Nutzung von GeoDaten in den Sozialwissenschaften - R-Recap

Jan-Philipp Kolb

07 April 2016

Hallo Welt

Pakete installieren und laden

Ein Paket installieren

```
install.packages("ggmap")
```

Ein Paket laden

```
library(ggmap)
```

```
## Loading required package: ggplot2
```

Funktionen durchführen

```
# BSP Funktion die kein Argument braucht:
date()
```

```
## [1] "Wed Apr 06 12:47:06 2016"
```

```
# BSP Funktion mit einem Argument
sqrt(5)
```

```
## [1] 2.236068
```

```
# BSP Funktion mit mehr Argumenten
sample(1:10,3)
```

```
## [1] 6 8 4
```

Der Zuweisungspfeil

So wird ein Objekt a erzeugt:

```
a <- 5
```

Dieses Objekt befindet sich dann im Workspace.

Objekt b - Zahlen von 1 bis 10:

```
b <- 1:10
```

So bekommt man Hilfe

Mit einem Fragezeichen vor dem Befehlsnamen bekommt man die Hilfe angezeigt.

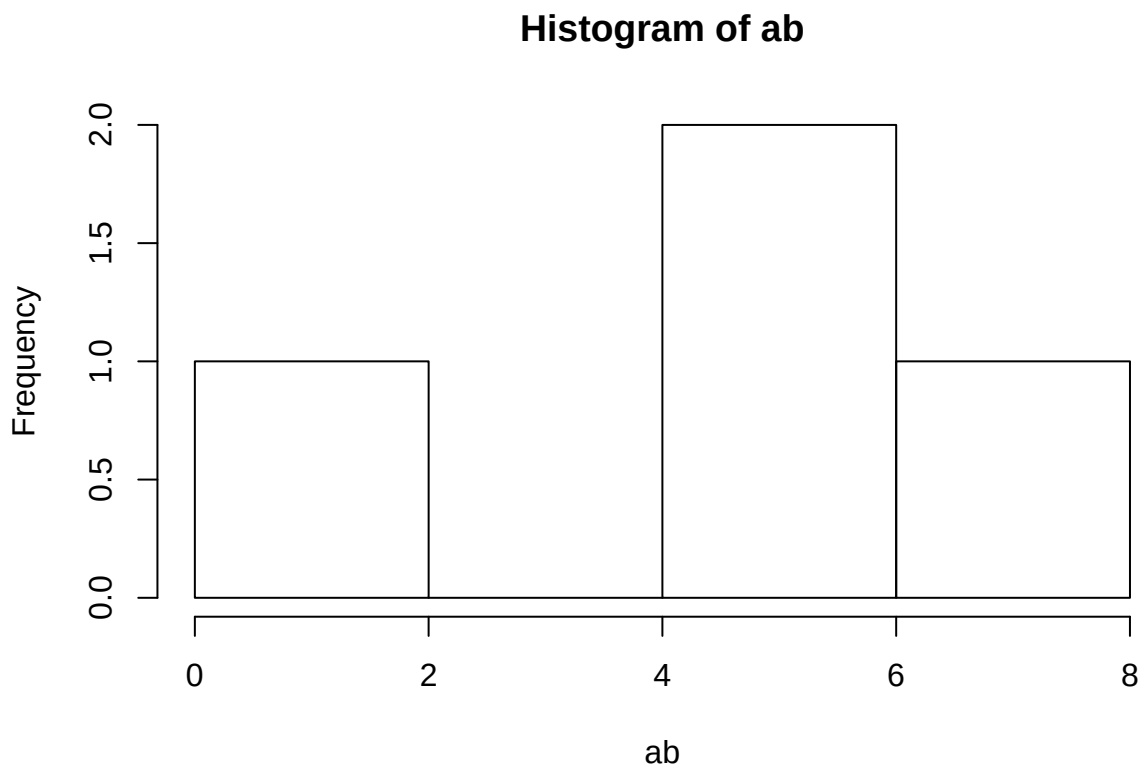
```
?mean
```

Mehrere Zahlen zu einem Vektor verbinden

```
ab <- c(1,5,6,7)
```

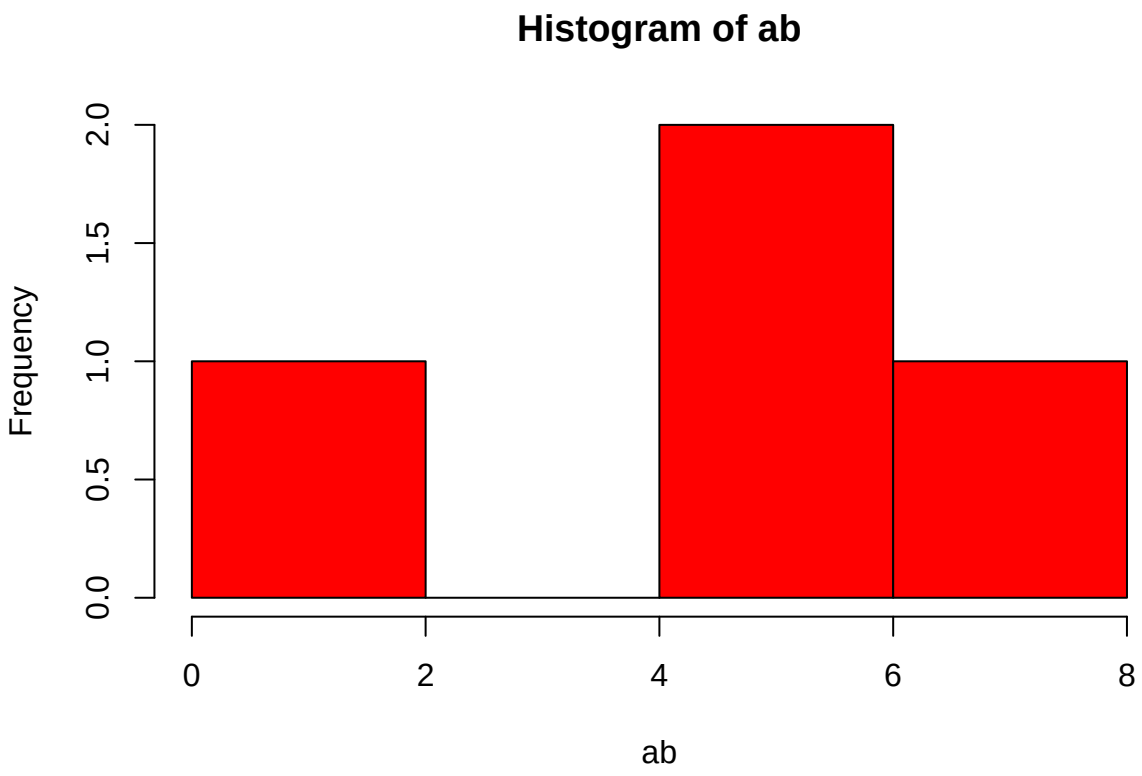
Eine Graphik erzeugen

```
hist(ab)
```



Farbe hinzu

```r
hist(ab,col="red")
```



Objekttypen und Indizierung

Rstudio installieren/updaten

Die aktuellste Version von Rstudio kann man hier herunterladen:

https://www.rstudio.com/products/rstudio/download/

Nutzung von AddIns in Rstudio

https://rstudio.github.io/rstudioaddins/

Um den Colourpicker zu nutzen muss bspw. folgendes Paket installiert werden:

```r
install.packages("shinyjs")
```

Objekttypen

```r
a <- c(1,2,3,4)
str(a)
```

