Intro Datenanalyse mit R - zweiter Teil

Jan-Philipp Kolb

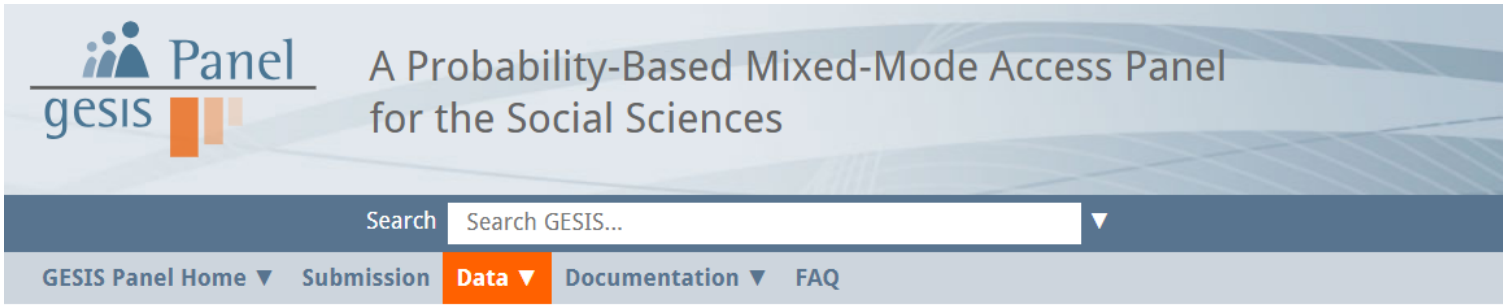
09 Mai, 2019

Das GESIS Panel

Das GESIS Panel

- Wahrscheinlichkeitsbasiertes Access Panel für Personen: - Allgemeine Bevölkerung in Deutschland, deutschsprachig, im Alter von 18-70 Jahren
- Panelisten wurden aus den Melderegistern rekrutiert - (270 Sampling Points) 7599 face-to-face Interviews (CAPI)
- Ungefähr 5000 Panelisten (Basis Stichprobe / erste Kohorte 2014)

Das GESIS Panel Campus File



GESIS Panel Campus File

The GESIS Panel Campus File is intended for teaching purposes only. It provides interested parties (e.g., students or lecturers) with an opportunity to work with an easily accessible, high quality panel dataset that should satisfy many requirements set forth in the interested parties' curriculum. In exchange for easy accessibility, the GESIS Panel Campus File contains only selected portions of the original GESIS Panel scientific use file. In order to ensure the anonymity of our panelists, the GESIS Panel Campus File contains only a **random 25% sample** of all panel members who were still active at the start of the respective year. For the current Campus File, the final sample size is N=1222. An exact documentation of studies and variables included in this year's GESIS Panel Campus File can be found in the data description and codebook.

Access to the GESIS Panel Campus File (ZA5666 / doi:10.4232/1.12749) can be acquired at the GESIS Data Archive.

Researchers interested in using GESIS Panel data for scientific publication purposes should use the full dataset (either the GESIS Panel Standard Edition or the GESIS Panel Extended Edition). Data users are strongly advised not to use the GESIS Panel Campus File for scientific publications other than for teaching purposes.

Download data

- Übersichtsseite: **GESIS Panel Campus File**
- Registrierung notwendig

Links für den Download:

- **Download** `.csv`
- **Download** `.sav`
- **Download** `**14.dta`


ZA5666_v1-0-0.csv


ZA5666_v1-0-0.sav


ZA5666_v1-0-0_Stata14.dta

Erster Eindruck der Daten

- bezf070a - Mitglied soziales Netzwerk
- bdao032a - Pflegeprodukte 1: Schminke
- bdao048a - Solariumbesuch
- bfam056a - MPMB Strategies: Dinge regelmäßig zur selben Zeit tun

bezf070a	bdao032a	bdao048a	bfam056a
Ja	Nicht genannt	Nie	Sehr oft
Nein	Nicht genannt	Nie	Gelegentlich
Nein	Nicht genannt	Nie	Eher selten
Nein	Genannt	Nie	Eher oft
Nein	Genannt	Nie	Unit nonresp
Unit nonresponse	Unit nonresponse	Unit nonresponse	Not in panel

Die Variablennamen im GESIS Panel

```
## [1] "bbaj096a" "bbak116a" "bczd034a" "bcam106a" "bczq017
```

- Die ersten beiden Buchstaben enthalten die Welle:

year	waves	numbers
2013	aa,ab,ac	1-3
2014	ba,bb,bc,bd,be,bf	4-9
2015	ca,cb,cc,cd,ce,cf	10-15
2016	da,db,dc,dd,de,df	16-21
2017	ea,eb,ec,ed,ee,ef	22-27
2018	fa,fb,fc,fd,fe,ff	28-33
2019	ga,gb	34-35

- Bis zum jetzigen Zeitpunkt sind 35 gelaufen

Die Variablennamen im GESIS Panel II

- Die Stellen drei und vier geben Information über die Studie:

aa: Lifestyles in everyday life.....

ab: How election outcomes shape public opinion .....

ac: Time perspective survey - Short German version.....

ad: European economic crisis' effect on support for the European Union.

ae: Scale label experiments.....

- Die Stellen fünf, sechs und sieben indizieren die Variablennummer
- Die letzte Stelle enthält die Information, ob es sich um eine originale Variable (a) oder eine synthetische Variable handelt (b,c,d,e,...)

Variablennamen im GESIS Panel

Beispiel Geburtsdatum - cfzh072c

```
## [1] "cfzh072c"

## [1] "Welle:  cf"

## [1] "Studie:  zh"

## [1] "Variablen Nr.:  072"

## [1] "Synthetische Variable?:  c"
```

Die Variablen im Campus File

https://rpubs.com/Japhilko82/VarsGesisPanel

RPubs brought to you by RStudio

Show 10 entries Search:

	Variablen	Variablen.Namen	Zahl.der.Ausprägungen	Häufigste.Ausprägung	Erhebungsjahr
	All	All	All	All	All
1	z000001z	Personen ID - Campus File	1222	111994770	2013
2	z000002z	Studiennummer des Archivs	1	ZA5666	2013
3	z000003z	Versionskennung und -datum des Archivs	1	1-0-0 2017-03-07	2013
4	z000005z	doi	1	10.4232/1.12749	2013
5	a11c019a	Zufriedenheit Leben in Wohnort	7	Item nonresponse	2013
6	a11c020a	Zufriedenheit Leben in Deutschland	7	Item nonresponse	2013

Wellen im Campus File

- Welche Wellen sind im Campus file?
- Anzahl Variablen pro Welle im Campus File:

```
## waves
##  a1  ba  bb  bc  bd  be  bf  z0
## 171 171 154 155 224 185 128   4
```

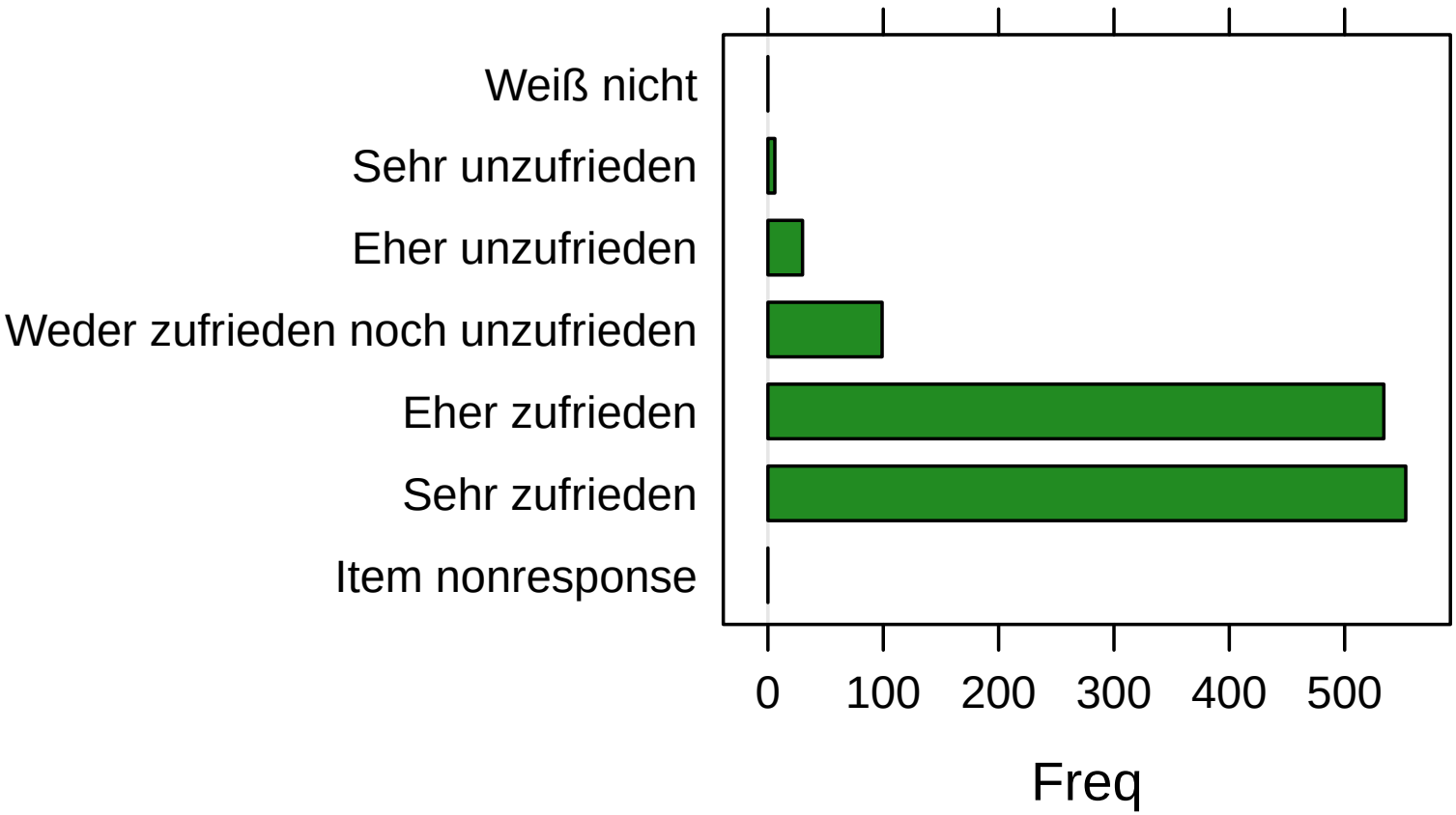
Studien im Campus File

Name	Code
GESIS Panel Core Study Module - Subjective Well-being	zb
Environmental Spatial Strategies	ag
Cross-National Replication of Question Design Experiments	ah
Survey Evaluation Items	ai
GESIS Panel Core Study Module - Survey Administration Variables	za
GESIS Panel Core Study Module - Monitoring quality: survey experience & mode characteristics	zq
GESIS Panel Core Study Module - Social and Political Participation	zc
Critical Elections in the European Union	aj
International panel comparison study	ak
Standardization of the Positive and Negative Affect Schedule (PANAS)	al
GESIS Panel Core Study Module - Environmental attitudes and behavior	zd
Short version of the Metacognitive Prospective Memory Battery (MPMBs)	am
Leisure travel and subjective well-being	an
GESIS Panel Core Study Module - Personality and Personal Values	ze
Social and individual predictors of Doing Beauty	ao
Citizens Conception of Democracy and their Political Participation	ap
GESIS Panel Core Study Module - Media Usage	zf

Die Missing Codes im GESIS Panel

Value	Value.label	Remark
-11	Not invited	only in recruitment waves - when profile survey not finished
-22	Not in panel	not willing to join the panel after recruitment or signing off
-33	Unit nonresponse	invited but not participating in corresponding wave
-44	Missing by m.o.p.	mode of participation (m.o.p.): online or offline
-55	Missing by technical error	e.g. questionnaire programming error
-66	Missing by design	experimental variation
-77	Not reached	only in online mode: panelist has not seen the item
-88	Missing by filter	filtered item
-99	Item nonresponse	due to nonresponse by the respondent
-111	Ambiguous answer	ambiguous answers in questionnaire

Zufriedenheit Leben in Wohnort (a11c019a)



Das Codebuch

- Das Codebuch kann man **hier** bekommen.

