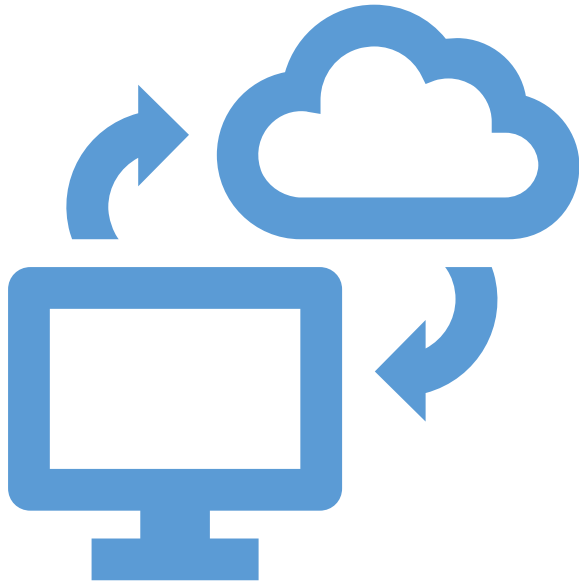


# Statistik 1 – Tutorate

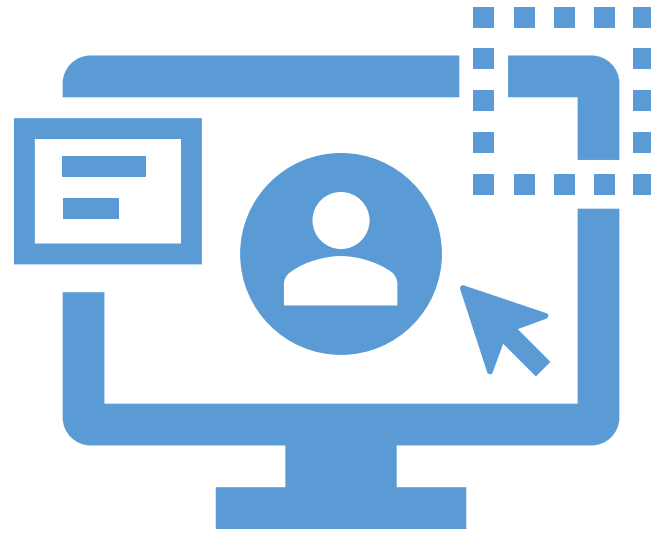
## Sitzung 2: Datenintegration in R

Marco Giesselmann, Lea Elina Hofer, Norma De Min, Mara Moos, Rémy Blum

# Lernziele dieser Sitzung



Installieren und  
aktivieren von Packages



Daten einlesen



Dateninspektion

# Schritt 1: Installation und Aktivierung von Packages



# 1

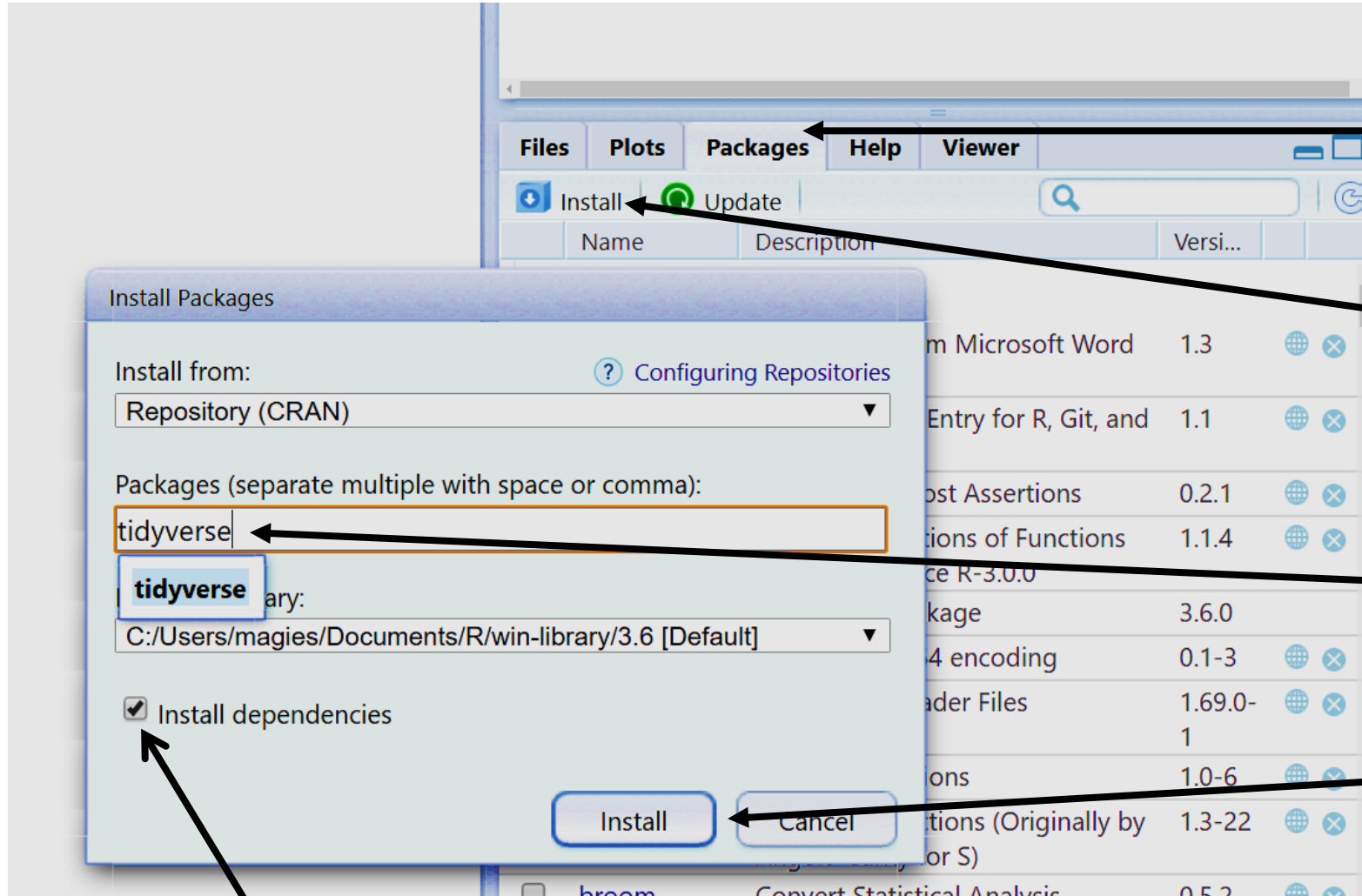
## Installation und Aktivierung von Packages

- R Studio bietet einen Basis-Pool an Kommandos und Funktionen an
- Dieser Pool lässt sich erweitern mit verschiedenen add-ons bzw. Apps: sog. «**Packages**»
- Besonders nützlich: Das Package-Set «**tidyverse**», bestehend aus mehreren Packages, die sehr häufig gebrauchte Befehle enthalten, darunter etwa **dplyr** oder **ggplot2**



# 1

## Installation und Aktivierung von Packages



Wähle „Packages“

Wähle „Install“

Suche nach gewünschtem Package

Installiere Package

Häkchen drin lassen

# 1

## Installation und Aktivierung von Packages

Installation ≠ Aktivierung

```
#Packages installieren  
install.packages("tidyverse")
```

Anführungszeichen nicht vergessen!

```
#Packages aktivieren  
library(tidyverse)
```

