Introduction to R

Import and export data

Jan-Philipp Kolb

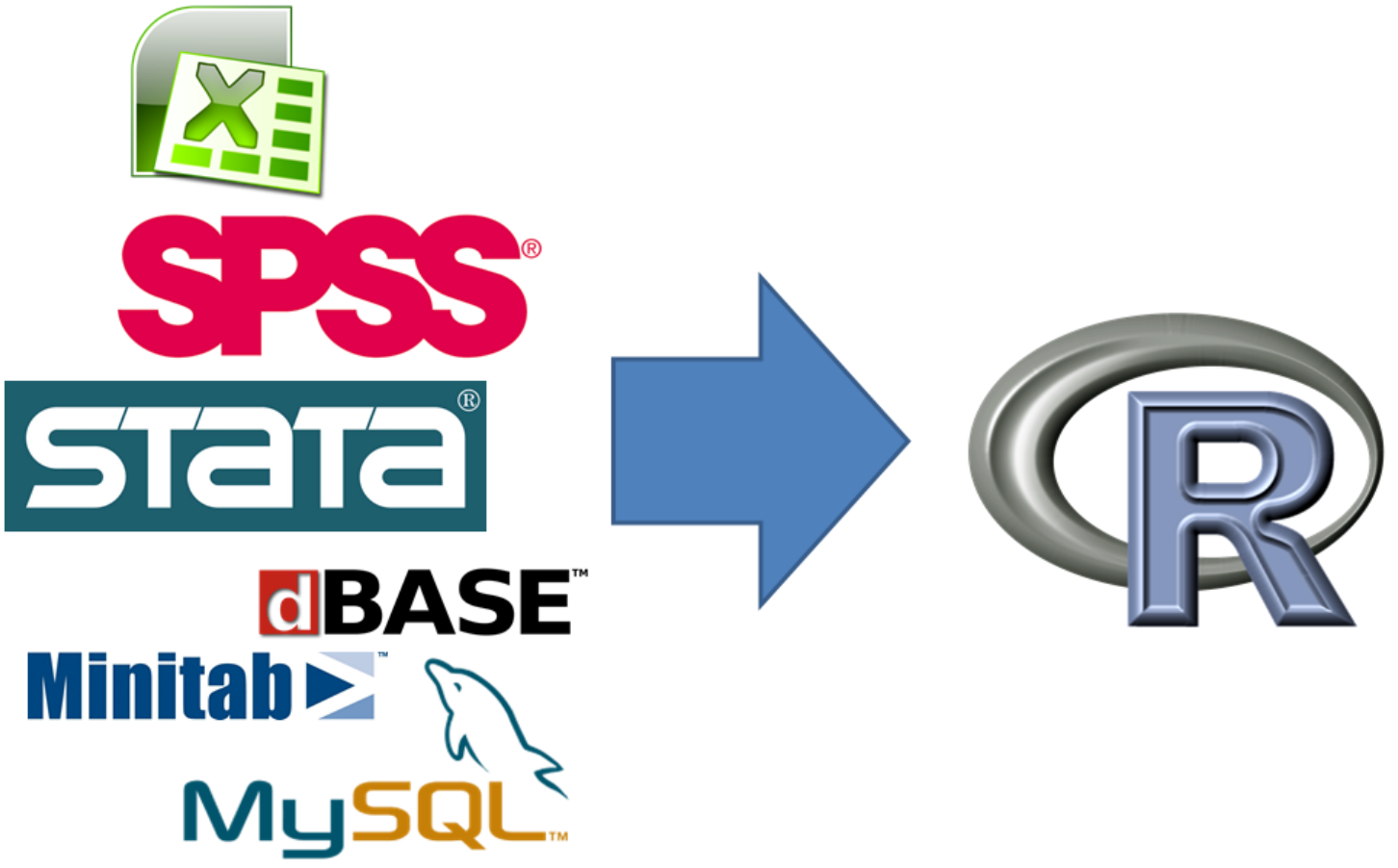
10 März, 2020

FIRST THINGS TO DO

Don’t try to kiss your data on the first date; rather, you just want to get to know the data:

1. **Import the data**
2. Review the codebook
3. Learn about the data
4. Quick visual understanding of the data