Variable name	bbak102a
Variable label	In 12 Monaten ausgeübt:Ehrenamtliche Tätigkeit
	Done in the past twelve months: voluntary or charity work
Publication status	standard edition
Dataset	ZA56664_a1_ba-bf_23-0-0 / ZA56665_a1_ba-bf_23-0-0
Item source	SHARE Wave 6 (AC035)
Question type	Multiple Choice
Question text	Welche der folgenden Aktivitäten haben Sie, falls überhaupt, in den letzten 12 Monaten ausgeübt?
	Which of the following activities - if any - have you done in the past twelve months?
Item text	Ehrenamtliche Tätigkeit
	Done voluntary or charity work
Value labels	

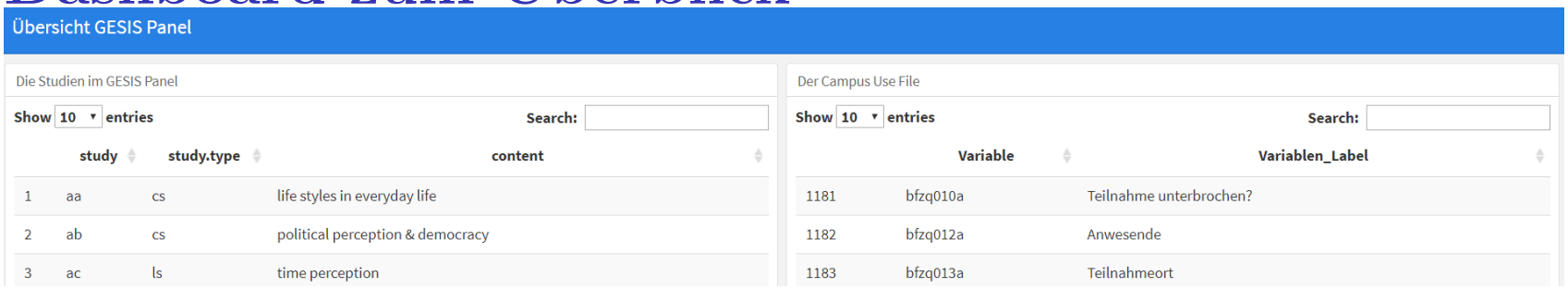
0	Nicht genannt
	Not quoted
1	Genannt
	Quoted
-11	Not invited
-22	Not in panel
-33	Unit nonresponse
-77	Not reached
-99	Item nonresponse

Within wave	bbak103a bbak104a bbak105a bbak106a bbak107a bbak108a bbak109a

Position within wave	Online	Offline
Question Order	35	30
Page ID/Page	2387	11

Übersicht GESIS Panel (Campus File)

Dashboard zum Überblick



Aufgabe - Download der GESIS Panel Daten

- Suche bei einer **Suchmaschine** nach GESIS Panel Campus file oder
- gehe auf die Seiten des **GESIS Datenbestandskatalogs** und
- lade die `**14.dta` Datei des GESIS Panel Campus file herunter.

Datenverarbeitung

Inhalt dieses Abschnitts

- Wie bekommt man einen Überblick über die Daten
- Indizieren von Vektoren, Datensätzen und Listen
- Wie geht man mit fehlenden Werten um
- Schleifen und Funktionen
- Zusammenhänge zwischen Variablen

data.frame's

- Beispieldaten importieren:

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

```
typeof(dat)
```

```
## [1] "list"
```

```
head(names(dat))
```

```
## [1] "z000001z" "z000002z" "z000003z" "z000005z" "a11c01
```

Anzahl Zeilen und Spalten

- Anzahl der Zeilen/Spalten ermitteln

```
nrow(gpdat) # Zeilen
```

```
## [1] 1222
```

```
ncol(gpdat) # Spalten
```

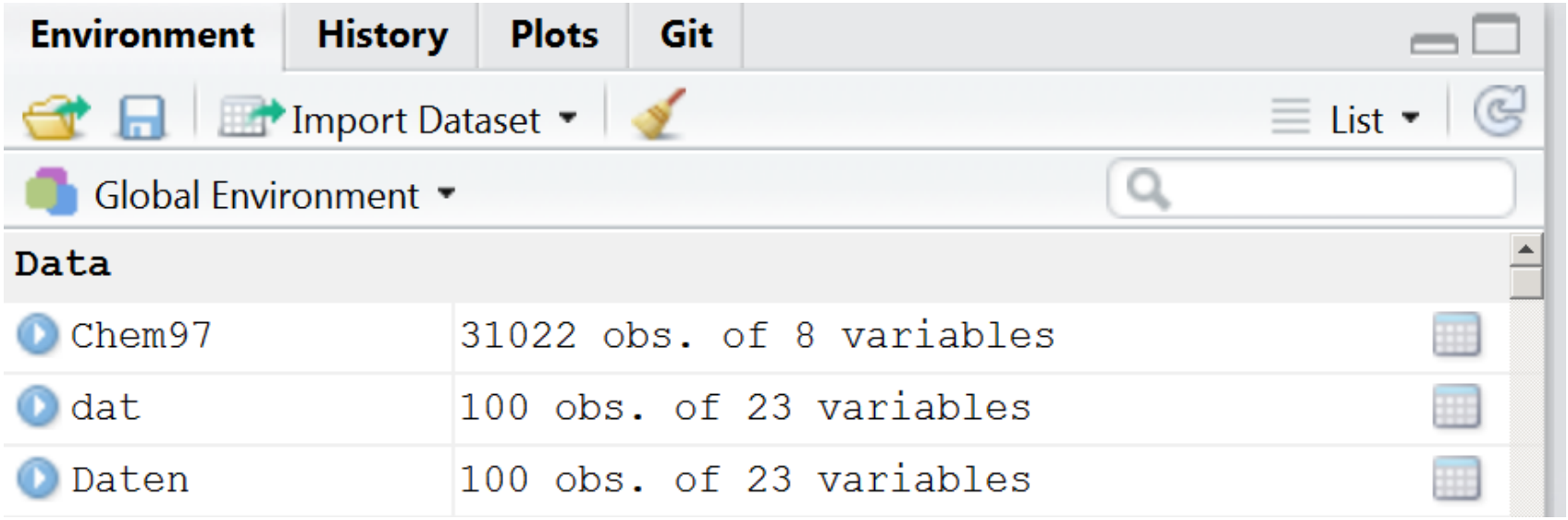
```
## [1] 1192
```

Die Daten ansehen

- Die ersten Zeilen sehen:

```
head(gpdat) # erste Zeilen
tail(gpdat) # letzte Zeilen
```

- Einen Überblick mit Rstudio bekommen:



Indizierung eines data.frame

```
dat[1,1] # das Element oben links bekommen
```

```
## [1] 198431880
```

```
dat[2,] # nur die zweite Zeile sehen
```

```
##    z000001z z000002z          z000003z          z000005z
## 2 436122330   ZA5666 1-0-0 2017-06-20 10.4232/1.12749
```

```
dat[,1] # sich nur die erste Spalte anzeigen lassen
```

```
## [1] 198431880 436122330 856844220 117346660 943433330 26
```

Weitere Möglichkeiten zur Indizierung eines data.frame

```
dat[1:2,] # getting the first two rows
```

```
##    z000001z z000002z        z000003z        z000005z
## 1 198431880   ZA5666 1-0-0 2017-06-20 10.4232/1.12749 Se
## 2 436122330   ZA5666 1-0-0 2017-06-20 10.4232/1.12749 Se
##         a11c020a       a11c021a       a11c022a
## 1 Sehr zufrieden Sehr zufrieden Stimme eher zu Stimme eh
## 2 Sehr zufrieden Sehr zufrieden Stimme eher zu Stimme eh
##         a11c024a
## 1 Stimme eher zu
## 2 Stimme eher zu
```

Indizierung

- Das Dollarzeichen kann auch zur Adressierung einzelner Spalten verwendet werden.

```
head(datf$a11c019a)
```

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

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

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

Zugriff auf Spalten

- Wie bereits beschrieben, können Sie über Zahlen auf die Spalten zugreifen.

```
head(datf[,5])
```

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

```
head(datf[,"a11c019a"]) # dasselbe Ergebnis
```

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

Logische Operatoren

```
(a <- 1:7) # Beispieldaten - numerisch
```

```
## [1] 1 2 3 4 5 6 7
```

```
a>4
```

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

```
a>=4
```

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

```
a<3
```

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

Logische Operatoren II

```
(b <- letters[1:7]) # Beispieldaten - Strings
```

```
## [1] "a" "b" "c" "d" "e" "f" "g"
```

```
b=="e"
```

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

```
b %in% c("e","f")
```

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

GESIS Panel Variable - Estimated duration (bazq020a)

Wie lange haben Sie den Fragebogen ausgefüllt?

```
duration <- as.numeric(datf$bazq020a)
```

```
summary(duration)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   -99.00   10.00   15.00   11.81   20.00 1440.00      23
```

Missing values

- Fehlende Werte sind in R als `NA` definiert
- Bei mathematische Funktionen gibt es in der Regel eine Möglichkeit, fehlende Werte auszuschließen.
- Bei `mean()`, `median()`, `colSums()`, `var()`, `sd()`, `min()` und `max()` gibt es das Argument `na.rm`.

```
mean(duration)
```

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

```
mean(duration,na.rm=T)
```

```
## [1] 11.81234
```

Die fehlenden Werte finden

```
is.na(head(duration))
```

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

```
which(is.na(duration))
```

```
##   [1]   30   63  103  182  184  258  415  424  441  527
##  [15]  766  861  917  923  962  995 1026 1037 1062
```

```
table(is.na(duration))
```

```
##
## FALSE  TRUE
##  1199    23
```

Die fehlenden Werte rekodieren

```
summary(duration)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   -99.00   10.00   15.00   11.81   20.00 1440.00      23
```

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

- **Quick-R zu fehlenden Werten**
- **Fehlende Werte rekodieren**

Eine einfache Funktion schreiben

```
tail(duration,n=10)
```

```
##   [1]  36  30 -33  10  20  15  17  15  10  15
```

```
transform_miss <- function(x){
  x[x==-99] <- NA
  return(x)
}
```

```
duration <- transform_miss(duration)
tail(duration,n=10)
```

```
##   [1]  36  30 -33  10  20  15  17  15  10  15
```

Aufgabe - eine Funktion erweitern

Erweitert die Funktion so,

- dass auch die anderen Missingwerte als NA umkodiert werden
- dass sie auch dann ihren Zweck erfüllt, wenn die Value Labels ausgegeben werden (Item nonresponse, Missing by filter, etc.).

