

# Introduction to R

## Import Data

Jan-Philipp Kolb

04 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

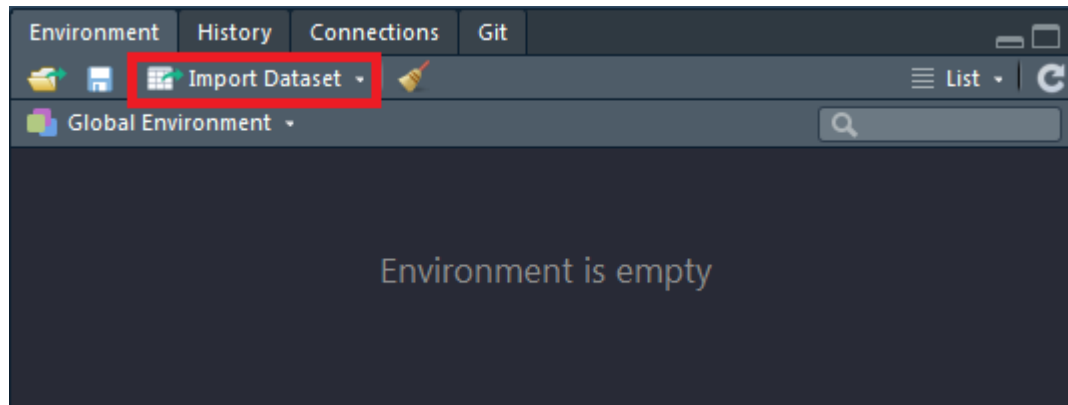
# Data import



# Import data with Rstudio

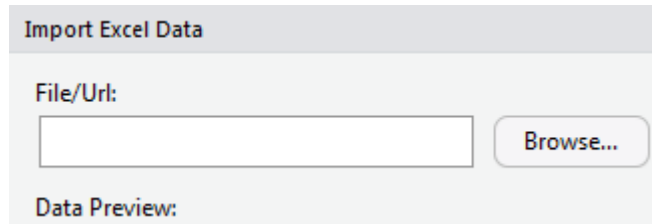
## Rstudio functionality to import data

- Environment - Import Dataset - choose file type



# Where to find data

## Browse Button in RStudio

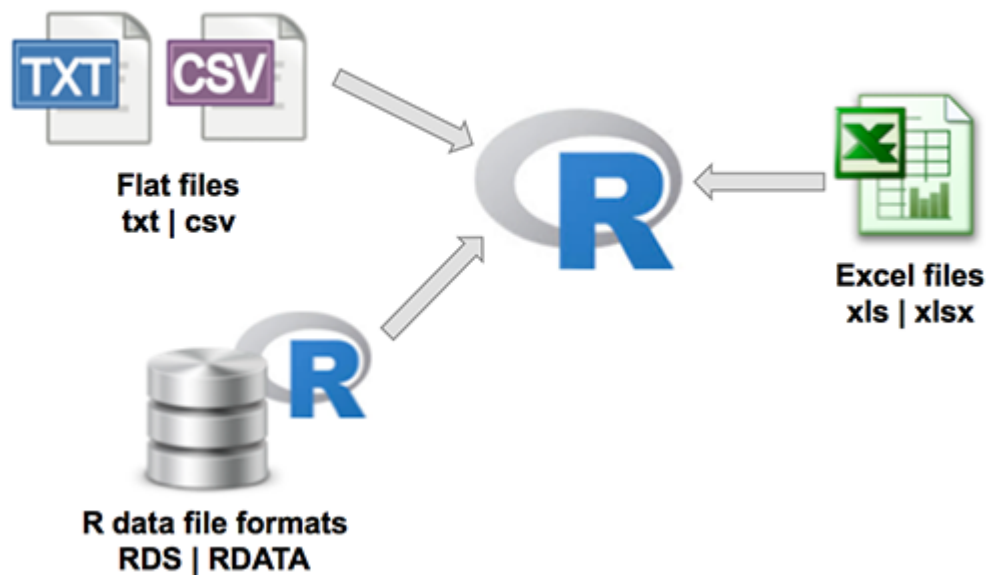


## Code preview in Rstudio



# Import .csv data

## Importing Data Into R



# 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("../data/ZA5666_v1-0-0.csv")
```

If it's German data:

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