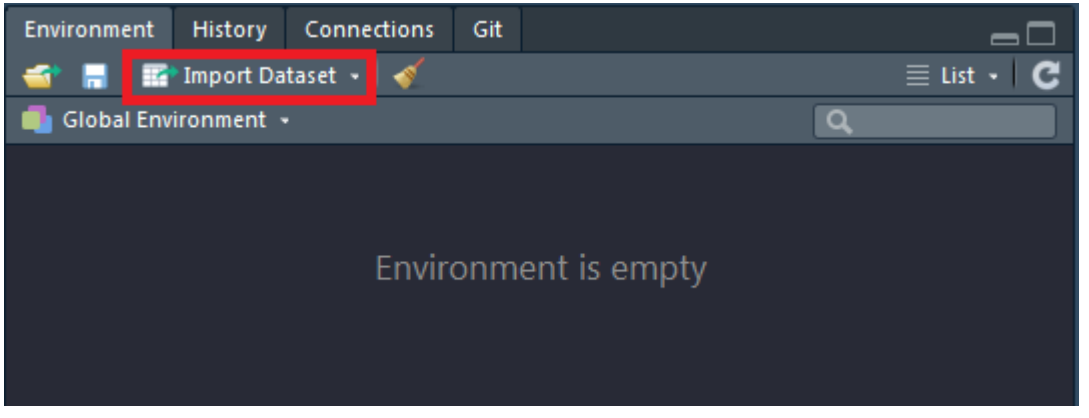
Data import



Import data with Rstudio

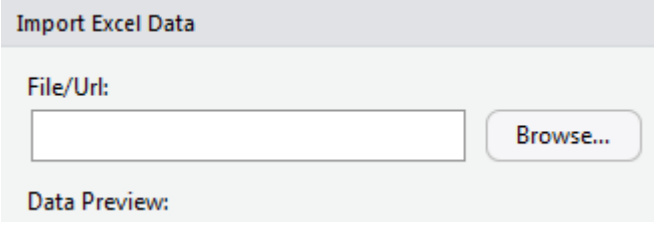
Rstudio functionality to import data

- Environment - Import Dataset - choose file type



Where to find data

Browse Button in RStudio

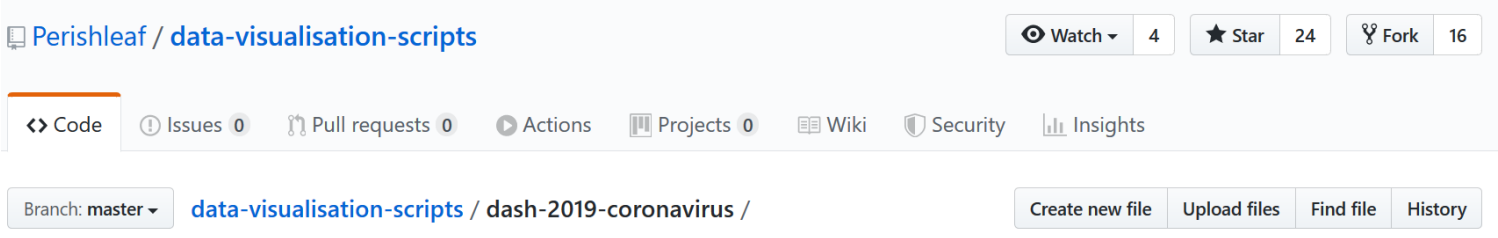


Code preview in Rstudio

- You can change the object name under Import Options
- Use short object names like e.g. `dat`

Exercise - download Corona data from github

- Go to the github repo of Perishleaf



- Click on the link for the data of 2020-01-21
- Click on the Raw button, you should see something like the following:

```
Province/State,Country/Region,Last Update,Confirmed,Deaths,Recovered,Suspected
Hubei,Mainland China,1/21/2020 00:00,270,6,,11.0
Guangdong,Mainland China,1/21/2020 00:00,17,0,,4.0
Beijing,Mainland China,1/21/2020 00:00,10,0,,
Shanghai,Mainland China,1/21/2020 00:00,9,0,,10.0
Zhejiang,Mainland China,1/21/2020 00:00,5,0,,16.0
Chongqing,Mainland China,1/21/2020 00:00,5,0,,
Sichuan,Mainland China,1/21/2020 00:00,2,0,,1.0
Tianjin,Mainland China,1/21/2020 00:00,2,0,,
Jiangxi,Mainland China,1/21/2020 00:00,2,0,,
```

- Right click with the mouse and *Save page as...*
- Remember the directory, we'll need it...

