**R Kurs Basler Versicherungen:**

**Einführung und Datenmanagement**

**von Simon Scott, Produkt- und Marktanalyse**

[1 Download 2](#_Toc448841016)

[2 Kurze Beschreibung von R Studio 2](#_Toc448841017)

[2.1 Skripts schreiben 2](#_Toc448841018)

[3 R Basics 3](#_Toc448841019)

[3.1 R als mächtiger!!!!!!!! Taschenrechner 3](#_Toc448841020)

[3.2 Sonderzeichen 3](#_Toc448841021)

[3.3 Variable zuordnen/speichern 3](#_Toc448841022)

[3.4 Zahlenreihen 4](#_Toc448841023)

[3.5 Relationen 4](#_Toc448841024)

[3.6 Hilfe 4](#_Toc448841025)

[3.7 Zusätzliche Pakete 4](#_Toc448841026)

[4 Objekte 5](#_Toc448841027)

[4.1 Wichtige Grundklassen von Objekten 5](#_Toc448841028)

[4.1.1 Integer, Numeric 5](#_Toc448841029)

[4.1.2 Character 5](#_Toc448841030)

[4.1.3 Factor 5](#_Toc448841031)

[4.2 Objekte: Vektor, Matrix und Data Frame 5](#_Toc448841032)

[4.2.1 Vektor: 5](#_Toc448841033)

[4.2.2 Matrix: 6](#_Toc448841034)

[4.2.3 Data Frame 6](#_Toc448841035)

[4.3 Indexierung von Objekten 6](#_Toc448841036)

[4.4 Rechenoperationen 8](#_Toc448841037)

[5 Funktionen 8](#_Toc448841038)

[5.1 Daten Import/Export 8](#_Toc448841039)

[5.1.1 .csv 8](#_Toc448841040)

[5.1.2 Excel (.xls, .xlsm) 9](#_Toc448841041)

[5.1.3 RData/rdata 9](#_Toc448841042)

[5.1.4 Zwischenablage 9](#_Toc448841043)

[5.2 Übersicht von Daten 9](#_Toc448841044)

[5.3 Nützliche Funktionen der Datenverabeitung 10](#_Toc448841045)

[5.3.1 Append 10](#_Toc448841046)

[5.3.2 Merge 10](#_Toc448841047)

[5.3.3 Aggregate 11](#_Toc448841048)

[5.4 Bearbeitung von Strings 11](#_Toc448841049)

[5.4.1 Länge eines Strings 11](#_Toc448841050)

[5.4.2 Zusammenfügen 11](#_Toc448841051)

[5.4.3 Ausschneiden 11](#_Toc448841052)

[6 Programmiersprache R 11](#_Toc448841053)

[6.1 If-Else Konstruktion 11](#_Toc448841054)

[6.2 Schlaufen 12](#_Toc448841055)

[6.3 Eigene Funktionen 12](#_Toc448841056)

[7 Literatur 12](#_Toc448841057)

[8 Funktionsverzeichnis 13](#_Toc448841058)

Download

R (Version 3.0.2 verwendet für diese Einführung):

<http://stat.ethz.ch/CRAN/>

R Studio (benötigt eine installierte R Version):

<https://www.rstudio.com/ide/download/desktop>

Kurze Beschreibung von R Studio

R Studio ist ein IDE (integrated development environment) und erleichtert den Umgang mit R. Die wichtigsten Fenster des R Studios sind:

Console:

* Direktes eintippen von Befehlen
* Befehlsverlauf mit ↑ ↓ Tasten oder Ctrl-↑ ↓
* Code completion mit Tab-Taste (klappt auch im Editor)
* Konsole löschen mit Ctrl-L

Environment:

* Anzeige der gespeicherten Objekte
* Klick auf Objekte öffnet sie im Editor Fenster

History:

* Benutzte Befehle

Editor:

* Schreiben von Skripts
* Anzeige von Datensätzen

Help:

* Funktionsbrowser

Skripts schreiben

Ein grosser Teil des Arbeitens mit R beinhaltet das Schreiben von Skripts, deshalb werden hier dem Skripten ein paar Zeilen gewidmet. Im Editor Fenster von R Studio kann ein neues R-Skript geöffnet werden.