Der Befehl complete.cases()

```
# Beispiel Datensatz
mydata <- data.frame(A=c(1,NA,9,6),B=c("A","B",1,NA))
```

- Der Befehl `complete.cases()` gibt einen logischen Vektor zurück, der angibt, welche Fälle vollständig sind.

```
# Datenzeilen mit fehlenden Werten auflisten
mydata[complete.cases(mydata),]
```

```
##   A B
## 1 1 A
## 3 9 1
```

Verschiedene Arten von fehlenden Werten (NAs) spezifizieren

- Spezifiziere verschiedene Arten von Fehlern mit dem Paket `memisc`.
- Benutze dazu den Befehl `include.missings()`

```
library(memisc)
?include.missings
```

- Es ist auch möglich, Codebuch-Einträge mit `memisc` zu erstellen.

```
codebook(dat$a11c019a)
```

Datensatz indizieren

```
SEX <- gpdat$a11d054a
table(SEX)
```

```
## SEX
## Männlich Weiblich
##      596      626
```

```
gpdat[SEX=="Männlich",]
# same result:
gpdat[SEX!="Weiblich",]
```

Weitere wichtige Optionen

- Speichern des Ergebnisses in einem Objekt

```
subDat <- gpdat[duration>20,]
```

- mehrere Bedingungen können mit & verknüpft werden

```
gpdat[duration>18 & SEX=="Männlich",]
```

- das oder das Argument - eine der beiden Bedingungen muss erfüllt sein

```
gpdat[duration>18 | SEX=="Männlich",]
```

Sequenzen bei der Indizierung

```
library("readstata13")
datf <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta",
                   convert.factors = F)
```

```
datf[15:20,10:14]
```

```
##    a11c024a a11c025a a11c026a a11c027a a11c028a
## 15        1        2        5        5        1
## 16        3        2        4        1        1
## 17        1        1        5        2        4
## 18        2        2        3        3        1
## 19        2        1        4        5        1
## 20        1        2        3        2        1
```

Variablen Labels

```
library(foreign)
dat <- read.dta("../data/ZA5666_v1-0-0_Stata12.dta")
```

```
attributes(dat)
```

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

- Das Gleiche gilt für das haven-Paket

```
library(haven)
dat2 <- read_dta("../data/ZA5666_v1-0-0_Stata14.dta")
var.labels2 <- attr(dat,"var.labels")
```

Umbenennen der Spaltennamen

- Mit dem Befehl `Colnames` erhält man die Spaltennamen

```
colnames(dat)
```

- Wir können die Spaltennamen umbenennen:

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

- Das gleiche gilt für die Zeilennamen

```
rownames(dat)
```

Private Internetnutzung (a11c034a)

Das Internet gewinnt eine immer größere Bedeutung in der Gesellschaft. Deshalb interessiert uns, ob Sie selbst zumindest gelegentlich das Internet für private Zwecke nutzen?

```
table(dat$a11c034a)
```

```
##
##                                                  Item nonresponse
##                                                                 0
##                        Ja, nutzt Internet für private Zwecke
##                                                              1044
## Nein, nutzt Internet nicht für private Zwecke
##                                                               177
##                                                  Weiß nicht
```

Faktorstufen

```
str(dat$a11c034a)
```

```
##  Factor w/ 4 levels "Item nonresponse",..: 2 2 2 2 3 2 2
```

```
levels(dat$a11c034a)
```

```
## [1] "Item nonresponse"
## [2] "Ja, nutzt Internet für private Zwecke"
## [3] "Nein, nutzt Internet nicht für private Zwecke"
## [4] "Weiß nicht"
```

```
levels(dat$a11c034a)[2:4] <- c("yes","no","don`t know")
levels(dat$a11c034a)
```

```
## [1] "Item nonresponse" "yes"                "no"
```

Exkurs - Wie man Labels verwendet

Werkzeuge für das Arbeiten mit kategorischen Variablen (Faktoren)

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

- `fct_collapse` - um Faktorstufen zu verdichten
- `fct_count` - um die Einträge in einem Faktor zu zählen
- `fct_drop` - Entferne unbenutzte Levels

Der Befehl fct_count

Freizeit Häufigkeit: Bücher lesen (a11c026a)

```
fct_count(f = dat$a11c026a)
```

```
## # A tibble: 8 x 2
##   f                                n
##   <fct>                        <int>
## 1 Item nonresponse                 0
## 2 Täglich                        239
## 3 Mehrmals die Woche             204
## 4 Mehrmals im Monat              154
## 5 Mindestens einmal im Monat      97
## 6 Seltener                       347
## 7 Nie                            181
## 8 Weiß nicht                       0
```

Der Befehl fct_collapse

```
a11c026a <- fct_collapse(.f = dat$a11c026a,
    Mehrmals=c("Mehrmals die Woche","Mehrmals im Monat"))
```

```
fct_count(a11c026a)
```

```
## # A tibble: 7 x 2
##   f                                n
##   <fct>                        <int>
## 1 Item nonresponse                 0
## 2 Täglich                        239
## 3 Mehrmals                       358
## 4 Mindestens einmal im Monat      97
## 5 Seltener                       347
## 6 Nie                            181
## 7 Weiß nicht                       0
```

recode Befehl im Paket car

```
library(car)
```

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

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

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

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

Aufgabe - Value Labels neu kodieren

- Übersetze die Deutschen Werte Labels der Variablen `bbzc022a` ins Englische (Man kann dafür https://www.deepl.com/ verwenden).
- Benenne die Ausprägungen der Variable so um, dass sie die englischen Value Label enthält.

Schleifen in R

- Der Befehl `for()` kennzeichnet den Start einer Schleife
- in Klammern, haben wir einen Index und die Anzahl der Läufe (in diesem Fall läuft die Schleife von 1 bis 4).
- in den geschweiften Klammern `{}` ist angegeben, was bei einer Iteration passiert.

```
for (i in 1:4){
  cat(i, "\n")
}
```

```
## 1
## 2
## 3
## 4
```

Beispiel für Schleifen in R

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

```
##  int [1:3] 198431880 436122330 856844220
```

- in diesem Fall läuft die Schleife von 1 bis zur Anzahl der Spalten in dat.

```
for (i in 1:ncol(dat)){
  dat[,i] <- as.character(dat[,i])
}
```

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

```
##  chr [1:3] "198431880" "436122330" "856844220"
```

Schleifen - die Ergebnisse behalten

- Wir können die Ergebnisse in einem Objekt speichern
- dieses kann bspw. ein Vektor oder eine Liste sein.

```
erg1 <- vector()
erg2 <- list()

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

Die apply Familie

```
(ApplyDat <- cbind(1:4,runif(4),rnorm(4))) #Example
```

```
##      [,1]      [,2]       [,3]
## [1,]    1 0.5885115  1.26394484
## [2,]    2 0.7847886  0.08826205
## [3,]    3 0.6864993 -0.70085905
## [4,]    4 0.5571181  0.04016490
```

```
apply(ApplyDat,1,mean)
```

```
## [1] 0.9508188 0.9576836 0.9952134 1.5324277
```

```
apply(ApplyDat,2,mean)
```

```
## [1] 2.5000000 0.6542294 0.1728782
```

Der Befehl apply()

```
apply(ApplyDat,1,var)
```

```
## [1] 0.1158666 0.9361050 3.4955677 4.6334951
```

```
apply(ApplyDat,1,sd)
```

```
## [1] 0.3403919 0.9675252 1.8696437 2.1525555
```

```
apply(X = ApplyDat,MARGIN = 1,FUN = range)
```

```
##           [,1]       [,2]       [,3]      [,4]
## [1,] 0.5885115 0.08826205 -0.7008591 0.0401649
## [2,] 1.2639448 2.00000000  3.0000000 4.0000000
```

Die Argumente des Befehls `apply()`

- Wenn `MARGIN=1` wird die Funktion `mean` auf die Reihen angewendet,
- Wenn `MARGIN=2` wird die Funktion `mean` auf die Spalten angewendet,
- Anstatt `mean` kann man auch `var`, `sd` oder `length` verwenden.

Der Befehl tapply()

```
ApplyDat <- data.frame(Income=rnorm(5,1400,200),
                       Sex=sample(c(1,2),5,replace=T))
```

Beispiel Befehl tapply()

```
tapply(ApplyDat$Income,
       ApplyDat$Sex,function(x)x)
```

```
## $`1`
## [1] 1494.279 1101.419 1374.248
##
## $`2`
## [1] 1075.3277  839.5478
```

Übung - tapply() Befehl verwenden

- Finde heraus, welche Variable Informationen über die Altersgruppe enthält.
- Berechne die durchschnittliche Dauer (Variable `bfzq020a`) für die Beantwortung des Fragebogens nach Altersgruppe.

Das reshape Paket

Beispiel Datensatz

```
(mydata <- data.frame(id=rep(1:2,each=2),
                      time=rep(c(1,2),2),
                      x1 = c(5,3,6,2),
                      x2 = c(6,5,1,4)))
```

```
##   id time x1 x2
## 1  1    1  5  6
## 2  1    2  3  5
## 3  2    1  6  1
## 4  2    2  2  4
```

Beispiel mit dem Befehl melt

```
library(reshape)
melt(mydata, id=c("id","time"))
```

```
##    id time variable value
## 1   1    1       x1     5
## 2   1    2       x1     3
## 3   2    1       x1     6
## 4   2    2       x1     2
## 5   1    1       x2     6
## 6   1    2       x2     5
## 7   2    1       x2     1
## 8   2    2       x2     4
```

Edgar Anderson's Iris Datensatz