Exercise - change the working directory

- Use the Button ... on the right side to browse your computer
- Click on the More button
- Choose Set as working directory

Import of csv data

- `read.csv` is a command available in base package
- Excel data can be saved as `.csv` in Excel
- Then `read.csv()` can be used to read in the data.
- For German data, you may need `read.csv2()` because of the comma separation.

```
dat <- read.csv("2020-01-21-00-00.csv")
```

If it's German data:

```
datd <- read.csv2("ZA5666_v1-0-0.csv")
```

- you can use the Tab key

The result - a `data.frame`

- the following `data.frame` is a small excerpt from the data:

```
head(dat)
```

Country.Name	Country.Code	Year	Value
Arab World	ARB	1990	4235545
Arab World	ARB	1991	3811595
Arab World	ARB	1992	4000509
Arab World	ARB	1993	4189545
Arab World	ARB	1994	4352945
Arab World	ARB	1995	4337009

The package `readxl`

```
install.packages("readxl")
```

- **`readxl` has no external dependencies**
- `readxl` supports both the legacy `.xls` format and the modern xml-based `.xlsx` format.

```
library(readxl)
ab <- read_excel("../data/ma_stadtteile.xlsx")
head(ab)
```

```
## # A tibble: 6 x 16
##   query   lat   lon lat_min lat_max lon_min lon_max place_id
##   <chr> <dbl> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>    <dbl>
## 1 Inne…  49.5  8.45    49.5    49.5    8.44    8.48   2.36e8
## 2 Neck…  49.5  8.47    49.5    49.5    8.42    8.48   2.36e8
## 3 Neck…  49.5  8.48    49.5    49.5    8.47    8.51   2.36e8
## 4 Osts…  49.5  8.48    49.5    49.5    8.46    8.50   1.69e7
## 5 Schw…  49.5  8.48    49.5    49.5    8.47    8.50   2.36e8
## 6 Lind…  49.5  8.47    49.5    49.5    8.46    8.48   2.36e8
## # … with 8 more variables: osm_type <chr>, osm_id <dbl>,
## #   place_rank <dbl>, display_name <chr>, class <chr>,
```

Import SPSS files

Import SPSS data

- library `haven` - import and export 'SPSS', 'Stata' and 'SAS' files
- the result of this import command is a tibble

```
library(haven)
dataset <- read_sav("../data/datahub_government_africa.sav")
```

```
> dataset
# A tibble: 53 x 5
            Country   Government                    Name          since  Term
          <dbl+lbl>    <dbl+lbl>                <dbl+lbl>      <dbl+lbl> <dbl>
 1 16 [Equatorial ~ 3 [Presiden~ 51 [Teodoro Obiang Ng~ 31 [3 August ~    39
 2  8 [Cameroon]    3 [Presiden~ 45 [Paul Biya]          44 [6 Novembe~    36
 3 51 [Uganda]      3 [Presiden~ 53 [Yoweri Museveni]    29 [29 Januar~    32
 4 47 [Sudan]       3 [Presiden~ 42 [Omar al-Bashir]     33 [30 June 1~    29
 5 10 [Chad]        3 [Presiden~ 24 [Idriss DÃ©by]       10 [2 Decembe~    28
 6 17 [Eritrea]     3 [Presiden~ 25 [Isaias Afwerki]     18 [24 May 19~    25
 7 12 [Congo]       3 [Presiden~ 12 [Denis Sassou Ngue~ 23 [25 Octobe~    21
```

Import data from the web

Austrian microcensus

Files can also be imported directly from the Internet:

```
link <- "http://www.statistik.at/web_de/static/mz_2013_sds_-_datensat
Dat <- rio::import(link)
```

Import `stata` files

Import newer `.dta` files

- With `read.dta13` stata files from version 13 (and higher) can be imported

```
library(readstata13)
dat_stata <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta")
```

Import `stata` files - older versions

```
library(foreign)
dat_stata12 <- read.dta("../data/example_gp_stata12.dta")
```

The library `readstata13`

readstata13 {readstata13} R Documentation

Import Stata Data Files

Description

Function to read the Stata file format into a data.frame.

Note

If you catch a bug, please do not sue us, we do not have any money.

Author(s)

Marvin Garbuszus jan.garbuszus@ruhr-uni-bochum.de

Sebastian Jeworutzki sebastian.jeworutzki@ruhr-uni-bochum.de

See Also

`read.dta` and `memisc` for dta files from Stata Versions < 13

Import - GESIS Panel data

`convert.factors` argument

```
library(readstata13)
```

```
datf <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta",
                   convert.factors = F)
head(datf$bbzc007a)
```

```
## [1] 1 1 1 1 3 1
```

For comparison - import without this argument

```
dat <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta")
head(dat$bbzc007a)
```

```
## [1] Nein          Nein          Nein          Nein
## [5] Ja, manchmal Nein
## 10 Levels: Ambiguous answer ... Ja, oft
```

The argument `convert.factors = F`

More information on `.dta` import

```
?read.dta13
```

- `convert.factors` - logical. If `TRUE`, factors from Stata value labels are created.
- It might be useful to import the dataset twice - with and without value labels...
- `nonint.factors`- logical. If `TRUE`, factors labels will be assigned to variables of type float and double.
- The import must be controlled, because otherwise errors can easily happen.