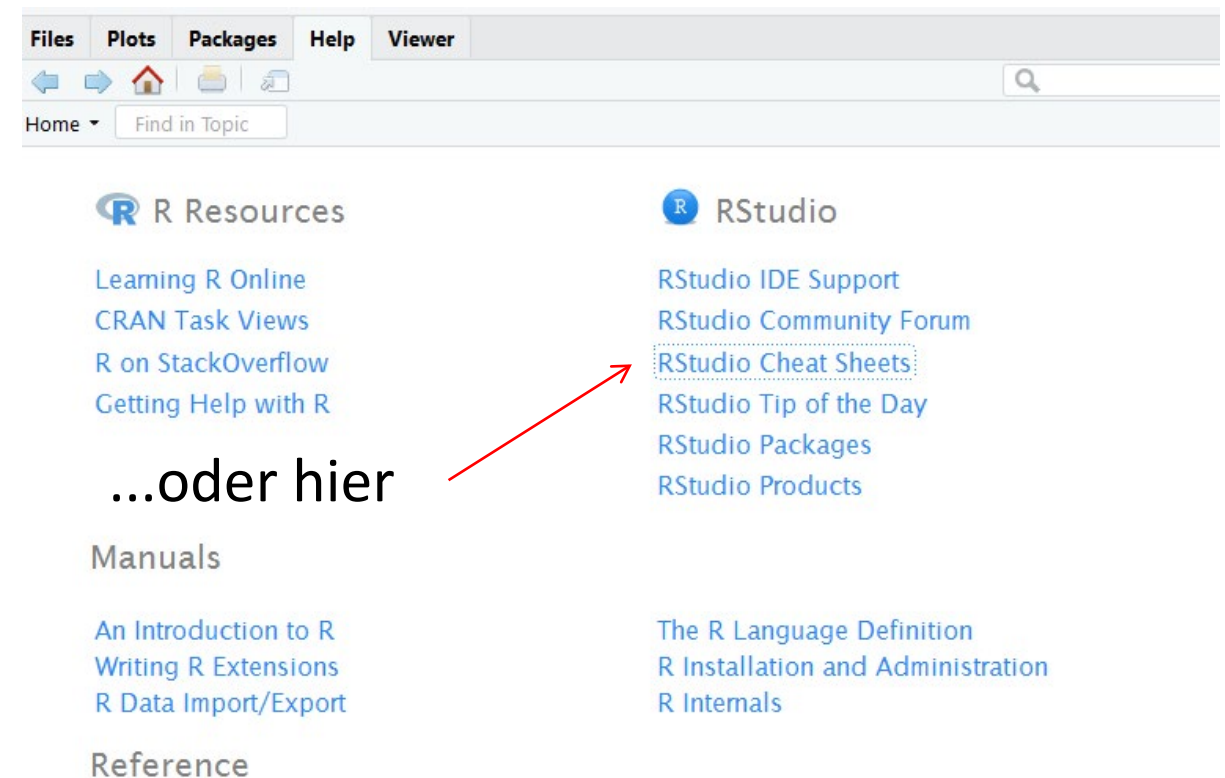
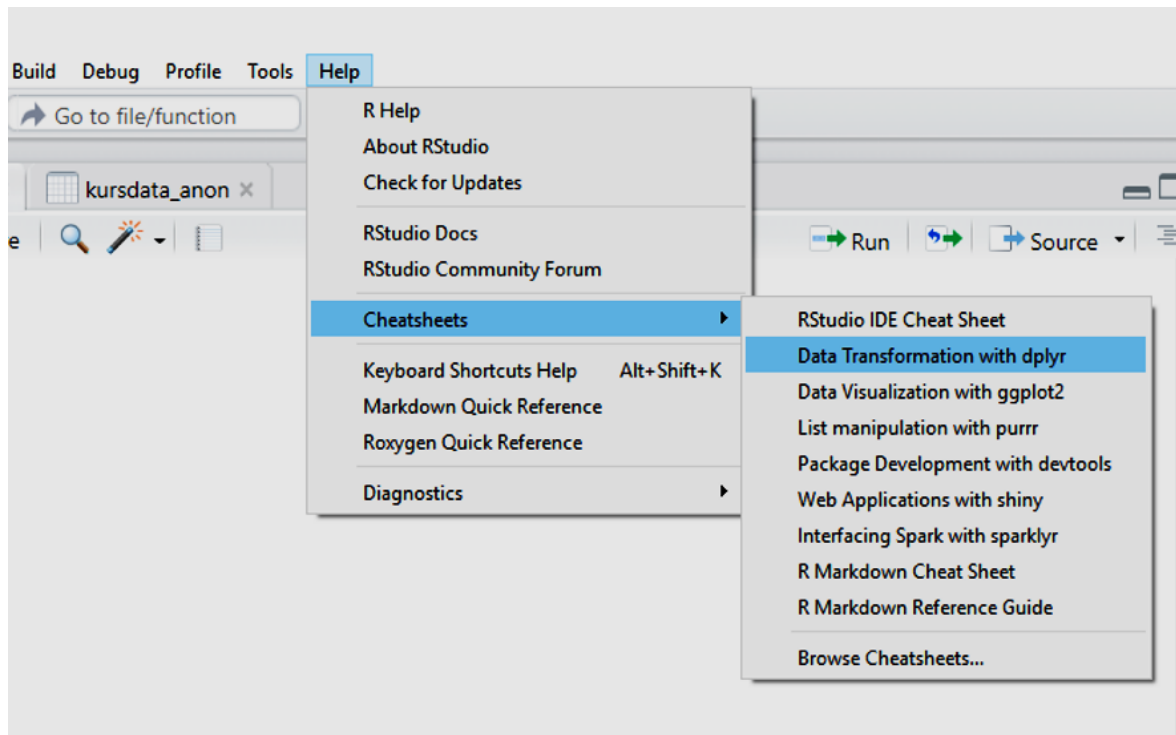
Ein Package muss nur einmal *installiert* werden!

Ähnlich einer App muss ein Package gestartet bzw. *aktiviert* werden, bevor es in einer aktuellen R Session genutzt werden kann!

# 1

# Installation und Aktivierung von Packages

Wichtige Informationsquelle zu den Packages: 1. Cheat Sheets



# 1

# Installation und Aktivierung von Packages

## Data Transformation with dplyr : : CHEAT SHEET

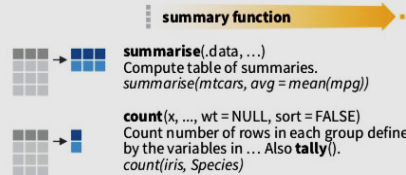


dplyr functions work with pipes and expect tidy data. In tidy data:



### Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).



#### VARIATIONS

`summarise_all()` - Apply funs to every column.  
`summarise_at()` - Apply funs to specific columns.  
`summarise_if()` - Apply funs to all cols of one type.

### Group Cases

Use `group_by()` to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.



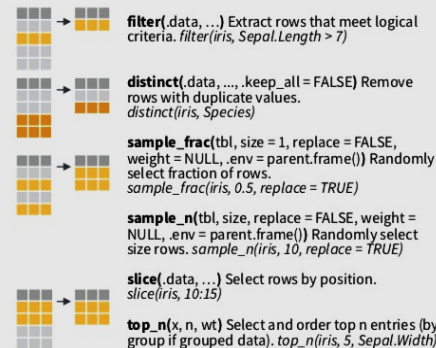
`group_by(data, ..., add = FALSE)`  
Returns copy of table grouped by ...  
`g_iris <- group_by(iris, Species)`

`ungroup(x, ...)`  
Returns ungrouped copy of table.  
`ungroup(g_iris)`

### Manipulate Cases

#### EXTRACT CASES

Row functions return a subset of rows as a new table.

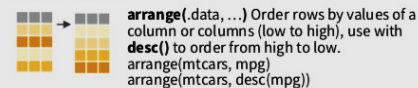


#### Logical and boolean operators to use with filter()

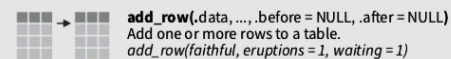
<	<=	is.na()	%in%		xor()
>	>=	is.na()	!	&	

See **?base::logic** and **?Comparison** for help.

