Statistische Datenanalyse mit R
Teil 1 online

Dr. Andrea Denecke
Leibniz Universität IT-Services

1

Vorteile

- kostenlos
- Open-source Software: Transparenz des „source-code“
- Flexibilität: eigene Funktionen können leicht geschrieben werden
- Große Anzahl an sog. Zusatzpaketen
- „support“ durch die R-community

Nachteile

- Schwierig zu Erlernen durch die fehlende graphische Oberfläche
- Die Ergebnisse der Berechnungen erscheinen unformatiert im Ausgabefenster
- Es ist schwierig, Fehler in den Kommandos zu finden

Installation von R

Die R-Datei (aktuell: R Version 4.2.2) ist verfügbar unter
http://www.r-project.org
Hier findet man auch weitere Informationen zu R, z.B. FAQ‘s, Handbücher etc.

CRAN: **C**omprehensive **R** **A**rchive **N**etwork
Auflistung der URL‘s, durch die R-Software zur Verfügung gestellt wird

Nützlich: R reference card

Anmerkung: Die im Folgenden vorgestellten Funktionen beziehen sich auf eine Installation auf Windows, einige sind bei R auf Linux oder Mac OS X nicht verfügbar

Besonderheiten von R

- Mit dem Öffnen von R öffnet sich die Konsole, eingegebene Befehle werden durch Drücken von ENTER ausgeführt
- R enthält die gängigen mathematischen Funktionen und kann als „Taschen“-Rechner benutzt werden (Funktionen s. übernächste Seite)
- Um Eingaben zu kommentieren, benutze „#“
- R ist „case sensitive“, d.h. Daten ≠ daten
- R ist objektorientiert
- Das Dezimaltrennzeichen ist immer ein Punkt

Anmerkung: im Folgenden werden Befehle über das Menü in **Arial fett** geschrieben, die Befehle in der Konsole/ Skriptfenster in `Courier New`

R-Struktur

Konsole → dokumentiert alle Ein- & Ausgaben sowie ablaufende Prozesse

> (sog. prompt oder Eingabeaufforderung, per Voreinstellung in Rot)

[1] Ausgabe, per Voreinstellung in Blau, die Zahl in den eckigen Klammern ist immer die Angabe der „Fallnummer“

mit dem Cursor (oben/ unten) können die vorherigen Befehle wiederholt und auch verändert werden

Workspace: sichert Daten

History sichert die Kommandos der aktuellen Sitzung

Skriptfenster ersetzt die Konsole als die Eingabe-Oberfläche → benutzer-optimiert, vor allem nützlich bei wiederkehrenden, gleichartigen Analysen

5 Einige mathematische Funktionen in R

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
^	Potenzierung
log(x)	natürlicher Logarithmus der Zahl x
log10(x)	Logarithmus der Zahl x zur Basis 10
log(x, Basis)	Logarithmus der Zahl x zu anzugebender Basis
sqrt(x)	Quadratwurzel der Zahl x
exp(x)	Exponentialfunktion der Zahl x
factorial(x)	x! (Fakultät von x)
choose(n,k)	Binomialkoeffizient (nk)
sin(x), cos(x), tan(x)	Sinus-, Cosinus-, Tangensfunktion
asin(x), acos(x), atan(x)	Arcsinus-, Arccosinus-, Arctangensfunktion

Erzeugen von Vektoren

Um Vektoren mit mehr als einer Zahl zu erzeugen, verwendet man die Funktion `c()` (concatenate oder combine)

Beispiel:

```
 > c(0,1,2,3,4)*5  #Eingabe, nach Drücken von ENTER erscheint
[1]  0  5 10 15 20  #Ergebnis
```

das gleiche Ergebnis erzielt man in diesem Fall mit

```
> c(0:4)*5          #fortlaufende Zahlen von 0 bis 4 verwenden
[1]  0  5 10 15 20  #gleiches Ergebnis
```

Oder noch einfacher

```
> 0:4*5             #fortlaufende Zahlen von 0 bis 4 verwenden
[1]  0  5 10 15 20
```

Merke: um Vektoren mit fortlaufende Zahlen zu erzeugen verwendet man „:“, ansonsten c(). Zeichenfolge-Vektoren werden in „“ gesetzt.