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

```r
b <- c("red","green","blue")
str(b)
```

```
##  chr [1:3] "red" "green" "blue"
```

```r
d <- c(1,2,"green")
str(d)
```

```
##  chr [1:3] "1" "2" "green"
```

Sequenzen

```r
1:4
```

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

```r
4:1
```

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

```r
seq(1,4,by=.5)
```

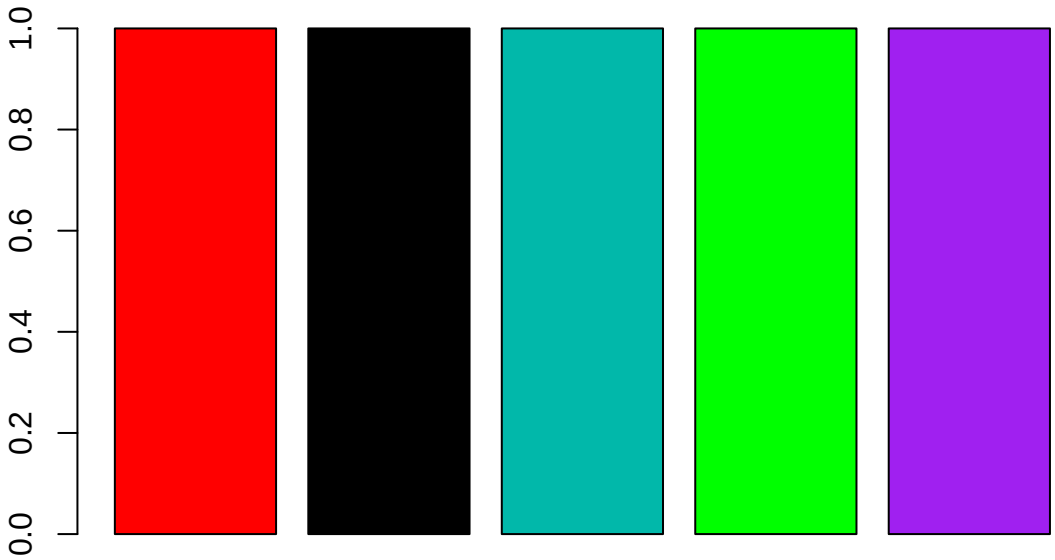
```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

```r
rep(1,5)
```

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

Farben in R

```r
barplot(rep(1,5),
        col=c("red",1,"#01B8AA",
              rgb(0,1,0),"purple"))
```



Logische Abfragen

```
vector1 <- 1:10
vector1==5
```

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

```
vector1>5
```

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

```
vector1<5
```

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

Ein `data.frame`

```
A <- 1:6
B <- rep(1:2,3)
C <- runif(6)
```

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

```
##   A B         C
## 1 1 1 0.1019685
## 2 2 2 0.2982027
## 3 3 1 0.5809665
## 4 4 2 0.7953655
## 5 5 1 0.6672730
## 6 6 2 0.2655281
```

Auswahl auf einem `data.frame`

Nur eine Zeile:

```
ABC[1,]
```

```
##   A B         C
## 1 1 1 0.1019685
```

Nur eine Spalte:

```
ABC[,1]
```

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

Gleiches Ergebnis:

```
ABC$A
```

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

Darstellung ohne erste Zeile