```
dat <- read.csv("../data/datahub_refugee.csv")
```

# The result - a data . frame

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

```
head(dat)
```

##	Country.Name	Country.Code	Year	Value
## 1	Arab World	ARB	1990	4235545
## 2	Arab World	ARB	1991	3811595
## 3	Arab World	ARB	1992	4000509
## 4	Arab World	ARB	1993	4189545
## 5	Arab World	ARB	1994	4352945
## 6	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/datahub_names_sa.xlsx")
head(ab)
```

```
## # A tibble: 6 x 3
##   ...1 X1          X2
##   <chr> <chr>      <chr>
## 1 1      0.0036749995 JOHN
## 2 2      0.0036418888 JOHANNES
## 3 3      0.0031182974 DAVID
## 4 4      0.0030492798 MICHAEL
## 5 5      0.0029608373 JACOBUS
## 6 6      0.0027139047 MARIA
```

# Import SPSS files

## Import GESIS Panel 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 <dbl+lbl>	Government <dbl+lbl>		Name <dbl+lbl>		since <dbl+lbl>	Term <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:

```
library(foreign)
link <- "http://www.statistik.at/web_de/static/
mz_2013_sds_-_datensatz_080469.sav"

?read.spss
Dat <- read.spss(link,to.data.frame=T)
```

# 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/example_gp.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](mailto:jan.garbuszus@ruhr-uni-bochum.de)

Sebastian Jeworutzki [sebastian.jeworutzki@ruhr-uni-bochum.de](mailto: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/example_gp.dta",  
                  convert.factors = F)  
head(datf$bbzc007a)
```

```
## NULL
```

## For comparison - import without this argument

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

```
## NULL
```

# 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)  
head(names(att_dat))
```

```
## NULL
```

## Example: the variable names

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

```
## NULL
```



# Get an initial overview of the data

```
View(datf)
```

	z000001z Personen ID - Campus File	z000002z Studennummer 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 menu

# 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

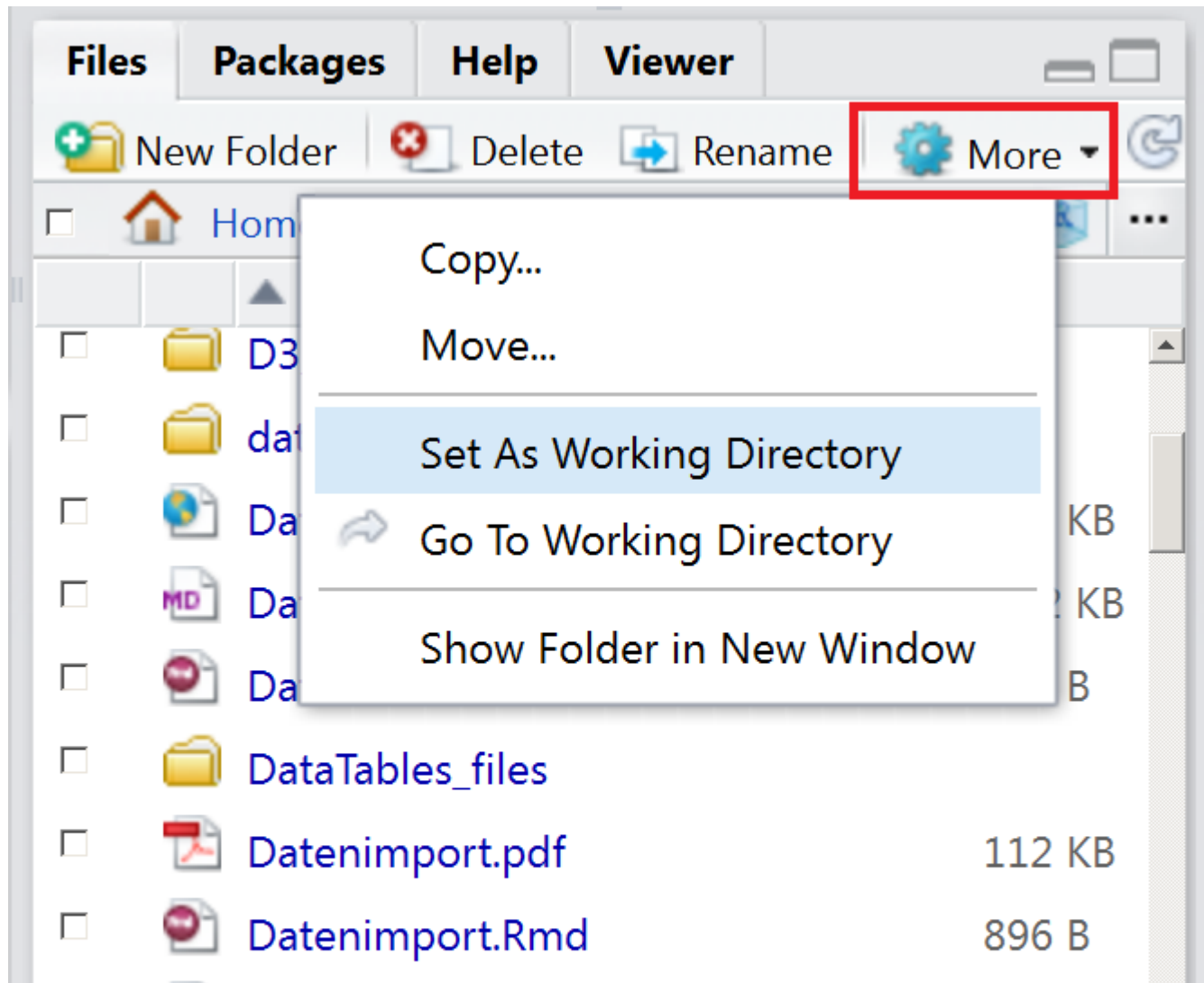
For SPSS and SAS I would recommend the Hmisc package for ease and functionality.