#### ARRANGE CASES



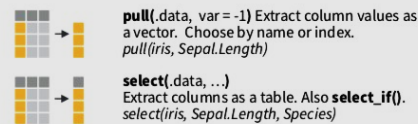
#### ADD CASES



### Manipulate Variables

#### EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

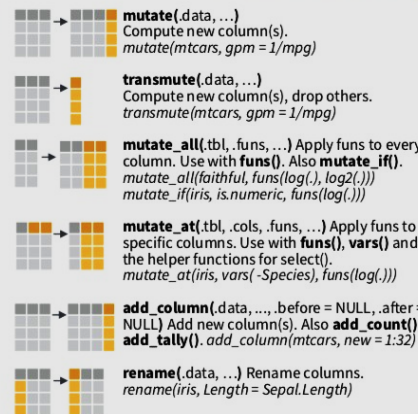


Use these helpers with `select()`, e.g. `select(iris, starts_with("Sepal"))`

<code>contains(match)</code>	<code>num_range(prefix, range)</code>	-, e.g. <code>mpg:cyl</code>
<code>ends_with(match)</code>	<code>one_of(...)</code>	-, e.g. <code>-Species</code>
<code>matches(match)</code>	<code>starts_with(match)</code>	

#### MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).



Was ist ein «Cheat Sheet»

Übersicht über die Befehle und Funktionen eines Packages

Wozu dienen die Befehle und Funktionen von dplyr?

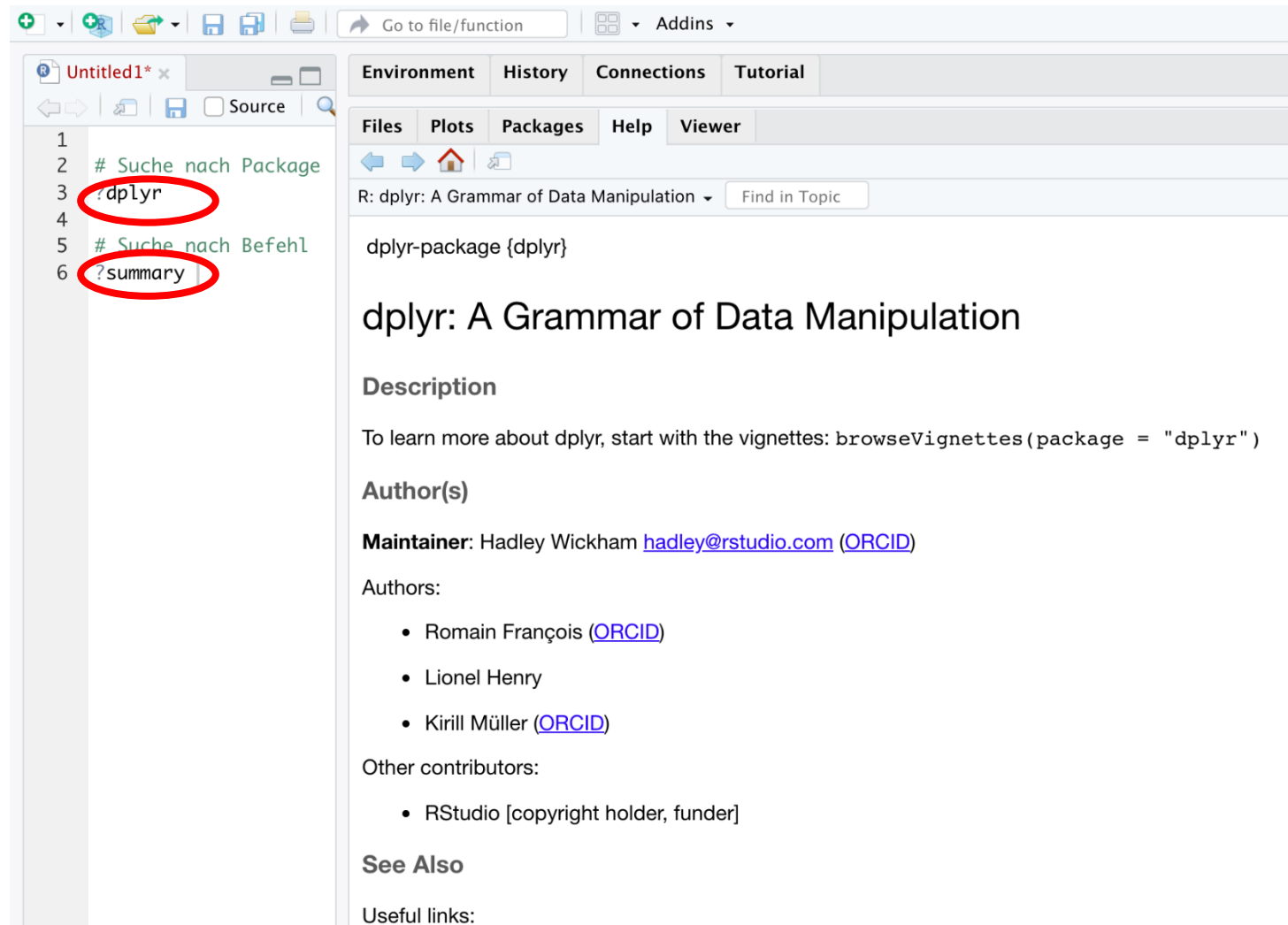
Datenmanagement!



# 1

## Installation und Aktivierung von Packages

Wichtige Informationsquelle zu den Packages: **2. R-Hilfe** (`?package` bzw. `?befehl`)



The screenshot shows the RStudio interface. On the left, the source editor contains the following code:

```
1  
2 # Suche nach Package  
3 ?dplyr  
4  
5 # Suche nach Befehl  
6 ?summary
```

The code on lines 3 and 6 is circled in red. On the right, the Help Viewer is open to the page for the 'dplyr' package. The search bar at the top of the Help Viewer contains the text 'R: dplyr: A Grammar of Data Manipulation'. The main content of the Help Viewer displays the following information:

**dplyr: A Grammar of Data Manipulation**

**Description**

To learn more about dplyr, start with the vignettes: `browseVignettes(package = "dplyr")`

**Author(s)**

**Maintainer:** Hadley Wickham [hadley@rstudio.com](mailto:hadley@rstudio.com) ([ORCID](#))

Authors:

- Romain François ([ORCID](#))
- Lionel Henry
- Kirill Müller ([ORCID](#))

Other contributors:

- RStudio [copyright holder, funder]

**See Also**

Useful links:

# Schritt 2: Einlesen der Daten





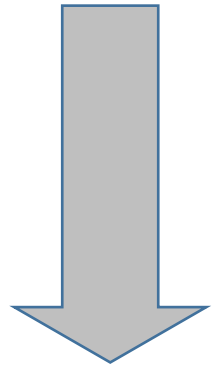
## 2

# Einlesen der Daten: Daten herunterladen

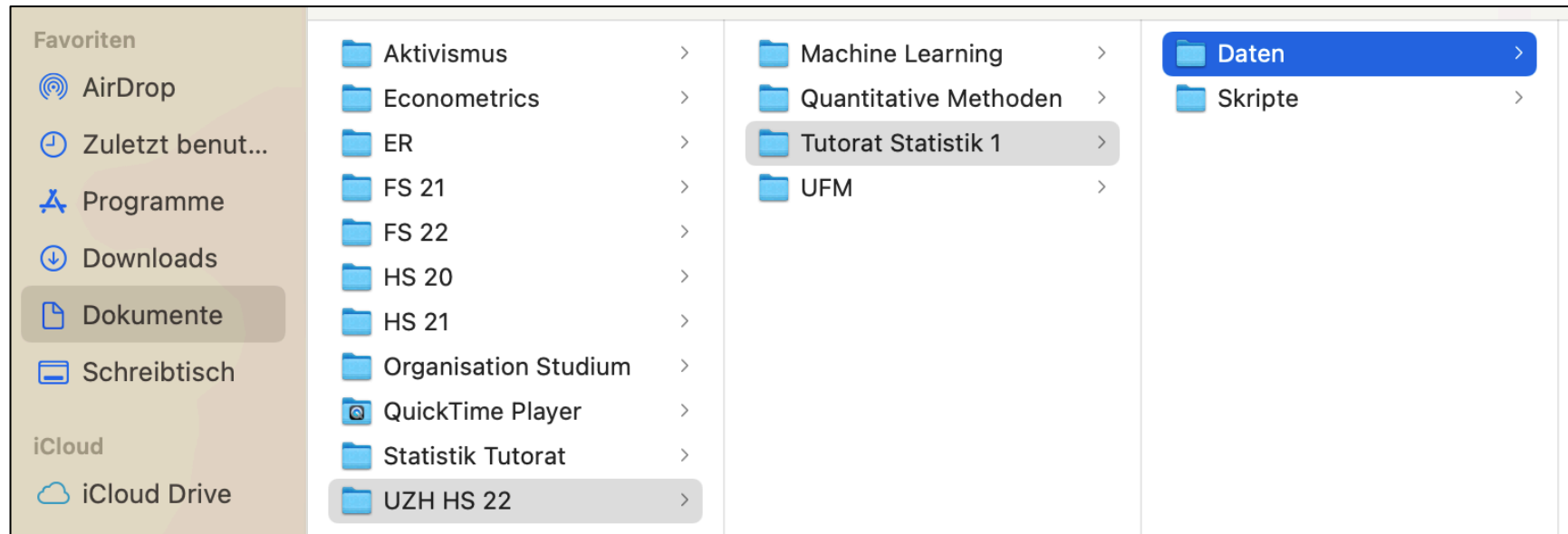
Datensatz  
bereitgestellt auf  
Olat...

Materialien / Daten

<input type="checkbox"/>	Name ▲	Grösse	Geändert
<input type="checkbox"/>	 kursdata_anon.csv	10.3 kB	06.10.22, 14:47
<input type="checkbox"/>	 kursdata_anon.dta	21.7 kB	06.10.22, 14:47



...speichern in  
persönlichem  
Datenordner

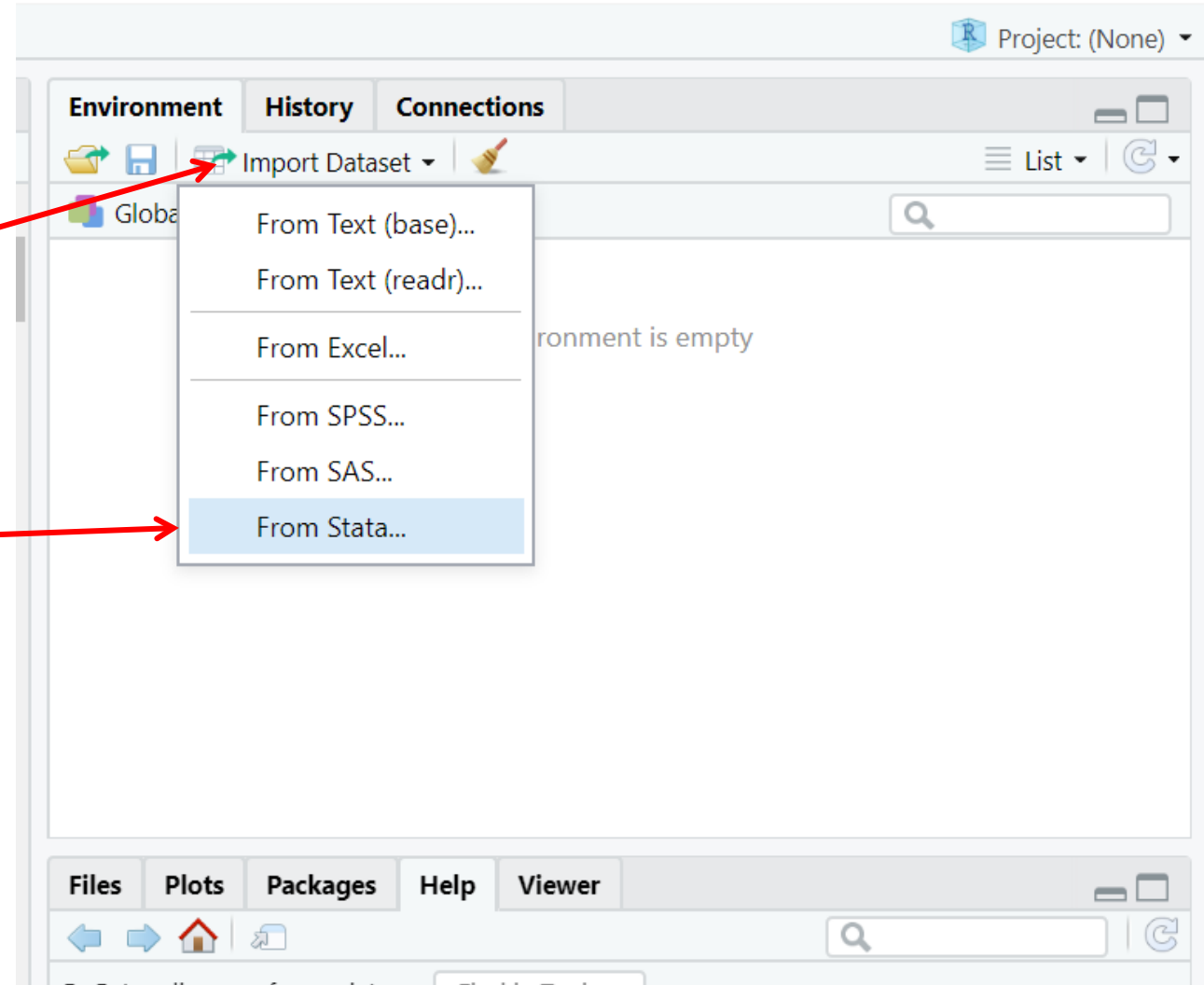


## 2

# Einlesen der Daten: Daten in R importieren

**Import Dataset** im Environment wählen

Datenformat **From Stata (.dta)...** wählen



## 2

## Einlesen der Daten: Daten in R importieren

Import Statistical Data

File/URL:  
C:/Daten/kursdata\_anon.dta Browse...

Data Preview:

id	dispcode	lastpage	lezufr Lebenszufriedenheit derzeit	alter	geschlecht	qm effektiver wohnraum	rauchen letzte Woche geraucht?	lezufr Lebenszufriedenheit vor einem Jahr	trust Kann man Menschen im Allgemeinen vertrauen
15	31	54689		78	19 weiblich	100	2		59
16	31	54689		81	21 weiblich	40	1		68
14	31	54689		60	23 weiblich	30	3		75
19	31	54689		30	21 weiblich	14	2		25
20	31	54689		16	23 maennlich	35	3		61
24	31	54689		73	20 weiblich	30	2		53
25	31	54689		-99	19 weiblich	50	3		17
22	31	54689		80	19 weiblich	NA	2		75
23	31	54689		69	21 weiblich	57	3		55
26	31	54689		70	19 weiblich	70	1		85
21	31	54689		92	24 weiblich	27	3		73
28	31	54689		100	25 maennlich	10	2		100
31	31	54689		94	24 weiblich	45	3		87

Previewing first 50 entries.

Import Options:  
Name: kursdata\_anon  
Model: Browse...  
Format: DTA  Open Data Viewer

Code Preview:

```
library(haven)
kursdata_anon <- read_dta("C:/Daten/kursdata_anon.dta")
View(kursdata_anon)
```

Import Cancel

[? Reading data using haven](#)

Navigiere zum Datenordner  
und öffne den Datensatz

R übersetzt Importvorgang in  
Befehlssyntax

Die Befehlssyntax sollte dann  
per **copy-paste** direkt ins Skript  
übertragen werden.

Hier könnt Ihr dann weitere  
Änderungen vornehmen (z.B.  
**Name ändern**, Pfad auf WD, etc.)