Vorteil des Skripteschreibens:

* Alles nur einmal schreiben
* Replizierbarkeit von Ergebnissen
* Einfache Erweiterung von Code
* Vereinfacht Fehlersuche
* Nachvollziehbarkeit
* Beschreibung von Code: alles was nach dem #-Zeichen in einer Zeile folgt, wird nicht als Programmcode interpretiert
* Abspeichern von Code
* Gesamter Code wird mit Source-Knopf ausgeführt, einzelne Zeilen mit Ctrl-Enter

Es empfiehlt sich, bei jedem neuen Skript, den gewünschten Workspace-Pfad zu definieren. So wird das gewünschte Verzeichnis als Default Speicher- und Ladeort verwendet.

setwd(**"L:\\bibliothek\\Lunchseminar\\R-Kurs\\Daten")**

R Basics

R als mächtiger!!!!!!!! Taschenrechner

**1+2**

**[1] 3**

**8^0.5**

**[1] 2.828427**

**exp(5)**

**[1] 148.4132**

**cos(pi)**

**[1] -1**

Sonderzeichen

pi : π

Inf : Unendlich (entsteht z. Bsp. bei Division durch 0)

NA : Not available (entsteht z. Bsp. beim einlesen von leeren Zellen, jede weitere Operation

mit einem NA führt zu einem NA)

NaN : Not a number (entsteht z. Bsp. Bei der Operation 0\*Inf oder Inf/Inf)

NULL : leere Menge

Variable zuordnen/speichern

mit <-, -> oder = in die gewünschte Variable oder Zeichenkette. Alle folgenden Operationen ordnen der Variable Var die Zahl 42 zu:

**Var<-42**

**Var=42**

**42->Var**

Wird in einer Zeile eine Zuordnung vorgenommen, wird in der Konsole nichts angezeigt. Um dann den Inhalt einer Variable angezeigt zu bekommen, kann man den Variablenamen eintippen.

**Var**

**[1] 42**

Wichtig: Zeichenketten dürfen nicht mit einer Ziffer (0-9) beginnen oder Operatoren (\*+/-…) enthalten.

Zahlenreihen

Zahlenreihen sind ein wichtiger Grundbaustein des Arbeitens mit R:

**1:5 #Zahlen von 1 bis 5**

**[1] 1 2 3 4 5**

**seq(5,1,by=-2) #Zahlen von 5 bis 1 in -2er Schritten**

**[1] 5 3 1**

**seq(-5,7,length=4) #Eine Sequenz von 4 Zahlen von -5 bis 7**

**[1] -5 -1 3 7**

**rep(1:3,2) #2 mal die Zahlenreihe von 1 bis 3**

**[1] 1 2 3 1 2 3**

Relationen

< : kleiner

> : grösser

<= : kleiner gleich

>= : grösser gleich

!= : ungleich

is.na() : ist ein NA (s. 3.2)

is.nan() : ist ein NaN (s. 3.2)

is.infinite() : ist unendlich (s. 3.2)

is.null() : ist NULL (s. 3.2)

Bemerkung: Das Ausrufezeichen ! negiert jeweils die Relation

**5>1**

**[1] TRUE**

**!is.na(c(1,3,NA,2))**

**[1] TRUE TRUE FALSE TRUE**

Hilfe

Braucht man Hilfe zu einer Funktion in R:

?Funktion

??Funktion #Falls die Funktion sich in einem nicht geladenen Paket befindet

Alternativ kann in R Studio im unteren rechten Fenster im Help Browser die Funktion gesucht werden.

Zusätzliche Pakete

R ist eine Open-Source Plattform was den Vorteil mit sich bringt, dass es für sehr viele spezifische Problemstellungen schon vorprogrammierte Lösungen gibt. Dazu muss nur das benötigte Paket installiert werden.

Zusätzliches Paket installieren:

R Studio (unten rechts): Packages -> Install Packages -> Package auswählen

R Konsole: install.packages("paketname")

Zusätzliches Paket laden:

Bei Beginn jeder neuen Session müssen die gewünschten Extrapakete neu geladen werden.

R Studio: Packages -> Häkchen machen

R Konsole: library("paketname")

Objekte

R ist eine objektbasierte Programmiersprache. Abhängig von der Klasse eines Objektes können Funktionen und Prozesse zu unterschiedlichen Outputs führen.

