UNIVERSITÄT LEIPZIG

Eine Einführung in R: Grundlagen I

Bernd Klaus, Verena Zuber
Institut für Medizinische Informatik, Statistik und Epidemiologie (IMISE),
Universität Leipzig

http://www.uni-leipzig.de/ zuber/teaching/ws12/r-kurs/

11. Oktober 2012

1. Grundlegendes zu R: Einstieg
   - Start und Hilfe
   - Was sind Pakete?
   - R als Taschenrechner

2. Grundlegendes zu R: Objekte und Klassen
   - Elementartypen
   - Vektoren
   - Matrizen
   - Datensätze
   - Benutzen von Funktionen in R

- Download unter *http://cran.r-project.org*
- R besteht aus einem Grundprogramm mit vielen Zusätzen den sogenannten *packages* oder Pakete
- Hilfe per `?<Name>` oder `help.search(suchbegriff)`
- Übersicht über die Hilfe `help.start()`
- Pakete speziell für Bioinformatik / Biostatistik: *http://bioconductor.org/*

R ist sensibel gegenüber Großschreibung

Was sind Pakete?

- R bietet eine Vielzahl frei verfügbarer Pakete
- Ein Paket enthält unterschiedlichste, spezielle Funktionen
- Beim Start von R ist nur eine Grundausstattung geladen, alle anderen Pakete müssen zusätzlich geladen werden
- Jeder kann sein eigenes Paket schreiben
- Derzeit gibt es 4076 Pakete (Stand Oktober 2009: 2112 Pakete)
- Es besteht aber KEINE GARANTIE für richtige Funktionsweise!

Was sind Pakete?

- Überblick über die geladenen Pakete `sessionInfo()`
- package laden `require(packagename)` oder `library(packagename)`
- package installieren `install.packages(packagename)`
- Wichtige Pakete:
  - `glmnet`: Lasso and elastic-net regularized generalized linear models
  - `mvtnorm`: Multivariate Normalverteilung
  - `samr`: Significance Analysis of Microarrays
  - `fda`: Functional Data Analysis (z.B. Kurvenglättung)
- Mögliche Pakete:
  - `sendmailR`: send email from inside `R`
  - `twitteR`: `R` based Twitter client
  - `sudoku`: Sudoku Puzzle Generator and Solver

R als Taschenrechner

- Wertzuweisungen mit `<-`
- Bsp. `a <- 5` => die Variable `a` hat jetzt den Wert 5.

R beherrscht die bekannten Rechenoperationen

- `a+b, a-b, a*b, a^b, a %% b (a MOD b)`
- aber auch `sqrt(a), sin(a)` ...
- und einfache Statistik:
  - Mittelwert `mean(a,b)`
  - Zusammenfassung `summary(a,b)`
  - Varianz `var(a,b,c)`
  - Minimum, Maximum `min(a,b), max(a,b)`

Daten: Klassen und Objekte I

- R ist objektorientiert
- Objekt = alles was in R vorkommt
- Objekte liegen verschiedene Baupläne zu Grunde, die sogenannten Klassen:
  - Elementartypen
  - Vektoren
  - Matrizen
  - Datensätze

- `text <- "IMISE"` das Objekt `text` speichert jetzt den Text IMISE
- `class(text)` gibt die Klasse von `text` an, in diesem Fall String
- `ls()` gibt eine Liste der aktuell im RAM vorhandenen Objekte an
- `rm(Objektname)` löscht ein bestimmtes Objekt
- `is.Klassenname()` überprüft, ob ein Objekt zu einer bestimmten Klasse gehört
- `as.Klassenname(Objekt)` überführt ein Objekt in eine andere Klasse (das sog. casten)
- `summary(Objekt)` oder `str(Objekt)` zeigt Informationen über das Objekt an

Elementartypen

- *numeric*: ganze Zahl oder reelle Zahl
- *string*: Zeichenkette, Text
- *factor*: String oder Zahl, der eine begrenzte Anzahl an Kategorien beschreibt; geeignet für kategoriale Variablen
- *logical*: Wahrheitswerte `TRUE FALSE`
- NA: fehlender Wert

