

R für die Sozialwissenschaften - Teil 1

Jan-Philipp Kolb

08 Juli, 2017

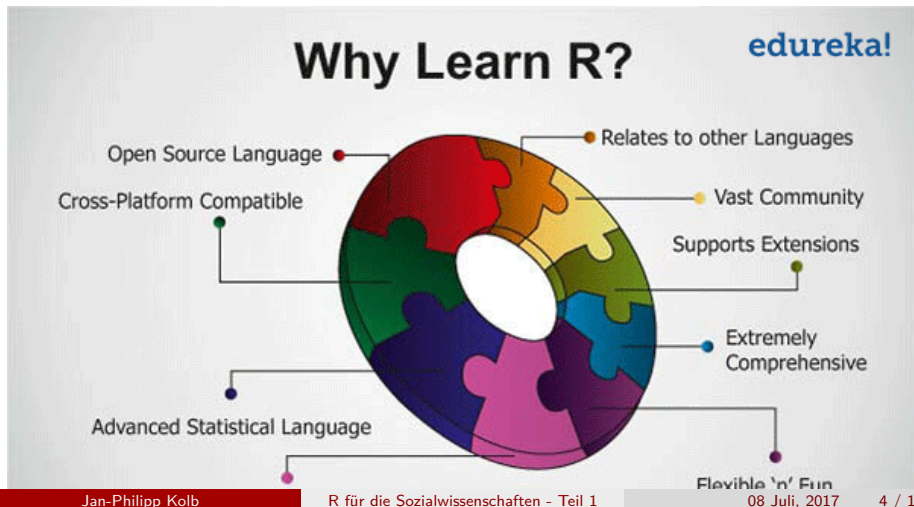
Einführung und Motivation

Pluspunkte von R

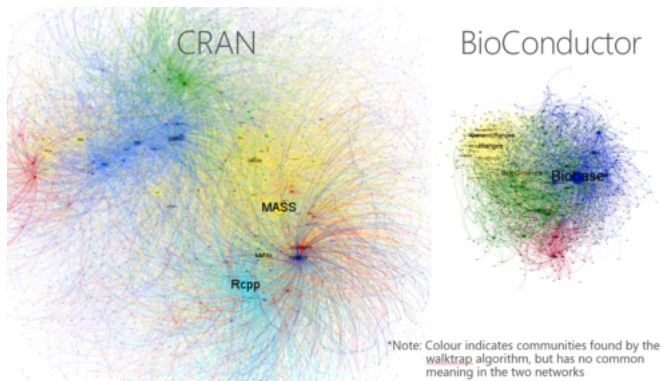
- Als Weg kreativ zu sein ...
- Graphiken, Graphiken, Graphiken
- In Kombination mit anderen Programmen nutzbar
- Zur Verbindung von Datenstrukturen
- Zum Automatisieren
- Um die Intelligenz anderer Leute zu nutzen ;-)
- ...

Gründe

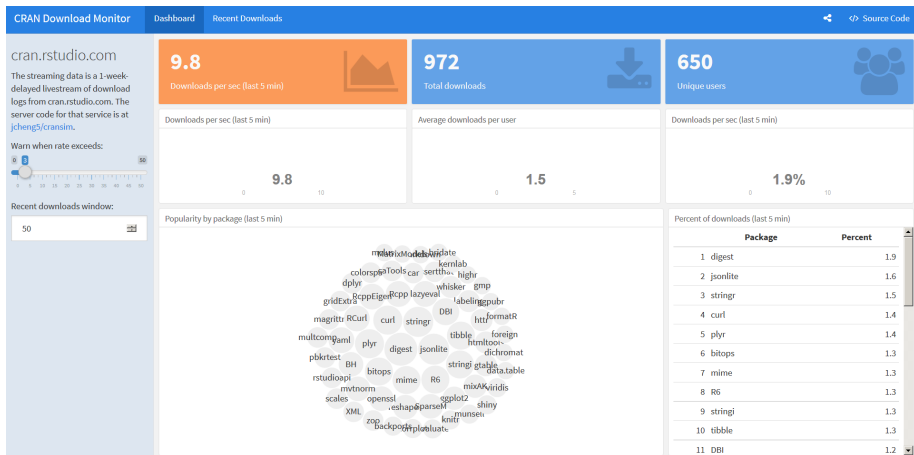
- R ist frei verfügbar. Es kann umsonst runtergeladen werden.
- R ist eine Skriptsprache / Popularität von R



Modularer Aufbau



Viel genutzte Pakete



Organisation des Kurses

- Unterlagen sind komplett auf Github hinterlegt, damit man den Kurs gleich mitverfolgen kann (mehr dazu gleich)
- Es werden viele verschiedene kleine Beispieldatensätze verwendet um spezifische Dinge zu zeigen
- Alle Funktionen in R sind mit diesen kleinen Beispielen hinterlegt
- An geeigneten Stellen verwende ich auch größere (sozialwissenschaftliche) Datensätze

Dem Kurs folgen

- <https://japhilko.github.io/RSocialScience/>



The screenshot shows the homepage of the RSocialScience website. The header is blue with the title 'RSocialScience' and the subtitle 'Kurs zur Nutzung von R in den Sozialwissenschaften'. Below the header is a yellow navigation bar with a 'Home' button and a GitHub logo. The main content area is divided into two columns. The left column lists topics: 'Einführung', 'Liebe auf den ersten Plot – Grafiken und Datenanalyse mit R', 'Regression mit R', 'Arbeitsorganisation mit Rstudio und git', and 'Präsentation von Daten – Reproducible Research'. The right column features a section titled 'Einführung' with a list of topics: 'Einführung und Motivation', 'Erste Schritte mit R', 'Wie bekommt man Hilfe?', 'Modularer Aufbau', 'Datenimport', 'Datenaufbereitung', and 'Datenexport'.

RSocialScience
Kurs zur Nutzung von R in den Sozialwissenschaften

Home

Einführung

Liebe auf den ersten Plot –
Grafiken und Datenanalyse mit
R

Regression mit R

Arbeitsorganisation mit
Rstudio und git

Präsentation von Daten –
Reproducible Research

Einführung

- › Einführung und Motivation
- › Erste Schritte mit R
- › Wie bekommt man Hilfe?
- › Modularer Aufbau
- › Datenimport
- › Datenaufbereitung
- › Datenexport

Komplette Foliensätze

Die kompletten Foliensätze kann man hier herunterladen:

- Teil 1 - Von der Einführung bis Graphiken mit `lattice`
- Teil 2 - Von den Paketen `ggplot2` und `ggmap` bis zu Mehrebenenmodellen
- Teil 3 - Die Arbeitsorganisation mit Rstudio und Rmarkdown
- Teil 4 - Präsentationen, Dashboards, Notebooks und Interaktivität

Der R-code

- Den R-code kann man direkt in die R-Konsole kopieren und ausführen.
- Begleitend zu den Folien wird meistens auch jeweils ein R-File angeboten.
- Der R-code befindet sich in folgendem Ordner:

<https://github.com/Japhilko/RSocialScience/tree/master/code>

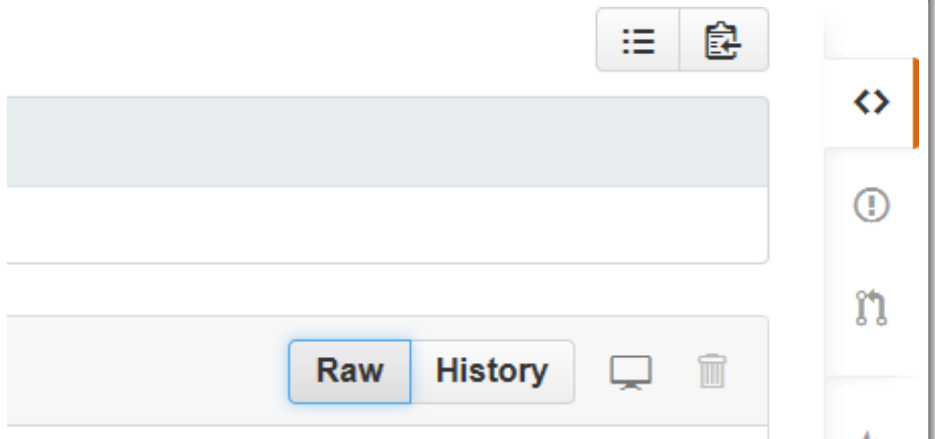
Daten herunterladen

- Vereinzelt sind auch Datensätze vorhanden.
- .csv Dateien können direkt von R eingelesen werden (wie das geht, werde ich noch zeigen).
- Wenn die .csv Dateien heruntergeladen werden sollen - den Raw Button verwenden.
- Alle anderen Dateien (bspw. .RData) auch mittels Raw Button herunterladen.

Ausdrucken

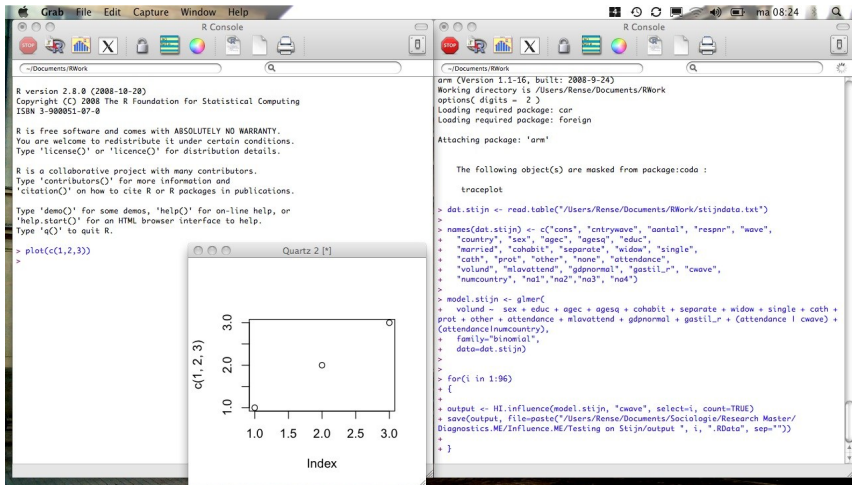
- Zum Ausdrucken eignen sich die pdf-Dateien am besten.
- Diese können mit dem Raw Button heruntergeladen werden.

Raw Button bei Github



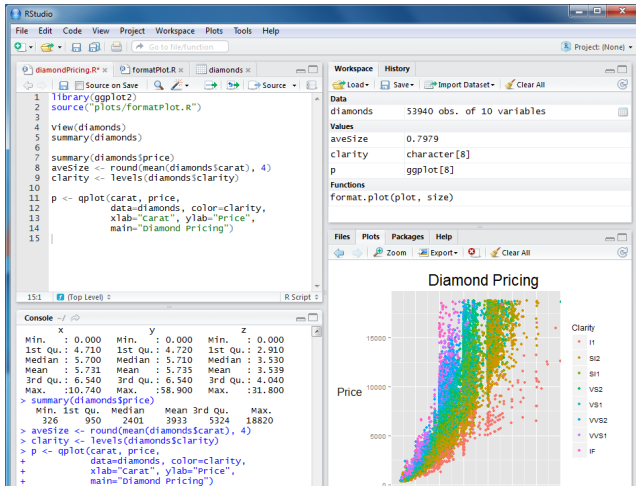
Basis R ...

- Wenn man nur R herunterlädt und installiert, sieht das so aus:
- So habe ich bis 2012 mit R gearbeitet.



... und Rstudio

- Rstudio bietet Heute sehr viel Unterstützung:
- und macht einige Themen dieses Workshops erst möglich



Aufgabe - Vorbereitung

- Prüfen Sie, ob eine Version von R auf Rechner installiert ist.
- Falls dies nicht der Fall ist, laden Sie R runter und installieren Sie R.
- Prüfen Sie, ob Rstudio installiert ist.
- Falls nicht - Installieren sie Rstudio.
- Laden Sie die R-Skripte von meinem GitHub-Account
- Erstellen Sie ein erstes Script und finden Sie das Datum mit dem Befehl `date()` und die R-version mit `sessionInfo()` heraus.

```
## [1] "Sat Jul 08 09:52:51 2017"
```

```
## R version 3.3.3 (2017-03-06)
```

```
## Platform: x86_64-w64-mingw32/x64 (64-bit)
```

```
## Running under: Windows 7 x64 (build 7601) Service Pack 1
```

```
##
```

```
## locale:
```

```
## [1] LC_COLLATE=German_Germany.1252 LC_CTYPE=German_Germany
```

Erste Schritte mit R

R ist eine Objekt-orientierte Sprache

Vektoren und Zuweisungen

- R ist eine Objekt-orientierte Sprache
- `<-` ist der Zuweisungsoperator (Shortcut: "Alt" + "-")

```
b <- c(1,2) # erzeugt ein Objekt mit den Zahlen 1 und 2
```

- Eine Funktion kann auf dieses Objekt angewendet werden:

```
mean(b) # berechnet den Mittelwert
```

```
## [1] 1.5
```

Mit den folgenden Funktionen können wir etwas über die Eigenschaften des Objekts lernen:

```
length(b) # b hat die Länge 2
```

```
## [1] 2
```

Objektstruktur - Datentypen

```
str(b) # b ist ein numerischer Vektor
```

```
## num [1:2] 1 2
```

- mehr zu den möglichen Datentypen später

Funktionen im base-Paket

Funktion	Bedeutung	Beispiel
<code>length()</code>	Länge	<code>length(b)</code>
<code>max()</code>	Maximum	<code>max(b)</code>
<code>min()</code>	Minimum	<code>min(b)</code>
<code>sd()</code>	Standardabweichung	<code>sd(b)</code>
<code>var()</code>	Varianz	<code>var(b)</code>
<code>mean()</code>	Mittelwert	<code>mean(b)</code>
<code>median()</code>	Median	<code>median(b)</code>

Diese Funktionen brauchen nur ein Argument.