Wichtige Grundklassen von Objekten

Zu bemerken ist, dass gewisse Befehle nur über gewisse Klassen ausgeführt werden können.

Wichtigstes Beispiel: Rechenoperationen nur mit Objekten möglich, die Integer oder Numeric als Grundklasse haben.

Funktion, um die Klasse eines Objektes zu erhalten:

**class()**

Funktion, um die Klasse eines Objektes zu ändern:

as."gewünschte Klasse"()

Integer, Numeric

Stetige und diskrete Zahlen

**Var<-42**

**class(Var)**

**[1] "numeric"**

Character

Strings bzw. alles zwischen Anführungszeichen was nicht als Factor definiert ist

**Namen<-"Bello"**

**class(Namen)**

**[1] "character"**

Factor

Kategoriale Variablen. Bei ihrer Handhabung ist Vorsicht geboten. Der Vorteil dieser Klasse ist erst bei ökonometrischen Modellen oder bei Plots ersichtlich.

**Geschlecht<-factor(c("m","f","f","f","m"))**

**Geschlecht**

**[1] m f f f m**

**Levels: f m**

**class(Geschlecht)**

**[1] "factor"**

Objekte: Vektor, Matrix und Data Frame

Vektor:

Eine Reihe von Werten mit identischer Grundobjektklasse. Vektoren werden mit der Funktion c() „combine“ gebildet.

**Vec<-c(1,3,6,8)**

**Vec**

**[1] 1 3 6 8**

Matrix:

Eine Matrix besteht aus Spalten und Zeilen. Die Grundobjektklasse jeder Zelle muss für die ganze Matrix identisch sein. Matrizen werden mit der Funktion matrix() gebildet. Dabei wird als erstes Argument der Zelleninhalt bestimmt, anschliessend die Anzahl Zeilen und Spalten.

**Mat<-matrix(1:12,3,4) #die Zahlen von 1 bis 12, 3 Zeilen und 4 Spalten**

**Mat**

**[,1] [,2] [,3] [,4]**

**[1,] 1 4 7 10**

**[2,] 2 5 8 11**

**[3,] 3 6 9 12**

Data Frame

Die einzelnen Werte in den Matrizen und Vektoren müssen immer die Selbe Grundobjektklasse besitzen. Um eine „Matrix-Repräsentation“ mit Spalten unterschiedlicher Objektklassen zu erhalten, wird die Objektklasse Data Frame verwendet. Dabei müssen die jeweiligen Spalten die gleiche Länge haben.

Man kann ein Data Frame mit dem entsprechenden Befehl (Beispiel folgt) zusammenflicken, im Normalfall wird der Datensatz aber aus einer Tabelle eingelesen.

himymBoys<-data.frame(Name=c("Ted","Barney","Marshall"),

Beruf=c("Architekt",NA,"Anwalt"),

Verheiratet=factor(c("n","y","y")),

Einkommen=c(40000,130020,92000))

himymBoys

Name Beruf Verheiratet Einkommen

1 Ted Architekt n 40000

2 Barney <NA> y 130020

3 Marshall Anwalt y 92000

Indexierung von Objekten

Grundidee: Objektname gefolgt von einer eckigen Klammer mit der Zeilen- und Spaltennummer der gewünschten Zelle.

Zelle eines Vektors adressieren:

Vec[3]

[1] 6

Matrix adressieren:

Einzelne Zellen:

Mat[2,4] #Zeile 2, Spalte 4

[1] 11

Einzelne Spalten oder Zeilen (wird der Eintrag für die Zeile oder Spalte leer gelassen, werden alle Zeilen bzw. Spalten adressiert):

Mat[,2] #Spalte 2

[1] 4 5 6

Mat[3,] #Zeile 3

[1] 3 6 9 12

Mat[c(1,3),c(1,3,4)] #Zeilen 1 und 3, Spalten 1,3 und 4

[,1] [,2] [,3]

[1,] 1 7 10

[2,] 3 9 12

Man kann auch alternativ die Indexierung vornehmen, indem man bestimmt, welche Werte man nicht betrachten will. Dabei wird das Minus-Zeichen verwendet:

Mat[,-1] #Die 1. Spalte nicht

[,1] [,2] [,3]