...und schliesslich den Code aktivieren

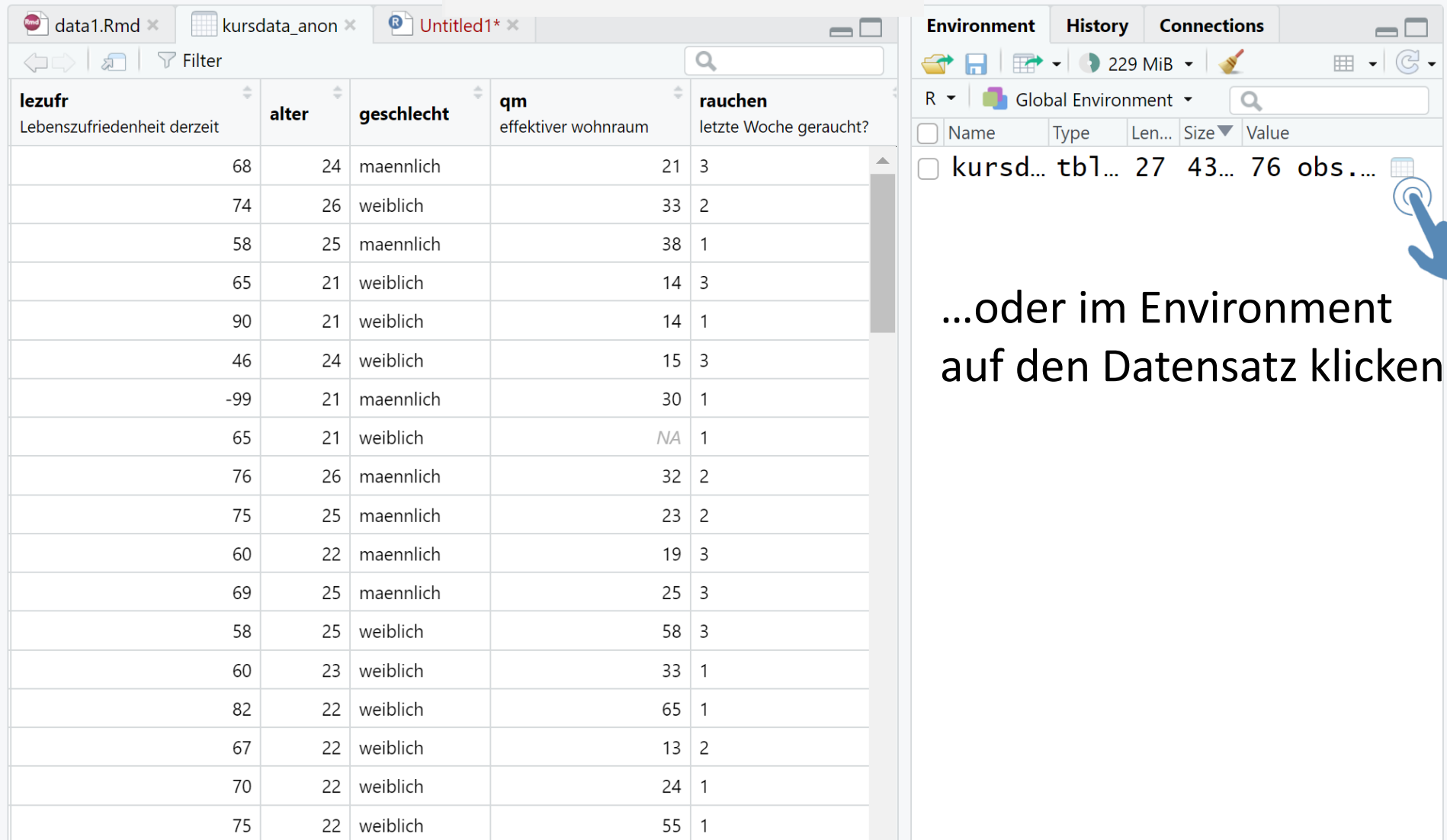
```
basics.Rmd x data1.Rmd* x Untitled1* x
Source on Save Run Source
1 # Lese Daten ein
2 library(haven)
3 kursdata_anon <- read_dta("C:/Daten/kursdata_anon.dta")
4 View(kursdata_anon)
```

# Schritt 3: Dateninspektion



### 3 Dateninspektion: (a) Visuelle Inspektion

Öffnen der Datenmatrix mit: `View(kursdata_anon)`



The screenshot shows the RStudio interface. On the left, a data table is displayed with columns: `lezufr` (Lebenszufriedenheit derzeit), `alter`, `geschlecht`, `qm` (effektiver wohnraum), and `rauchen` (letzte Woche geraucht?). The table contains 20 rows of data. On the right, the Environment pane shows a table object named `kursd... tbl...` with 27 columns and 43 rows. A blue hand icon is pointing to the table icon next to the object name. Below the Environment pane, the text "...oder im Environment auf den Datensatz klicken." is displayed.

lezufr	alter	geschlecht	qm	rauchen
Lebenszufriedenheit derzeit			effektiver wohnraum	letzte Woche geraucht?
	68	24	maennlich	21 3
	74	26	weiblich	33 2
	58	25	maennlich	38 1
	65	21	weiblich	14 3
	90	21	weiblich	14 1
	46	24	weiblich	15 3
	-99	21	maennlich	30 1
	65	21	weiblich	NA 1
	76	26	maennlich	32 2
	75	25	maennlich	23 2
	60	22	maennlich	19 3
	69	25	maennlich	25 3
	58	25	weiblich	58 3
	60	23	weiblich	33 1
	82	22	weiblich	65 1
	67	22	weiblich	13 2
	70	22	weiblich	24 1
	75	22	weiblich	55 1

## 3

## Dateninspektion: (a) Visuelle Inspektion

- Merkmale in Spalten
- Merkmalsträger in Zeilen
- Werte in Zellen
- Fehlende Werte == «NA»?

Nicht ok! (...warum?)

lezufr Lebenszufriedenheit derzeit	alter	geschlecht	qm effektiver wohnraum	rauchen letzte Woche geraucht?
68	24	maennlich	21	3
74	26	weiblich	33	2
58	25	maennlich	38	1
65	21	weiblich	14	3
90	21	weiblich	14	1
46	24	weiblich	15	3
-99	21	maennlich	30	1
65	21	weiblich	NA	1
76	26	maennlich	32	2
75	25	maennlich	23	2
60	22	maennlich	19	3
69	25	maennlich	25	3
58	25	weiblich	58	3
60	23	weiblich	33	1
82	22	weiblich	65	1
67	22	weiblich	13	2
70	22	weiblich	24	1
75	22	weiblich	55	1

Showing 1 to 18 of 76 entries, 27 total columns

Alle (für dich) relevanten  
Variablen anschauen: ggf.  
browsen & blättern!



### 3 Dateninspektion: (b) Systematische Inspektion

1. Ist die **Anzahl an Fällen und Variablen** plausibel?
2. **Variablenübersicht**: Sind alle relevanten Variablen enthalten?
3. **Übersichtstatistiken** (*Summary Statistics*): Plausible Werte?
4. Sind die **Klassen** der relevanten Variablen stimmig definiert?



### 3 Dateninspektion: (b) Systematische Inspektion

1. Ist die **Anzahl an Fällen und Variablen** plausibel?

```
dim(kursdata_anon)
```

```
> dim(kursdata_anon)  
[1] 76 27
```

Wie interpretiert ihr den Output?

### 3 Dateninspektion: (b) Systematische Inspektion

## 2. Variablenübersicht: Sind alle relevanten Variablen enthalten?

Aufgabe: Versucht den Befehl **look\_for()** dafür anzuwenden

```
> look_for(kursdata_anon)
Fehler in look_for(kursdata_anon) :
  konnte Funktion "look_for" nicht finden
```

... RStudios findet den **look\_for()** Befehl nicht. Was nun?

```
> ?look_for
No documentation for 'look_for' in specified packages and libraries:
you could try '??look_for'
```

... Rstudios findet zwar keine Informationen über **look\_for()**, gibt uns jedoch eine Hilfestellung.