```
data(iris)
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
## 4          4.6         3.1          1.5         0.2
## 5          5.0         3.6          1.4         0.2
## 6          5.4         3.9          1.7         0.4
```

- petal length and width - Länge und Breite der Blütenblätter
- sepal length and width - Kelchlänge und -breite
- **Wikipedia Artikel zum IRIS Datensatz**

Zusammenhang zwischen kontinuierliche Variablen

```
# Pearson correlation coefficient
cor(iris$Sepal.Length,iris$Petal.Length)
```

```
## [1] 0.8717538
```

- Zusammenhang zwischen Blütenblattlänge und Blütenblattlänge ist 0,87
- Der Pearson-Korrelationskoeffizient ist die Standardmethode in `cor()`.

Verschiedene Korrelationskoeffizienten

```
# Pearson correlation coefficient
cor(iris[,1:4])
```

```
##                Sepal.Length Sepal.Width Petal.Length Petal
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0
```

```
# Kendall's tau (rank correlation)
cor(iris[,1:4], method = "kendall")
```

```
##                Sepal.Length Sepal.Width Petal.Length Petal
## Sepal.Length   1.00000000 -0.07699679    0.7185159   0.6
## Sepal.Width   -0.07699679  1.00000000   -0.1859944  -0.1
```

Eine zweidimensionale Kreuztabelle erstellen

Bedeutung Variablen

```
att_dat <- attributes(dat)
att_dat$var.labels[which(colnames(dat)=="a11c025a")]
att_dat$var.labels[which(colnames(dat)=="a11c024a")]
```

- a11c025a - Lebensstandard junge Generation
- a11c024a - Vertrauen: Vorsichtig sein bei Fremden

Tabelle erstellen

```
tab <- table(dat$a11c025a,dat$a11c024a)
```

Kreuztabelle anschauen

Variablen

- a11c025a - Lebensstandard junge Generation
- a11c024a - Vertrauen: Vorsichtig sein bei Fremden

Tabelle

	Item nonresponse	Stimme voll und ganz zu	Stimme eher zu
Item nonresponse	0	0	0
Eher höheren Lebensstandard	0	157	98
Eher niedrigeren Lebensstandard	0	325	224
Denselben Lebensstandard	1	137	137
Weiß nicht	1	17	11

	Stimme eher nicht zu	Stimme überhaupt nicht zu	Weiß nicht
Item nonresponse	1	0	0
Eher höheren Lebensstandard	15	4	0
Eher niedrigeren Lebensstandard	53	8	2
Denselben Lebensstandard	21	6	1
Weiß nicht	2	1	0

Beziehung zwischen kategorialen Variablen

- `chisq.test()` prüft, ob zwei kategoriale Merkmale stochastisch unabhängig sind.
- Der Test wird gegen die Nullhypothese der Gleichverteilung durchgeführt.

```
chisq.test(tab)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab
## X-squared = 45.411, df = 20, p-value = 0.0009703
```

Übung - eine interaktive Tabelle

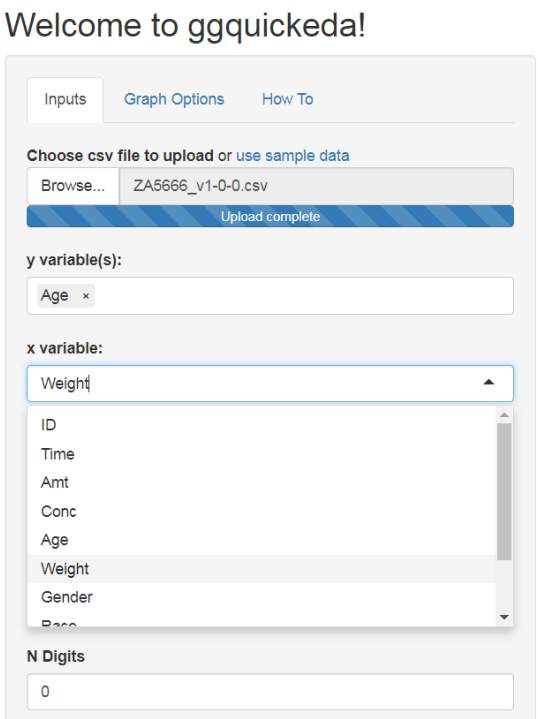
- Lade den Datensatz `dat_cf2.RData` vom **Github Verzeichnis** herunter.
- Importiere den Datensatz in R
- Erstelle eine interaktive Tabelle mit den folgenden Befehlen:

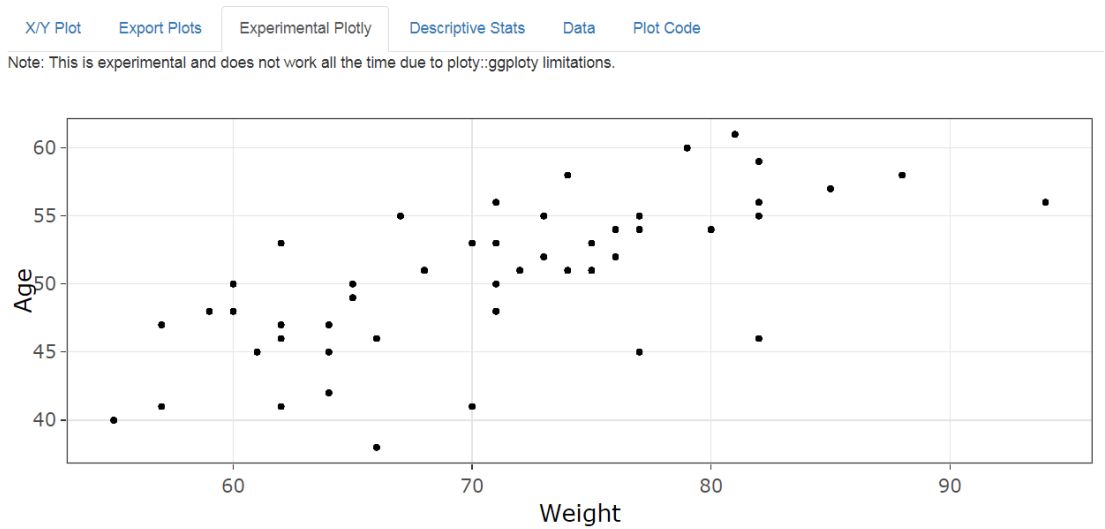
```
library(DT)
DT::datatable(dat_cf2)
```

- Probiere witere Argumente der Funktion `datatable` aus.

Shiny App für eine schnelle explorative Datenanalyse

https://pharmacometrics.shinyapps.io/ggplotwithyourdata/





Weitere Links

- **Tidy data** - das Paket `tidyr`
- Homepage für: **the `tidyverse` collection**
- **Data wrangling mit R und RStudio**
- Hadley Wickham - **Tidy Data**
- Hadley Wickham - **Advanced R**
- Colin Gillespie and Robin Lovelace **Efficient R programming**

Einfache Graphiken erstellen

Ein Plot sagt mehr als 1000 Worte

- Grafisch gestützte Datenanalyse ist toll
- Gute Plots können zu einem besseren Verständnis beitragen
- Einen Plot zu generieren geht schnell
- Einen guten Plot zu machen kann sehr lange dauern
- Mit R Plots zu generieren macht Spaß
- Mit R erstellte Plots haben hohe Qualität
- Fast jeder Plottyp wird von R unterstützt
- R kennt eine große Menge an Exportformaten für Grafiken

Plot ist nicht gleich Plot

- Bereits das base Package bringt eine große Menge von Plot Funktionen mit
- Das lattice Packet erweitert dessen Funktionalität
- Eine weit über diese Einführung hinausgehende Übersicht findet sich in Murrell, P (2006): R Graphics.

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

Task View zu Thema Graphiken

CRAN Task View: Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

Maintainer: Nicholas Lewin-Koh
Contact: nikko at hailmail.net
Version: 2015-01-07
URL: https://CRAN.R-project.org/view=Graphics

R is rich with facilities for creating and developing interesting graphics. Base R contains functionality for many plot types including coplots, mosaic plots, biplots, and the list goes on. There are devices such as postscript, png, jpeg and pdf for outputting graphics as well as device drivers for all platforms running R. lattice and grid are supplied with R's recommended packages and are included in every binary distribution. lattice is an R implementation of William Cleveland's trellis graphics, while grid defines a much more flexible graphics environment than the base R graphics.

Grafiken für bedingte, bi- und multivariate Verteilungen - Scatterplots

- Funktion `plot()` ist eine generische Funktion - bspw. kann ein einfacher Scatterplot erstellt werden
- Für einen solchen muss `plot()` ein `x` und ein `y` Beobachtungsvektor übergeben werden
- Um die Farbe der Plot-Symbole anzupassen gibt es die Option `col` (Farbe als character oder numerisch)
- Die Plot-Symbole selbst können mit `pch` (plotting character) angepasst werden (character oder numerisch)
- Die Achenbeschriftungen (labels) werden mit `xlab` und `ylab` definiert

Beispiel - iris Datensatz

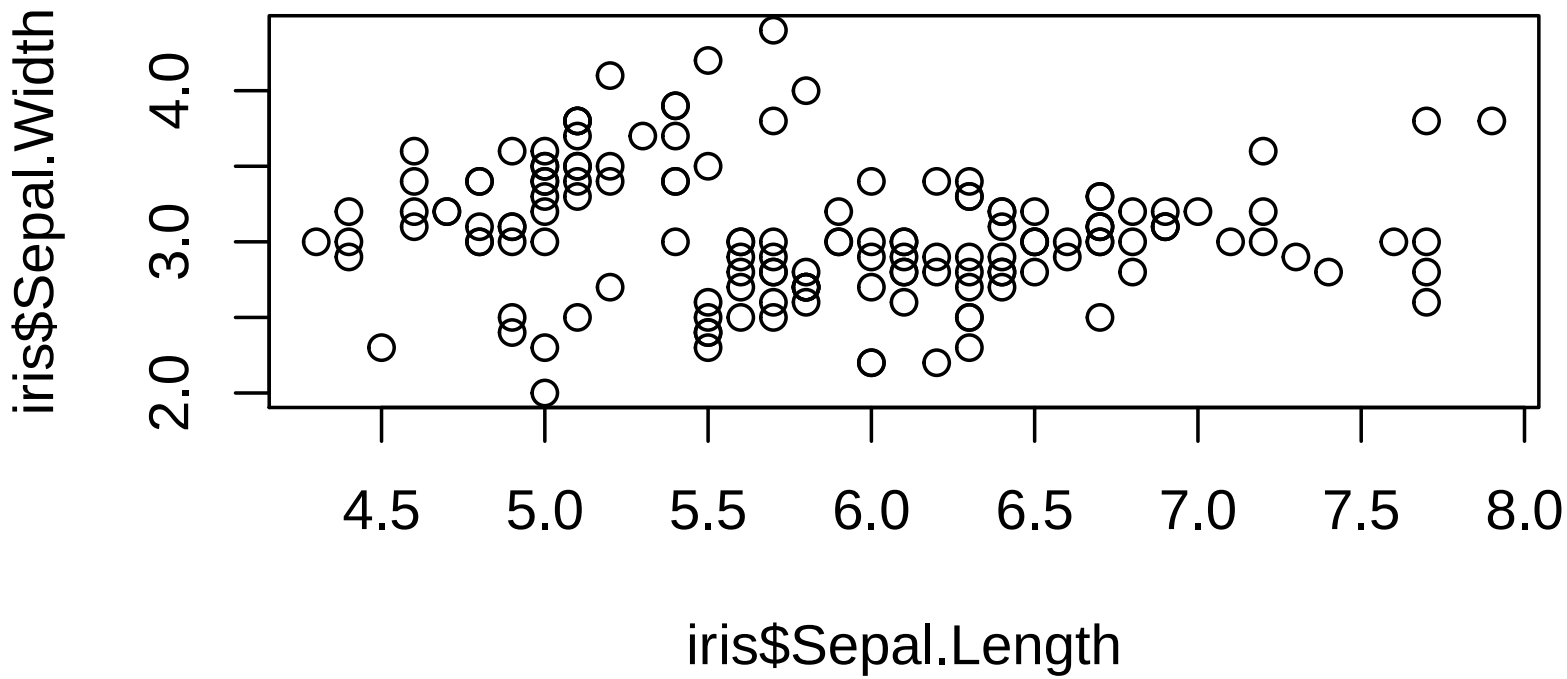
```
data(iris)
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Spec
## 1          5.1         3.5          1.4         0.2  set
## 2          4.9         3.0          1.4         0.2  set
## 3          4.7         3.2          1.3         0.2  set
## 4          4.6         3.1          1.5         0.2  set
## 5          5.0         3.6          1.4         0.2  set
## 6          5.4         3.9          1.7         0.4  set
```

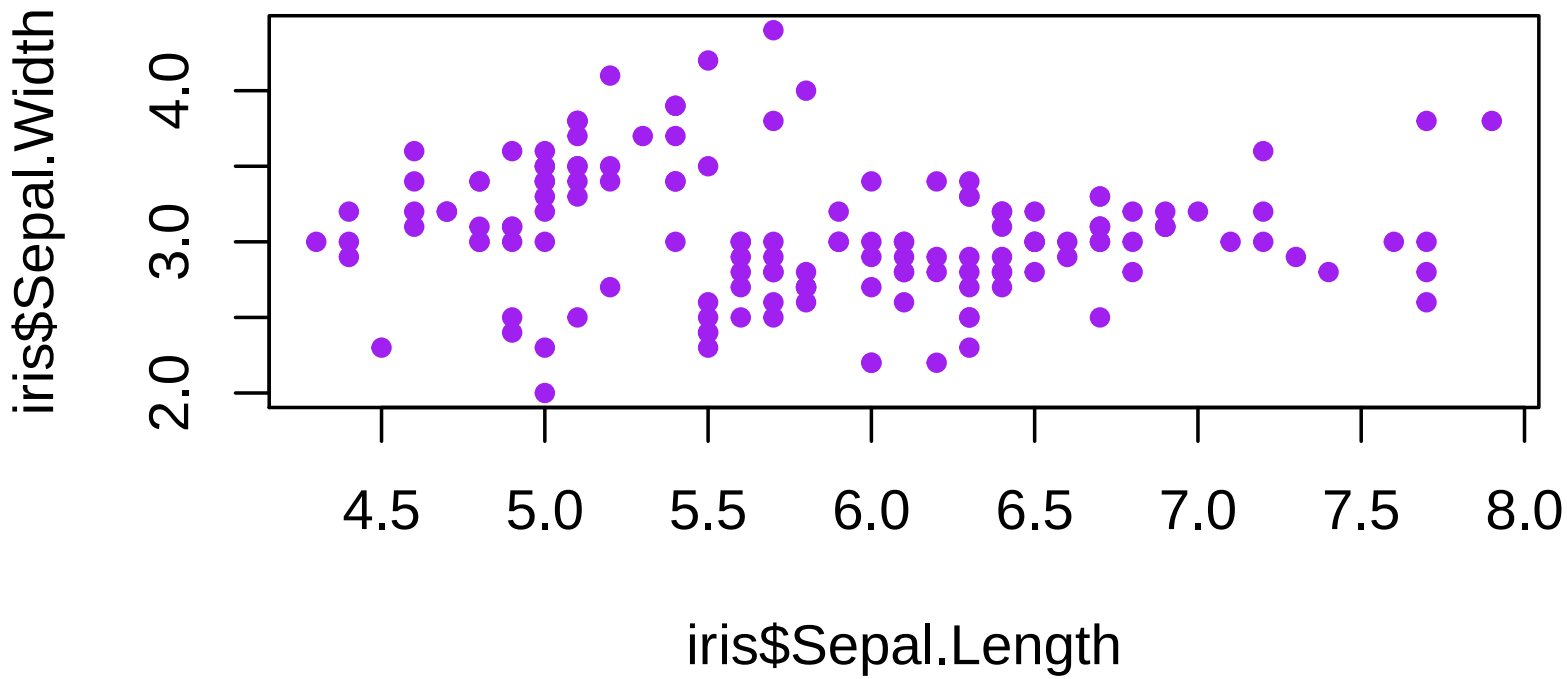
Ein erster Scatterplot