[1,] 4 7 10

[2,] 5 8 11

[3,] 6 9 12

Mat[-2,-c(2,4)] #Zeile 2 nicht, Spalte 2 und 4 nicht

[,1] [,2]

[1,] 1 7

[2,] 3 9

Indexierung von Vektoren, Matrizen und Data Frames kann durch Nummerierung (wie oben) durchgeführt werden. Dabei darf die Zahl nicht grösser als die zu indexierende Dimension (Zeile, Spalte) sein. Alternativ kann mittels eines logischen Vektors/Matrix indexiert werden, dabei müssen die Dimensionen des Indexierungsvektors/Indexierungsmatrix mit dem zu indexierenden Vektor/Matrix übereinstimmen.

indexMat<-matrix(

c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,FALSE,TRUE,TRUE,FALSE,TRUE),3,4)

indexMat

[,1] [,2] [,3] [,4]

[1,] TRUE FALSE FALSE TRUE

[2,] TRUE FALSE FALSE FALSE

[3,] TRUE TRUE TRUE TRUE

Mat[indexMat] #Matrix aus 4.2.2 mit indexMat indexieren

[1] 1 2 3 6 9 10 12

Für Data Frame Objekte ist es möglich, ganze Spalten mit dem $ Zeichen zu adressieren. Dabei wird der Spaltenname mit einem $-Zeichen an den Objektnamen gehängt.

himymBoys$Beruf

[1] Architekt <NA> Anwalt

himymBoys$Beruf[3]

[1] Anwalt

Die logische Indexierungsform ermöglicht auch die Auswahl von Subsets eines Datensatzes:

himymBoys[himymBoys$Verheiratet=="y",] #Nur die Zeilen wo Verheiratet=y

Name Beruf Verheiratet Einkommen

2 Barney <NA> y 130020

3 Marshall Anwalt y 92000

himymBoys[himymBoys$Einkommen==min(himymBoys$Einkommen),] #Kleinstes Eink.?

**Name Beruf Verheiratet Einkommen**

**1 Ted Architekt n 40000**

Rechenoperationen

Mit allen Objekten dieses Kapitels können Rechenoperation durchgeführt werden. Dabei ist zu Beachten, dass der Vektor, die Matrix oder die verwendeten Spalten des Data Frame die Grundklasse Numeric oder Integer besitzen muss/sollte.

\*/+-: Grundoperation, Zellenweise

Mat[,1]\*himymBoys$Einkommen

[1] 40000 260040 276000

t: Transponierte Matrix

t(Mat)

[,1] [,2] [,3]

[1,] 1 2 3

[2,] 4 5 6

[3,] 7 8 9

[4,] 10 11 12

%\*%: Matrixmultiplikation

Mat%\*%t(Mat)

[,1] [,2] [,3]

[1,] 166 188 210

[2,] 188 214 240

[3,] 210 240 270

Funktionen

An dieser Stelle werden nur ein paar wenige Funktionen vorgestellt, die für das Datenmanagement von Nutzen sein können. Um eine Funktion auszuführen, folgen auf den Funktionsname in einer runden Klammer die Argumente. Es ist meist möglich, einer Funktion, neben den benötigten Argumenten, auch weitere „freiwillige“ Information mitzugeben. Diese sogenannten Handle können das Verhalten der Funktion steuern (z. Bsp.: was mit NA‘s geschehen soll, ob die Kopfzeile als Titel zu interpretieren ist,…). In den Beispielen werden hin und wieder solche Handles verwendet, für Details einfach die Hilfe der Funktion aufrufen.

Daten Import/Export

.csv

Import

Am einfachsten in .csv form

Test.csv<-read.csv2("test.csv")

Test.csv<-read.csv("test.csv",sep=";")

Test.csv<-read.table("test.csv",sep=";",header=TRUE)

Wenn man den Pfad des gewünschten Files nicht kennt:

Test.csv<-read.csv(choose.files(),sep=";")

oder über R Studio Environment -> Import Dataset

Export

write.table(himymBoys, "himymBoys.csv",sep=";",row.names=FALSE)

Excel (.xls, .xlsm)