Zuweisungen (1)

Die berechneten Werte werden in der Konsole angezeigt. Um sie als Objekt für weitere Berechnungen zur Verfügung zu haben, muss man die Ergebnisse zuweisen.
Beispiel:

```
> a <- c(0,1,2,3,4)*5
```
#die Eingabe wird als Objekt „a“ gespeichert

Merke: Innerhalb des Zuweisungspfeils dürfen keine Leerzeichen stehen!

```
> a
```
#Aufrufen oder Ansehen des Objekts

```
[1] 0  5 10 15 20
```

Das Objekt kann dann in weiteren Berechnungen verwendet werden.

```
> log10(a)
```
#Logarithmus aller Zahlen des Objekts a zur Basis 10

```
[1] -Inf 0.698970 1.000000 1.176091 1.301030
```

Zuweisungen (2)

Man möchte den dritten Wert des Objekts a wissen

```
> a[3]
[1] 10
```

Es können auch höhere Strukturen als Vektoren erzeugt werden, z.B. Matrizen über die Funktion `matrix()` mit den Argumenten „Vektor“ und „Anzahl der Reihen bzw. Spalten“ (nrow oder ncol) . Beispiel:

```
> b <- matrix(1:12, nrow=3)
> b
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

Zuweisungen (3)

Man möchte den Wert der dritten Zeile und der zweiten Spalte wissen

```
> b[3,2]
[1] 6
```

Oder alle Werte der ersten Zeile

```
> b[1, ]
[1]  1  4  7 10
```

Eine Matrix entspricht quasi den gängigen Datensätzen, bestehend aus Variablen (Spalten) und Fällen (Zeilen). Es gibt auch noch höher-dimensionale Strukturen, die Arrays.

Funktionen

Alle Prozesse in R laufen über Funktionen. Der Aufbau ist immer

```
funktionsname(argument1, argument2,…)
```

wobei je nach Funktion manchmal kein Argument benötigt wird, nach oben sind oft keine Grenzen gesetzt.

Beispiele:

```
log10(2)          #die Funktion ist hier der Zehnerlogarithmus, das einzige Argument die Zahl 2

mean(x, na.rm=FALSE)  #diese Funktion benötigt als Argument „x“, das zweite Argument ist optional und besagt, dass fehlende Werte nicht gelöscht werden.
```

Woher kann man den Aufbau der Funktionen wissen?

R-reference card, online Hilfe,…

Hilfe!

- Die integrierte Hilfefunktion `?help`
- Hilfe, wenn Funktionsname bekannt ist: `?mean` oder `help(mean)` oder `help(„mean“)`
- `example(mean)` gibt ein Beispiel für diese Funktion an
- Wenn man den korrekten Namen der Funktion nicht weiß: `help.search(„mean“)` → Liste aller Funktionen, die die Zeichenfolge „mean“ enthalten
- `apropos(„mean“)` zeigt ebenso die Funktionen mit der Zeichenfolge „mean“ in der Konsole an
- Auf www.r-project.org gibt es Handbücher, FAQ‘s, WIKI (letzte Möglichkeit: mailing list R-help)
- Über Suchmaschinen
- QuickR: www.statmethods.net
- R Documentation: www.rdocumentation.org

Zusammenfassung Zeichen

- **Eckige Klammern** [] ermöglichen Zugriff auf Teile eines Objekts (Vektor, Matrix,…), erste Zahl Zeilennummer, zweite Zahl Spaltennummer, getrennt durch eine Komma
- **Runde Klammern** () folgen immer auf Funktionen
- Argumente bei Funktionen werden innerhalb der runden Klammern geschrieben und durch **Kommata getrennt**
- Pro Zeile ein Befehl, stehen mehrere Befehle in einer Zeile, trennt man diese durch ein **Semikolon ;**
- Zeichenfolgen (character objects, Namen) werden (fast) immer in doppelten **Anführungsstrichen** „“ geschrieben
- Neuen Objekten in R wird über den **Zuweisungspfeil** <- ein Objektname zugewiesen
- Fortlaufende Reihen ganzer Zahlen können über den **Doppelpunkt** (Anfangszahl : Endzahl) definiert werden

Übung 0.1

Führen Sie die folgenden Berechnungen durch:
1 plus 3 plus 5 plus 7
1 geteilt durch 30
Logarithmus der Zahl 2 zur Basis 10
Fakultät von 3
Verdopplung der Zahlen 1 bis 10
Verdopplung der geraden Zahlen von 1 bis 10

Anmerkung: Leerzeichen innerhalb der Befehle werden toleriert!
Tipp: vorher eingegebene Befehle können über die Cursor-Tasten wiederholt und verändert werden.

Übung 0.2

Wir hatten uns eine Matrix erzeugt über

```
> b <- matrix(1:12, nrow=3)
> b
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

Erreichen Sie, dass die Zahlen von 1 bis 12 nicht spaltenweise, sondern reihenweise in die Matrix gelegt werden!
Tipp: Schauen Sie, ob es unter `?matrix` einen geeigneten Hinweis gibt!
Verändern Sie die Zahl 8 in den Wert 11.

