Grundlegende R Kommandos

Funktionen

- `quit()` – beendet R
- `source("test.R")` – führt `test.R` aus
- `save.image(file="file.RData")` – speichert den aktuellen Zustand ab
- `load("file.RData")` – lädt abgespeicherten Zustand
- `rm( obj )` – löscht `obj`
- `length( obj )` , `class( obj )` , `mode( obj )` , `attributes( obj )` – liefern Informationen über das Objekt
- `str( obj )`, `structure(obj)` – zeigen die Struktur des Objektes an
- `summary( obj )` – statistische Zusammenfassung

Mathematische Funktionen

- Trigonometrische Funktionen: `sin(x), cos(x), tan(x)`
- Inverse trigonometrische Funktionen: `asin(x), acos(x), atan(x), atan2(x,y)`
- Hyperbolische Funktionen: `sinh(x), cosh(x), tanh(x)`
- Inverse hyperbolische Funktionen: `asinh(x), acosh(x), atanh(x)`
- Exponential- und Logarithmusfunktionen: `log(x), log(x,base=2), log2(x), log10(x), exp(x)`, ausserdem: `x^y`
- Diverse mathematische Funktionen: `sqrt(x), abs(x)`

Logische Operatoren

- `< > <= >= == !=` – Relationale Operatoren
- `& | !` – Logische Operatoren

Vektoren

- `x<-numeric(250)` – erzeugt 0,0,0,...,0
- `x<-c(1,2,3)` – erzeugt 1,2,3
- `x<-1:10` – erzeugt 1,2...,10
- `x<-seq(1,10)` – erzeugt 1,2...,10
- `x<-seq(1,10,0.1)` – erzeugt 1,1.1,1.2,1.3..,9.9,10
- `x<-seq(1,10.01,0.1)` – erzeugt 1,1.1,1.2,1.3..,9.9,10,10.0
- `x<-seq(1,10,len=5)` – erzeugt 1,3.25,5.5,7.75,10
- `x<-rep( 1:2 , times=5)` – erzeugt 1,2,1,2,1,2,1,2,1,2
- `x<-rep( 1:2 , each=5)` – erzeugt 1,1,1,1,1,2,2,2,2,2
- `x<-rep( c(1,2,3,4) , c(1,2,3,4) )` – erzeugt 1,2,2,3,3,3,4,4,4,4

Subsets von Vektoren

- `y<-x[x>1.0]` – erzeugt einen Vektor der alle Element aus `x` mit $x > 1$ enthält
- `y<-x[1:10]` – die ersten 10 Elemente aus `x`
- `y<-sample(x,10)` – wählt zufällig 10 Werte aus x

File In/Output

- `x<-read.table("file")` – liest `file` ein und erstellt daraus ein data.frame
- `x<-scan("file") , list(0,"",0)` – liest Daten nach einem speziellen Format ein, und erstellt daraus einen Vektor oder eine Liste
- `write.table( obj , file="file.dat")` – schreibt `obj` in `file.dat`

Grundlegende Statistikfunktionen

- `sum(x)` – die Summe von x
- `mean(x)` – das arithmetische Mittel von x
- `weighted.mean(x,w)` – das gewichtete Mittel von x mit den Gewichten w
- `median(x)` – der Median von x
- `var(x)` – die Varianz von x
- `sd(x)` – Die Standardabweichung von x
- `cov(x,y)` – die Kovarianz von x,y
- `cor(x,y)` – die Korrelation zwischen x,y
- `quantile(x,0.25)` – berechnet das 25% Quantile von x

Zufallszahlen – Wahrscheinlichkeiten

- `x<-runif(1000,min=-1,max=1)` – erzeugt 1000 gleichverteilte Zufallszahlen zwischen -1 und 1
- `x<-rnorm(1000,mean=2,sd=4)` – erzeugt 1000 normalverteilte Zufallszahlen mit Mittelwert 0 und Standardabweichung 4
- `x<-rexp(1000,rate=1)` – erzeugt 1000 exponentialverteilte Zufallszahlen mit "Halbwertszeit" 1
- `rpois, rbinom, rchisq, rt, rlnorm, ...` – Poisson-, Binomial-, χ^2-, Student's t-, log-normal- Verteilung

R als Programmiersprache

Funktionen definieren

- `func <- function(x) { anweisungen }` – erzeugt die Funktion `func`
- Anweisungen werden durch Semikolons getrennt.
- Die letzte Ausgabe ist der Rückgabewert
- Beispiel: `func<-function(x) { x ; 2*x }` – gibt $2 \cdot x$ zurück

Bedingungen

- `if( logical ) { Anweisungen1 } else { Anweisungen2 }` – falls logical wahr ist werden die Anweisungen1 ausgeführt ansonsten Anweisungen2
- Auch hier: mehre Anweisungen werden durch Semikolon getrennt.
- Beispiel `if( x<10 ) { "X ist kleiner als 10" } else { "X ist groesser als 10" }`

Schleifen

- `for( i in 1:10 ) { Anweisungen }` – for Schleife
- `while( logical ) { Anweisungen }` – while Schleife
- `repeat { Anweisungen }` – Repeat Schleife, hat keine Abbruchbedingung, muss mit `break` unterbrochen werden

Grafik mit R

Ausgabe in pdf

- Sieht meistens so aus
```
pdf("plot.pdf")
plot( c )
dev.off()
```

Diagrammtypen

- `plot(x,y)` – Scatterplot
- `plot(x)` – Scatterplot gegen die Indices von `x`
- `points(x,y)` – fügt Punkte hinzu
- `lines(x,y)` – fügt Linien hinzu
- `qqplot(x,y)` – Plottet die Quantile von `x` und `y` gegeneinander auf
- `hist(x)` – erstellt und plottet das Histogramm von `x`

Multiplots

-
```
oldpar<-par
par(mfcol=c(1,2))
plot(x,y)
plot(x,z) par<-oldpar
```