Get stata attributes

```
att_dat <- attributes(dat)
names(att_dat)
```

```
##  [1] "row.names"        "names"            "datalabel"
##  [4] "time.stamp"       "formats"          "types"
##  [7] "val.labels"       "var.labels"       "version"
## [10] "label.table"      "expansion.fields" "byteorder"
## [13] "orig.dim"         "class"
```

Example: the variable names

```
head(att_dat$names)
```

```
## [1] "z000001z" "z000002z" "z000003z" "z000005z" "a11c019a"
## [6] "a11c020a"
```

Get an initial overview of the data

```
View(datf)
```

Filter

	z000001z Personen ID - Campus File	z000002z Studiennummer des Archivs	z000003z Versionskennung und -datum des Archivs	z000005z doi	a11c019a Zufriedenheit Leben in Wohnort	a11c020a Zufriedenheit
1	198431880	ZA5666	1-0-0 2017-06-20	10.4232/1.12749	1	
2	436122330	ZA5666	1-0-0 2017-06-20	10.4232/1.12749	1	
3	856844220	ZA5666	1-0-0 2017-06-20	10.4232/1.12749	2	
4	117346660	ZA5666	1-0-0 2017-06-20	10.4232/1.12749	1	
5	943433330	ZA5666	1-0-0 2017-06-20	10.4232/1.12749	1	

- You can get the same in RStudio if you click on the dataset icon in the environment menue

The library `rio`

```
install.packages("rio")
```

```
library("rio")
x <- import("../data/ZA5666_v1-0-0.csv")
y <- import("../data/ZA5666_v1-0-0_Stata12.dta")
z <- import("../data/ZA5666_v1-0-0_Stata14.dta")
```

- **rio: A Swiss-Army Knife for Data I/O**

The package `Hmisc`

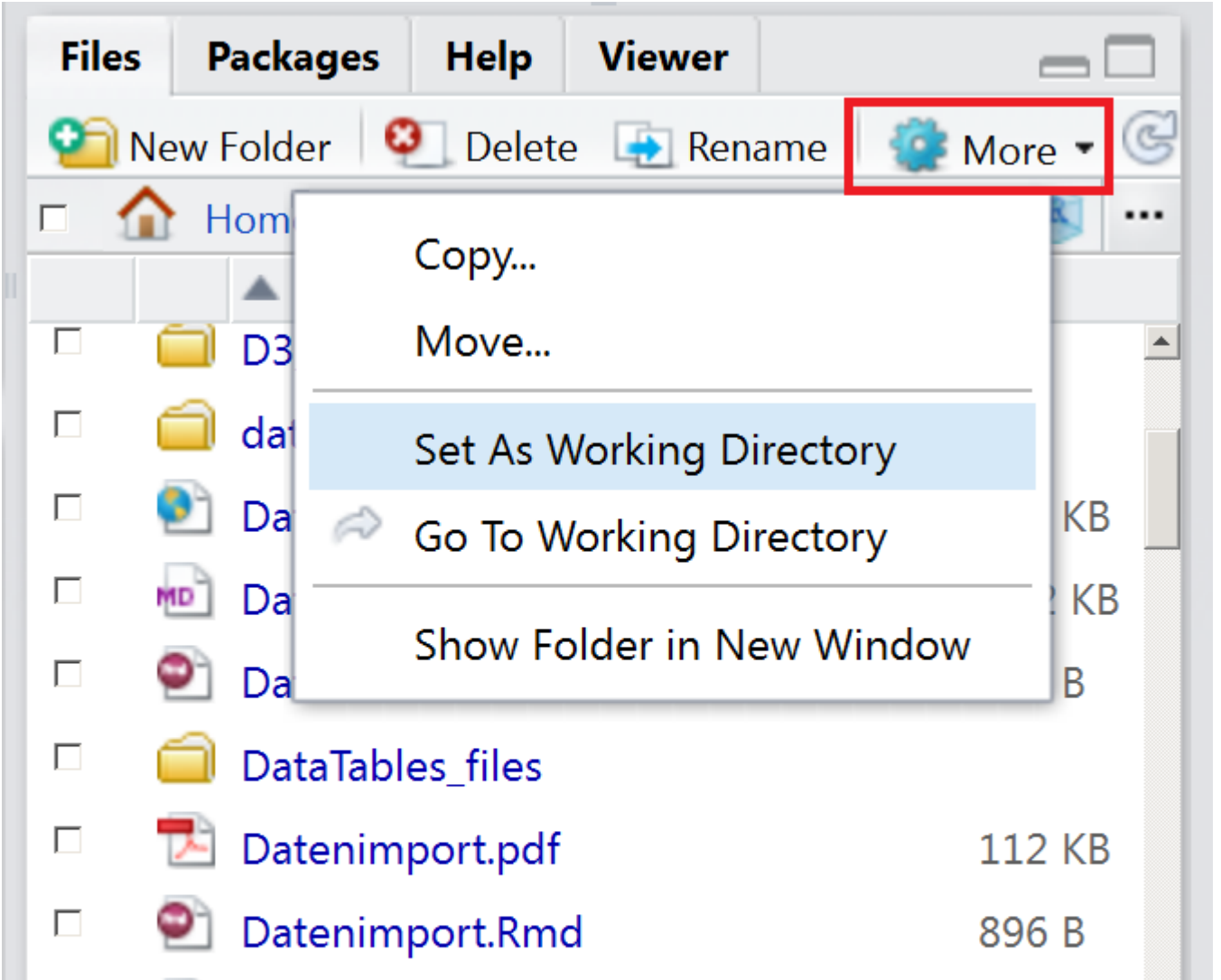
> For SPSS and SAS I would recommend the Hmisc package for ease and functionality.

```
library(Hmisc)
mydata <- spss.get("c:/mydata.por", use.value.labels=TRUE)
# last option converts value labels to R factors
```