Der Workspace

- Speichert alle zugewiesenen Objekte / Daten einer Sitzung
  Anzeigen des Inhalts des Workspace: `ls()`
  (oder Menü: **Verschiedenes → Liste Objekte auf**)
  Entfernen eines Objekts aus dem Workspace: `rm(„Name“)`
  (oder Menü: **Verschiedenes → Entferne alle Objekte** (!))
- Speichern des Workspace: `save.image(„Pfad“)`
  (oder **Datei → Sichere Workspace)**.
  Beim Verlassen von R wird automatisch abgefragt, ob man den Workspace speichern möchte.
- Einen vorher gesicherten Workspace (Endung .Rdata) hochladen: `load(„Pfad“)` oder **Datei → Lade Workspace**

History

- Eine History-Datei enthält eine komplette Auflistung aller Befehle der letzten Sitzung → dient vor allem der Dokumentation
- Wenn der Workspace gespeichert wird, werden die Kommandos der Sitzung automatisch in der Datei .Rhistory gespeichert und können später über `history()` abgerufen werden
- Kann auch über das Menü gespeichert werden **Datei→ Speichere History**
- Kann auch im Editor geöffnet werden und dann als Textdatei (.txt) gespeichert werden

Das Skript Fenster

- Um ein neues oder bereits existierendes Skript-Fenster zu öffnen: **Datei → Neues Skript** beziehungsweise **→Öffne Skript**
- Befehle können über STRG-R an die Konsole übergeben werden (Beachten: geht der Befehl über eine Zeile, reicht es den Cursor in die entsprechende Zeile zu setzen und STRG-R zu drücken. Geht der Befehl über mehrere Zeilen, muss er komplett markiert werden)
- Längere, komplizierte Befehle oder Befehlsfolgen können immer wieder verwendet werden
- Dokumentation: verglichen mit der History-Datei, die alle Befehle speichert, können hier nur die benötigten gespeichert werden

Dateneingabe

- Über die Eingabe von `neue.daten <- data.frame()` erzeugt man einen neuen Datensatz, nach Eingabe von `fix(neue.daten)` (oder **Bearbeiten → Dateneditor**) kann man die Werte und Variablennamen händisch eingeben.
- Hauptweg wird der Import aus anderen Programmen sein (z.B. Excel)

Regeln für die Erzeugung von Datensätzen

- **Struktur**: Variablen in Spalten, Fälle/ Beobachtungseinheiten in Zeilen
- Keine **freien Zellen** zur besseren Erkennbarkeit
- **Variablennamen** in der ersten Reihe möglich (Sonderzeichen und Leerzeichen sind zu vermeiden, z.B. Frage_1 anstatt Frage 1)
- Keine Vermischung von Text und Zahl innerhalb der Werte einer Variable, jedes Objekt kann nur einen **Datentyp** haben
- Keine Angabe von **Einheiten** in den Zellen
- Keine **Kommentare** innerhalb des Datensatzes
- Zellen mit **fehlenden Werten** frei lassen oder „NA“ benutzen

Struktur der Datensätze

falsch

	Männer		Frauen	
	Größe	Gewicht	Größe	Gewicht
1	180	78	161	53
2	166	86	161	58
3	186	80	157	59
4	191	88	170	75
5	179	85	166	57
6	188	95	168	k.A.
7	175	70	166	65
8	186	77	175	61
9	180	86	168	62
10	190	90	170	61

richtig

Geschlecht	Größe	Gewicht
m	180	78
m	166	86
m	186	80
m	191	88
m	179	85
m	188	95
m	175	70
m	186	77
m	180	86
m	190	90
w	161	53
w	161	58
w	157	59
w	170	75
w	166	57
w	168	
w	166	65
w	175	61
w	168	62
w	170	61

Öffnen von Dateien

- Anleitungen gibt es unter `help(read)` oder im Menü **Help→manuals (pdf) → R-data import**
- Für .txt-Dateien: `read.table()`
- `spss.get()` (Paket {Hmisc})
- Pakete, um Excel-Dateien direkt zu öffnen: {xls}, {xlsx}, {RODBC}, {openxlsx} oder {xlsReadWrite}
- empfohlen: Excel-Dateien als .csv speichern, dann `read.csv2()` (wenn Deutsch als Systemsprache eingestellt ist) benutzen. (Englisch: `read.csv()`)
- Mit `file.choose()` kann man die einzulesende Datei per Klick aussuchen, ansonsten muss man den kompletten Pfad eintippen

Vorsicht: das Öffnen von Dateien anderer Formate in R ist fehleranfällig und sollte nach Möglichkeit nicht unnötig oft durchgeführt werden!

Import von Excel-Dateien

Die Daten der Datei „test“ sollen in R übertragen werden.

1. sind die Regeln für den Datensatz erfüllt? (s. 3 Seiten vorher)
2. In Excel **Datei → Speichern unter** → Dateityp wählen: „CSV Trennzeichen getrennt (*.csv)“, Speichern
3. In R in die Konsole/ Skript eingeben:

```
test <- read.csv2(file.choose())
```

Es öffnet sich der Explorer, die soeben abgespeicherte .csv-Datei heraus suchen, anklicken, Öffnen

Man hat so die eingelesenen Daten im Workspace als Datensatz „test“ zur Verfügung

Dateninspektion/ Deskriptive Statistiken

- `test` zeigt den kompletten Datensatz an
- Es existieren eine Reihe von Befehlen/Funktionen, um einen Überblick über die Daten zu bekommen:
  - `summary(test)`: gibt einen allgemeinen Überblick
  - `str(test)`: zeigt die Struktur des Datensatzes
  - `names(test)`: Zeigt die Variablennamen an
  - `describe(test)`: ähnlich wie `summary`, aber etwas ausführlicher und formatiert (Paket {Hmisc})

Installation von Paketen

- Durch die Installation von R hat man Zugriff auf die Basispakete. Um festzustellen, welche Pakete installiert sind: `library()`
- Ein neues Paket installieren: `install.packages(„Name“)` oder **Pakete → Installiere Paket(e)**
  um es zu entfernen: `remove.packages(„Name“)`
- Bevor man die installierten Pakete verwenden kann, muss man sie hochladen: `library(„Name“)` oder **Pakete → Lade Paket**
- `library(help=„Name“)` liefert Informationen über die im Paket enthaltenen Funktionen

Datenmanagement

Um nachträglich Daten in einem existierenden Datensatz hinzu zu fügen: Öffnen des Dateneditors mit `fix(test)` oder über das Menü mit **Edit →Data Editor**

Beispiel:
In dem Datensatz „test“ sollen die Variablennamen geändert werden. Die Eingabe von `fix(test)` öffnet den Dateneditor, Überschreiben von var1=nr, var2=geschl, var3=groes, var4=gew, var5=alter.

Vorsicht! Einmal geänderte Daten können nicht wieder zurück geholt werden. Die Änderungen werden mit dem Schließen des Editors im Datensatz aktiv.

Datenauswahl

- Auswahl einzelner Fälle: `test[Fallnr,Variablennr]`
- Auswahl einer Variable : `test[,„Variablenname“]`
- Auswahl von >1 Variable/Fall: benutze Funktion `c()`

Beispiele:

- Von dem Datensatz „test“ sollen die Werte der ersten und dritten Variable angegeben werden: `test[,c(1,3)]`
- Es sollen die Werte der Fälle 1 bis 5 angegeben werden: `test[1:5,]`
- Als Kombination: es sollen die ersten 10 Beobachtungen der Variablen 2 und 4 angegeben werden: `test[1:10,c(2,4)]`

Beachten: bei der Auswahl mit eckigen Klammern wird der Variablenname in Anführungszeichen geschrieben!

Auswahl von Daten

Mit der Funktion `subset()` können Daten nach Falleigenschaften ausgesucht werden, z.B. kann man sich aus einem Datensatz nur die männlichen Versuchsteilnehmer heraus suchen.

Operatoren

== gleich	!= ungleich		
> größer	>= größer gleich		
< kleiner	<= kleiner gleich		
& und	| oder	`xor()`	entweder-oder

<u>Beispiele</u>:

Von dem test Datensatz sollen nur die männlichen Teilnehmer angezeigt werden `subset(test,geschl==1)`

Man möchte nur die mit einer Körpergröße über 180 cm haben und nur das Geschlecht und das Alter ausgeben lassen

```
subset(test,groes>180, select= c(geschl, alter))
```

Übung 1

- Erzeugen Sie in R einen Datensatz „bundesliga“ basierend auf der CSV-Datei „bundesliga.csv“ (Folie 22)
- Verschaffen Sie sich einen Überblick über die Daten: welche Variablen gibt es?, wie viele Fälle?, etc. (Folie 23)
- Lassen Sie sich die Werte der Variable „wochtag“ anzeigen (Folie 26)
- Lassen Sie sich die ersten 10 Werte der Heimtore und Gasttore anzeigen (Folie 26)
- Lassen Sie sich die Spiele anzeigen, in denen die Gäste mindestens zwei Tore geschossen haben (Folie 27)