```
ABC[-1,]
```

```
##   A B         C
## 2 2 2 0.2982027
## 3 3 1 0.5809665
## 4 4 2 0.7953655
## 5 5 1 0.6672730
## 6 6 2 0.2655281
```

Indizierung

```
ABC[ABC$B==2,]
```

```
##   A B         C
## 2 2 2 0.2982027
## 4 4 2 0.7953655
## 6 6 2 0.2655281
```

Manuals zur Einführung in R

- R-Intro auf cran.

Schleifen

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

Daten Import

Import von csv-Dateien

- csv-Dateien aus dem Internet direkt einlesen

```
link <- "https://raw.githubusercontent.com/Japhilko/GeoData/master/data/TrustData.csv"

trust_data <- read_csv2(link)
```

- Für Deutsche Daten sollte `read_csv2` verwendet werden

Sich die Daten anzeigen lassen

```
head(trust_data)
```

```
colnames(trust_data)
```

Import Excel-files

```
library("readxl")
```

```
ab <- read_excel("dataset.xlsx",sheet=1)
```

- Blatt Index als zweites Argument nötig

Übersicht zum Weltkulturerbe Datensatz

```
table(whc$category)
```

```
colnames(whc)
```

Ein Balkendiagramm erzeugen

```
barplot(table(whc$category))
```

Import .sav-Daten

```
library(foreign)
mz10 <- read.spss("mz2010_cf.sav",to.data.frame=T)
```

Der Befehl read.spss

```
?read.spss
```

```
## No documentation for 'read.spss' in specified packages and libraries:
## you could try '??read.spss'
```

stata-Dateien lesen

- Datensatz über Studierende (2000) vom Forschungsdatenzentrum
- Datensatz hat 206867 Einträge

```
studis <- read.dta("studenten_cf_2000.dta")
```

- Blog post zum lesen und schreiben von stata Dateien auf is.R

Quick R zum Datenimport

http://www.statmethods.net/input/index.html

Datenexport

```
x <- runif(1000)
save(x, file="WasAuchImmer.RData")
```

Datenmanagement ein wenig wie bei SPSS

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

Datentypen, Graphiken, Schleifen etc.

Ein `data.frame`

```
?data.frame
```

```
L3 <- LETTERS[1:3]
fac <- sample(L3, 10, replace = TRUE)
d <- data.frame(x = 1, y = 1:10, fac = fac)
```

x	y	fac
1	1	B
1	2	B
1	3	A
1	4	A
1	5	B
1	6	A
1	7	A
1	8	C
1	9	A
1	10	B

Einen Überblick über die Daten

```
head(d)
```

```
##   x y fac
## 1 1 1   B
## 2 1 2   B
## 3 1 3   A
## 4 1 4   A
## 5 1 5   B
## 6 1 6   A
```

Datentypen

- Wenn ein Vektor als `factor` dargestellt wird, gibt es manchmal Probleme diesen mit einem anderen Vektor zu matchen.
- Deshalb muss man den Datentyp verändern.

- Am Besten ist es `character` zu matchen.

`integer` in `character` umwandeln

```
A <- 1:10
A
```

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

```
str(A)
```

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

```
B <- as.character(A)
B
```

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

```
str(B)
```

```
##  chr [1:10] "1" "2" "3" "4" "5" "6" "7" "8" "9" ...
```

Ein `factor`

```
AB <- sample(LETTERS,4)
AB <- as.factor(AB)
levels(AB) <- LETTERS
table(AB)
```

```
## AB
## A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
## 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Das Matchen

```
?match
```

```
1:10 %in% c(1,3,5,9)
```

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

- Ergebnis ist ein `logical`.
- Man bekommt die Aussagen wahr und falsch.

```r
match(1:10,c(1,3,5,9))
```

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

- Als Ergebnis bekommt man die Stelle an der sich die Zahl im zweiten Vektor wiederfindet.

Fehlende Werte

- Fehlende Werte sind in R mit `NA` definiert.
- Man bekommt die Information, ob es sich um einen fehlenden Wert handelt mit der Funktion `is.na`.

```r
A <- 1:10
is.na(A)
```

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

```r
A[5] <- NA
is.na(A)
```

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

Die Länge eines Vektors

```r
A <- 1:10
length(A)
```

```
## [1] 10
```

if-Abfrage

- Nur wenn Bedingung erfüllt ist, wird das Statement in den geschweiften Klammern ausgeführt.

```r
a <- 5
if (a>4){
  cat("Dies stimmt")
}
```

```
## Dies stimmt
```

```r
a <- 3
if (a>4){
  cat("Dies stimmt")
}
```

Schleifen

```r
A <- 1
for (i in 1:5){
  A <- A + i
  cat(A,"\n")
}
```

```
## 2
## 4
## 7
## 11
## 16
```

Matchen mit agrep

```r
?agrep
```

```r
agrep("lasy", "1 lazy 2")
```

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

Die Funktion which

```r
A <- 1:10
A
```

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

```r
which(A==5)
```

```
## [1] 5
```

```r
which(A>5)
```

```
## [1]  6  7  8  9 10
```

Tabellieren

```r
A <- sample(1:10,100,replace=T)
table(A)
```

```
## A
##  1  2  3  4  5  6  7  8  9 10
## 10 12 13  9  9  8  7  9 12 11
```