Funktionen mit mehr Argumenten

Andere Funktionen brauchen mehr:

Argument	Bedeutung	Beispiel
<code>quantile()</code>	90 % Quantile	<code>quantile(b,.9)</code>
<code>sample()</code>	Stichprobe ziehen	<code>sample(b,1)</code>

Beispiel - Funktionen mit einem Argument

```
max(b)
```

```
## [1] 2
```

```
min(b)
```

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

```
sd(b)
```

```
## [1] 0.7071068
```

```
var(b)
```

```
## [1] 0.5
```

Funktionen mit einem Argument

```
mean(b)
```

```
## [1] 1.5
```

```
median(b)
```

```
## [1] 1.5
```

Funktionen mit mehr Argumenten

```
quantile(b,.9)
```

```
## 90%
```

```
## 1.9
```

```
sample(b,1)
```

```
## [1] 2
```

Übersicht Befehle

<http://cran.r-project.org/doc/manuals/R-intro.html>

An Introduction to R

Table of Contents

[Preface](#)

[1 Introduction and preliminaries](#)

[1.1 The R environment](#)

[1.2 Related software and documentation](#)

[1.3 R and statistics](#)

[1.4 R and the window system](#)

[1.5 Using R interactively](#)

[1.6 An introductory session](#)

[1.7 Getting help with functions and features](#)

[1.8 R commands, case sensitivity, etc.](#)

[1.9 Recall and correction of previous commands](#)

[1.10 Executing commands from or diverting output to a file](#)

[1.11 Data permanency and removing objects](#)

Aufgabe - Zuweisungen und Funktionen

Erzeugt einen Vektor `b` mit den Zahlen von 1 bis 5 und berechnet. . .

- ① den Mittelwert
- ② die Varianz
- ③ die Standardabweichung
- ④ die quadratische Wurzel aus dem Mittelwert

Verschiedene Datentypen

Datentyp	Beschreibung	Beispiel
numeric	ganze und reele Zahlen	5, 3.462
logical	logische Werte	FALSE, TRUE
character	Buchstaben und Zeichenfolgen	"Hallo"

Quelle: R. Munnich und M. Knobelspieß (2007): Einführung in das statistische Programmpaket R

Verschiedene Datentypen

```
b <- c(1,2) # numeric
log <- c(T,F) # logical
char <- c("A","b") # character
fac <- as.factor(c(1,2)) # factor
```

Mit `str()` bekommt man den Objekttyp.

```
str(fac)
```

```
## Factor w/ 2 levels "1","2": 1 2
```

Indizieren eines Vektors:

```
A1 <- c(1,2,3,4)
```

```
A1
```

```
## [1] 1 2 3 4
```

```
A1[1]
```

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

```
A1[4]
```

```
## [1] 4
```

```
A1[1:3]
```

```
## [1] 1 2 3
```

```
A1[-4]
```

Logische Operatoren

```
# Ist 1 größer als 2?
```

```
1>2
```

```
## [1] FALSE
```

```
1<2
```

```
## [1] TRUE
```

```
1==2
```

```
## [1] FALSE
```

Sequenzen

```
# Sequenz von 1 bis 10
```

```
1:10
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
# das gleiche Ergebnis
```

```
seq(1,10)
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

Weitere Sequenzen

```
seq(-2,8,by=1.5)
```

```
## [1] -2.0 -0.5  1.0  2.5  4.0  5.5  7.0
```

```
a <-seq(3,12,length=12)
```

```
a
```

```
## [1] 3.000000 3.818182 4.636364 5.454545 6.272727 7.090909  
## [8] 8.727273 9.545455 10.363636 11.181818 12.000000
```

```
b <- seq(to=5,length=12,by=0.2)
```

```
b
```

```
## [1] 2.8 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6 4.8 5.0
```

Reihenfolge von Argumenten

```
1:10
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
seq(1,10,1)
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
seq(length=10,from=1,by=1)
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```


Wiederholungen

```
# wiederhole 1 10 mal
```

```
rep(1,10)
```

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

```
rep("A",10)
```

```
## [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
```

Die Funktion paste

```
?paste
```

```
paste(1:4)
```

```
## [1] "1" "2" "3" "4"
```

```
paste("A", 1:6, sep = "")
```

```
## [1] "A1" "A2" "A3" "A4" "A5" "A6"
```

- Ein weiteres Beispiel:

```
paste0("A", 1:6)
```

```
## [1] "A1" "A2" "A3" "A4" "A5" "A6"
```

Wie bekommt man Hilfe

Wie bekommt man Hilfe?

- Um generell Hilfe zu bekommen:

```
help.start()
```

- Online Dokumentation für die meisten Funktionen:

```
help(name)
```

- Nutze `?` um Hilfe zu bekommen.

```
?mean
```

- `example(lm)` gibt ein Beispiel für die lineare Regression

```
example(lm)
```

- Dokumente zur Veranschaulichung und Erläuterung von Funktionen im Paket

```
browseVignettes()
```

- zu manchem Paketen gibt es Demonstrationen, wie der Code zu verwenden ist

```
demo()
```

```
demo(nlm)
```

Die Funktion apropos

- sucht alles, was mit dem eingegebenen String in Verbindung steht

```
apropos("lm")
```

```
## [1] ".__C__anova.glm"          ".__C__anova.glm.null" ".__C__g
## [4] ".__C__glm.null"          ".__C__lm"             ".__C__m
## [7] ".__C__optionalMethod"    ".colMeans"            ".lm.fit
## [10] "colMeans"                "confint.lm"           "contr.b
## [13] "dummy.coef.lm"           "getAllMethods"        "glm"
## [16] "glm.control"             "glm.fit"              "KalmanF
## [19] "KalmanLike"              "KalmanRun"            "KalmanS
## [22] "kappa.lm"                "lm"                   "lm.fit"
## [25] "lm.influence"            "lm.wfit"              "model.m
## [28] "nlm"                     "nlminb"               "predict
## [31] "predict.lm"              "residuals.glm"        "residua
## [34] "summary.glm"             "summary.lm"
```

Suchmaschine für die R-Seite

```
RSiteSearch("glm")
```


Nutzung Suchmaschinen


- Ich nutze meistens google
- Tippe:

R-project + Was ich schon immer wissen wollte



- Das funktioniert natürlich mit jeder Suchmaschine!

Stackoverflow

- Für Fragen zum Programmieren
- Ist nicht auf R fokussiert, es gibt aber viele Diskussionen zu R
- Sehr detaillierte Diskussionen

 **stackoverflow**

Questions Jobs Documentation BETA Tags Users

  [Log In](#) [Sign Up](#)

Tagged Questions

info newest **8** featured frequent votes active unanswered

R is a free, open-source programming language and software environment for statistical computing, bioinformatics, and graphics. Please supplement your question with a minimal reproducible example. Use `dput()` for data and specify all non-base packages with library calls. For statistical questions ...

[learn more...](#) [top users](#) [synonyms \(2\)](#) [r jobs](#)

1776 votes

22 answers

147k views

How to make a great R reproducible example?


When discussing performance with colleagues, teaching, sending a bug report or searching for guidance on mailing lists and here on SO, a reproducible example is often asked and always helpful. What ...

[r](#) [r-faq](#)

[community wiki](#)
11 revs, 8 users 54%
[Hack-R](#)

22,187 frequent questions tagged

[r](#) [about »](#)

 **R Language**
DOCUMENTATION

[Find a request to handle](#) or [browse 121 topics](#).