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

## Import SAS data

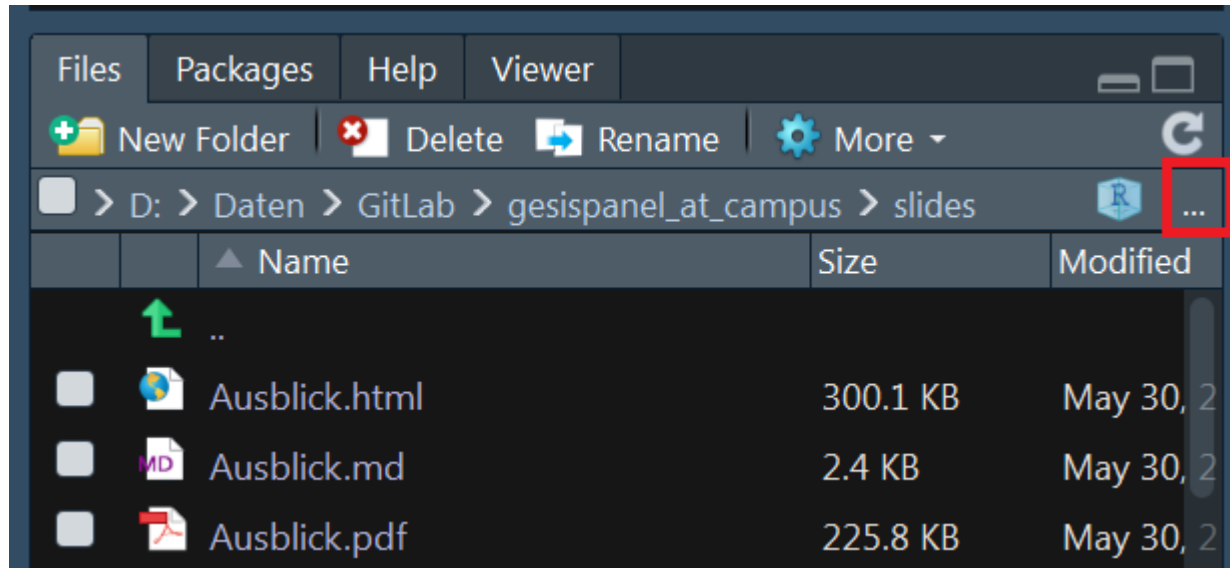
```
mydata <- sasxport.get("c:/mydata.xpt")
# character variables are converted to R factors
```

# 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 slashes instead of backslashes.

# Change working directory

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

```
getwd()
```

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

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

```
## [1] "D:/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)
```