- Wir weisen dem Objekt `a` den Wert Neun zu: `a<-9`
- Ist `a` ein String?
  `is.character(a)`
  `FALSE`
- Ist `a` eine Zahl?
  `is.numeric(a)`
  `TRUE`
- Nun soll `a` in einen Faktor verwandelt werden:
  `a<-as.factor(a)`
- Ist `a` ein Faktor?
  `is.factor(a)`
  `TRUE`
- Wir weisen dem Objekt `a` einen Namen zu: `a<-"NAME"`
- Was ist `a`?
  `summary(a)`
  `Length 1 Class character`

vector = eindimensionale Sammlung von atomaren Objekten

- `a <- c(5, 6, 7)` `a` ist jetzt ein Vektor
  mit den drei Elementen 5, 6, 7
- Wie lange ist `a`? `length(a)`
- Zugriff auf bestimmtes Objekt erfolgt mit `a[Stelle]`
  `a[2]` ergibt als Ausgabe also 6
- Mit Vektoren kann man fast wie mit Zahlen rechnen
- `3*a` ergibt z.B. den Vektor `(15,18,21)`
- `a[2] = a[2]*3` ergibt dagegen den Vektor `(5,18,7)`
- `R` behandelt Vektoren wie Spaltenvektoren
- Sortieralgorithmen:
  `sort(a)`, `order(a)`, `rank(a)`

matrix = Mehrdimensionaler Vektor

- Bsp: `mat <- matrix(c(1,0,0,0,1,0,0,0,1), nrow = 3)` erzeugt die Einheitsmatrix und speichert sie in dem Objekt `mat`
- Zugriff auf die Einträge mit `[Zeile, Spalte]`,
  `mat[1,1]` ergibt z.B. die Zahl 1,
  `mat[,1]` die erste Spalte
  und `mat[,]` die ganze Matrix
- Es können auch einzelne Zeilen `mat[Zeile,]`
  oder Spalten `mat[,Spalte]` gewählt werden
- Mehrere konsekutive Zeilen bzw Spalten sind mittels `von:bis` zu erhalten
- Ausschließen einer Zeile erfolgt über `mat[-Zeile,]`

Wichtige Klassen - Matrizen II: Zugriff

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

- `A[1,3]`=3
- `A[3,1]`=7
- `A[1,]`=1,2,3
- `A[,1]`=1,4,7
- `A[,-1]`= $\begin{pmatrix} 2 & 3 \\ 5 & 6 \\ 8 & 9 \end{pmatrix}$ = `A[,2:3]`

Wichtige Klassen - Matrizen III: Definition

Nützliche Befehle:

- Abfrage der Dimension `dim(A)`
- Zugriff auf die Diagonalelemente `diag()`
- Zusammenfügen von Matrizen über:
  - Nebeneinander (*columnbind*): `cbind(A,B)`
  - Untereinander (*rowbind*): `rbind(A,B)`
- Erzeugen einer Sequenz von 1 bis 10: `seq(from=1,to=10)`
- Erzeugt einen Vektor mit 10 Einsen: `rep(1,10)`

Wichtige Klassen - Matrizen IV: Operationen

Sei `a` eine Zahl und `A,B` geeignete Matrizen

- Multiplikation mit einem Skalar `a*A`
- Matrizenmultiplikation `A%*%B`
- Invertieren `solve(A)`
- Transponieren `t(A)`
- Determinante `det(A)`
- Spur `sum(diag(A))`
- Eigenwertzerlegung `eigen(A)`

dataframe = Tabelle, z.B. Spalten repräsentieren Variablen, Zeilen Beobachtungen

- `head()` liefert die ersten Einträge des Datensatzes, `names()` bringt die Namen der Spalten
- Der Zugriff auf einzelne Spalten ist per `dataframe$Spaltenname` möglich
- Per `attach( )` kann direkt auf die Variablen zugegriffen werden
- Zum Schluss `detach( )` nicht vergessen (Verwechslungsgefahr)!

In der R-Hilfe finden sich zum Befehl matrix folgende Informationen

matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)

- In den runden Klammern stehen die Variablen, für die Werte angegeben werden können
- Falls keine Werte angegeben werden, benutzt R die Standardwerte nach dem " = "
- Der Befehl `matrix(c(1,0), nrow = 2 )` erzeugt einen Matrix mit 2 Zeilen und einer Spalte (also einen Vektor)
- Der Befehl `matrix(c(1,0,3,4), nrow = 2, ncol = 2 )` dagegen erzeugt eine 2 x 2 Matrix
- Äquivalent dazu aber nicht empfohlen: `matrix(c(1,0,3,4), 2, 2)`