#### Vignettes:

[labelled::look\\_for](#) Generate a data dictionary and search for variables with `look\_for()` [HTML](#) [source](#) [R code](#)

#### Help pages:

[labelled::look\\_for](#) Look for keywords variable names and descriptions / Create a data dictionary

... **look\_for()** ist im Package **labelled** enthalten, welches wir zuerst noch installieren und aktivieren müssen.

```
install.packages("labelled")
library(labelled)
look_for(kursdata_anon)
```

... geschafft 😊

### 3 Dateninspektion: (b) Systematische Inspektion

## 2. Variablenübersicht: Sind alle relevanten Variablen enthalten?

```
varlist <- look_for(kursdata_anon)
varlist
```

Erstellt mit look\_for ein Codebuch!

pos	variable	label	col_type	levels	value_labels
1	id	NA	dbl	NULL	NULL
2	dispcode	NA	dbl	NULL	NULL
3	lastpage	NA	dbl	NULL	NULL
4	lezufr	Lebenszufriedenheit derzeit	dbl	NULL	NULL
5	alter	NA	dbl	NULL	NULL
6	geschlecht	NA	chr	NULL	NULL
7	qm	effektiver wohnraum	dbl	NULL	NULL
8	rauchen	letzte Woche geraucht?	dbl+lbl	NULL	c(ja = 1, `nein, aber früher` = 2, `nein,
9	llezufr	Lebenszufriedenheit vor einem Jahr	dbl	NULL	NULL
10	trust	Kann man Menschen im Allgemeinen vertrauen? (5-volle Zustimmung, 1-volle Ablehnu	dbl	NULL	NULL
11	eltern	(fach)hochschulabschluss eltern	dbl+lbl	NULL	c(beide = 1, einer = 2, keiner = 3)
12	mathe	letzte erzielte schulnote im fach mathematik	dbl	NULL	NULL
13	deutsch	letzte erzielte schulnote im fach deutsch	dbl	NULL	NULL
14	fach	studiertes hauptfach	dbl+lbl	NULL	c(Soziologie = 1, Sonstiges = 2)

- Beschreibt das mit «look\_for» erstellte Objekt
- Nutzt das mit «look\_for» erstellte Objekt zur Variablensuche

## 3 Dateninspektion: (b) Systematische Inspektion

### 3. Übersichtstatistiken (*Summary Statistics*): Plausible Werte?

Aufgabe: Tippe «**summary**(kursdata\_anon).

Welche relevanten Inspektionsergebnisse liefert dieser Befehl?

```
> summary(kursdata_anon)
   id          dispcode      lastpage      lezufr
Min.   : 28.00   Min.   :31.00   Min.   :95873   Min.   : -99.00
1st Qu.: 74.75   1st Qu.:31.00   1st Qu.:95873   1st Qu.:  60.00
Median : 98.50   Median :31.00   Median :95873   Median :  75.00
Mean   :101.91   Mean   :31.09   Mean   :95873   Mean   :  65.53
3rd Qu.:138.25   3rd Qu.:31.00   3rd Qu.:95873   3rd Qu.:  80.00
Max.   :173.00   Max.   :32.00   Max.   :95873   Max.   :100.00

   alter          geschlecht          qm          rauchen
Min.   :19.00   Length:76   Min.   :10.00   Min.   :1.000
1st Qu.:21.00   Class :character   1st Qu.:25.50   1st Qu.:1.000
Median :22.00   Mode  :character   Median :35.00   Median :2.000
Mean   :22.42                                     Mean   :2.171
3rd Qu.:24.00                                     3rd Qu.:3.000
Max.   :26.00                                     Max.   :3.000
NA's   :1
```

... ihr könnt den Befehl auch auf eine einzelne Variable einschränken, indem ihr diese mittels **\$** ansteuert

### 3 Dateninspektion: (b) Systematische Inspektion

#### 3. Übersichtstatistiken (*Summary Statistics*): Plausible Werte?

Aufgabe: Benutze den **summary()** Befehl für Übersichtstatistiken zu den Variablen **alter**, **lezufr** und **leftright**. Deute jeweils die Ergebnisse.

```
summary(kursdata_anon$alter)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
19.00	21.00	22.00	22.42	24.00	26.00

```
summary(kursdata_anon$lezufr)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-99.00	60.00	75.00	65.53	80.00	100.00

```
summary(kursdata_anon$qm)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
10.00	25.50	35.00	38.48	55.00	65.00

NA's
1

### 3 Dateninspektion: (b) Systematische Inspektion

#### 3. Übersichtstatistiken (*Summary Statistics*): Plausible Werte?

Aufgabe: Benutze den Befehl **table()** um Häufigkeitstabellen für kategoriale Variablen herzustellen und deute das Ergebnis.

```
library(dplyr)
table(kursdata_anon$konsum)
```

1	2
26	44

?

Aufgabe: Weise die Häufigkeiten bzw. Verteilung der «Semester»-Variable aus


```
table(kursdata_anon$semester)
```

1	2	3	4	5	6
12	16	20	15	3	7

Ermittle Anteilswerte zu den Kategorien der Semesterzahl (gerundet auf 2 oder 3 Nachkommastellen). Suche sowohl einen händischen als auch befehlsbasierten Weg.


### 3 Dateninspektion: (b) Systematische Inspektion

MA `prop.table(table(kursdata_anon$semester), digits=2) # how can I reduce decimals`

 If you want to reduce the number of decimal places displayed in the output of `prop.table()`, you can use the `digits` argument to specify the number of decimal places you want to retain. Here's how you can modify your code to achieve that:

```
R prop.table(table(kursdata_anon$semester), digits = 2) Copy code
```

MA Error in `prop.table(table(kursdata_anon$semester), digits = 2)`:  
unused argument (`digits = 2`)

 I apologize for the confusion. It seems I made an error in my previous response. The `prop.table()` function in R does not have a `digits` argument to control the number of decimal places in the output.

To control the number of decimal places in the output, you can use the `round()` function to round the proportions to the desired number of decimal places. Here's how you can modify your code to achieve this:

```
> round(prop.table(table(kursdata_anon$semester)), 2)
```

```
 1      2      3      4      5      6  
0.16 0.22 0.27 0.21 0.04 0.10
```



# 3 Dateninspektion: (b) Systematische Inspektion

Search results for "r frequency table".

Top result: <https://www.programmingr.com/statistics/frequency-table>  
**How To Make Frequency Table in R - Programming R Tutorials**  
 In this article, we'll explore how to create frequency tables in R using both base R functions and the tidyverse packages. We'll cover the basic concepts of frequency tables, such as counts, percentages, and cumulative frequencies, as well as some advanced techniques for visualizing and analyzing categorical data.

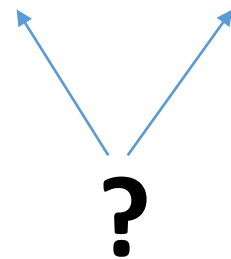
Second result: <https://www.statology.org/frequency-table-in-r>  
**How to Create Frequency Tables in R (With Examples) - Statology**  
 A frequency table is a table that displays the frequencies of different categories. This type of table is particularly useful for understanding the distribution of values in a dataset. This tutorial explains how to create frequency tables in R using the following data frame:

Carb	Frequency	Percent	Cum. percent
1	14	43.8	43.8
2	11	34.4	78.1
3	7	21.9	100.0
4	32	100.0	100.0