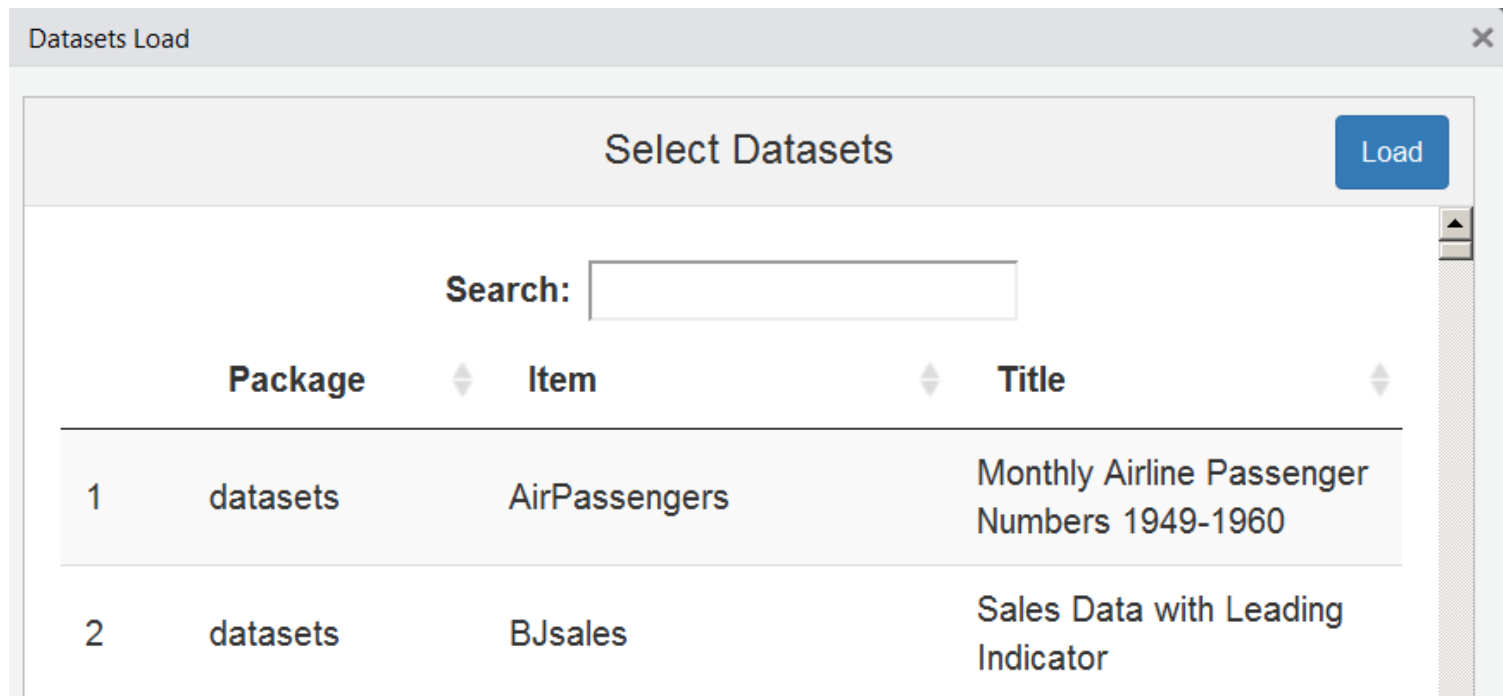
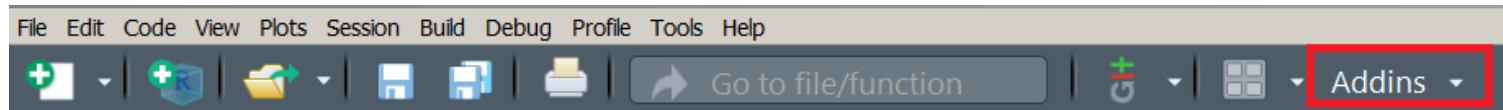
- There is also an **RStudio add-in** that helps to find a dataset

```
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 <- c(1,2,3,4)
B <- c("A","B","C","D")

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

```
mydata
```

A B

---

1 A  
2 B  
3 C  
4 D

# 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

# 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**