Related Tags

[ggplot2](#) × 2875

[dataframe](#) × 1351

[plot](#) × 1105

Quick R

- Immer eine Seite mit Beispielen und Hilfe zu einem Thema
- Beispiel: Quick R - Getting Help

Weitere Links

- Übersicht - Hilfe bekommen in R
- Eine Liste mit HowTo's
- Eine Liste der wichtigsten R-Befehle

Ein Schummelzettel - Cheatsheet

<https://www.rstudio.com/resources/cheatsheets/>

Base R Cheat Sheet

Getting Help

Accessing the help files

?mean
Get help of a particular function.

help.search("weighted.mean")
Search the help files for a word or phrase.

help(package = "dplyr")
Find help for a package.

More about an object

str(object)
Get a summary of an object's structure.

class(object)
Find the class an object belongs to.

Using Libraries

install.packages("dplyr")
Download and install a package from CRAN.

library(dplyr)
Load the package into the session, making all R/Functions available to use.

dplyr::select
Use a particular function from a package.

detach()
Load a loaded in-database into the environment.

Working Directory

getwd()
Find the current working directory (where inputs are found and outputs are sent).

setwd("C:/Users/you/path")
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

<code>c(1, 4, 8)</code>	<code>1 4 8</code>	Arithmetic vector
<code>1:4</code>	<code>1 2 3 4</code>	Integer sequence
<code>rep(1, 4, length=3)</code>	<code>1 1 1 1</code>	A complex sequence
<code>rep(1:4, times=2)</code>	<code>1 2 3 4 1 2 3 4</code>	Repeating vector
<code>rep(1:4, each=2)</code>	<code>1 1 2 2 3 3 4 4</code>	Repeating elements

Vector Functions

<code>sort(x)</code>	Return sorted
<code>unique(x)</code>	Return unique values
<code>length(x)</code>	Get length of vector

Selecting Vector Elements

By Position

<code>x[4]</code>	The fourth element
<code>x[-4]</code>	All but the fourth
<code>x[2:4]</code>	Elements two to four
<code>x[-1:4]</code>	All elements except first to four
<code>x[c(1, 5)]</code>	Elements one and five

By Value

<code>x[x == 10]</code>	Elements which are equal to 10
<code>x[x < 0]</code>	All elements less than zero
<code>x[x %>% 2]</code>	Elements in the set 1, 2, 5

Named Vectors

<code>x["apple"]</code>	Element with name 'apple'
-------------------------	---------------------------

Programming

For Loop

```
for (something in something) {  
  do something  
}
```

Example

```
for (i in 1:10) {  
  x[i] = 5 * i  
  print(i)  
}
```

While Loop

```
while (condition) {  
  do something  
}
```

Example

```
while (i < 10) {  
  print(i)  
  i = i + 1  
}
```

If Statements

```
if (condition) {  
  do something  
} else {  
  do something different  
}
```

Example

```
if (i < 5) {  
  print("Yes")  
} else {  
  print("No")  
}
```

Functions

```
function_name <- function(parameters) {  
  do something  
  return(something_returned)  
}
```

Example

```
squares <- function(x) {  
  squared <- x^2  
  return(squared)  
}
```

Reading and Writing Data

Input	Output	Description
<code>df <- read.table("file.csv")</code>	<code>write.table(df, "file.csv")</code>	Read and write a delimited text file.
<code>df <- read.csv("file.csv")</code>	<code>write.csv(df, "file.csv")</code>	Read and write a comma-separated value file. This is a special case of read/write delimited.
<code>load("file.Rsave")</code>	<code>save(df, file = "file.Rsave")</code>	Read and write an R data file, a R-specific save file.

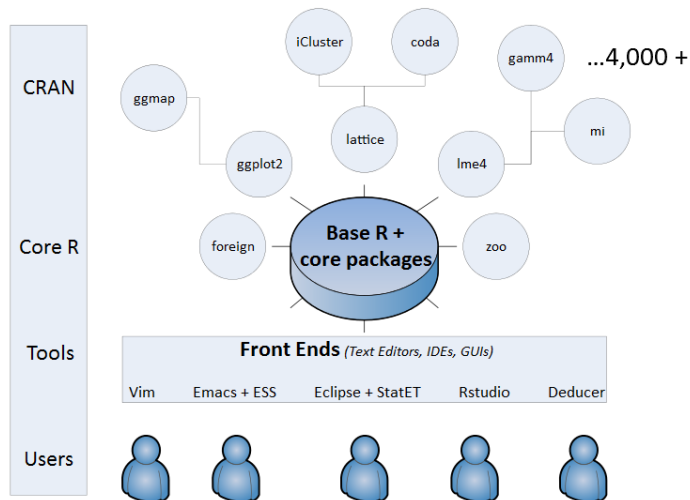
<code>is.na(x)</code>	Are equal	<code>is.null(x)</code>	Is null	<code>is.nan(x)</code>	Is NaN
<code>is.infinite(x)</code>	Is infinite	<code>is.na(x)</code>	Is not a number	<code>is.nan(x)</code>	Is NaN

Modularer Aufbau

Wo sind die Routinen enthalten

- Viele Funktionen sind im Basis-R enthalten
- Viele spezifische Funktionen sind in zusätzlichen Bibliotheken integriert
- R kann modular erweitert werden durch sog. packages bzw. libraries
- Auf CRAN werden die wichtigsten packages gehostet (im Moment 10430)
- Weitergehende Pakete finden sich z.B. bei bioconductor

Übersicht R-Pakete



Installation

```
install.packages("lme4")
```

```
library(lme4)
```


Installation von Paketen mit RStudio

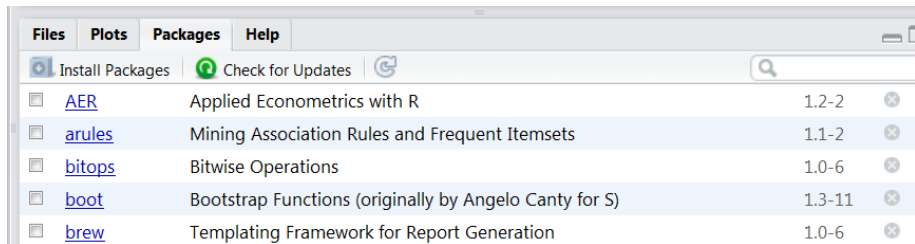
The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains the R code `setwd("D:/Projekte/Rpackages/germanwehr/Rfunctions")`.
- Environment:** Shows "Global Environment" and the message "Environment is empty".
- Files:** A list of installed and available packages is shown in a table.
- Console:** Displays the output of the `setwd()` command and a message about the R collaborative project.

Package Name	Description	Version
AER	Applied Econometrics with R	1.2-2
arules	Mining Association Rules and Frequent Itemsets	1.1-2
bitops	Bitwise Operations	1.0-6
boot	Bootstrap Functions (originally by Angelo Canty for S)	1.3-11
brew	Templating Framework for Report Generation	1.0-6
car	Companion to Applied Regression	2.0-19
caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17
class	Functions for Classification	7.3-10
cluster	Cluster Analysis Extended Rousseeuw et al.	1.15.2
codetools	Code Analysis Tools for R	0.2-8
colorspace	Color Space Manipulation	1.2-4
compiler	The R Compiler Package	3.1.0
DAAG	Data Analysis And Graphics data and functions	1.18

R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on how to cite R or R packages in publications.