```
> #install.packages('epiDisplay')
> library(epiDisplay)
> tab1(kursdata_anon$semester, cum.percent = TRUE)
```

kursdata\_anon\$semester :

	Frequency	%(NA+)	cum.%(NA+)	%(NA-)	cum.%(NA-)
1	12	15.8	15.8	16.4	16.4
2	16	21.1	36.8	21.9	38.4
3	20	26.3	63.2	27.4	65.8
4	15	19.7	82.9	20.5	86.3
5	3	3.9	86.8	4.1	90.4
6	7	9.2	96.1	9.6	100.0
<NA>	3	3.9	100.0	0.0	100.0
Total	76	100.0	100.0	100.0	100.0



# 3 Dateninspektion: (b) Systematische Inspektion

Search results for "r frequency table".

Top result: <https://www.programmingr.com/statistics/frequency-table>  
**How To Make Frequency Table in R - Programming R Tutorials**  
 In this article, we'll explore how to create frequency tables in R using both base R functions and the tidyverse packages. We'll cover the basic concepts of frequency tables, such as counts, percentages, and cumulative frequencies, as well as some advanced techniques for visualizing and analyzing categorical data.

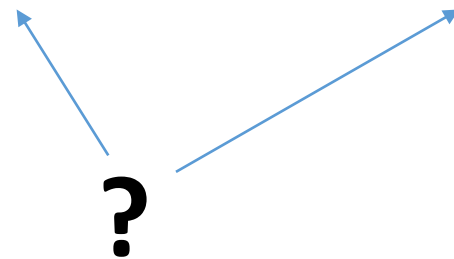
Second result: <https://www.statology.org/frequency-table-in-r>  
**How to Create Frequency Tables in R (With Examples) - Statology**  
 A frequency table is a table that displays the frequencies of different categories. This type of table is particularly useful for understanding the distribution of values in a dataset. This tutorial explains how to create frequency tables in R using the following data frame:

Carb	Frequency	Percent	Cum. percent
1	14	43.8	43.8
2	11	34.4	78.1
3	7	21.9	100.0
4	32	100.0	100.0

```
> #install.packages('epiDisplay')
> library(epiDisplay)
> tab1(kursdata_anon$semester, cum.percent = TRUE)
```

kursdata\_anon\$semester :

	Frequency	%(NA+)	cum.%(NA+)	%(NA-)	cum.%(NA-)
1	12	15.8	15.8	16.4	16.4
2	16	21.1	36.8	21.9	38.4
3	20	26.3	63.2	27.4	65.8
4	15	19.7	82.9	20.5	86.3
5	3	3.9	86.8	4.1	90.4
6	7	9.2	96.1	9.6	100.0
<NA>	3	3.9	100.0	0.0	100.0
Total	76	100.0	100.0	100.0	100.0

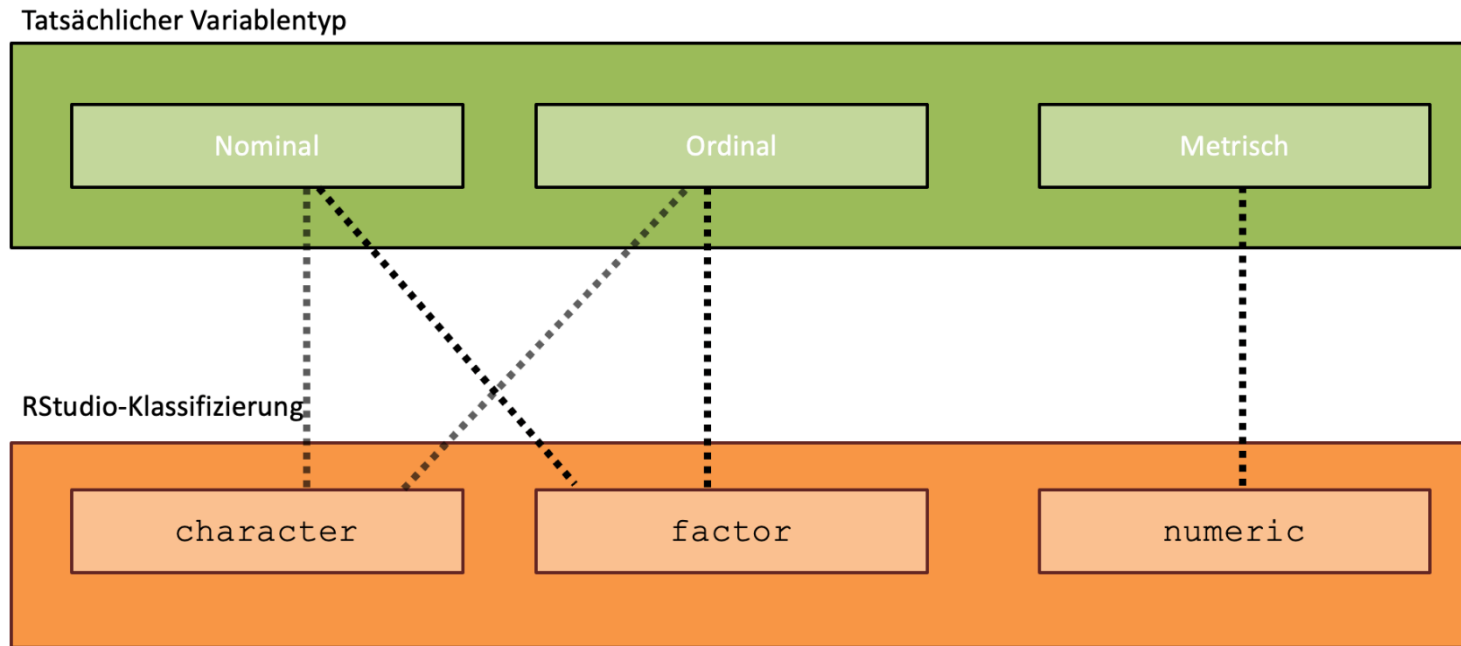


### 3 Dateninspektion: (b) Systematische Inspektion

4. Sind die **Klassen** der relevanten Variablen stimmig definiert?

Aufgabe: Überprüft den Typ der Variablen: **geschlecht**, **konsum**, **leftright** - Ist die Zuordnung stimmig im Sinne des Korrespondenzmusters?

```
> class(kursdata_anon$geschlecht)
[1] "character"
> class(kursdata_anon$leftright)
[1] "numeric"
> class(kursdata_anon$konsum)
[1] "haven_labelled" "vctrs_vctr"      "double"
```



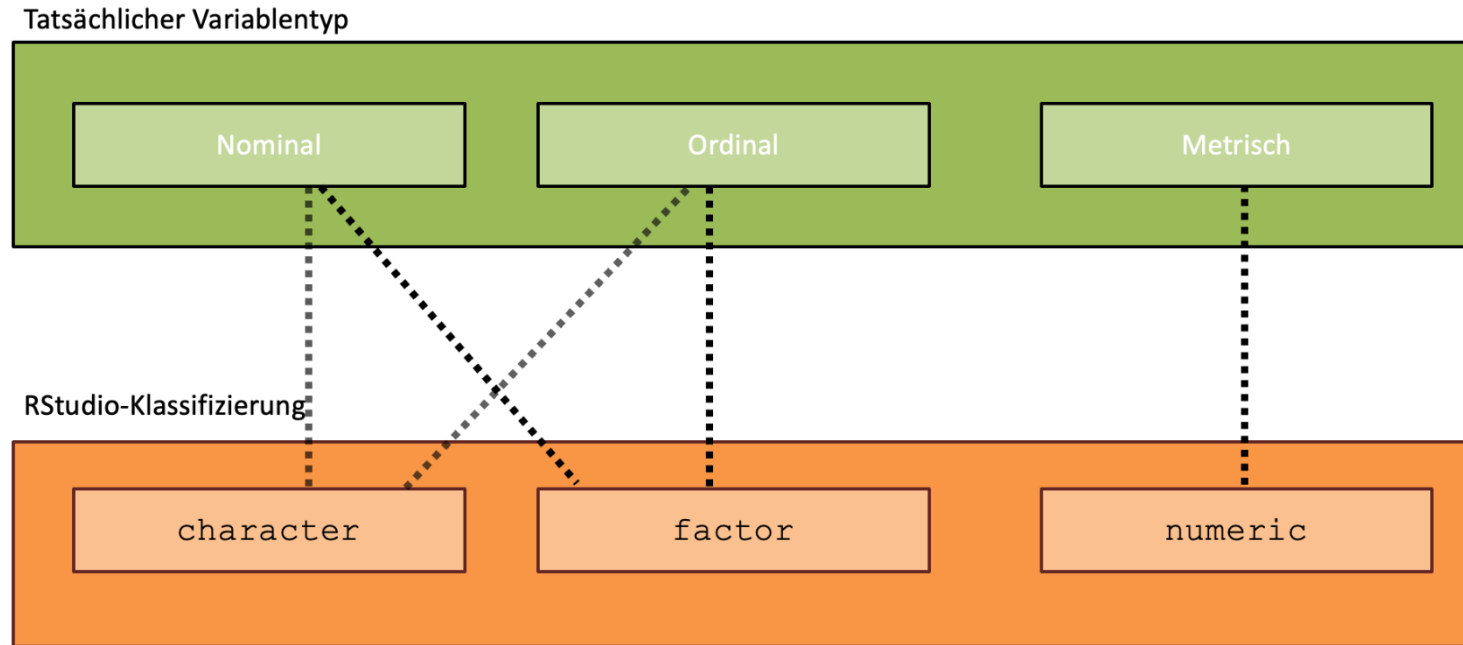
Antwort: Nein. Die kategoriale (nominale) Variable **konsum** ist als «**numeric**» klassifiziert. Dies ist in vielen Anwendungen nicht optimal, wie in der vorigen Aufgabe schon bemerkt (Hinweis: das Attribut **numeric** wird hier nicht direkt dargestellt, aber indirekt durch die Kennzeichnung «**double**» angezeigt.)

### 3 Dateninspektion: (b) Systematische Inspektion

4. Sind die **Klassen** der relevanten Variablen stimmig definiert?

Aufgabe: Überprüft den Typ der Variablen: **geschlecht**, **fach**, **leftright**, **lezufr** - Ist die Zuordnung stimmig im Sinne des Korrespondenzmusters?