```
plot(iris$Sepal.Length,iris$Sepal.Width)
```



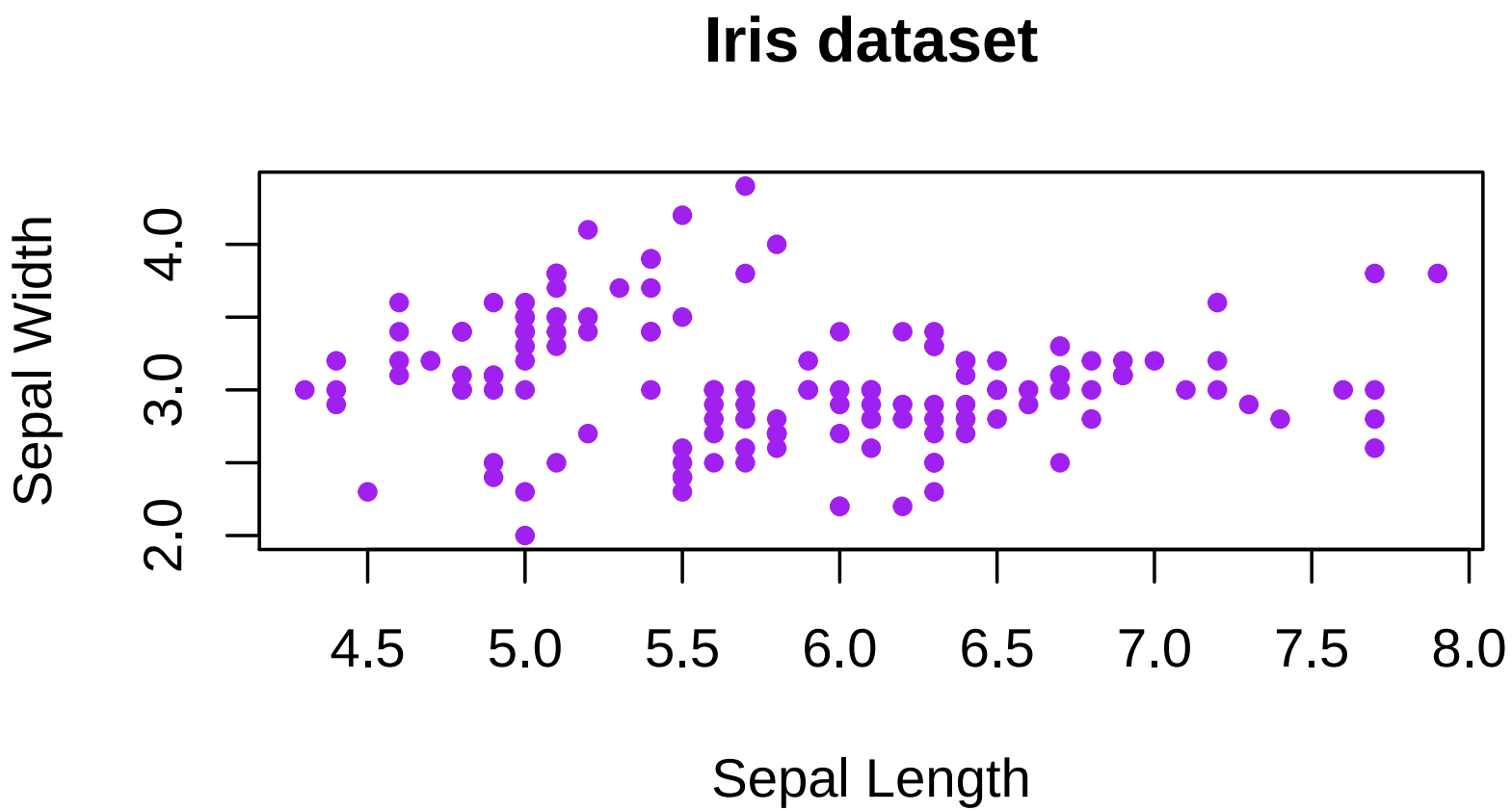
Andere Farbe und Punkttyp

```
plot(iris$Sepal.Length,iris$Sepal.Width,pch=20,col="purple"
```



Beschriftung hinzufügen