Import SAS data

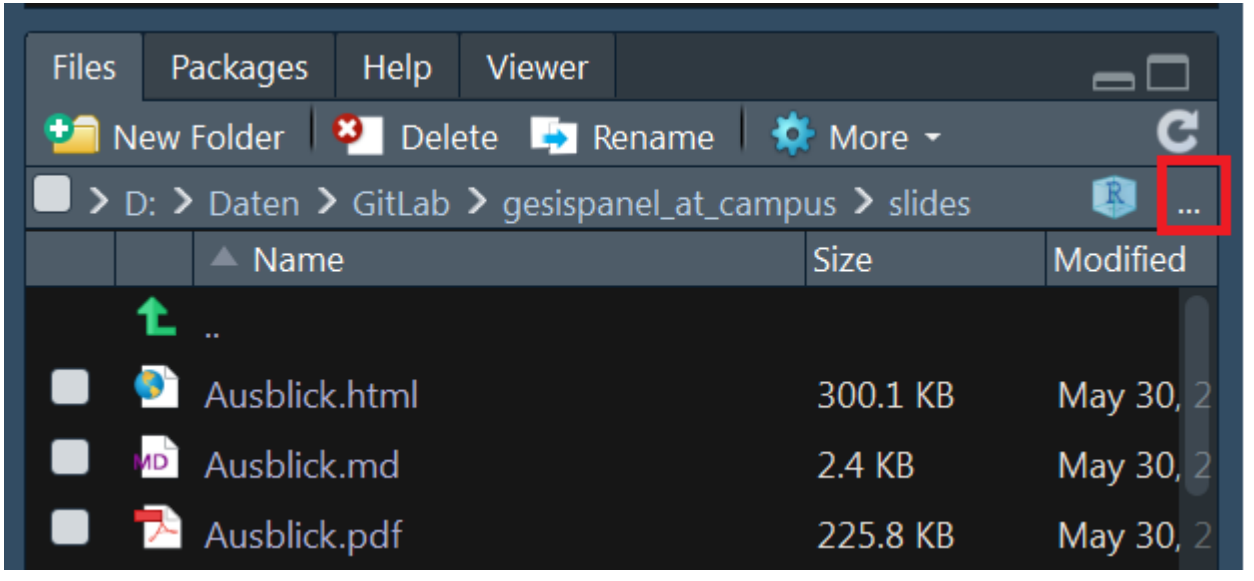
```
mydata <- sasxport.get("c:/mydata.xpt")
# character variables are converted to R factors
```

The working directory



...