```
> class(kursdata_anon$geschlecht)
[1] "character"
> class(kursdata_anon$leftright)
[1] "numeric"
> class(kursdata_anon$konsum)
[1] "haven_labelled" "vctrs_vctr"      "double"
```



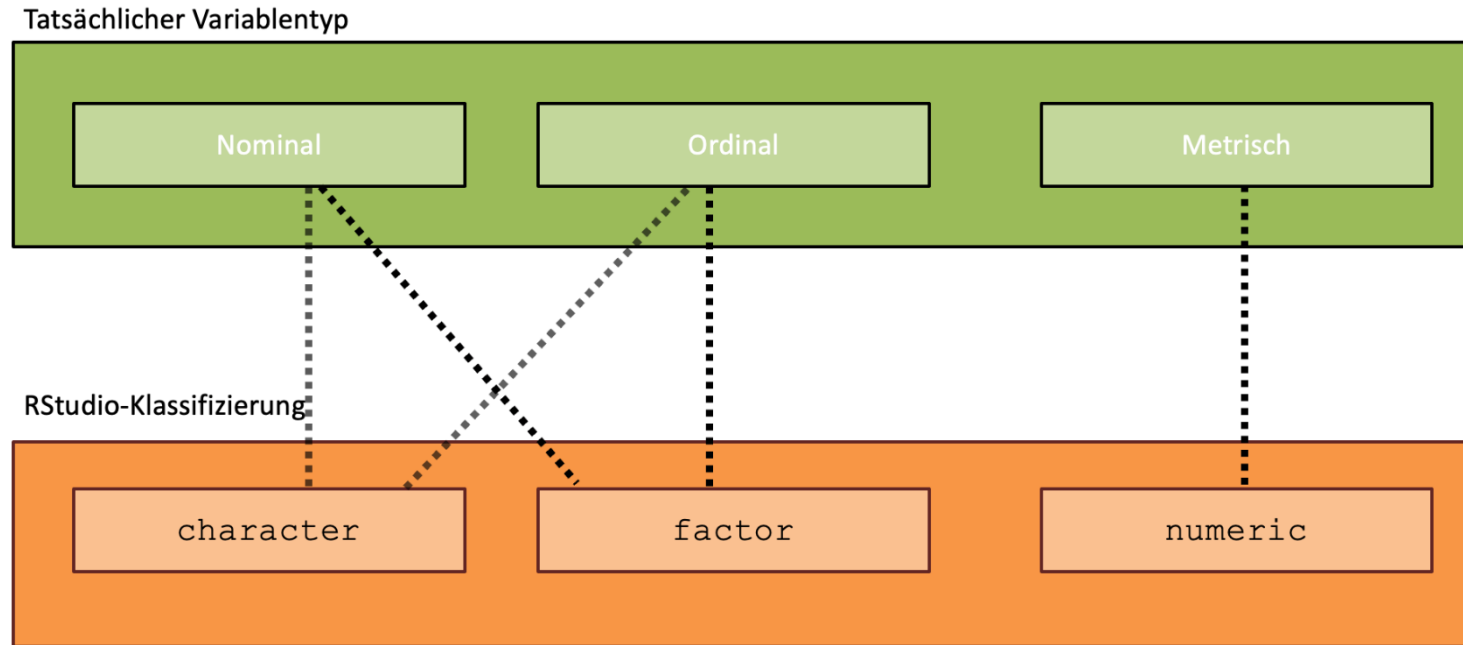
Ausserdem: Die Geschlechtervariable ist als **character**, nicht als **factor** angelegt. Dies ist zwar ok, für viele Anwendungen ist es trotzdem sinnvoll, eine kategoriale Variable als «**factor**» zu klassifizieren.

### 3 Dateninspektion: (b) Systematische Inspektion

4. Sind die **Klassen** der relevanten Variablen stimmig definiert?

Aufgabe: Überprüft den Typ der Variablen: **geschlecht**, **fach**, **leftright**, **lezufr** - Ist die Zuordnung stimmig im Sinne des Korrespondenzmusters?

```
> class(kursdata_anon$geschlecht)
[1] "character"
> class(kursdata_anon$leftright)
[1] "numeric"
> class(kursdata_anon$konsum)
[1] "haven_labelled" "vctrs_vctr"      "double"
```



*Grundsätzlich* empfehlen wir euch also, vor der Analyse alle analyserelevanten kategorialen Variablen als «**factor**» zu klassifizieren.

## 4 Faktorisierung in R

Für die Faktorisierung einer Variable gibt es mehrere Befehle. Was sind die Unterschiede?

```
kursdata_anon$konsum_factor1 <- factor(kursdata_anon$konsum)
table (kursdata_anon$konsum_factor1)
```

```
kursdata_anon$konsum_factor2 <- as.factor(kursdata_anon$konsum)
table (kursdata_anon$konsum_factor2)
```

```
kursdata_anon$konsum_factor3 <- as_factor(kursdata_anon$konsum)
table (kursdata_anon$konsum_factor3)
```

# Dateninspektion mit dem ESS

- Falls noch nicht erledigt: Ladet den kompletten Datensatz des ESS (**Round 8, 2016**) als **Stata-File (.dta)** herunter auf <https://www.europeansocialsurvey.org/data/>
- Ladet ebenfalls das Codebook (Fragekatalog) als PDF herunter.
- Importiert/ öffnet den Datensatz in RStudio.
- Wieviele Fälle und Variablen enthalten die Daten?
- Wie heisst die zu deinem Konzept (siehe unten) korrespondierende Variable im ESS?
  - Vornamen A-E: *Vertrauen in das Europäische Parlament*
  - Vornamen F-L: *Vertrauen in das Nationale Parlament*
  - Vornamen M-P: *Diskriminierungserfahrung aufgrund des Geschlechtes*
  - Vornamen R-Z: *Höchster Bildungsabschluss der Mutter*
- Welchem Typ und welcher R-Klasse gehört die Variable an?
- Wieviele fehlende Werte gibt es? Wie sind diese codiert?
- Nenne und bewerte die grundlegenden Verteilungseigenschaften bzw statistischen Kennwerte der Variable
- Homework: <https://www.suz.uzh.ch/dataforstat/data1ueb.html>

Fragen ?