Vorhandene Pakete und Installation



Übersicht viele nützliche Pakete:

- Luhmann - Tabelle mit vielen nützlichen Paketen

Weitere interessante Pakete:

- Paket für den Import/Export - foreign
- Pakete für Survey Sampling
- xtable Paket für die Integration von Latex und R (xtable Galerie)
- Paket zur Erzeugung von Dummies
- Multivariate Normalverteilung
- Paket für Karten

Pakete installieren

Pakete von CRAN Server installieren

```
install.packages("lme4")
```

Pakete von Bioconductor Server installieren

```
source("https://bioconductor.org/biocLite.R")  
biocLite(c("GenomicFeatures", "AnnotationDbi"))
```

Pakete von Github installieren

```
install.packages("devtools")  
library(devtools)  
  
install_github("hadley/ggplot2")
```

Wie bekomme ich einen Überblick

- Pakete entdecken, die neulich auf CRAN hochgeladen wurden
- Pakete die in letzter Zeit von CRAN heruntergeladen wurden
- Quick-list nützlicher Pakete
- Beste Pakete für Datenbearbeitung und Analyse
- Die 50 meist genutzten Pakete

CRAN Task Views

- Zu einigen Themen sind alle Möglichkeiten in R zusammengestellt. (Übersicht der Task Views)
- Zur Zeit gibt es 35 Task Views
- Alle Pakete eines Task Views können mit folgendem Befehl installiert werden:

```
install.packages("ctv")  
library("ctv")  
install.views("Bayesian")
```

CRAN Task Views

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data

Aufgabe - Zusatzpakete

Gehen Sie auf <https://cran.r-project.org/> und suchen Sie in dem Bereich, wo die Pakete vorgestellt werden, nach Paketen,...

- die für die deskriptive Datenanalyse geeignet sind.
- um Regressionen zu berechnen
- um fremde Datensätze einzulesen (z.B. SPSS-Daten)
- um mit großen Datenmengen umzugehen

Die Exportformate von R

- In R werden offene Dateiformate bevorzugt
- Genauso wie `read.X()` Funktionen stehen viele `write.X()` Funktionen zur Verfügung
- Das eigene Format von R sind sog. Workspaces (`.RData`)

Beispieldatensatz erzeugen

```
A <- c(1,2,3,4)
```

```
B <- c("A","B","C","D")
```

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

A	B
1	A
2	B
3	C
4	D

Überblick Daten Import/Export

- wenn mit R weitergearbeitet wird, eignet sich das .RData Format am Besten:

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

- Der Datensatz kann dann mit load wieder eingelesen werden

```
load("mydata.RData")
```

Daten in .csv Format abspeichern

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

- Wenn mit Deutschem Excel weitergearbeitet werden soll, eignet sich write.csv2 besser

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

- Sonst sieht das Ergebnis so aus:

	A	
1	,"A","B"	
2	1,1,"A"	
3	2,2,"B"	
4	3,3,"C"	
5	4,4,"D"	
6		

Das Paket xlsx

R xlsx package : A quick start guide to manipulate Excel files in R

 AdChoices

[Microsoft Excel](#)

[Download for Java](#)

[Excel Tutorial](#)

- 
- [Install and load xlsx package](#)
 - [Read an Excel file](#)
 - [Write data to an Excel file](#)

```
library(xlsx)
write.xlsx(mydata, file="mydata.xlsx")
```

Reading/Writing Stata (.dta) files with Foreign

December 4, 2012

By `is.R()`

- Funktionen im Paket foreign

R topics documented:

lookup.xport	2
read.arff	3
read.dbf	4
read.dta	5
read.epiinfo	7
read.mtp	8
read.octave	9
read.spss	10
read.ssd	12

Daten in stata Format abspeichern

```
library(foreign)  
write.dta(mydata, file="data/mydata.dta")
```

```
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.

Daten als .sav abspeichern (SPSS)

```
library("rio")  
# create file to convert  
  
export(mtcars, "data/mtcars.sav")
```

Dateiformate konvertieren

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