- If the data is on a different drive in Windows



The working directory II

This way you can find out which directory you are currently in

```
getwd()
```

So you can change the working directory:

You create an object in which you save the path:

```
main.path <- "C:/" # Example for Windows
main.path <- "/users/Name/" # Example for Mac
main.path <- "/home/user/" # Example for Linux
```

And then change the path with `setwd()`

```
setwd(main.path)
```

On Windows it is important to use slashs instead of backslashes.

Change working directory

- You can also use the tab key to get the autocompletion.

```
getwd()
```

```
## [1] "E:/github/intror2020/slides"
```

```
setwd("..")
getwd()
```

```
## [1] "E:/github/intror2020"
```

Built-In datasets

- Often an example dataset is provided to show the functionality of a package
- These datasets can be loaded with the command `data`

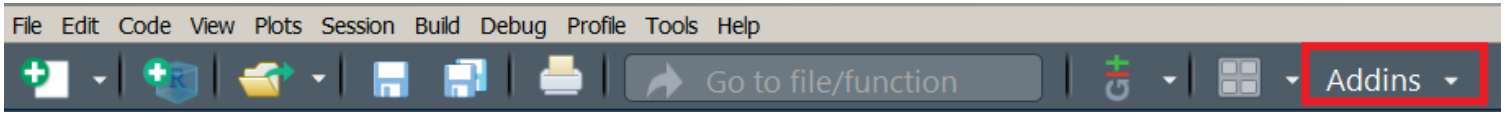
```
data(iris)
```

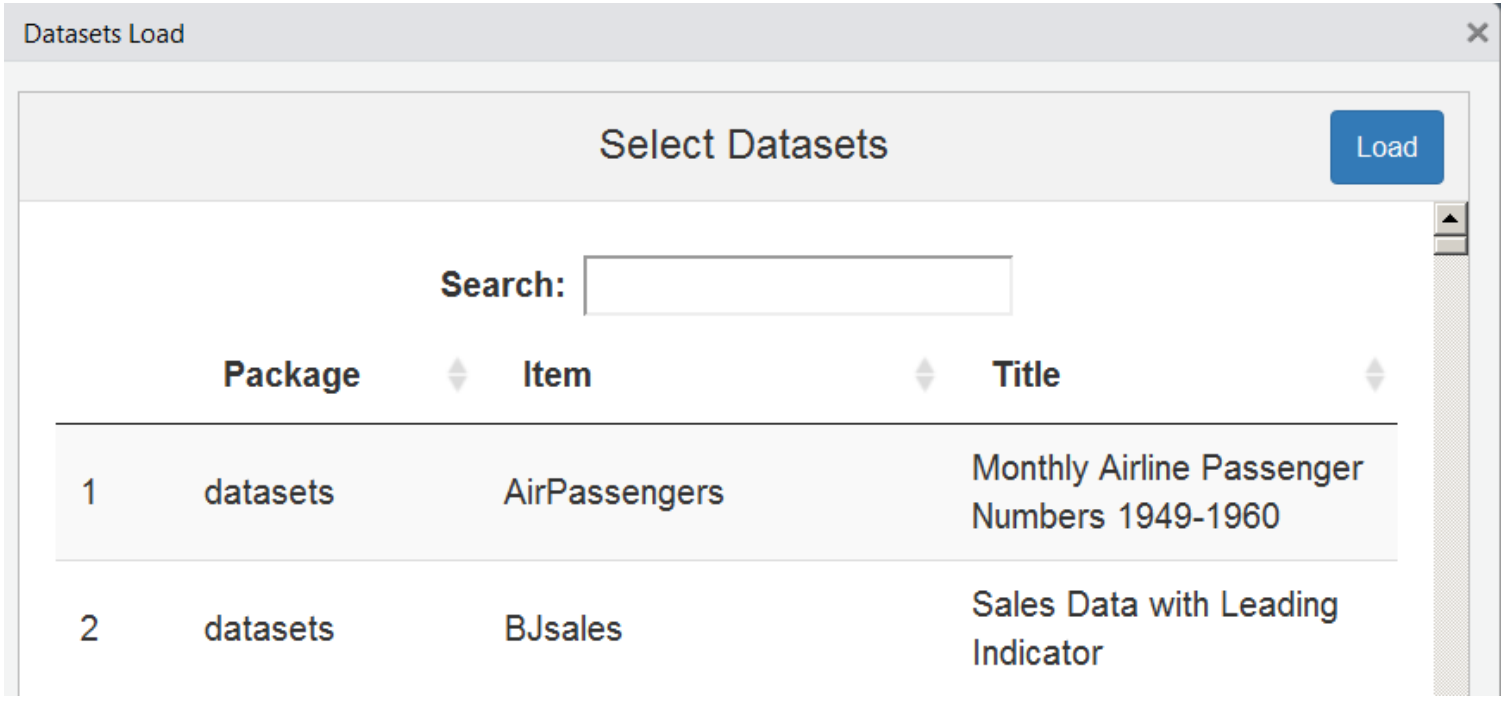
- There is also an **RStudio add-in** that helps to find a dataset

```
install.packages("datasets.load")
```

Excursus RStudio Addins

- In the upper right corner there is a button Addins





Exercise: load built-in data

Load the the built-in dataset `mtcars`

1) How many observations and variables are available?