1. Excel Sheet als csv abspeichern, dann weiter siehe oben.

2. Alternativ gibt es Pakete, die Excel Dateien lesen können, hier ein Beispiel eines solchen: Paket **XLConnect:**

Der Vorteil dieses Paketes ist, dass es gesamte Workbooks abspeichern kann und somit ein xlsm mit mehreren sheets nur einmal eingelesen werden muss. (Paket installieren und laden s. 2.7)

Workbook laden:

wb.xlsm<-loadWorkbook("test.xlsm")

Sheet himym aus Workbook lesen

Test.xlsm<-readWorksheet(wb,"himym")

RData/rdata

Export

Objekte in RData Format speichern. Dabei können eine beliebige Anzahl von Objekte aufgelistet werden

save(himymBoys,Mat,Vec,file="himym.RData")

Import

load("himym.RData")

Zwischenablage

Export

Daten in Zwischenablage speichern

write.table(himymBoys, "clipboard",sep="\t",row.names=FALSE)

Import

Daten von Zwischenablage importieren

**Test.clip<-read.table("clipboard",sep="\t",header=TRUE)**

Übersicht von Daten

Es gibt mehrere Möglichkeiten um einen schnellen Überblick von Daten zu erhalten, Bsp.:

summary(): Gibt abhängig von der Grundklasse der Spalten eine Übersicht

summary(himymBoys)

Name Beruf Verheiratet Einkommen Geschlecht

Barney :1 Anwalt :1 n:1 Min. : 40000 Length:3

Marshall:1 Architekt:1 y:2 1st Qu.: 66000 Class :character

Ted :1 NA's :1 Median : 92000 Mode :character

Mean : 87340

3rd Qu.:111010

Max. :130020

str(): Gibt eine Übersicht der Klassen/Struktur eines Datensatzes

str(himymBoys)

'data.frame': 3 obs. of 5 variables:

$ Name : Factor w/ 3 levels "Barney","Marshall",..: 3 1 2

$ Beruf : Factor w/ 2 levels "Anwalt","Architekt": 2 NA 1

$ Verheiratet: Factor w/ 2 levels "n","y": 1 2 2

$ Einkommen : num 40000 130020 92000

$ Geschlecht : chr "m" "m" "m"

head(), tail(): Die letzten oder ersten Zeile eines Datensatzes

head(himymBoys) #die ersten 6 Zeilen des Datensatzes

tail(himymBoys) #die letzten 6 Zeilen des Datensatzes

Nützliche Funktionen der Datenverabeitung

Append

Zeilen anhängen:

rbind(Vec,Mat) #row bind

[,1] [,2] [,3] [,4]

Vec 1 3 6 8

1 4 7 10

2 5 8 11

3 6 9 12

Beispiel himymBoys um Daten der Girls erweitern:

himymGirls<-data.frame(Name=c("Lilly","Robin"), #Neuer Datenframe bauen

Beruf=c("Kindergärtnerin","Journalistin"),

Verheiratet=factor(c("y","y")),

Einkommen=c(22600,51300),

Geschlecht=rep("f",2))

himymGirls

Name Beruf Verheiratet Geschlecht Einkommen

1 Lilly Kindergärtnerin y f 22600

2 Robin Journalistin y f 51300

himym<-rbind(himymBoys,himymGirls) #Boys und Girls Daten Zusammenhängen

himym

Name Beruf Verheiratet Einkommen Geschlecht

1 Ted Architekt n 40000 m

2 Barney <NA> y 130020 m

3 Marshall Anwalt y 92000 m

4 Lilly Kindergärtnerin y 22600 f

5 Robin Journalistin y 51300 f

Spalten anhängen

cbind(Mat,Vec(1:3)) #column bind

[,1] [,2] [,3] [,4] [,5]

[1,] 1 4 7 10 1

[2,] 2 5 8 11 3

[3,] 3 6 9 12 6

Alternativ kann bei Data Frames eine neue Spalte mit dem $-Zeichen angehängt werden:

himymBoys$Geschlecht<-"m"

himymBoys

Name Beruf Verheiratet Einkommen Geschlecht

1 Ted Architekt n 40000 m

2 Barney <NA> y 130020 m

3 Marshall Anwalt y 92000 m

Merge

Zwei Datensätze über gemeinsame id zusammenführen.

himymPartner<-data.frame(Name=c("Barney","Marshall","Ted"),Partner=c("231","1","38"))

merge(himymBoys,himymPartner,by="Name") #merge über id "Name"

Name Beruf Verheiratet Einkommen Geschlecht Partner

1 Barney <NA> y 130020 m 231

2 Marshall Anwalt y 92000 m 1

3 Ted Architekt n 40000 m 38

Aggregate

Funktion Gruppenweise ausführen. Dabei ist zu beachten, dass der Gruppierungsvektor als Liste weitergegeben wird (deshalb by=list()).