```
# convert Stata to SPSS
```

```
convert("data/mtcars.dta", "data/mtcars.sav")
```

- Quick R für das Exportieren von Daten:
- Hilfe zum Export auf dem cran Server
- Daten aus R heraus bekommen

Datenimport

Datenimport



Dateiformate in R

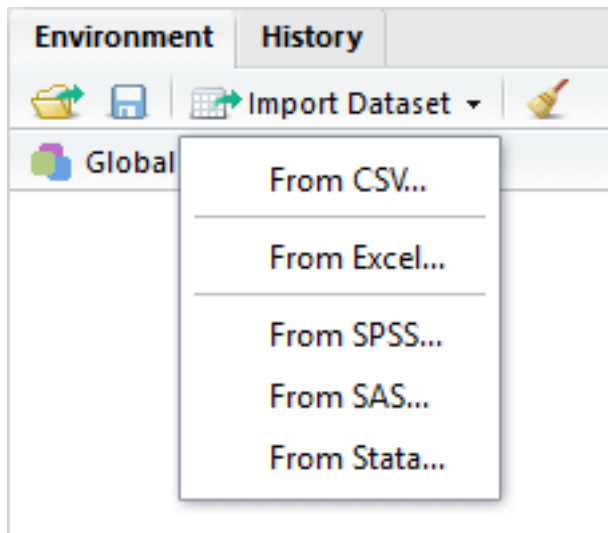
- Von R werden quelloffene, nicht-proprietäre Formate bevorzugt
- Es können aber auch Formate von anderen Statistik Software Paketen eingelesen werden
- R-user speichern Objekte gerne in sog. Workspaces ab
- Auch hier jedoch gilt: (fast) alles andere ist möglich

Formate - base package

R unterstützt von Haus aus schon einige wichtige Formate:

- CSV (Comma Separated Values): `read.csv()`
- FWF (Fixed With Format): `read.fwf()`
- Tab-getrennte Werte: `read.delim()`

Datenimport leicht gemacht mit Rstudio



CSV aus dem Web einladen

- Datensatz:

<https://data.montgomerycountymd.gov/api/views/6rqk-pdub/rows.csv?accessType=DOWNLOAD>

- Datenimport mit Rstudio

Import Text Data

File/URL

<https://data.montgomerycountymd.gov/api/views/6rqk-pdub/rows.csv?accessType=DOWNLOAD>

Data Preview:

Full Name (character) *	Gender (character) *	Current Annual Salary (character) *	2015 Gross Pay Received (character) *	2015 Overtime Pay (character) *	Department (character) *	Department Name (character) *	Division (character) *	Assignment Category (character) *	Position Title (character) *	Underfilled Job Title (character)
Aarhus, Pam J.	F	\$68876.16	\$72336.79	NA	POL	Department of Police	MIS Information Management and Technology Divisi...	Fulltime-Regular	Office Services Coordinator	NA
Aaron, David J.	M	\$96908.09	\$101857.00	\$6640.99	POL	Department of Police	ISB Major Crimes Division Fugitive Section	Fulltime-Regular	Master Police Officer	NA
Aaron, Marsha M.	F	\$104196.06	\$103019.73	NA	HHS	Department of Health and Human Services	Adult Protective and Case Management Services	Fulltime-Regular	Social Worker IV	NA
Abalos, Coffred A.	M	\$50697.79	\$54181.46	\$6445.15	COR	Correction and Rehabilitation	PRRS Facility and Security	Fulltime-Regular	Resident Supervisor II	NA
Ababu, Essays	M	\$59391.00	\$93468.35	NA	HCA	Department of Housing and Community Affairs	Single Family Housing Program	Fulltime-Regular	Planning Specialist III	NA
Abdmonem, Drea B.	M	\$67715.00	\$81392.40	\$10027.11	POL	Department of Police	PSB 6th District Special Assignment Team	Fulltime-Regular	Police Officer III	NA
Abdelmonem, Marwan M.	M	\$62286.30	\$59663.27	NA	HHS	Department of Health and Human Services	Head Start	Fulltime-Regular	Administrative Specialist II	NA
Abdul-Chan, Hasiyah J.	F	\$45828.92	\$46783.23	\$6.38	POL	Department of Police	PSB Traffic Division Automated Traffic Enforcement S...	Fulltime-Regular	Police Aide	NA
Abduljabbar, Saeed	M	\$61040.57	\$66861.98	\$6569.81	DCS	Department of General Services	Facilities Maintenance	Fulltime-Regular	Electrician I	NA
Abdur-Rahem, Mikael A.	M	\$56404.96	\$71943.08	\$15342.84	DOT	Department of Transportation	Transit Silver Spring Ride On	Fulltime-Regular	Bus Operator	NA
Abene, Hiruth	F	\$151585.60	\$164945.06	NA	HHS	Department of Health and Human Services	STD and HIV Services	Parttime-Regular	Medical Doctor III - Physician	NA
Abene, Zekarias S.	M	\$44825.99	\$51693.47	\$5240.75	DOT	Department of Transportation	Transit Nicholson Ride On	Fulltime-Regular	Bus Operator	NA
Abedin, Amiraza	M	\$39062.00	\$450.00	NA	DOT	Department of Transportation	Transportation Management	Fulltime-Regular	Traffic Management Technician II	Traffic Management Technici...
Abelove, Sherry R.	F	\$93436.50	\$90833.10	NA	HHS	Department of Health and Human Services	Adult Protective and Case Management Services	Fulltime-Regular	Social Worker III	NA
Aben, Yoseph M.	M	\$117811.00	\$115786.22	NA	DTS	Department of Technology Services	EASD - ERP Applications Support	Fulltime-Regular	Senior Information Technology Specialist	NA
Abi Jamaa, Rania F.	F	\$53009.99	\$46850.36	NA	LIB	Department of Public Libraries	Olney Library	Fulltime-Regular	Library Assistant I	NA
Abijomas, Ryan Z.	M	\$17288.00	\$14655.53	NA	LIB	Department of Public Libraries	Silver Spring Library	Parttime-Regular	Library Desk Assistant	NA
Abita, Lydia B.	F	\$40429.58	\$41746.34	\$5500.53	DOT	Department of Transportation	Transit Caldersburg Ride On	Fulltime-Regular	Bus Operator	NA
Abikarim, Maral	F	\$20925.51	\$9976.52	\$45.28	POL	Department of Police	PSB Traffic Division School Safety Section	Parttime-Regular	Crossing Guard	NA
Abouraya, Nadia L.	F	\$16602.01	\$15592.92	NA	HHS	Department of Health and Human Services	Community Support Network for People with Disabilities	Parttime-Regular	Office Clerk	NA

Previewing first 50 entries.

Der Arbeitsspeicher

So findet man heraus, in welchem Verzeichnis man sich gerade befindet

```
getwd()
```

So kann man das Arbeitsverzeichnis ändern:

Man erzeugt ein Objekt in dem man den Pfad abspeichert:

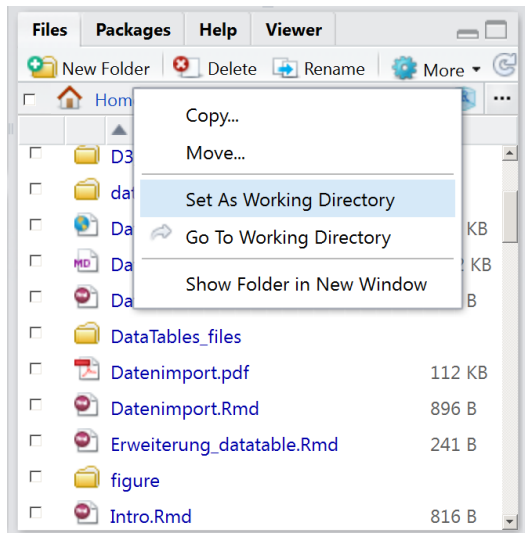
```
main.path <- "C:/" # Beispiel für Windows  
main.path <- "/users/Name/" # Beispiel für Mac  
main.path <- "/home/user/" # Beispiel für Linux
```

Und ändert dann den Pfad mit `setwd()`

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

Bei Windows ist es wichtig Slashes anstelle von Backslashes zu verwenden.

Alternative - Arbeitsspeicher



Import von Excel-Daten

- `library(foreign)` ist für den Import von fremden Datenformaten nötig
- Wenn Excel-Daten vorliegen - als `.csv` abspeichern
- Dann kann `read.csv()` genutzt werden um die Daten einzulesen.
- Bei Deutschen Daten kann es sein, dass man `read.csv2()` wegen der Komma-Separierung braucht.

```
library(foreign)
```

```
?read.csv
```

```
?read.csv2
```

CSV Dateien einlesen

Zunächst muss das Arbeitsverzeichnis gesetzt werden, in dem sich die Daten befinden:

```
Dat <- read.csv("schuldaten_export.csv")
```

Wenn es sich um Deutsche Daten handelt:

```
Dat <- read.csv2("schuldaten_export.csv")
```

Das Paket readr

```
install.packages("readr")
```

```
library(readr)
```

- [readr auf dem Rstudio Blogg](#)

Import von Excel-Daten

- `library(readr)` ist für den Import von fremden Datenformaten hilfreich
- Wenn Excel-Daten vorliegen - als .csv abspeichern

```
url <- "https://raw.githubusercontent.com/Japhilko/  
GeoData/master/2015/data/whcSites.csv"
```

```
whcSites <- read.csv(url)
```

Der Beispieldatensatz

```
head(data.frame(whcSites$name_en,whcSites$category))
```

```
##                               whcSit  
## 1 Cultural Landscape and Archaeological Remains of the Bami  
## 2                               Minaret and Archaeological Rema  
## 3                               Historic Centres of Berat and G  
## 4  
## 5                               Al Qal'a of F  
## 6                               M  
##   whcSites.category  
## 1      Cultural  
## 2      Cultural  
## 3      Cultural  
## 4      Cultural  
## 5      Cultural  
## 6      Cultural
```


Das Paket haven

Import and Export 'SPSS', 'Stata' and 'SAS' Files