```
plot(iris$Sepal.Length,iris$Sepal.Width,pch=20,col="purple"
```



Datensatz

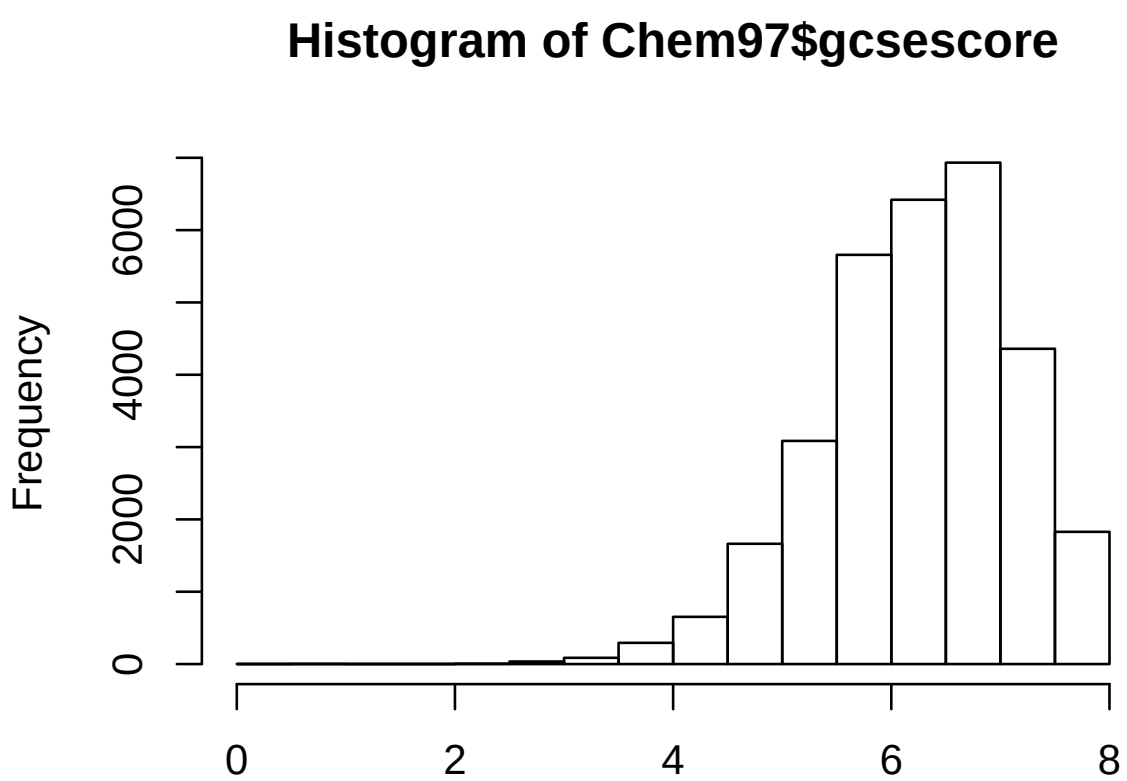
```
library(mlmRev)
data(Chem97)
```

- [lea] Local Education Authority - a factor
- [school] School identifier - a factor
- [student] Student identifier - a factor
- [score] Point score on A-level Chemistry in 1997
- [gender] Student's gender
- [age] Age in month, centred at 222 months or 18.5 years
- [gcsescore] Average GCSE score of individual.
- [gcsecnt] Average GCSE score of individual, centered at mean.

Histogramm - Die Funktion hist()

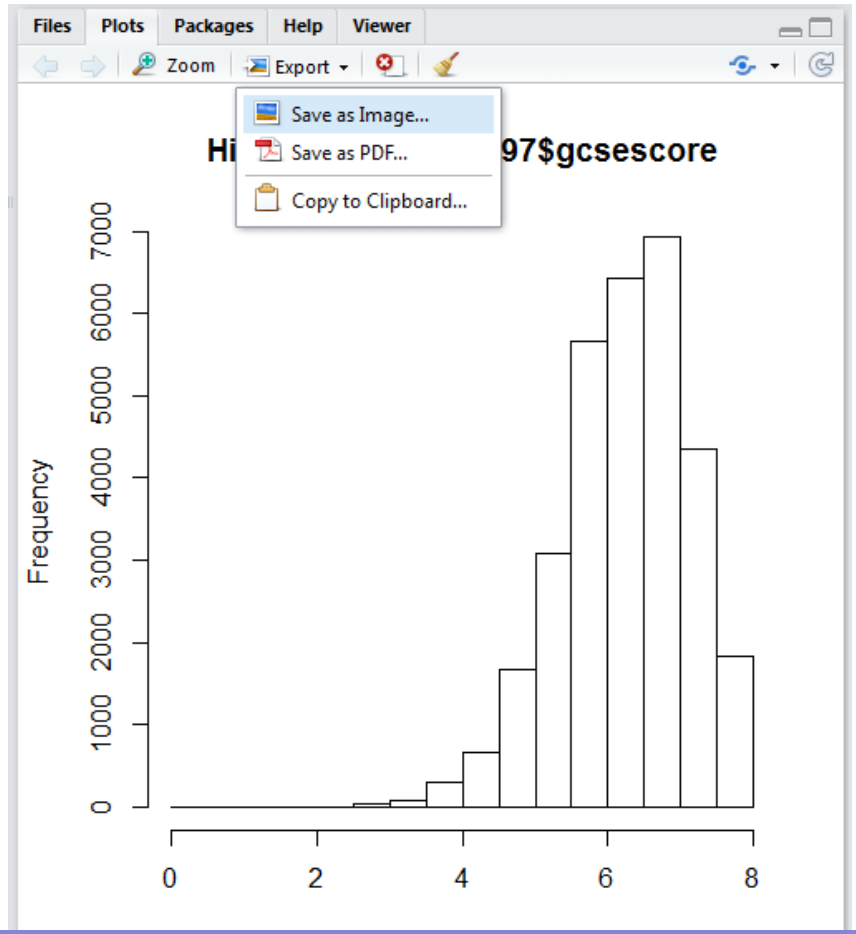
Wir erstellen ein Histogramm der Variable gcsescore:

```
hist(Chem97$gcsescore)
```



Graphik speichern

- Mit dem button Export in Rstudio kann man die Grafik speichern.



Befehl um Graphik zu speichern

- Alternativ auch bspw. mit den Befehlen `png`, `pdf` oder `jpeg`

```
png("Histogramm.png")
hist(Chem97$gcsescore)
dev.off()
```

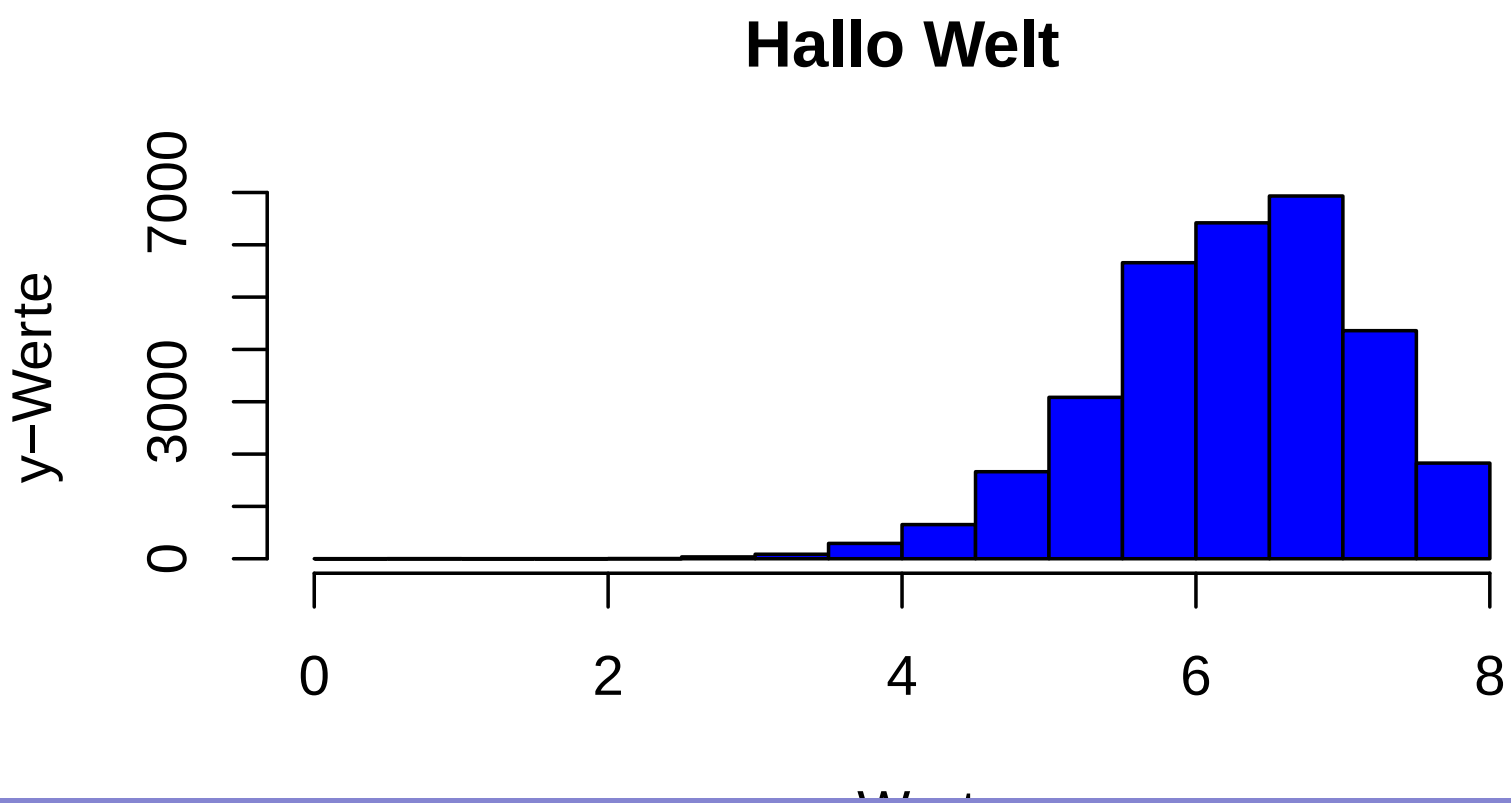
Histogramme

- Die Funktion `hist()` plottet ein Histogramm der Daten
- Der Funktion muss mindestens ein Beobachtungsvektor übergeben werden
- `hist()` hat noch sehr viel mehr Argumente, die alle (sinnvolle) default values haben

Argument	Bedeutung	Beispiel
main	Überschrift	main='Hallo Welt'
xlab	x-Achsenbeschriftung	xlab='x-Werte'
ylab	y-Achsenbeschriftung	ylab='y-Werte'
col	Farbe	col='blue'

Histogramm