Die Funktion cut

- Diese Funktion kann verwendet werden um zu kategorisieren

```
?cut
```

```
Z <- rnorm(10000)
head(Z)
```

```
## [1] -1.8133738  1.1347637 -2.3641291 -0.3203864 -0.2203685 -0.1515251
```

```
Zc <- cut(Z, breaks = -6:6)
head(Zc)
```

```
## [1] (-2,-1] (1,2]   (-3,-2] (-1,0]  (-1,0]  (-1,0]
## 12 Levels: (-6,-5] (-5,-4] (-4,-3] (-3,-2] (-2,-1] (-1,0] (0,1] ... (5,6]
```

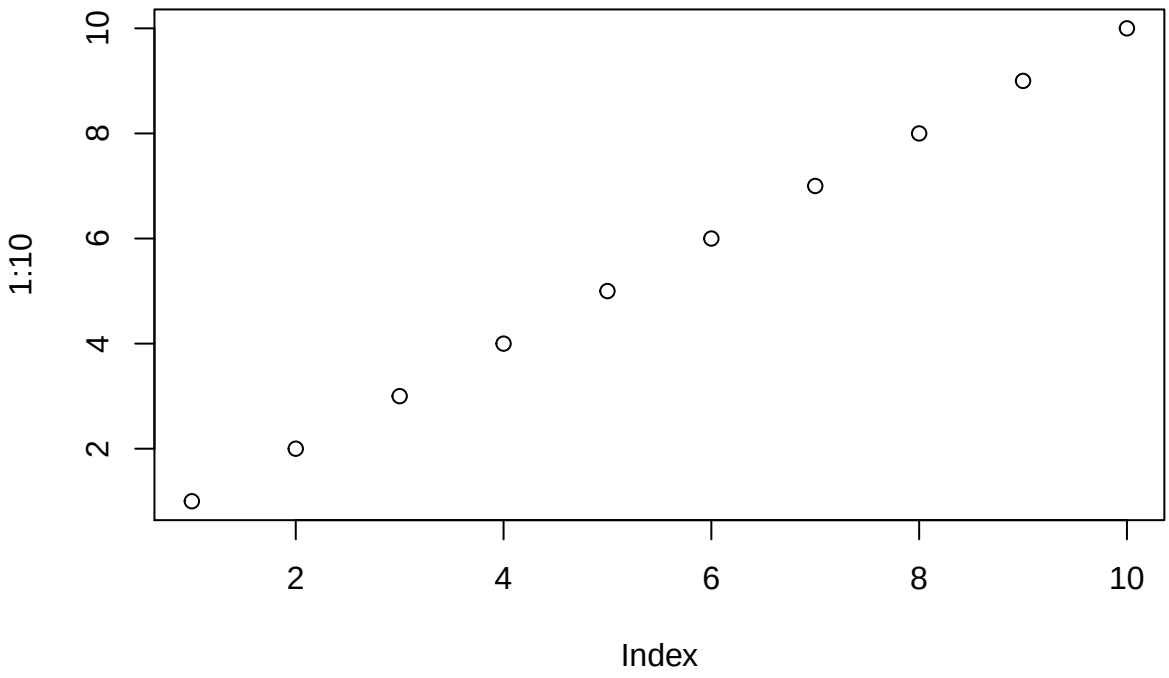
```
table(Zc)
```

```
## Zc
## (-6,-5] (-5,-4] (-4,-3] (-3,-2] (-2,-1]  (-1,0]   (0,1]   (1,2]   (2,3]
##       0       0      13     221    1391    3473    3399    1297     193
##   (3,4]   (4,5]   (5,6]
##      13       0       0
```

Die Funktion `plot`

- `plot` ist eine generische Funktion
- d.h. je nachdem welches Objekt man rein steckt kommt ein anderes Ergebnis heraus.