2) What is the object structure of the variables?

Interactive data table

3) Create an interactive data table

Inserting data

- **RStudio addin for inserting data**

```
devtools::install_github("lbusett/insert_table")
```

The `file.choose` option

- You can browse through the directory with `file.choose`:

```
dat <- read.csv(file.choose())
```

- If you run the command line above a window is opened and you can browse in the file system.
- That also works with other import functions

Creating an example data record

```
A <- 1:4
B <- LETTERS[1:4]
C <- letters[1:4]
D <- runif(4)

mydata <- data.frame(A,B,C,D)
```

```
mydata
```

A	B	C	D
1	A	a	0.4801876
2	B	b	0.1887678
3	C	c	0.7733946
4	D	d	0.8233726

Overview data import/export

- if you continue working with R, `.RData` or **`rds`** format is the best choice:

```
save(mydata, file="mydata.RData")
saveRDS(mydata, "mydata.rds")
```

- The data set can be imported with `load`.

```
load("mydata.RData")
mydata <- readRDS("mydata.rds")
```

- `saveRDS()` doesn't save the both the object and its name it just saves a representation of the object

Overview import functions

Package	Function	.CSV	.TSV	.TXT	FIXED WIDTH	SPECIAL SEPARATOR
utils (Base R)	read.csv	x				
	read.delim		x			
	read.table			x		x
readr	read_csv	x				
	read_tsv		x			
	read_table			x	x	
	read_fwf				x	
	read_delim					x
data.table	fread	x	x	x	x	x

Export as Excel

- Create an example **tibble**:

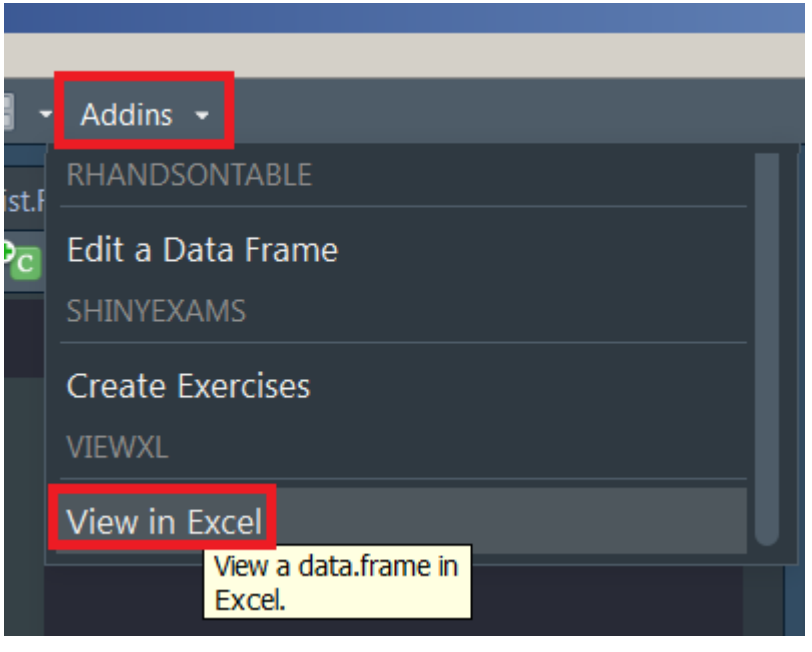
```
library(tibble)
ab <- tibble(a=1:4,b=4:1)
```

```
library(xlsx)
setwd("D:/Daten/GitLab/IntroDataAnalysis/data")
write.xlsx(ab,file="ab.xlsx")
```

Addin to open dataset in Excel

```
devtools::install_github("dreamRs/viewxl")
```

- select a `data.frame` in script -> it is opened in Excel.