```
hist(Chem97$gcsescore,col="blue",
     main="Hallo Welt",ylab="y-Werte", xlab="x-Werte")
```



Barplot

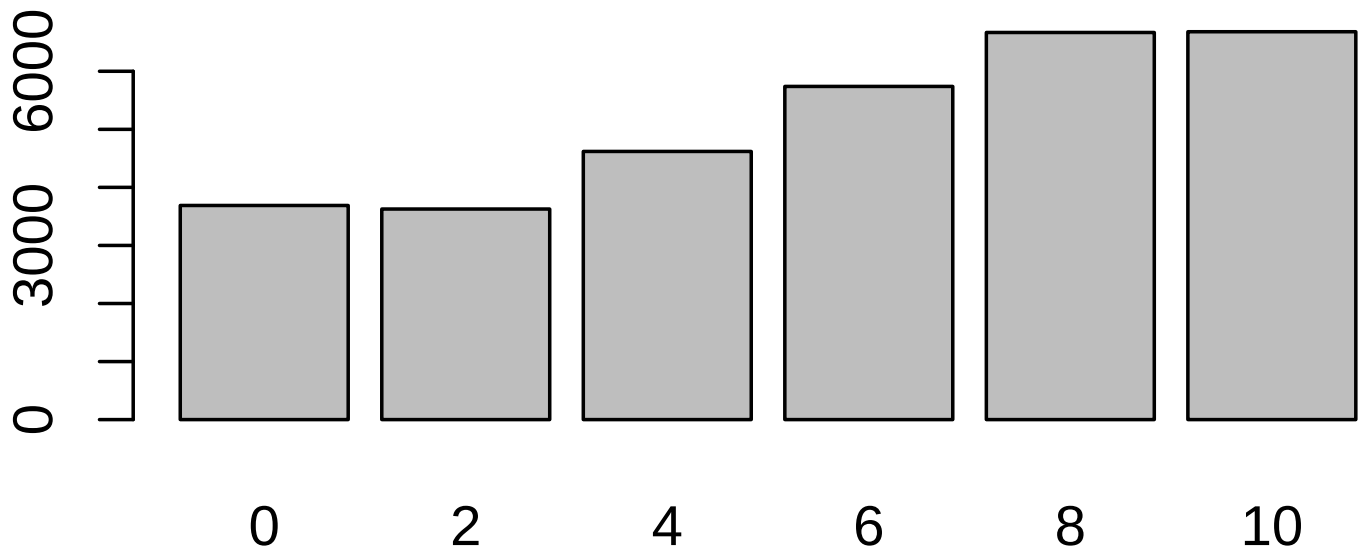
- Die Funktion `barplot()` erzeugt aus einer Häufigkeitstabelle einen Barplot
- Ist das übergebene Tabellen-Objekt zweidimensional wird ein bedingter Barplot erstellt

```
tabScore <- table(Chem97$score)
```

```
barplot(tabScore)
```

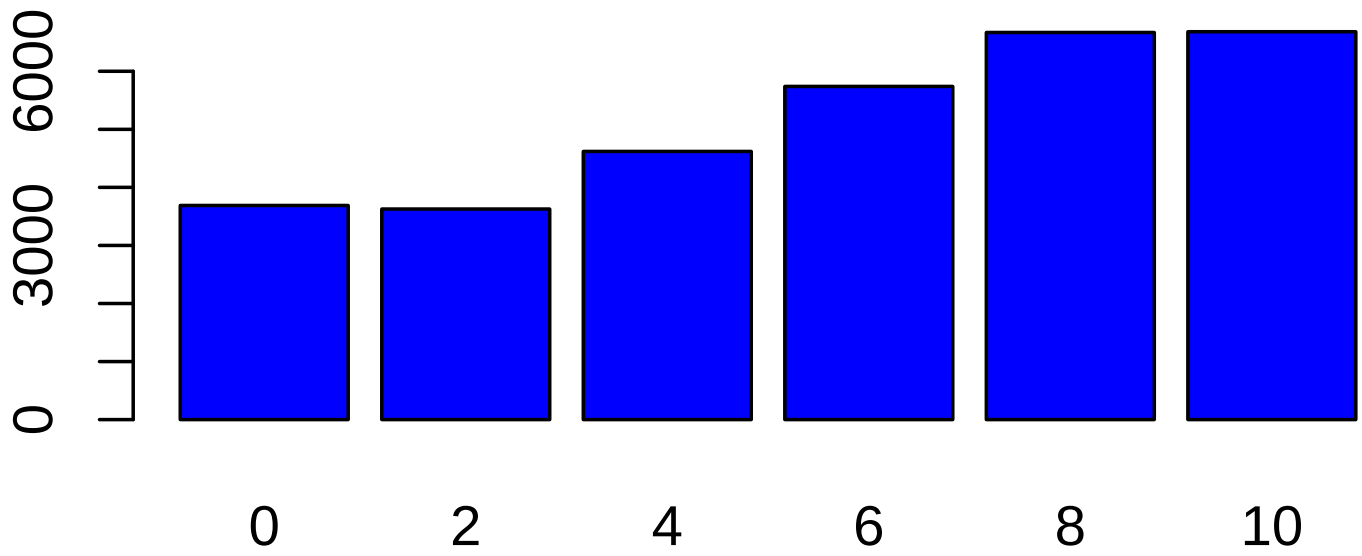
Barplots und barcharts

```
barplot(tabScore)
```



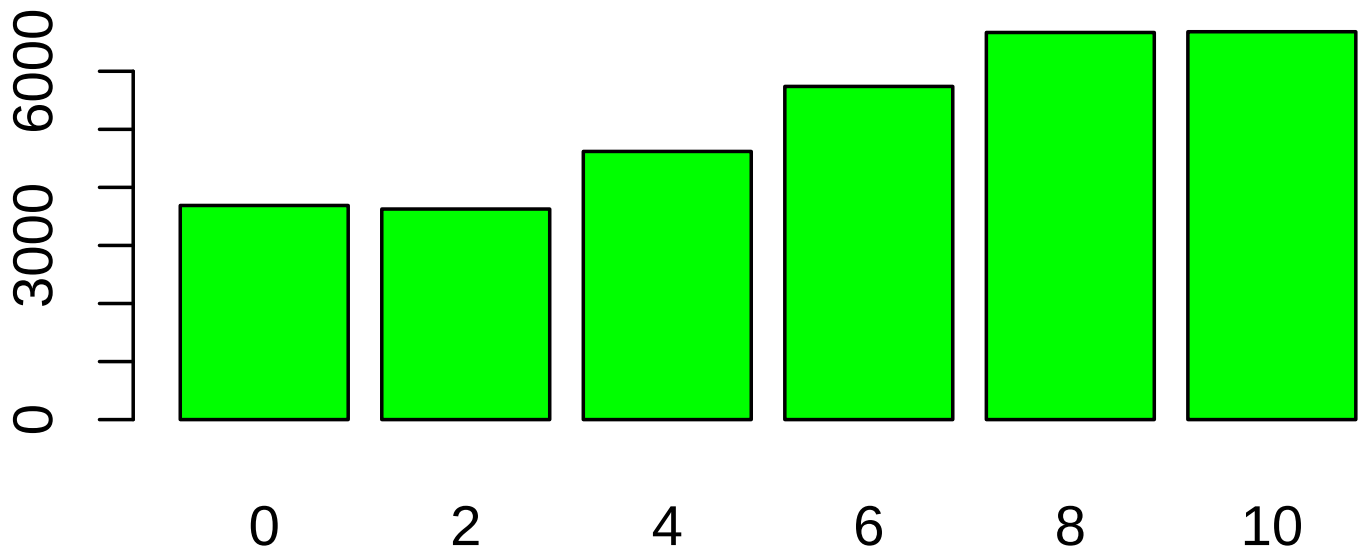
Mehr Farben:

```
barplot(tabScore,col=rgb(0,0,1))
```



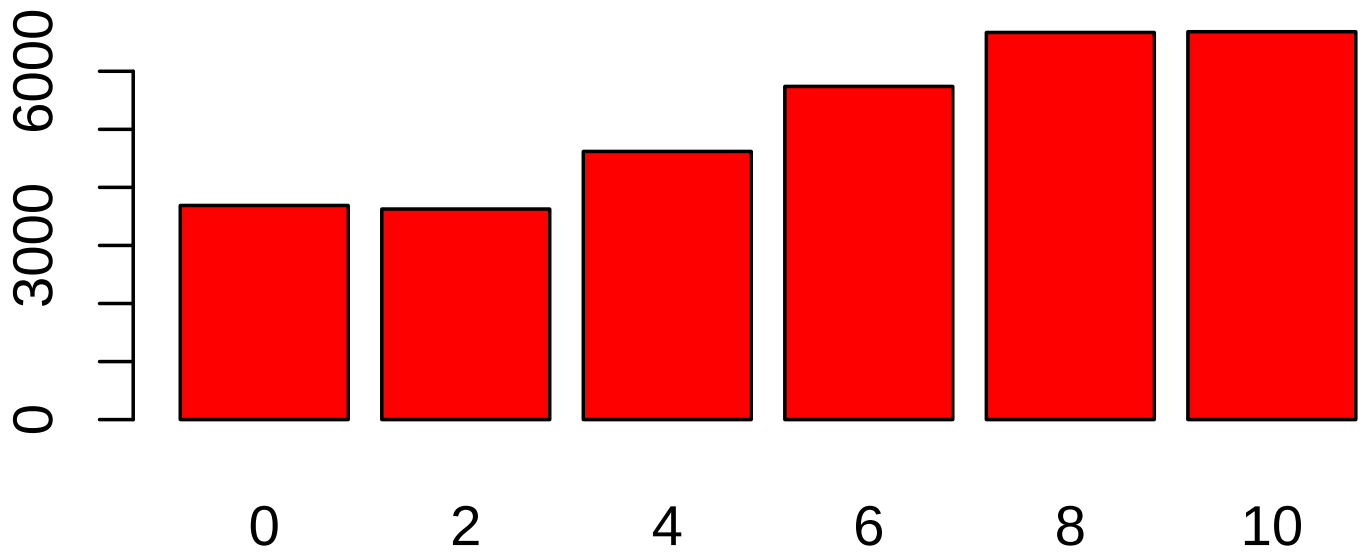
Grüne Farbe

```
barplot(tabScore,col=rgb(0,1,0))
```



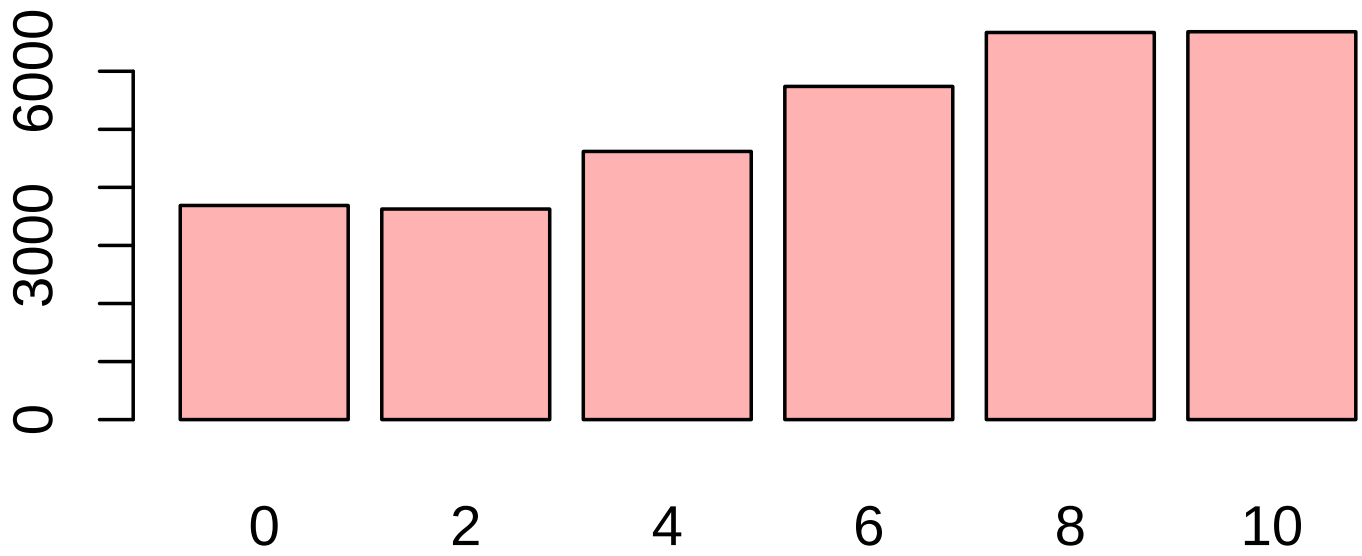
Rote Farbe