Beispiel: Das Durchschnittseinkommen des himym-Datensatzes (s. 5.3.1) nach Geschlecht ermitteln:

aggregate(himym$Einkommen,by=list(Geschlecht=himym$Geschlecht),mean)

Geschlecht x

1 f 36950

2 m 87340

Bearbeitung von Strings

Ein String ist eine Zeichenkette beliebiger form. Sie kann Text aber auch Zahlen oder Sonderzeichen enthalten. Hier wird ein String als Zeichenkette die in Anführungszeichen eingeschlossen ist definiert. Es ist manchmal unumgänglich, solche Variablen auseinanderzuschneiden oder zu verarbeiten.

Länge eines Strings

Funktion nchar

nchar("Mein Name ist")

[1] 13

Zusammenfügen

Strings oder Variablen zusammenfügen mit paste:

Name<-"Bond"

paste("Mein Name ist",Name)

[1] "Mein Name ist Bond"

Ausschneiden

Teile von einem String herausschneiden mit substr:

IBTitel<-c("NNCHA42","XPSWEA21","NNTotA9798")

substr(IBTitel,1,2) #Stelle 1 bis 2 herausschneiden

[1] "NN" "XP" "NN"

Position von Zeichenketten bestimmen.

pos<-regexpr("A",IBTitel) #An welcher Stelle steht das A?

pos[1:3]

[1] 5 6 6

substr(IBTitel,pos,nchar(IBTitel)) #Branchen herausschneiden

[1] "A42" "A21" "A9798"

substr(IBTitel,3,pos-1) #Länder herausschneiden

[1] "CH" "SWE" "Tot"

Programmiersprache R

If-Else Konstruktion

Bedingte Ausführung von Programmcode:

if (himymBoys[1,4]>himymBoys[2,4]) {

print("Ted verdient mehr als Barney")

}else {print("Ted verdient nicht mehr als Barney")

}

[1] "Ted verdient nicht mehr als Barney"

Schlaufen

For Schlaufe:

Wiederholtes Ausführen von Programmcode für jedes Element i einer Menge M:

for (i in M) { Programmcode }

count<-0

for (i in 1:5){

count<-count+i

}

count

[1] 15

While Schlaufe:

Wiederholtes Ausführen von Programmcode solange ein Bedingung erfüllt ist:

while (Bedingung) { Programmcode }

i<-1

count<-0

while(i<=5){

count<-count+i

i<-i+1

}

count

[1] 15

Repeat-Schlaufe

Wiederholtes Ausführen von Programmcode bis Abbruchbedingung erfüllt ist:

repeat { Programmcode if (condition) { break } }

i<-1

count<-0

repeat {

count<-count+i

i<-i+1

if(i>5){break}

}

count

[1] 15

Eigene Funktionen

Genau wie eine Variable kann man auch eine eigene Funktion speichern. Um eine eigene Funktion zu schreiben: function(Argumente) { Funktion }

TestFkt<-function(x){

add<-0

for(i in 1:x){

add<-add+i

}

return(add)

}

TestFkt(5)

[1] 15

Literatur

Torfs P., Brauer C.: “A (very) short introduction to R”, 2014.

<http://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf>

Venables W. N., Smith D. M., R Core Team: “An introduction to R”, 2014

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

Spector P.: ”Data Manipulation with R”, Springer, 2008.