```
install.packages("haven")
```

```
library(haven)
```

- Das R-Paket haven auf dem Rstudio Blogg

SPSS Dateien einlesen

- Zunächst muss wieder der Pfad zum Arbeitsverzeichnis angegeben werden.
- SPSS-Dateien können auch direkt aus dem Internet geladen werden:

```
library(haven)
mtcars <- read_sav(
  "https://github.com/Japhilko/RInterfaces/raw/master/
  data/mtcars.sav")
```

foreign kann ebenfalls zum Import genutzt werden

```
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)
```

stata Dateien einlesen

```
library(haven)
oecd <- read_dta("https://github.com/Japhilko/IntroR/  
raw/master/2017/data/oecd.dta")
```

- Einführung in Import mit R (is.R)

Das Paket rio

```
install.packages("rio")  
  
library("rio")  
x <- import("mtcars.csv")  
y <- import("mtcars.rds")  
z <- import("mtcars.dta")
```

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

Datenmanagement ähnlich wie in SPSS oder Stata

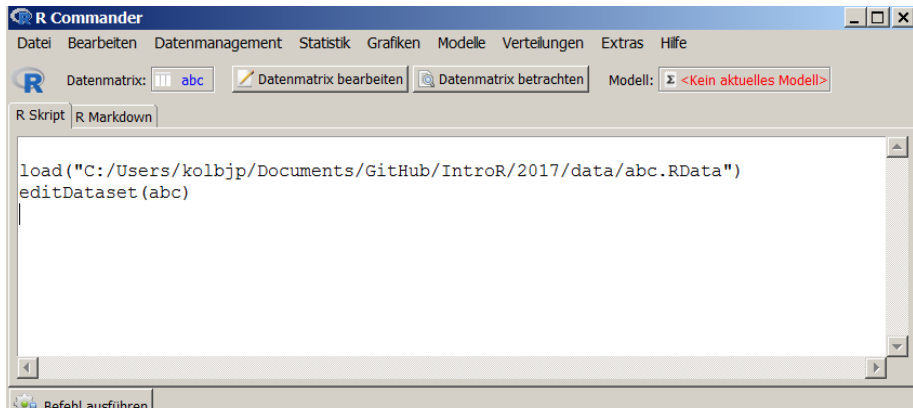
```
install.packages("Rz")  
library(Rz)
```

Weitere Alternative Rcmdr

```
install.packages("Rcmdr")
```

- Funktioniert auch mit Rstudio

```
library(Rcmdr)
```



Aufgabe

- Gehen Sie auf meine Github Seite

`https://github.com/Japhilko/RSocialScience/tree/master/data`

- Importieren Sie den Datensatz `GPanel.dta`

Data Frames

Beispieldaten einlesen:

```
library(foreign)
dat<-read.dta("https://github.com/Japhilko/RSocialScience/
              raw/master/data/GPanel.dta")

typeof(dat)

## [1] "list"
```

In Dataframe übertragen

Diese beiden Vektoren zu einem data.frame verbinden:

```
Daten <- data.frame(dat)
```

Anzahl der Zeilen/Spalten herausfinden

```
nrow(Daten) # Zeilen
```

```
## [1] 100
```

```
ncol(Daten) # Spalten
```

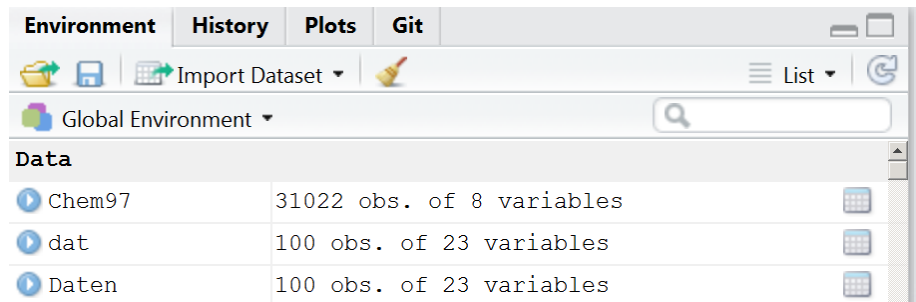
```
## [1] 23
```

Die Daten anschauen

- die ersten zeilen anschauen

`head(Daten)`

- Übersicht mittels Rstudio



The screenshot shows the RStudio interface's Environment pane. At the top, there are tabs for 'Environment', 'History', 'Plots', and 'Git'. Below these tabs is a toolbar with icons for file operations and a search bar. The main area of the pane is titled 'Global Environment' and contains a table of loaded data objects.

Data	
▶ Chem97	31022 obs. of 8 variables
▶ dat	100 obs. of 23 variables
▶ Daten	100 obs. of 23 variables

Indizieren

Indizieren eines dataframe:

```
Daten[1,1]
```

```
## [1] Eher zufrieden
```

```
## 7 Levels: Item nonresponse Sehr zufrieden ... Weiß nicht
```

```
Daten[2,]
```

```
##          a11c019a          a11c020a          a11c021a
```

```
## 2 Sehr zufrieden Eher unzufrieden Eher unzufrieden Stimme e
```

```
##          a11c023a          a11c024a
```

```
## 2 Stimme eher zu Stimme voll und ganz zu Eher niedrigeren I
```

```
##          a11c026a          a11c027a a11c028a a1
```

```
## 2 Mehrmals die Woche Mindestens einmal im Monat Täglich T
```

```
##          a11c031a          a11c032a a11c033a
```

```
## 2 Mindestens einmal im Monat Mehrmals im Monat Seltener
```

```
##          a11c034a b3g0020a a11c054a
```

Operatoren um Subset für Datensatz zu bekommen

Diese Operatoren eignen sich gut um Datensätze einzuschränken

```
Dauer <- as.numeric(Daten$bazq020a)
head(Daten[Dauer>20,])
```

```
##          a11c019a          a11c020a          a11c021a
## 2  Sehr zufrieden Eher unzufrieden Eher unzufrieden Stimme
## 3  Eher zufrieden  Sehr zufrieden Eher unzufrieden Stimme
## 5  Eher zufrieden  Eher zufrieden  Eher zufrieden
## NA          <NA>          <NA>          <NA>
## 9  Sehr zufrieden  Eher zufrieden  Sehr zufrieden
## 15 Sehr zufrieden  Sehr zufrieden  Sehr zufrieden
##          a11c023a          a11c024a
## 2          Stimme eher zu Stimme voll und ganz zu
## 3  Stimme eher nicht zu          Stimme eher zu
## 5          Stimme eher zu          Stimme eher zu
## NA          <NA>          <NA>
```

Einschränken mit dem Paket tidyverse

- einfacher geht es mit dem Paket tidyverse

```
library(tidyverse)
filter(Daten, Dauer>20)
```

```
##                               a11c019a
## 1      Sehr zufrieden          Eher un
## 2      Eher zufrieden          Sehr
## 3      Eher zufrieden          Eher
## 4      Sehr zufrieden          Eher
## 5      Sehr zufrieden          Sehr
## 6      Eher zufrieden          Sehr
## 7      Sehr zufrieden          Sehr
## 8      Sehr zufrieden          Sehr
## 9      Eher zufrieden          Eher
## 10     Sehr zufrieden          Eher
## 11     Eher zufrieden          Eher
```

Datensätze einschränken