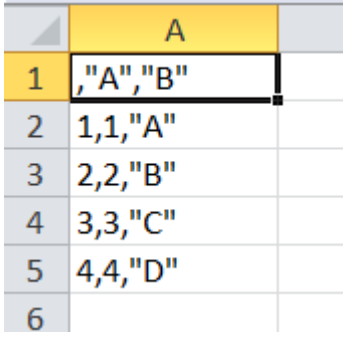
Save data in `.csv` format

```
write.csv(mydata,file="mydata.csv")
```

- If you want to continue working with German Excel, it is better to use `write.csv2`

```
write.csv2(mydata,file="mydata.csv")
```

- Otherwise, the result looks like this:



Argument `row.names`

Prevent row names to be written to file when using write.csv

```
write.csv(mydata,file="mydata.csv", row.names=FALSE)
```

- or for German data:

```
write.csv2(mydata,file="mydata.csv", row.names=FALSE)
```

The package `rio`

```
install.packages("rio")
```

Import, Export, and Convert Data Files

The idea behind **rio** is to simplify the process of importing data into R and exporting data from R. This process is, probably unnecessarily, extremely complex for beginning R users. Indeed, R supplies an entire manual describing the process of data import/export. And, despite all of that text, most of the packages described are (to varying degrees) out-of-date. Faster, simpler, packages with fewer dependencies have been created for many of the file types described in that document. **rio** aims to unify data I/O (importing and exporting) into two simple functions: `import()` and `export()` so that beginners (and experienced R users) never have to think twice (or even once) about the best way to read and write R data.

Save data as `.sav` (SPSS)

```
library("rio")
# create file to convert

export(mtcars, "data/mtcars.sav")
```

Exercise: Export dataset

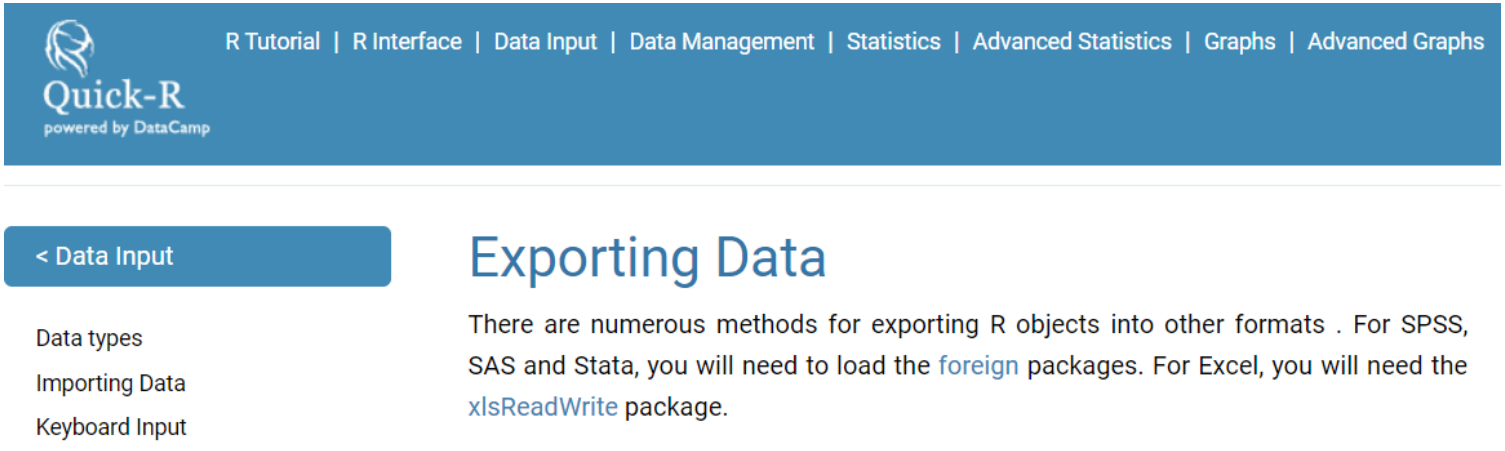
- Please load the `iris` built-in dataset
- Export the `iris` dataset to Excel

Links and resources

- Introduction to import with R (**is.R**)
- **Youtube video** on importing data
- Statistical tools for high-throughput data analysis (STHDA) - **Importing Data Into R**
- Karlijn Willems - **This R Data Import Tutorial Is Everything You Need**
- **R for data science book**
- The **R-package** `labelled` to work with labelled data imported from SPSS or stata
- Overview - **all import functionalities**

Links Export

- **Quick R** for the export of data



- Help for exporting on the **CRAN Server**
- **Export data from R**
- Youtube video - **Export data from R**
- Quick R - **Exporting data**
- dummies - **How to Get Your Data Out of R**
- R Core Team - **R Data Import/Export**