Funktionsverzeichnis

Diese Liste ist weit entfernt von vollständig. Sie soll den Einstieg erleichtern und eine Hilfestellung für die ersten Übungen bieten. Die kommenden Übungen sollten anhand dieser Funktionsliste zu bewältigen sein.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Funktion** | **Beschreibung** | **Beispiel** | | --- | --- | --- | | %\*% | Matrixmultiplikation | matrix1%\*%matrix2 | | aggregate | Funktion gruppiert über einen Vektor ausführen | aggregate (DF,by=list(DF$gruppe),sum) | | as.matrix  as.character  as.factor  as.numeric  as.integer  as.data.matrix  as.data.frame | Klasse eines Objektes ändern | as.character(vektor) | | c | Zusammenfassen von Einträgen zu Vektor | c(1,2,3,4,5) | | cbind | Spalten anhängen | cbind(objekt1,objekt2) | | choose.files | Öffnet ein Auswahlfenster. Die Ausgabe ist der ausgewählte Pfad | choose.files() | | class | Klasse eines Objektes ermitteln | class(objekt) | | colnames | Name der Spalten eines Data Frame/Matrix | colnames(DF) | | colSums | Summe der Spalten eines Data Frame/Matrix | colSums(DF) | | cumsum | Kumulierte Summe eines Vektors | cumsum(vektor) | | head | Die ersten Sechs Zeilen eines Objektes | head(objekt) | | identical | Test, ob zwei Objekte identisch sind | identical(objekt1,objekt2) | | install.package | Zusätzliches Paket installieren | install.package("paketname") | | is.element | Test, ob a Teil einer Menge b | is.element(a,b) | | length | Länge eines Vektors | length(vektor) | | library | Laden eines zusätzlichen Paketes | library("paketname") | | load | Laden eines RData/rdata Files | load("file") | | matrix | Matrix mit a=Inhalt, b=Reihen und c=Spalten | matrix(a,b,c) | | max | Maximum eines Vektors | max(Vektor) | | mean | Durchschnitt eines Vektors | mean(vektor) | | median | Median eines Vektors | median(vektor) | | **Funktion** | **Beschreibung** | **Beispiel** | | merge | Zwei Data Frames über gemeinsame id zusammenführen | merge(objekt1,objekt2,by="id") | | min | Minimum | min(Vektor) | | ncol | Anzahl Spalten eines Data Frame/Matrix | ncol(DF) | | nrow | Anzahl Zeilen eines Data Frame/Matrix | nrow(DF) | | order | Data Frame nach n-ter Spalte ordnen | DF[order(DF[,n]),] | | paste | Verbindet Strings/Variablen miteinander | paste(string1,string2,string3) | | plot | plot von Punkten nach x und y Koordinaten | plot(x,y) | | print | Ausgabe von Text | print("text") | | rbind | Zeilen anhängen | rbind(objekt1,objekt2) | | read.csv | Einlesen eines .csv Files | read.csv("Tab") | | read.table | Einlesen einer Tabelle | read.table("Tab",sep=";") | | regexpr | Überprüft, wo Zeichenkette in String vorkommt | regexpr  ("Zeichenkette","string") | | rep | n-fache Wiederholung eines Vektor | rep(vektor,n) | | rm | Entfernen von Objekten aus dem Workspace | rm(objekt) #nur Objekt  rm(list=ls()) #alle | | rownames | Name der Zeilen eines Data Frame/Matrix | rownames(DF) | | rowSums | Summe der Zeilen eines Data Frame | rowSums(DF) | | save | Speichern von Objekten als RData/rdata File | save(objekt,file="filename") | | sd | Standardabweichung eines Vektors | sd(vektor) | | seq | Zahlenreihe von a bis b, in c-Schritten | seq(a,b,c) | | setwd | Ort des Workspace definieren | setwd("workspacePfad") | | solve | Inverse einer Matrix | solve(matrix) | | str | Übersicht von Objekt (Klasse) | str(objekt) | | subset | Teil eines Objektes nach Bedingung auswählen | subset(objekt,bedingung) | | substr | String von der der a-ten Stelle bis zur b-ten Stelle herausschneiden | substr("string",a,b) | | sum | Summe eines Vektors | sum(vektor) | | summary | Übersicht von Objekt (Deskriptive Stat.) | summary(objekt) | | t | Transponiert Matrix | t(matrix) | | tail | Die letzten sechs Zeilen eines Objektes | tail(objekt) | | unique | Gibt alle unterschiedlichen Merkmale eines Vektors an. | unique(vektor) | | which | An welcher Stelle eines Vektors trifft die Kondition zu | which(vektor==1) | | write.table | Tabelle Tab unter Save Speichern | write.table(Tab,"Save") | |
| |  | | --- | |  | |