```
SEX <- Daten$a11d054a
```

```
Daten[SEX=="Männlich",]
```

```
##                                a11c019a
## 1      Eher zufrieden Weder zufrieden noch un
## 2      Sehr zufrieden                                Eher un
## 3      Eher zufrieden                                Sehr
## 4      Eher zufrieden                                Sehr
## 5      Eher zufrieden                                Eher
## 7      Eher zufrieden                                Eher
## 9      Sehr zufrieden                                Eher
## 12     Sehr zufrieden                                Eher
## 15     Sehr zufrieden                                Sehr
## 16     Sehr zufrieden                                Sehr
## 17 Weder zufrieden noch unzufrieden                Eher un
```


Weitere wichtige Optionen

Ergebnis in ein Objekt speichern

```
subDat <- Daten[Dauer>20,]
```

mehrere Bedingungen können mit

& verknüpft werden:

```
Daten[Dauer>18 & SEX=="Männlich",]
```

```
##                                a11c019a
```

```
## 2                            Sehr zufrieden                            Eher un
```

```
## 3                            Eher zufrieden                            Sehr
```

```
## 5                            Eher zufrieden                            Eher
```

```
## 9                            Sehr zufrieden                            Eher
```

```
## 15                           Sehr zufrieden                            Sehr
```

```
## 18                           Eher zufrieden                            Sehr
```

```
## 20                           Eher zufrieden                            Eher
```

```
## 29                           Sehr zufrieden                            Sehr
```

```
## 41                           Sehr zufrieden                            Eher
```

Die Nutzung einer Sequenz

Daten[1:3,]

```
##          a11c019a          a11c020a          a
## 1 Eher zufrieden Weder zufrieden noch unzufrieden Sehr zu
## 2 Sehr zufrieden          Eher unzufrieden Eher unzu
## 3 Eher zufrieden          Sehr zufrieden Eher unzu
##          a11c022a          a11c023a
## 1 Stimme eher nicht zu      Stimme eher zu      Stimme
## 2 Stimme eher nicht zu      Stimme eher zu Stimme voll und
## 3 Stimme eher nicht zu Stimme eher nicht zu      Stimme
##          a11c025a          a11c026a
## 1 Eher niedrigeren Lebensstandard      Seltener
## 2 Eher niedrigeren Lebensstandard Mehrmals die Woche
## 3      Denselben Lebensstandard      Täglich
##          a11c027a a11c028a a11c029a          a
## 1      Mehrmals die Woche      Täglich      Täglich      T
```

Variablen Labels

```
library(foreign)
dat <- read.dta("https://github.com/Japhilko/RSocialScience/blob/master/data/1998.dta")

attributes(dat)

var.labels <- attr(dat, "var.labels")
```

- Genauso funktioniert es auch mit dem Paket haven

```
library(haven)
dat2 <- read_dta(
  "https://github.com/Japhilko/RSocialScience/blob/master/data/1998.dta")
var.labels2 <- attr(dat, "var.labels")
```

Die Spaltennamen umbenennen

- Mit `colnames` bekommt man die Spaltennamen angezeigt

```
colnames(dat)
```

```
## [1] "a11c019a" "a11c020a" "a11c021a" "a11c022a" "a11c023a"  
## [7] "a11c025a" "a11c026a" "a11c027a" "a11c028a" "a11c029a"  
## [13] "a11c031a" "a11c032a" "a11c033a" "a11c034a" "bazq020a"  
## [19] "a11d056z" "a11d092a" "a11c100a" "a11c111a" "a11c109a"
```

- So kann man die Spaltennamen umbenennen:

```
colnames(dat) <- var.labels
```

- Analog geht das für die Reihennamen

```
rownames(dat)
```

```
## [1] "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

Indizieren

- Das Dollarzeichen kann man auch nutzen um einzelne Spalten anzusprechen

```
head(dat$a11c019a)
```

```
## [1] Eher zufrieden Sehr zufrieden Eher zufrieden Eher zufried  
## [5] Eher zufrieden Sehr zufrieden  
## 7 Levels: Item nonresponse Sehr zufrieden ... Weiß nicht
```

```
dat$a11c019a[1:10]
```

```
## [1] Eher zufrieden Sehr zufrieden Eher zufrieden Eher zufried  
## [5] Eher zufrieden Sehr zufrieden Eher zufrieden Eher zufried  
## [9] Sehr zufrieden Sehr zufrieden  
## 7 Levels: Item nonresponse Sehr zufrieden ... Weiß nicht
```

Auf Spalten zugreifen

- Wie bereits beschrieben kann man auch Zahlen nutzen um auf die Spalten zuzugreifen

```
head(dat[,1])
```

```
head(dat[, "a11c019a"]) # liefert das gleiche Ergebnis
```

Tools for Working with Categorical Variables (Factors)

```
library("forcats")
```

- `fct_collapse` - um Faktorlevel zusammenzufassen
- `fct_count` - um die Einträge in einem Faktor zu zählen
- `fct_drop` - Unbenutzte Levels raus nehmen

Rekodieren

```
library(car)
```

```
head(dat$a11c020a)
```

```
## [1] Weder zufrieden noch unzufrieden Eher unzufrieden  
## [3] Sehr zufrieden                      Sehr zufrieden  
## [5] Eher zufrieden                      Eher zufrieden  
## 7 Levels: Item nonresponse Sehr zufrieden ... Weiß nicht
```

```
head(recode(dat$a11c020a,"'Eher unzufrieden'='A';else='B'"))
```

```
## [1] B A B B B B  
## Levels: A B
```


Das Paket tibble

```
install.packages("tibble")
```

```
library(tibble)
```

```
gpanel1 <- as_tibble(dat)
```

```
gpanel1
```

```
## # A tibble: 100 × 23
```

```
##           a11c019a                               a11c020a
```

```
## *           <fctr>                               <fctr>
```

```
## 1  Eher zufrieden Weder zufrieden noch unzufrieden    Sehr z
```

```
## 2  Sehr zufrieden                               Eher unzufrieden Eher unz
```

```
## 3  Eher zufrieden                               Sehr zufrieden Eher unz
```

```
## 4  Eher zufrieden                               Sehr zufrieden Eher z
```

```
## 5  Eher zufrieden                               Eher zufrieden Eher z
```

```
## 6  Sehr zufrieden                               Eher zufrieden Eher z
```

```
## 7  Eher zufrieden                               Eher zufrieden Eher z
```

```
## 8  Eher zufrieden                               Eher zufrieden Eher z
```

Schleifen

```
erg <- vector()

for (i in 1:ncol(dat)){
  erg[i] <- length(table(dat[,i]))
}
```

Fehlende Werte ausschließen

- Mathe-Funktionen haben in der Regel einen Weg, um fehlende Werte in ihren Berechnungen auszuschließen.
- `mean()`, `median()`, `colSums()`, `var()`, `sd()`, `min()` und `'max()` all take the `na.rm` argument.

Fehlende Werte umkodieren

```
Daten$bazq020a[Daten$bazq020a==-99] <- NA
```

- Quick-R zu fehlenden Werten
- Fehlende Werte rekodieren

Weitere Links

- Tidy data - das Paket `tidyr`
- Die tidyverse Sammlung
- Data wrangling with R and RStudio