LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Statistische Software

Micha Schneider

Institut für Statistik
Ludwig-Maximilians-Universität München

WS 2016/17, R Teil 1

Einführung

Was ist R?

R ist eine kostenlose Software-Umgebung für statistische Datenanalyse und Graphiken. Es beruht auf einer Implementation der Sprache S. Anfänglich wurde R von Ross Ihaka und Robert Gentleman (Univ. Auckland) entwickelt und wird seit Mitte der 90er Jahre von einem Entwickler-Kollektiv (R-Core) betreut.

Informationen zu R:

`http://www.R-project.org`

Wie installiere ich R?

Fertige R Distributionen sind für Windows, MacOS X und viele Linux-Versionen auf CRAN (**C**omprehensive **R** **A**rchive **N**etwork) erältlich:

- Die Website `http://www.R-project.org` aufrufen und auf „download R“ klicken
- Einen nahegelegenen Server auswählen
- Dem Link mit dem benötigtem Betriebsystem folgen
- Bei Windows: „base“ auswählen und R installieren

R als Taschenrechner

Ok. Aller Anfang ist schwer, also zunächst mal was Bekanntes...

```
> 1 + 1
[1] 2
> 1 + 2 * 3
[1] 7
> (1 + 2) * 3
[1] 9
> 2^3
[1] 8
> 4^0.5
[1] 2
> 200/10
[1] 20
```

Die Grundrechnungsarten folgen in der Abarbeitungsreihenfolge den üblichen Regeln („Punkt vor Strich“). Im Zweifelsfall immer besser ein Klammernpaar zu viel als eins zu wenig verwenden.

Hinweis: In R wird immer der Punkt als Dezimaltrennzeichen verwendet.

Arbeiten mit R

- Kern von R ist das Kommandofenster (Console), in dem alle Befehle ausgeführt werden. Die Eingabeaufforderung wird mit “>“ angezeigt.
- Die Befehle ansich sollten mit einem Editor in eine reine Textdatei geschrieben werden. Die Befehle werden anschließend in der Console ausgeführt.
- Graphiken werden in einem seperaten Fenster angezeigt.

Editoren für R

Für das Erstellen längerer Funktionen und Code-Blöcken ist die R Konsole ungeeignet. Mögliche Editoren sind beispielsweise:

R-Editor: Interner Skript-Editor ohne Zusatzfunktionen wie Syntax-Highlihgting

RStudio: Der Editor funktioniert unter Windows, Mac und Linux und bietet ebenso Syntax-Highlighting und eine Anbindung an R (empfehlenswert):
`http://www.rstudio.com/`

RStudio

Die Oberfläche des RStudios teilt sich in 4 Bereiche:

Skriptfenster (links oben): R-Code, der über Symbole in der Kopfleiste oder mittels Tastenkombinationen ausgeführt werden kann.

Console (links unten): Entspricht der herkömmlichen R-Console. Neben Ausgaben finden sich hier auch Warnungen (Bearbeitung läuft weiter) und Fehlermeldungen (Abbruch der Bearbeitung).

Workspace/History (rechts oben): Anzeige der existierenden Objekte und der zuletzt ausgeführten Befehle.

Files/Plots/Packages/Help: Ordnerstruktur (ausgehend vom Arbeitsverzeichnis), bisher erzeugte Graphiken, Installieren und Updaten von Packages, Zugriff auf die Hilfe

RStudio

Anlegen einer neuen Script-Datei:

```
File -> New -> R Script
```

Öffnen von bereits vorhandenen Script-Dateien:

```
File -> Open File ...
```

Änderungen können über das Menü oder über das Diskettensymbol in der Kopfleiste gespeichert werden. Senden von Befehlen an die Console funktioniert mit Strg+Enter oder dem entsprechendem Symbol 

.

Zuweisungen

Eine Zuweisung erfolgt mit „<-“ (auch „=“ möglich):

```
> a <- 5
> a
[1] 5
> b <- 10
```

Man sagt auch „dem Objekt „a“ wurde der Wert „5“ zugewiesen“. Objekte können leicht kombiniert werden:

```
> a*b
[1] 50
```

R überschreibt Objekte ohne Vorwarnung:

```
> a <- 1
> a
[1] 1
```

Alle Objekte, die wir während einer Sitzung direkt in der R-Console erzeugen, werden im sogenannten „Workspace“ gespeichert (in RStudio rechts oben zu sehen).

Operatoren

Vergleich von Objekten mit Hilfe logischer Operatoren:

```
> 2 > 3
[1] FALSE
> 3 > 2
[1] TRUE
> a == b
[1] FALSE
> a != b
[1] TRUE
```

+ - * /	Grundrechenarten
^	Potenzieren
<-	Zuweisungen
== != < > <= >=	logische Vergleiche
#	Kommentarzeichen

Objekte

- Jede Auswertung kann in einem Objekt abgespeichert werden.
- Ein Objekt in R kann (fast) alles sein: Variablen, Matrizen, Funktionen,...
- Als Objektnamen können fast beliebige Kombinationen aus Buchstaben, Ziffern, Punkt und Unterstrich verwendet werden.
- Objektnamen sollten mit einem Buchstaben beginnen.
- R unterscheidet zwischen Groß- und Kleinschreibung (d.h. case-sensitve).
- Jedes Objekt hat bestimmte Eigenschaften wie z. B. Datentyp.

Datentypen und Datenstrukturen

In R gibt es eine Vielzahl an Datentypen und Datenstrukturen (detaillierter nächstes Mal). Übliche Datentypen sind:

- **numeric**: ganzzahlige oder Gleitkomma-Werte (z.B. 1),
- **character**: beliebige Zeichen (z.B. "Hallo"),
- **logical**: die Zustände `TRUE` und `FALSE`,

Unter Datenstrukturen versteht man eine Beschreibung dessen, wie die Daten dargestellt werden und angeordnet sind: z.B. Vektor, Matrix, ... Ein numerischer Vektor lässt sich einfach erstellen, in dem man Zahlen mit `c()` verbindet:

```
> vec <- c(1,2,3)
> vec
[1] 1 2 3
```


Funktionen

R ist eine vollwertige Programmiersprache, und der größte Teil von R ist selber in der Sprache R geschrieben. Auch der Benutzer kann zu jeder Zeit neue Funktionen definieren (keine Angst, nicht Teil dieser LVA).

Funktionsaufrufe sind von der Form `funktionsname(arg1=wert1, arg2=wert2)` mit beliebig vielen Argumenten.

Die Namen der Argumente können auch weggelassen werden, jedoch muss die Reihenfolge dann exakt mit der Funktionsdefinition übereinstimmen: `funktionsname(wert1, wert2)`. Aufgrund der Fehleranfälligkeit ist dies in der Regel nicht zu empfehlen.

Achtung: Funktionen werden an den auf den Namen folgenden runden Klammern erkannt. Die Zuweisungen der Form `arg=wert` erfolgt immer mit dem Operator `=`, nie mit `<-` .

Funktionen

Beispiele:

```
> a <- 2
> b <- 2^a
> b
[1] 4
```

```
> log(b)
[1] 1.386
> log(b, 2)
[1] 2
> log(base=2, x=b)
[1] 2
> log(2, b)
[1] 0.5
```


Hilfe

Zu einer Funktion (z. B. `log()`) kann die entsprechende Hilfeseite mit

```
> ?log
```

oder

```
> help(log)
```

aufgerufen werden. In RStudio kann die Hilfe von einem markierten Befehl auch mit der Taste F1 aufgerufen werden.

Ist der Name der benötigten Funktion unbekannt, kann vorher mit `apropos()` danach gesucht werden. Eine gute Übersicht über R interne Handbücher findet man mit `help.start()`.

Pakete

Eine Vielzahl von Funktionalitäten sind nicht im Basissystem, sondern in Zusatzpaketen implementiert. Diese werden zentral auf http://CRAN.R-project.org gehalten und können direkt von R aus installiert werden.

```
> install.packages("colorspace")
```

installiert ein Paket mit Funktionen für die Modifizierung von Farben in Grafiken. Um das Paket benützen zu können, muss es (in jeder neuen Sitzung) mit

```
> library(colorspace)
```

geladen werden. Die Hilfe zu einem Paket kann mit `help(package=colorspace)` aufgerufen werden.

Datenanalyse Teil I

Münchner Mietspiegel 2003

Zahlreiche deutsche Städte erstellen sogenannte Mietspiegel, um Mietern, Vermietern, Mietberatungsstellen und Sachverständigen eine möglichst objektive Entscheidungshilfe in Mietfragen zur Verfügung zu stellen. Die Mietspiegel werden dabei insbesondere zur Ermittlung der „ortsüblichen Vergleichsmiete" (Nettomiete in Abhängigkeit von Wohnungsgröße, -ausstattung, -alter, etc.) herangezogen. Bei der Erstellung von Mietspiegeln wird aus der Gesamtheit aller in Frage kommenden Wohnungen eine repräsentative Zufallsstichprobe gezogen (im Fall der Stadt München durch Infratest), und die interessierenden Daten werden von Interviewern anhand von Fragebögen ermittelt. Der vorliegende Datensatz stellt einen Ausschnitt aus dem Mietspiegel München des Jahres 2003 dar und enthält die Daten von 2053 Wohnungen.

http://www.stat.uni-muenchen.de/service/datenarchiv/miete/miete03.html ⇒ `miete03.asc`

Arbeitsverzeichnis

Falls nichts anderes angegeben wird, nimmt R an, dass sich zu ladende Datensätze in diesem Verzeichnis befinden und speichert Graphiken in dieses Verzeichnis ab. Deshalb sollte man dies immer am Anfang überprüfen! Mit der Funktion `getwd()` kann das aktuelle Arbeitsverzeichnis (Working Directory) abgefragt und mit `setwd()` geändert werden.

```
> setwd("Z:/REinfuehrung")
```

oder

```
> setwd("Z:\\REinfuehrung")
```

Import von Daten

Am einfachsten können Textdateien in R eingelesen werden. Folgendes ist zu beachten:

- Struktur: Eine Zeile pro Beobachtung, Merkmale durch eindeutiges Trennzeichen separieren, z. B. Tabulator oder Semikolon.
- Normalerweise eignet sich das Format „Comma Separated Values“ (CSV) am Besten. Auch Excel-Tabellen lassen sich in diesem Format abspeichern.
- Achtung: Je nach Ursprung der Daten kann Punkt oder Komma das Dezimaltrennzeichen sein. Beim Lesen/Schreiben von Daten kann das Dezimaltrennzeichen entsprechend bestimmt werden. In der R-Console wird immer der Punkt verwendet.
- Kontrolle des Formats in einem Editor vor dem Import kann helfen, Fehler zu vermeiden.

Trennzeichen in Zeichenkette

Kommt das Trennzeichen in einer Zeichenkette vor, so muss diese durch einfaches oder doppeltes Hochkomma geschützt werden.

Leerzeichen als Trennzeichen:

```
Name Geburtsjahr Geschlecht
Sting 1951 m
"John Lennon" 1940 m
```

Semikolon: Hochkomma hier nicht notwendig (schadet aber auch nicht):

```
Name;Geburtsjahr;Geschlecht
Sting;1951;m
John Lennon;1940;m
```

Soll auch in der Zeichenkette ein Hochkomma vorkommen, dann am einfachsten jeweils das andere (einfach, doppelt) zum Schützen nehmen. Beides geht auch (durch Backslash schützen).

Einlesen nach R

```
> #setwd("Z:\\REinfuehrung")
> miete <- read.table("miete03.asc", header=TRUE)
```

Die wichtigsten Argumente:

- **file:** Name der einzulesenden Datei (gegebenenfalls mit Verzeichnis)
- **header:** enthält die erste Zeile die Namen der Variablen?
- **na.strings:** Kodierung der fehlende Werte (z. B. “NA“)
- **sep:** Trennzeichen (z. B. “ “ oder “;“ oder “,“ oder “\t“)
- **dec:** Dezimaltrennzeichen (z. B. “.“ oder “,“)
- **skip:** Anzahl der Zeilen, die übersprungen werden bevor die Daten eingelesen werden

Alternativ empfiehlt sich häufig die Verwendung von `read.csv()` oder `read.csv2()`.

Einlesen nach R

Danach unbedingt prüfen, ob der Import erfolgreich war:

```
> names(miete)
 [1] "nm"       "nmqm"     "wfl"      "rooms"    "bj"       "bez"      "wohngut"
 [8] "wohnbest" "ww0"      "zh0"      "badkach0" "badextra" "kueche"
> head(miete)
     nm  nmqm wfl rooms   bj bez wohngut wohnbest ww0 zh0 badkach0 badextra kueche
1 741.4 10.90  68     2 1918   2       1        0   0   0        0        0      0
2 715.8 11.01  65     2 1995   2       1        0   0   0        0        0      0
3 528.2  8.38  63     3 1918   2       1        0   0   0        0        0      0
4 554.0  8.52  65     3 1983  16       0        0   0   0        0        1      0
5 698.2  6.98 100     4 1995  16       1        0   0   0        0        1      1
6 935.6 11.55  81     4 1980  16       0        0   0   0        0        0      0
```

Sind die Variablennamen korrekt erkannt worden?

Mittels `summary()` erhählt man eine schnelle Übersicht über die einzelnen Variablen:

Einlesen nach R

```
> summary(miete)
       nm              nmqm             wfl            rooms           bj
 Min.   :  77.3   Min.   : 1.47   Min.   : 17.0   Min.   :1.0   Min.   :1918
 1st Qu.: 389.9   1st Qu.: 6.80   1st Qu.: 53.0   1st Qu.:2.0   1st Qu.:1948
 Median : 534.3   Median : 8.47   Median : 67.0   Median :3.0   Median :1960
 Mean   : 570.1   Mean   : 8.39   Mean   : 69.6   Mean   :2.6   Mean   :1958
 3rd Qu.: 700.5   3rd Qu.:10.09   3rd Qu.: 83.0   3rd Qu.:3.0   3rd Qu.:1973
 Max.   :1789.5   Max.   :20.09   Max.   :185.0   Max.   :6.0   Max.   :2001
      bez           wohngut         wohnbest          ww0             zh0
 Min.   : 1.0   Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.: 5.0   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :10.0   Median :0.000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :11.3   Mean   :0.391   Mean   :0.0219   Mean   :0.0351   Mean   :0.0852
 3rd Qu.:17.0   3rd Qu.:1.000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :25.0   Max.   :1.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
    badkach0        badextra         kueche
 Min.   :0.000   Min.   :0.000   Min.   :0.0000
 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.0000
 Median :0.000   Median :0.000   Median :0.0000
 Mean   :0.185   Mean   :0.093   Mean   :0.0731
 3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.0000
 Max.   :1.000   Max.   :1.000   Max.   :1.0000
```


Univariate Deskription

Zugriff auf Variablen/Elemente

`miete` ist ein dataframe. Üblicher Weise gibt es für jede Beobachtung eine Zeile und die Variablen stehen in den Spalten. Auf benannte Spalten (also in der Regel Variablen) eines dataframes kann mit dem $-Operator zugegriffen werden:

```
> miete$nm
 [1] 741.4 715.8 528.2 554.0 698.2 935.6 204.8 426.9 446.3 381.4
[11] 337.3 756.7 945.9 264.9 540.6 757.7 538.7 796.1 461.2 752.3
[21] 446.6 548.2 421.0 759.6 390.0 436.9 174.8 354.9 491.6 695.4
[31] 540.2 325.1
 [ reached getOption("max.print") -- omitted 2021 entries ]
```

Man kann auch den Operator [] benutzen, um Zeilen oder Spalten (oder beides) aus dataframes zu extrahieren. Die ersten drei Beobachtungen der ersten Spalte (nm) erhält man mit

```
> miete[c(1,2,3),1]
[1] 741.4 715.8 528.2
> miete[c(1,2,3),"nm"]
[1] 741.4 715.8 528.2
```


Häufigkeiten

Verteilung der Anzahl der Zimmer:

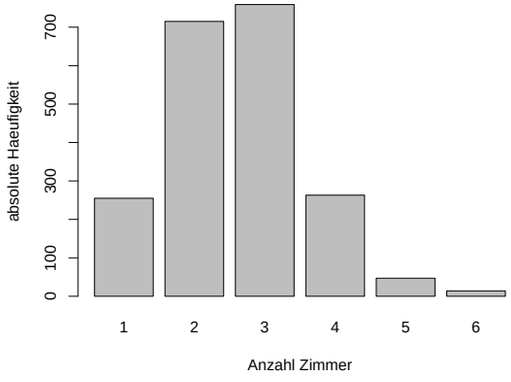
```
> TableZimmer <- table(miete$rooms)
> TableZimmer
  1   2   3   4   5   6
255 715 759 263  47  14
```

und ihre relativen Häufigkeiten:

```
> TableZimmer/sum(TableZimmer)
       1        2        3        4        5        6
0.124208 0.348271 0.369703 0.128105 0.022893 0.006819
> prop.table(TableZimmer) # Alternative
       1        2        3        4        5        6
0.124208 0.348271 0.369703 0.128105 0.022893 0.006819
```

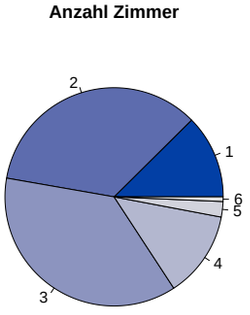
Balkendiagramm: Anzahl Zimmer

```
> TableZimmer <- table(miete$rooms)
> barplot(TableZimmer, xlab="Anzahl Zimmer", ylab="absolute Haeufigkeit")
```



Kreisdiagramm: Anzahl Zimmer

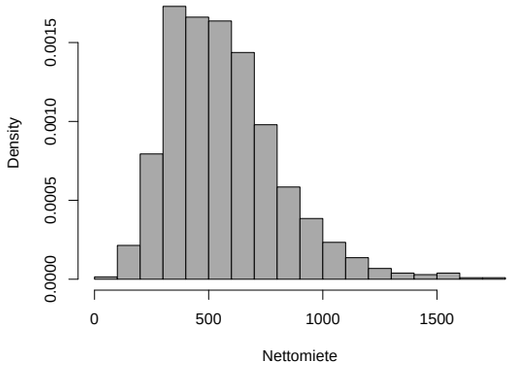
```
> library(colorspace)
> pie(TableZimmer, col=sequential_hcl(6), main="Anzahl Zimmer")
```



Histogramm: Nettomiete

Automatische Klassenanzahl:

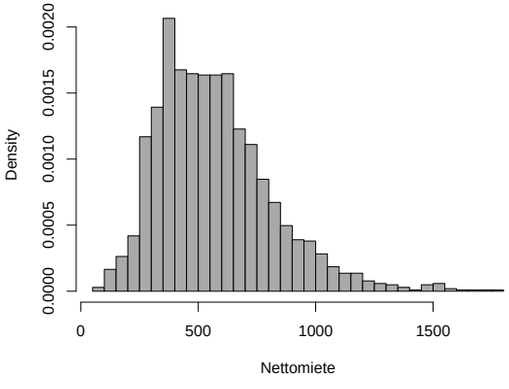
```
> hist(miete$nm, col="darkgray", freq=FALSE, main="", xlab="Nettomiete")
```



Histogramm: Nettomiete

Vorgegebene Klassenanzahl von 50 (wird nur ungefähr eingehalten):

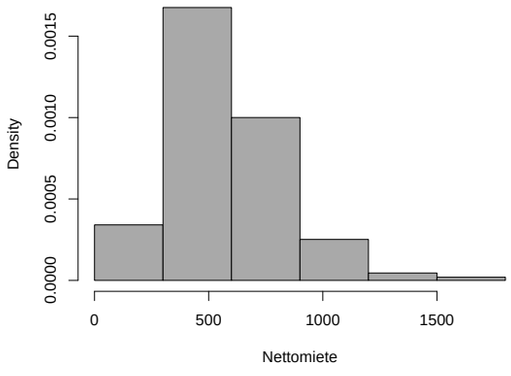
```
> hist(miete$nm, breaks=50, col="darkgray", freq=FALSE, main="", xlab="Nettomiete")
```



Histogramm: Nettomiete

Direkte Vorgabe der Klassen:

```
> hist(miete$nm, breaks=c(0, 300, 600, 900, 1200, 1500, 1800), col="darkgray",
+ freq=FALSE, main="", xlab="Nettomiete")
```



Speichern von Grafiken

Die in R erzeugten Grafiken können nicht nur über das Menü, sondern direkt durch eine Anweisung im Code im gewünschten Format abgespeichert werden, was gerade bei einer größeren Anzahl von Grafiken deutlich effizienter ist. Beispielsweise lässt sich durch

```
> pdf("Grafiken/Grafik1.pdf")
> barplot(TableZimmer, xlab="Anzahl Zimmer", ylab="abs. Haeufigkeit")
> dev.off() # Erst hier wird die Grafik gespeichert!
```

der erzeugte Boxplot als .pdf-Datei namens „Grafik1.pdf“ im Unterordner „Grafiken“ abspeichern. Weitere Funktionen neben `pdf()` sind beispielsweise `postscript()`, `win.metafile()`, `jpeg()`, `png()`, `tiff()` und `bmp()`.

Man kann in den Funktionen auch die genaue Breite oder Höhe der Grafik einstellen.

Speichern von Daten und Objekten

Mit der Funktion `save` können Objekte in einer .RData-Datei abgespeichert werden.

```
> save(miete, file="Daten/Kopie_miete.RData")
```

Datensätze können mit den Funktionen `write.table()`, `write.csv()` und `write.csv2()` auch als Text- oder .csv-Datei exportiert werden. Die Befehle unterscheiden sich insbesondere darin, welche (default-)Werte z.B. für das Trennzeichen eingestellt sind.

```
> write.csv2(miete, file="Daten/Kopie_miete.csv")
```


Literatur zum Einstieg

- Richard Cotton: *Learning R*
- Peter Dalgaard: *Introductory Statistics with R*
- Uwe Ligges: *Programmieren mit R*
- Daniel Wollschläger: *Grundlagen der Datenanalyse mit R*