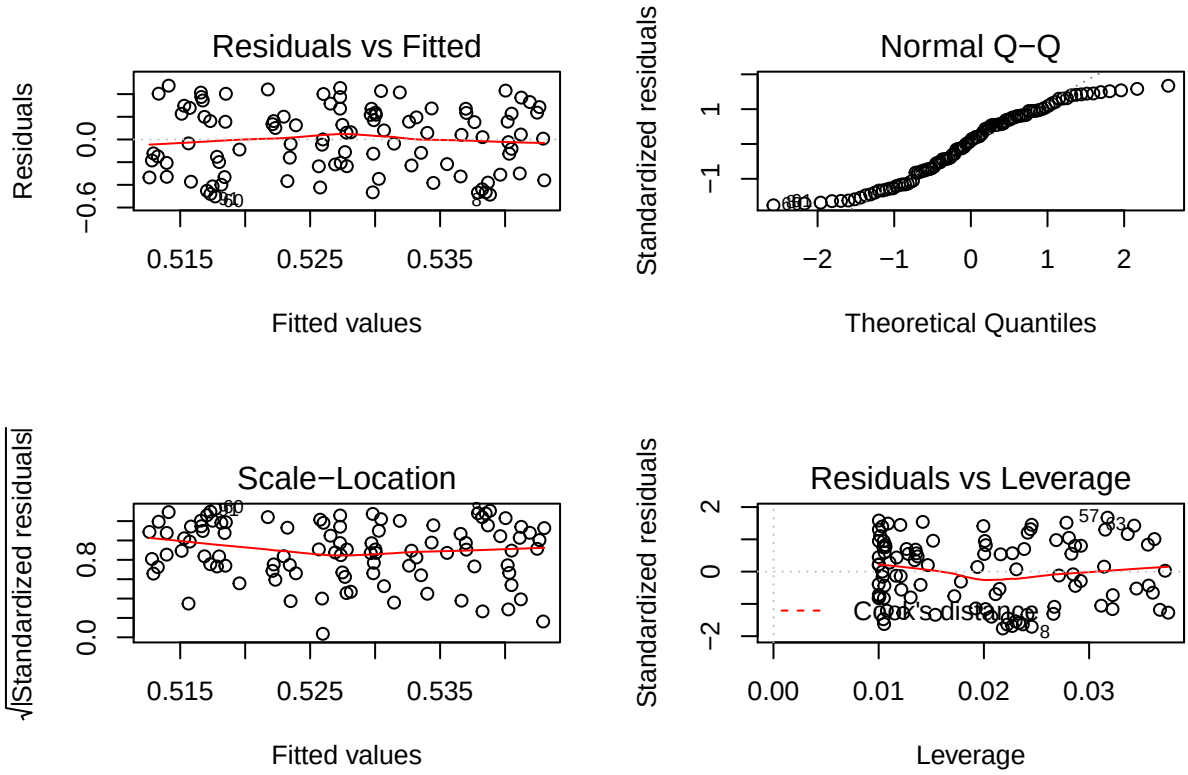
```
plot(1:10)
```



Ein weiterer plot

```
A <- runif(100)
B <- runif(100)
mod1 <- lm(A~B)

par(mfrow=c(2,2))
plot(mod1)
```

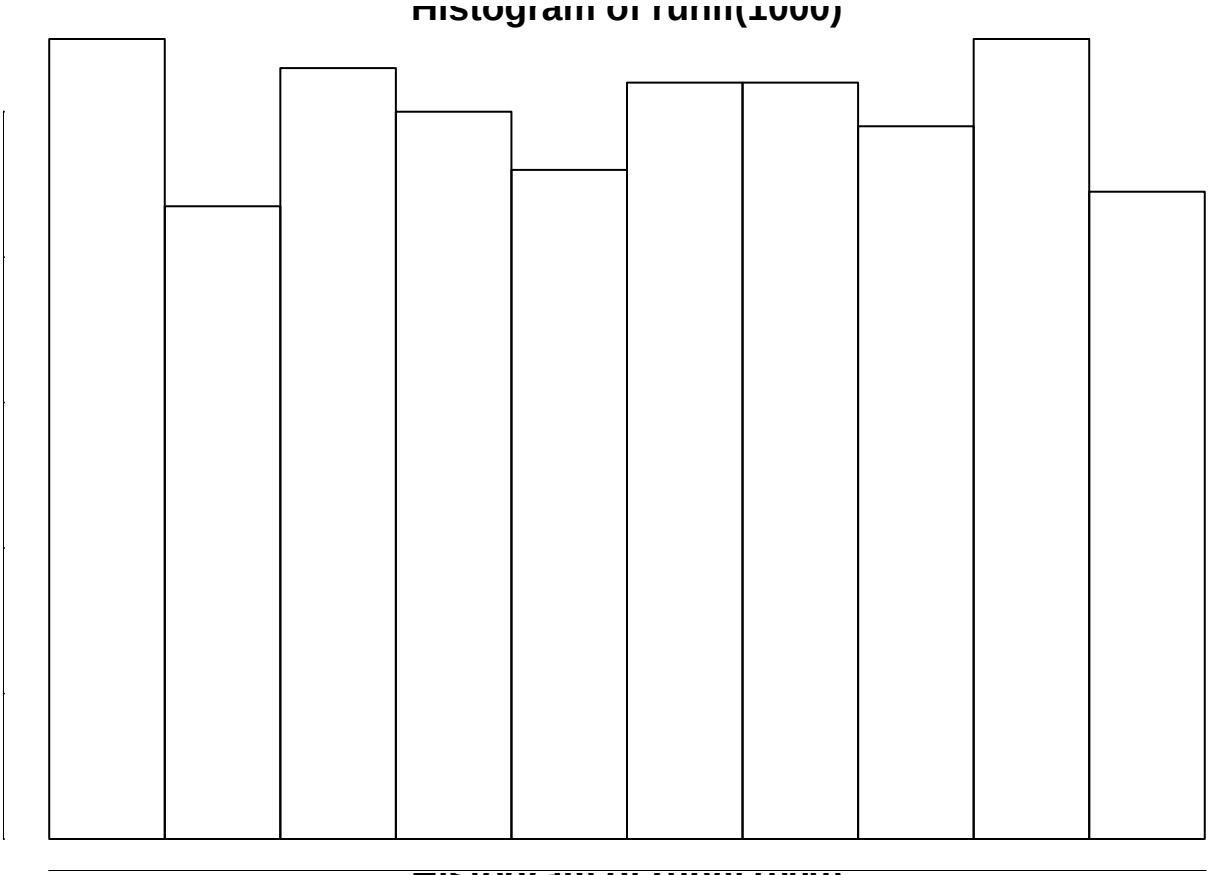


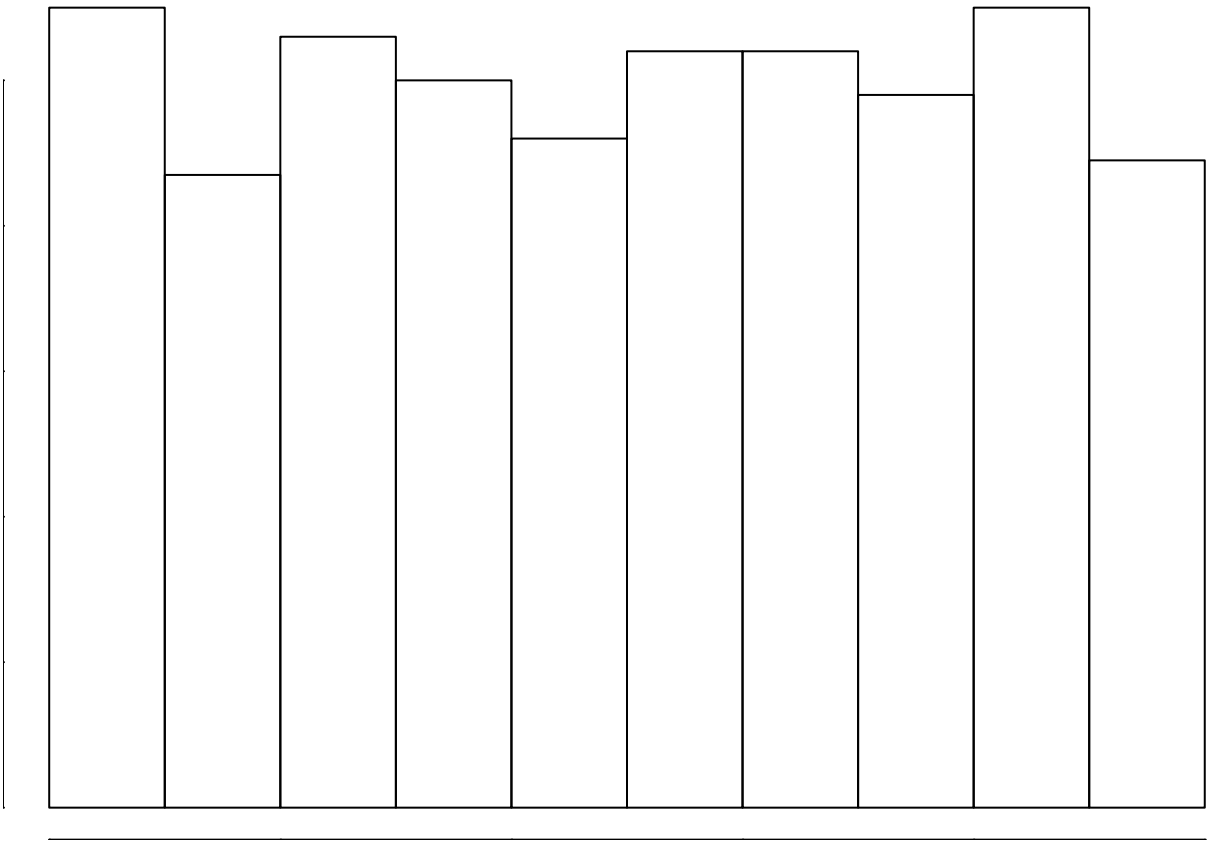
Eine Graphik ohne Rand

- Optionen bei der Gestaltung von Basis-Graphiken

```
?par
```

```
par(mai=c(0,0,0,0))
plot(hist(runif(1000)))
```

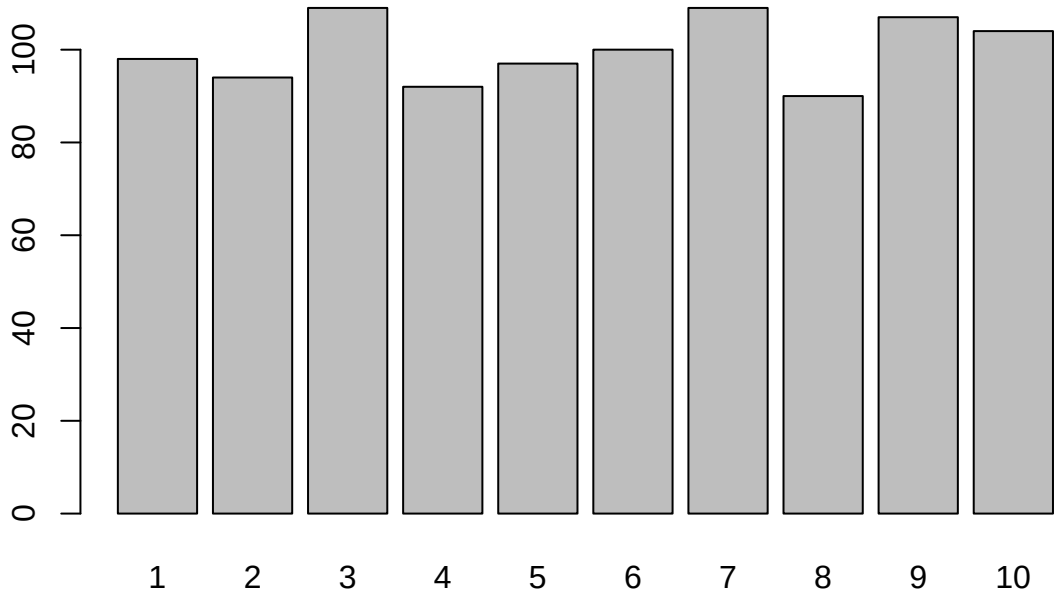

Histogram of runif(1000)


Histogram of runif(1000)

Ein Balkendiagramm

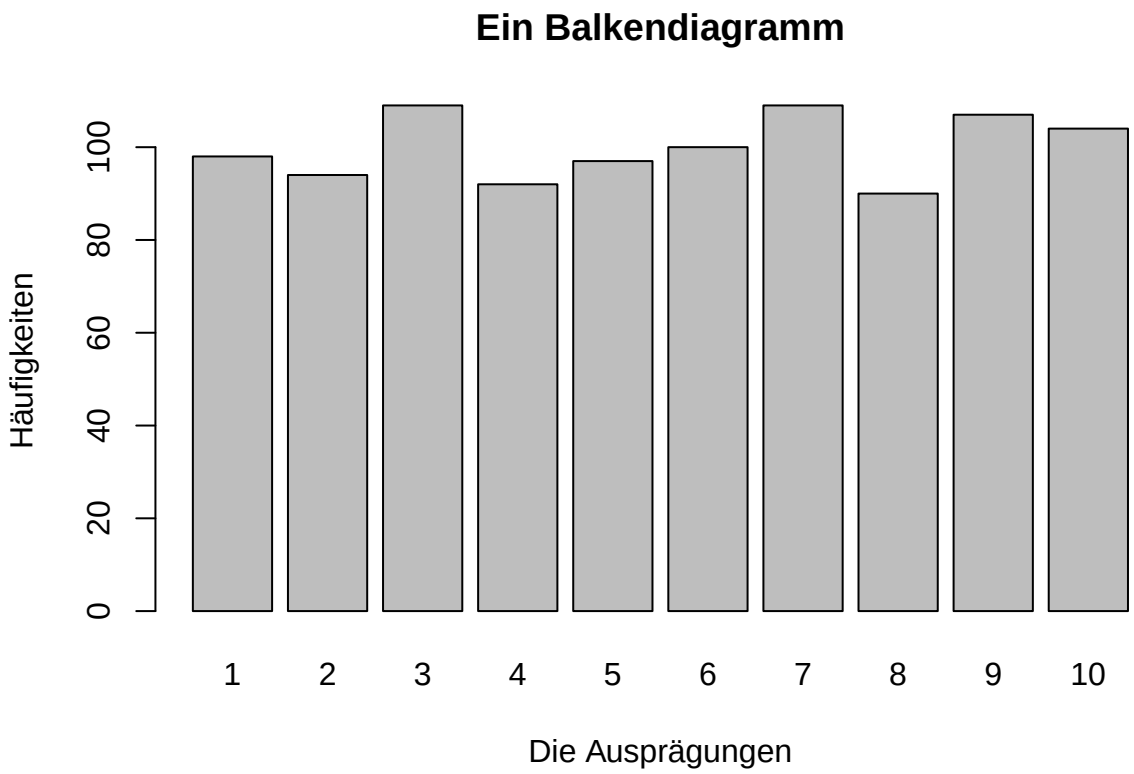
```
A <- sample(1:10,1000,replace=T)
tabA <- table(A)

barplot(tabA)
```



Die Achsenbezeichnung

```
barplot(tabA,xlab="Die Ausprägungen",
        ylab="Häufigkeiten",
        main="Ein Balkendiagramm")
```



Zufallszahlen

Erzeugung von ganzzahligen Zufallszahlen

```
Asample <- sample(1:5,1000,replace=T)
head(Asample)
```

```
## [1] 4 2 5 5 1 5
```

```
table(Asample)
```

```
## Asample
##   1   2   3   4   5
## 202 208 184 210 196
```

Gleichverteilte Zufallszahlen

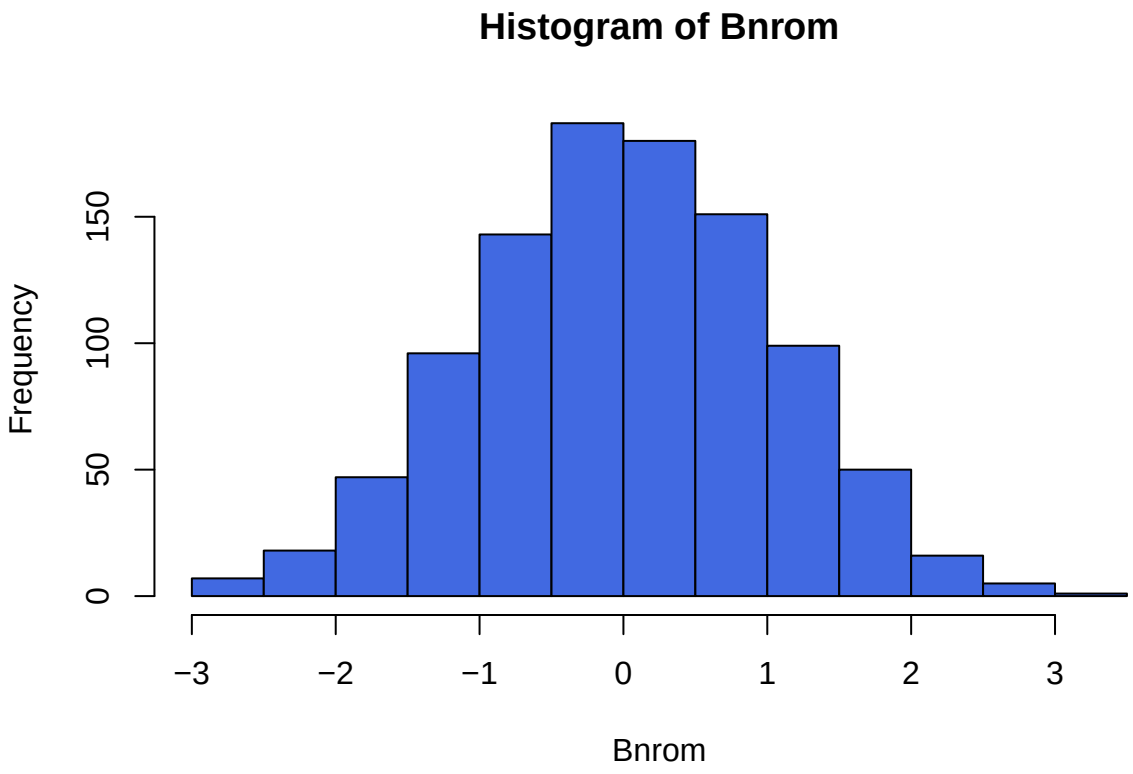
```
A <- runif(100)
head(A)
hist(A)
```

Gleichverteilte Zufallszahlen in einem Rahmen

```
A2 <- runif(100,100,200)
head(A2)
hist(A2)
```

Normalverteilte Zufallszahlen

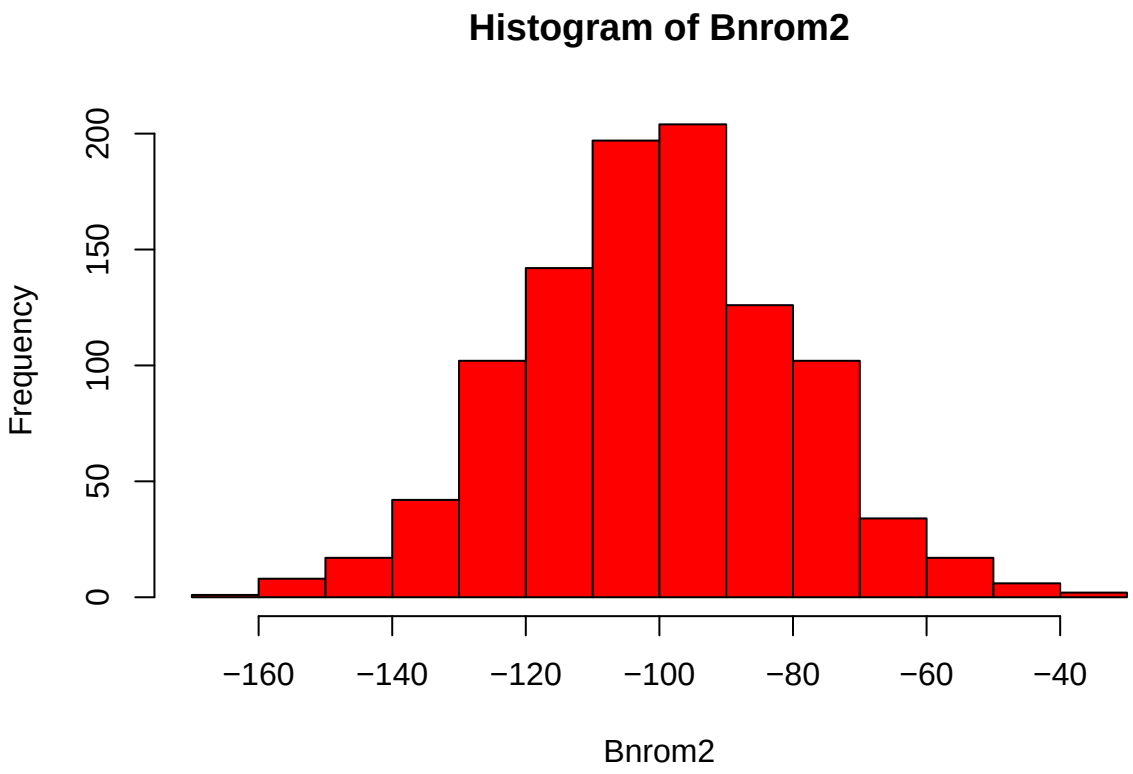
```
Bnrom <- rnorm(1000)
hist(Bnrom,col="royalblue")
```



Normalverteilte Zufallszahlen

```
?rnorm
```

```
Bnrom2 <- rnorm(1000,mean=-100,sd=20)
hist(Bnrom2,col="red")
```



Das Runden

Zahlen runden:

```
(A <- rnorm(10))
```

```
##  [1] -0.5329951 -1.0313481  0.8892700  0.9531233 -2.5507226  1.4517915
##  [7] -0.6345737  0.6534621 -2.0725455 -0.5246061
```

```
round(A)
```

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

```
round(A,digits=2)
```

```
##  [1] -0.53 -1.03  0.89  0.95 -2.55  1.45 -0.63  0.65 -2.07 -0.52
```

R als Taschenrechner

```
2/4
```

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

```
2*4
```

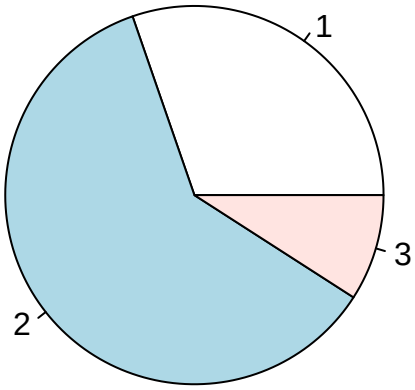
```
## [1] 8
```

```
sqrt(5)
```

```
## [1] 2.236068
```

Ein Kreisdiagramm

```
Students <- c(100, 200,30)
pie(Students)
```



Graphiken speichern

```
pdf("pie_Students.pdf")
pie(Students)
dev.off()
```

```
## pdf
##   2
```