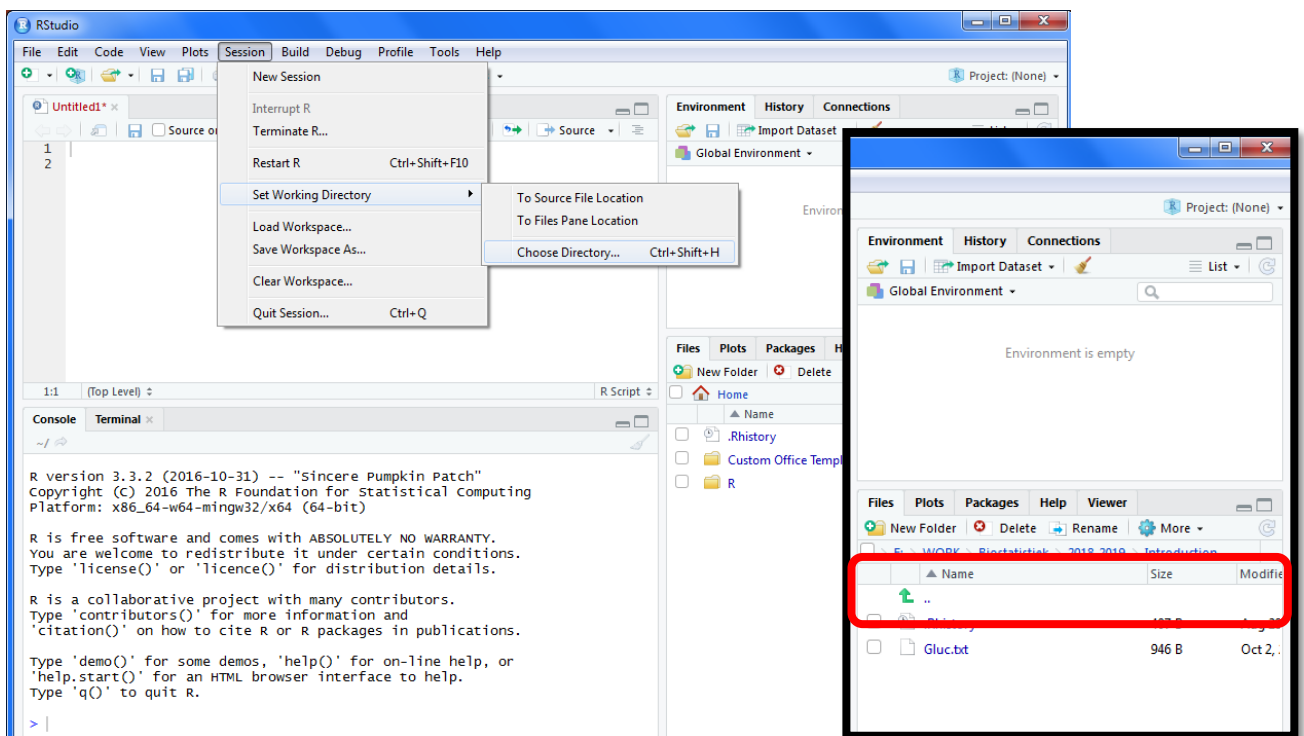
The image illustrates the steps to save an Excel spreadsheet as a text file. It shows the Excel spreadsheet with columns for 'farmac', 'bact', 'glucose', 'survival', and 'food'. The 'Save As' button in the ribbon is highlighted. The 'Save As' dialog box is open, showing the file name 'gluc.txt' and the 'Save as type' set to 'Text (Tab delimited) (*.txt)'. The resulting text file content is shown in a Notepad window, displaying the data from the spreadsheet in a tab-delimited format.

	farmac	bact	glucose	survival	food
1	farmac	bact	glucose	survival	food
2	sal	p	163	43	11
3	sal	p	157	37	23
4	sal	p	177	57	24
5	sal	p	139	19	12
6	sal	p	148	28	34
7	sal	p	144	24	32
8	sal	n	330	5	41
9	sal	n	302	2	23
10	sal	n	283	1	14
11	sal	n	273	9	12
12	sal	n	307	3	34
13	sal	n	279	2	32
14	adr	p	94	78	31
15	adr	p	109	50	42
16	adr	p	146	30	12
17	adr	p	141	25	23
18	adr	p	124	10	21
19	adr	p	114	2	24
20	adr	n	221	6	43
21	adr	n	200	9	23
22	adr	n	233	10	33
23	adr	n	180	12	11
24	adr	n	198	16	12
25	adr	n	213	6	12

2. To read a data frame, first create a new R-script within RStudio as shown here:



3. You can tell R where to find the data on your computer by setting the correct “working directory” as shown here and then browsing for the directory where you saved your files on your computer.



After choosing the directory, you will see your dataset on the right side within the “Files” folder.

4. Write the command line in the script to “read” your selected dataset as shown here. Make sure that your selected dataset is written exactly the same as the file you stored in the working directory including capitalization. So, if the file is stored as “Gluc.txt”, “gluc.txt” will give you an error message.

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Contains the R code:


```
1 gluc=read.table("gluc.txt", header=TRUE)
2 gluc
3
```
- Environment Pane:** Shows the 'Global Environment' with a 'Data' section containing the object 'gluc' with '24 obs. of 5 variables'. This pane is highlighted with a red box.
- Files Pane:** Shows the working directory 'F:\WORK\Biostatistiek\2018-2019\Introduction\' containing files like '.Rhistory' and 'Gluc.txt'.
- Console:** Shows the execution of the code:


```
> setwd("F:/WORK/Biostatistiek/2018-2019/Introduction/")
> gluc=read.table("Gluc.txt", header=TRUE)
> gluc
  pharm bact glucose survival food
1    sal    p    163      43    11
2    sal    p    157      37    23
3    sal    p    177      57    24
4    sal    p    139      19    22
5    sal    p    148      28    23
6    sal    p    144      28    23
7    sal    n    330      28    23
8    sal    n    302      28    23
9    sal    n    283      28    23
10   sal    n    273      28    23
11   sal    n    307      28    23
12   sal    n    279      28    23
13   adr    p     94      70    24
14   adr    p    109      50    24
15   adr    p    146      30    12
16   adr    p    141      25    23
17   adr    p    124      10    21
18   adr    p    114       2    24
```

Annotations in blue boxes:

- "Your selected dataset." points to the file name 'gluc.txt' in the script.
- "This expression confirms that the first row in the dataset contains column names" points to the 'header=TRUE' argument.
- "This is your 'working environment' which now contains your dataset." points to the Environment pane.
- "If you run the command, R will read in the dataset as shown here & will store it in your 'working environment'" points to the console output.

5. Once you have read in your file, you can start writing commands (“code”) to perform several actions on your data.

Here are some examples of things you might want to do with your data

- a. You can select specific parts of your dataset = subsets
- b. You can perform calculations such as finding the mean, the standard deviation or the standard error
- c. You can perform statistical analyses such as t-tests, ANOVA or regression analyses
- d. You can create a variety of graphs such as X-Y plots, barplots, boxplots or many others...

Packages

Many useful R functions come in packages, free libraries of code written by R's active user community. To install an R package, open an R session and type at the command line

```
install.packages("<the package's name>")
```

R will download the package from CRAN, so you'll need to be connected to the internet. Once you have a package installed, you can make its contents available to use in your current R session by running

```
library("<the package's name>")
```

You can also install packages through the "packages panel on the right side of your interface as shown here:

The screenshot shows the RStudio interface. The 'Packages' panel on the right lists installed and available packages. A dialog box titled 'Install Packages' is open, showing the 'Repository (CRAN, CRANextra)' dropdown, the 'Packages (separate multiple with space or comma):' field containing 'ggplot2', the 'Install to Library:' dropdown, and the 'Install dependencies' checkbox checked. The 'Install' button is circled in red. A blue callout box with an arrow points to the 'Install' button, containing the text: 'After installing the package, you also have to check the box with the functions you want on the right here'.

Here are some examples of useful packages for different purposes:

To manipulate data

[dplyr](#) - Essential shortcuts for subsetting, summarizing, rearranging, and joining together data sets. dplyr is our go to package for fast data manipulation.

[tidyr](#) - Tools for changing the layout of your data sets. Use the gather and spread functions to convert your data into the [tidy format](#), the layout R likes best.

[stringr](#) - Easy to learn tools for regular expressions and character strings.

[lubridate](#) - Tools that make working with dates and times easier.

To visualize data

[ggplot2](#) - R's famous package for making beautiful graphics. ggplot2 lets you use the [grammar of graphics](#) to build layered, customizable plots.

[ggvis](#) - Interactive, web based graphics built with the grammar of graphics.

[rgl](#) - Interactive 3D visualizations with R

[htmlwidgets](#) - A fast way to build interactive (javascript based) visualizations with R. Packages that implement htmlwidgets include:

- [leaflet](#) (maps)
- [dygraphs](#) (time series)
- [DT](#) (tables)
- [diagrammeR](#) (diagrams)
- [network3D](#) (network graphs)
- [threeJS](#) (3D scatterplots and globes).

[googleVis](#) - Let's you use Google Chart tools to visualize data in R. Google Chart tools used to be called Gapminder, the graphing software Hans Rosling made famous in his TED talk.

To model data

[car](#) - car's [Anova](#) function is popular for making type II and type III Anova tables.

[mgcv](#) - Generalized Additive Models

[lme4/nlme](#) - Linear and Non-linear mixed effects models

[randomForest](#) - Random forest methods from machine learning

[multcomp](#) - Tools for multiple comparison testing

[vcd](#) - Visualization tools and tests for categorical data

[glmnet](#) - Lasso and elastic-net regression methods with cross validation

[survival](#) - Tools for survival analysis

[caret](#) - Tools for training regression and classification models