```
barplot(tabScore,col=rgb(1,0,0))
```



Transparent

```
barplot(tabScore,col=rgb(1,0,0,.3))
```



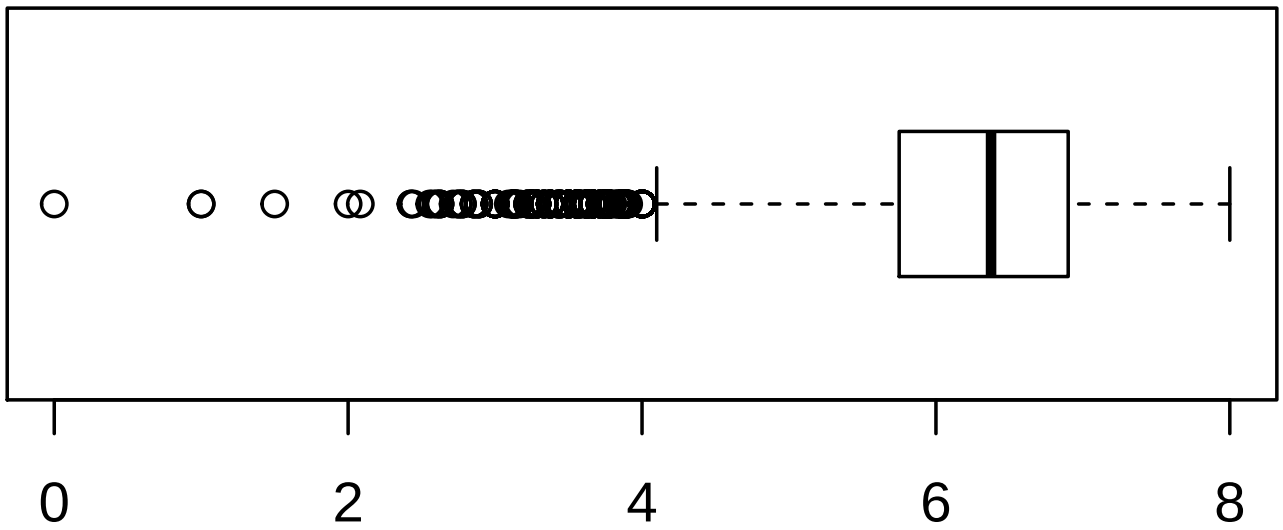
Boxplot

- Einen einfachen Boxplot erstellt man mit `boxplot()`
- Auch `boxplot()` muss mindestens ein Beobachtungsvektor übergeben werden

```
?boxplot
```

Horizontaler Boxplot

```
boxplot(Chem97$gcsescore,
horizontal=TRUE)
```

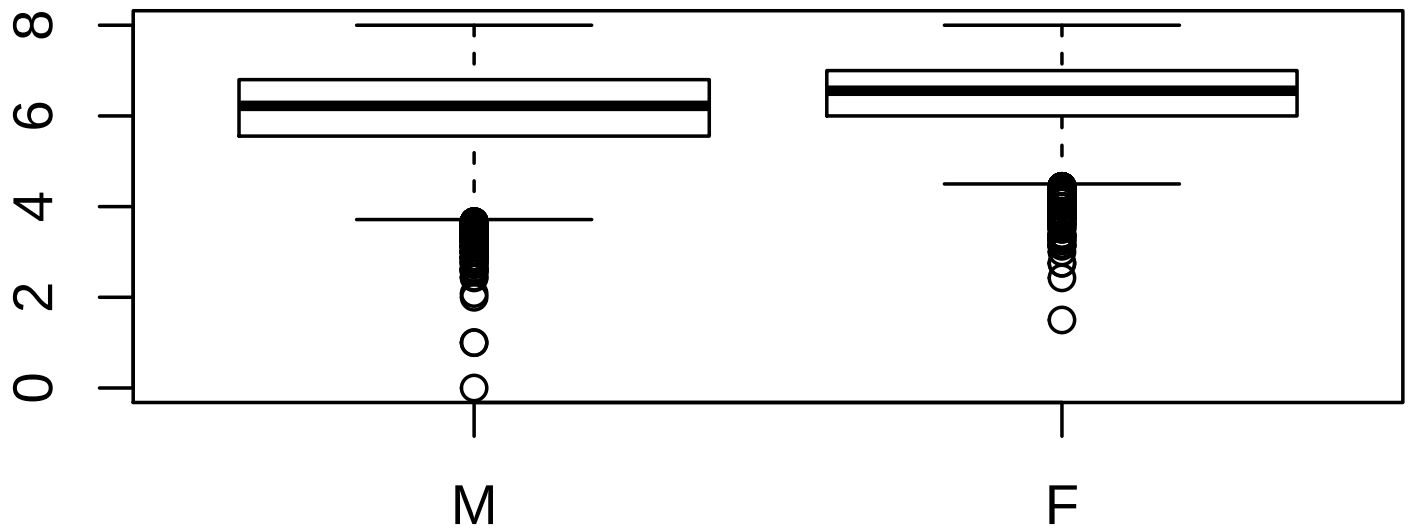


Gruppierte Boxplots

- Ein sehr einfacher Weg, einen ersten Eindruck über bedingte Verteilungen zu bekommen ist über sog. Gruppierte notched Boxplots
- Dazu muss der Funktion `boxplot()` ein sog. Formel-Objekt übergeben werden
- Die bedingende Variable steht dabei auf der rechten Seite einer Tilde

Beispiel grupierter Boxplot

```
boxplot(Chem97$gcsescore~Chem97$gender)
```



Alternativen zu Boxplot

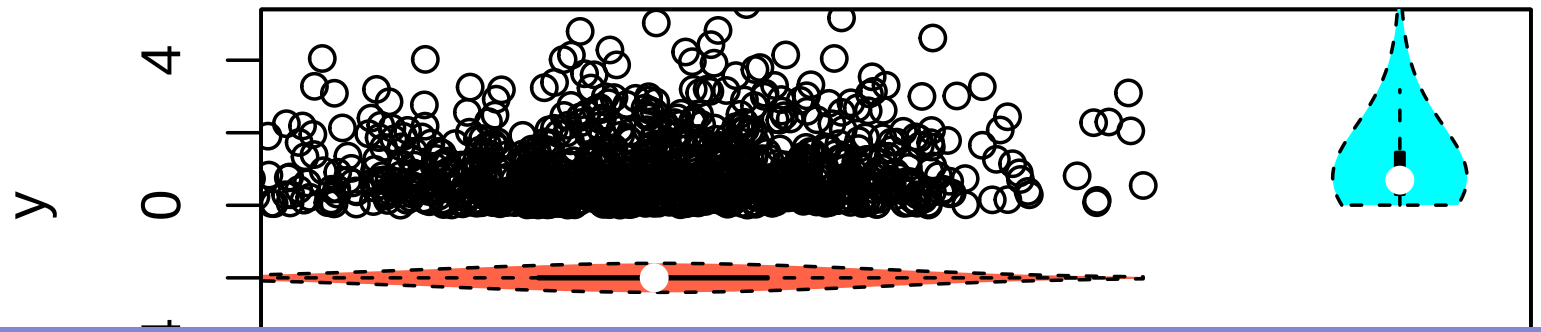
vioplot

- Baut auf Boxplot auf
- Zusätzlich Informationen über Dichte der Daten
- Dichte wird über Kernel Methode berechnet.
- weißer Punkt - Median
- Je weiter die Ausdehnung, desto größer ist die Dichte an dieser Stelle.

```
# Beispieldaten erzeugen
x <- rnorm(1000)
y <- rexp(1000,1)
```

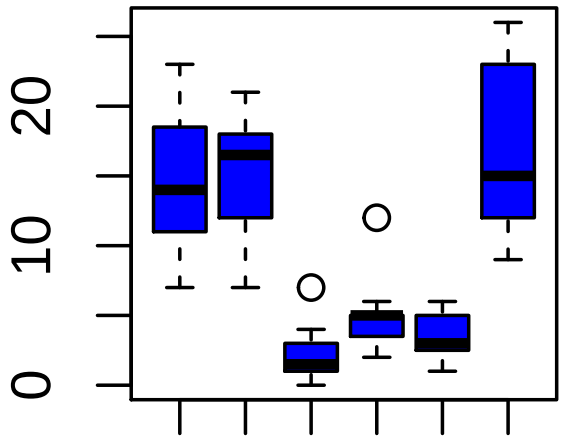
Beispiel vioplot

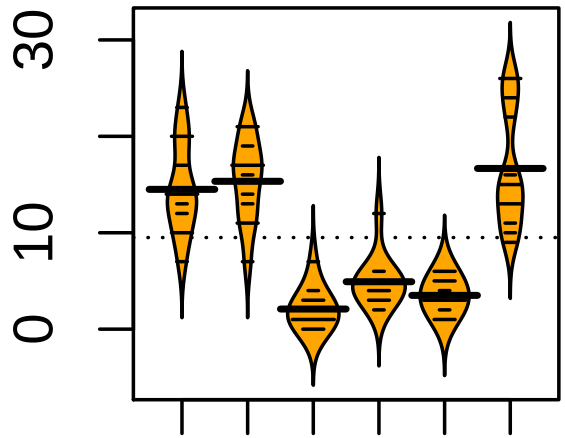
```
library(vioplot)
plot(x, y, xlim=c(-2,5), ylim=c(-5,5))
vioplot(x, col="tomato", horizontal=TRUE, at=-2,
        add=TRUE,lty=2, rectCol="gray")
vioplot(y, col="cyan", horizontal=FALSE, at=4.5,
        add=TRUE,lty=2)
```



Alternativen zum Boxplot

```
library(beanplot)
par(mfrow = c(1,2))
boxplot(count~spray,data=InsectSprays,col="blue")
beanplot(count~spray,data=InsectSprays,col="orange")
```





Aufgabe Balkendiagramm

Aufgabe - einfache Grafiken

- Laden Sie den Datensatz `VADeaths` und erzeugen Sie den folgenden plot: