LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

R Einstieg

Manuel Eugster, Armin Monecke, Faban Scheipl

Institut für Statistik
Ludwig-Maximilians-Universität München

Einführung in R

Was ist S?

S ist eine Sprache für Datenanalyse und Graphik, entwickelt von John Chambers und Kollegen in den Bell Laboratories ab den 70ern um

- interaktives Rechnen mit Daten in einer
- vollwertigen interpretierten Programmiersprache, die
- den Benutzern Zugang zu allen Strukturen und
- Graphik für explorative Datenanalyse und Publikationszwecke bietet,

zu ermöglichen.

Was ist R?

R ist eine Implementation der Sprache S, anfänglich von Ross Ihaka und Robert Gentleman (Univ. Auckland) entwickelt und seit Mitte der 90iger Jahre von einem Entwickler-Kollektiv (R-Core) betreut. Da R einfach erweiterbar ist, hat sich über die Jahre eine weltweite, extrem aktive Entwicklergemeinschaft gebildet.

Informationen zu R:

http://www.R-project.org

Die Sprache R

R als Tischrechner

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

```
> 1 + 1
[1] 2
> a = 0.5
> b = a*2
> exp(b)
[1] 2.718282
> log(exp(b))
[1] 1
> c = (cos(log(exp(b))) - a) / b + 1
> c
[1] 1.040302
```

Wichtige Operatoren

`+ - * /`	Grundrechnungsarten
`^`	Potenzieren
`= <- ->`	Zuweisungen
`:`	Sequenz ganzer Zahlen erzeugen
`== != < > <= >=`	logische Vergleiche
`#`	Kommentarzeichen

```
> a <- 1:5
> a
[1] 1 2 3 4 5
> -a
[1] -1 -2 -3 -4 -5
> a * 10
[1] 10 20 30 40 50
> a^3
[1]   1   8  27  64 125
> 2^a
[1]  2  4  8 16 32
```

Wichtige Operatoren

Die Grundrechnungsarten folgen in der Abarbeitungsreihenfolge den üblichen Regeln („Punktrechnung vor Strichrechnung“), Zuweisungen und logische Vergleiche haben relativ niedrige Rangordnung. Im Zweifelsfall immer besser ein Klammernpaar mehr als weniger verwenden.

```
> 2 * 3^4 + 5
[1] 167
> (2 * (3^4)) + 5
[1] 167
> 2^a < 6
[1]  TRUE  TRUE FALSE FALSE FALSE
> 100 + 2^a < 6
[1] FALSE FALSE FALSE FALSE FALSE
```

Variablen

- Jede Auswertung kann in einer Variablen abgespeichert werden.

- Es gibt keine vorgegebene Maximalanzahl von Variablen (stimmt nicht ganz).

- Als Variablennamen können beliebige Kombinationen aus Buchstaben, Ziffern, Punkt und Unterstrich verwendet werden (stimmt auch nicht ganz).

- Variablennamen dürfen nicht mit einer Ziffer beginnen.

- R unterscheidet zwischen Groß- und Kleinschreibung.

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 (nicht Teil dieses Kurses).

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

Die Namen der Argumente können auch explizit angegeben werden, dann muß die Reihenfolge nicht mit der Funktionsdefinition übereinstimmen:

```
funktionsname(arg2=wert2, arg1=wert1)
```

Achtung: Funktionen werden an den auf den Namen folgenden *runden Klammern* erkannt, eckige Klammern haben eine andere Bedeutung. Die Zuweisungen der Form `arg=wert` immer mit dem Operator =, nicht <-.

Funktionen

Beispiele:

```
> b <- 2^a
> b
[1]  2  4  8 16 32
> log(b)
[1] 0.6931472 1.3862944 2.0794415 2.7725887 3.4657359

> log(b, 2)
[1] 1 2 3 4 5
> log(2, b)
[1] 1.0000000 0.5000000 0.3333333 0.2500000 0.2000000

> log(base=2, x=b)
[1] 1 2 3 4 5
```

Datentypen in R

Das einfachste Datenobjekt ist ein Vektor mit Elementen des Typs

- **numeric**: ganzzahlige oder Gleitkomma-Werte,
- **character**: beliebige Zeichen,
- **logical**: die Zustände `TRUE` und `FALSE`,
- **list**: ein Objekt beliebigen Typs, auch wieder eine Liste (rekursive Datenstrukturen!).

Jeder Vektor hat eine Länge (`length()`) und der Typ kann mittels `mode()` festgestellt werden.

Numerische Vektoren

Üblicherweise wird `c()` (concatenate: verbinden) zum Erstellen von mehrelementigen Vektoren verwendet. Weitere wichtige Funktionen sind `seq()` (Sequenzen) und `rep()` (Repeat).

```
> v1 <- c(1, 3.14, 17)
> v1
[1]  1.00  3.14 17.00
> v2 <- seq(from = -pi, to = pi, length = 5)
> v2
[1] -3.141593 -1.570796  0.000000  1.570796  3.141593
> v3 <- rep(2, 5)
> v3
[1] 2 2 2 2 2
> length(v3)
[1] 5
> mode(v3)
[1] "numeric"
```

Zeichen-Vektoren

Die Elemente von Zeichen-Vektoren bestehen aus einer Folge von beliebigen Zeichen, die lexikographisch geordnet werden. Zur Unterscheidung von Variablen werden Zeichenketten durch einfache oder doppelte Hochkomma am Anfang und Ende markiert.

```
> LETTERS
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O"
[16] "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
> mode(LETTERS)
[1] "character"
> Benutzer <- c("Hansi", "Seppl")
> Benutzer
[1] "Hansi" "Seppl"
> "Benutzer"
[1] "Benutzer"
> Benutzer < "R"
[1]  TRUE FALSE
> "Benutzer" < "R"
[1] TRUE
```

Logische Vektoren

Es gibt die zwei logischen Zustände `TRUE` und `FALSE`. In den meisten Fällen werden logische Vektoren nicht direkt eingegeben, sondern durch die logischen Operatoren `==`, `!=`, ... erzeugt. Der Operator `!` invertiert einen logischen Vektor.

```
> WoIstSeppl <- (Benutzer == "Seppl")
> WoIstSeppl
[1] FALSE  TRUE
> ! WoIstSeppl
[1]  TRUE FALSE
> which(WoIstSeppl)
[1] 2
```

Zugriff auf Vektorelemente

Generell erfolgt in R der Zugriff auf Teile einer Struktur durch *eckige Klammern*. Bei einem Vektor können Zahlen und logische Vektoren für den Zugriff auf einzelne Elemente verwendet werden, negative Zahlen schließen einzelne Elemente aus.

```
> v2
[1] -3.141593 -1.570796  0.000000  1.570796  3.141593
> v2[5]
[1] 3.141593
> v2[2:4]
[1] -1.570796  0.000000  1.570796
> v2[-(2:4)]
[1] -3.141593  3.141593
> v2[v2<0]
[1] -3.141593 -1.570796
```

Namen für Vektorelemente

Jedes Vektorelement kann einen Namen bekommen, diese können dann für den einfacheren Zugriff verwendet werden.

```
> konst <- c(e = exp(1), pi = pi, zweipi = 2 * pi)
> konst
       e       pi   zweipi
2.718282 3.141593 6.283185
> names(konst)
[1] "e"      "pi"     "zweipi"
> names(konst)[3] <- "2pi"
> konst
       e       pi      2pi
2.718282 3.141593 6.283185
> konst["2pi"]
     2pi
6.283185
> konst[c("pi","2pi")]
      pi      2pi
3.141593 6.283185
```

Faktoren

Nominale oder ordinale Daten werden in R „Faktoren“ genannt. Intern ist dies ein ganzzahliger Vektor, wo jeder Zahl ein „Label“ zugeordnet ist. Zeichen-Vektoren sind keine nominale Variablen, können aber mit der Funktion `factor()` leicht in eine solche verwandelt werden.

```
> behandlung <- rep(c("Kontrolle", "Mittelchen"), c(2,3))
> behandlung
[1] "Kontrolle"  "Kontrolle"  "Mittelchen" "Mittelchen"
[5] "Mittelchen"
> summary(behandlung)
   Length     Class      Mode
        5 character character
> behandlung <- factor(behandlung)
> behandlung
[1] Kontrolle  Kontrolle  Mittelchen Mittelchen Mittelchen
Levels: Kontrolle Mittelchen
> summary(behandlung)
 Kontrolle Mittelchen
         2          3
```

Geordnete Faktoren

Die Ordnung von ordinalen Meßgrößen wird durch geordnete Faktoren dargestellt.

```
> zu <- ordered(c("Gut", "Schlecht", "Super", "Super", "Frag nicht", "Gut"),
+               levels = c("Frag nicht", "Schlecht", "Gut", "Super"))
> zu
[1] Gut        Schlecht   Super      Super      Frag nicht
[6] Gut
Levels: Frag nicht < Schlecht < Gut < Super
```

Es ist wichtig, nominale und ordinale Variablen als solche zu codieren, da viele Statistik-Funktionen diese von Zeichen-Vektoren unterscheiden. Beim Datenimport wird dies aber ohnedies fast immer automatisch erledigt.

Datenmatrix

Die wohl wichtigste Struktur zur Haltung von Daten im üblichen Rechteckschema, wo die Beobachtungen in den Zeilen und die Variablen in den Spalten dargestellt werden, ist die Datenmatrix. In R wird diese `data.frame` genannt.

```
> data("iris")
> class(iris)
[1] "data.frame"
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Datenmatrix

Die Spalten von data frames beinhalten in der Regel numerische Vektoren oder Faktoren. Mittels `summary()` kann man feststellen, ob nominale oder ordinale Meßgrößen tatsächlich als solche gespeichert wurden.

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
       Species
 setosa    :50
 versicolor:50
 virginica :50
```

Teilmengen von Datenmatrizen

Man kann wieder den Operator [benutzen, um Zeilen oder Spalten (oder beides) aus data frames zu extrahieren. Dazu sind zwei Index-Ausdrücke nötig: einer für die Zeilen und einer für die Spalten, die durch ein Komma getrennt werden.

Sowohl für Zeilen wie auch Spalten können dieselben Indizierungen wie für einfache Vektoren verwendet werden (positive und negative Zahlen, logisch, Namen).

Teilmengen von Datenmatrizen

```
> data("iris")
> dim(iris)
[1] 150   5
> summary(iris[1:25,1:2])
  Sepal.Length    Sepal.Width
 Min.   :4.300   Min.   :2.90
 1st Qu.:4.800   1st Qu.:3.20
 Median :5.000   Median :3.40
 Mean   :5.028   Mean   :3.48
 3rd Qu.:5.400   3rd Qu.:3.70
 Max.   :5.800   Max.   :4.40
> summary(iris[,-(1:2)])
  Petal.Length    Petal.Width          Species
 Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :4.350   Median :1.300   virginica :50
 Mean   :3.758   Mean   :1.199
 3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :6.900   Max.   :2.500
```

Teilmengen von Datenmatrizen

Auf benannte Spalten eines data frames kann auch einfacher mit dem $-Operator zugegriffen werden:

```
> iris$Sepal.Width
  [1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 3.4 3.0 3.0 4.0
 [16] 4.4 3.9 3.5 3.8 3.8 3.4 3.7 3.6 3.3 3.4 3.0 3.4 3.5 3.4 3.2
 [31] 3.1 3.4 4.1 4.2 3.1 3.2 3.5 3.6 3.0 3.4 3.5 2.3 3.2 3.5 3.8
 [46] 3.0 3.8 3.2 3.7 3.3 3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.4 2.9 2.7
 [61] 2.0 3.0 2.2 2.9 2.9 3.1 3.0 2.7 2.2 2.5 3.2 2.8 2.5 2.8 2.9
 [76] 3.0 2.8 3.0 2.9 2.6 2.4 2.4 2.7 2.7 3.0 3.4 3.1 2.3 3.0 2.5
 [91] 2.6 3.0 2.6 2.3 2.7 3.0 2.9 2.9 2.5 2.8 3.3 2.7 3.0 2.9 3.0
[106] 3.0 2.5 2.9 2.5 3.6 3.2 2.7 3.0 2.5 2.8 3.2 3.0 3.8 2.6 2.2
[121] 3.2 2.8 2.8 2.7 3.3 3.2 2.8 3.0 2.8 3.0 2.8 3.8 2.8 2.8 2.6
[136] 3.0 3.4 3.1 3.0 3.1 3.1 3.1 2.7 3.2 3.3 3.0 2.5 3.0 3.4 3.0
```

Fehlende Werte

Die Codierungen `NA` (Not Available) und `NaN` (Not a Number) bezeichnen fehlende Werte in einem Vektor oder anderen Datenstruktur. `NA` kann sich aus der Datenerhebung ergeben, kann das Ergebnis einer (fehlgeschlagenen) Berechnung sein oder auch zugewiesen werden.

```
> rn <- rnorm(5)
> rn
[1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078
> log(rn)
[1]        NaN -1.6947599        NaN  0.4670498 -1.1101553

Warning message:
In log(rn) : NaNs produced
```