Institut für Stochastik
Universität Marburg
Prof. Dr. H. Holzmann
Dipl.-Math. oec. F. Ketterer

SS 2009

Ergänzungsmaterial zum

Statistischen Praktikum

Kurze Learning by Doing Einführung in R

Vorbemerkungen
Das Ziel dieser kurzen Einführung ist es, Ihnen ein Basiswissen über R zu vermitteln, das Sie bei der Bearbeitung der Praktikumsaufgaben immer wieder benötigen werden.

R ist für viele Betriebssysteme verfügbar und unter

http://cran.r-project.org

frei erhältlich. Dort findet man auch eine große Zahl von Software-Bibliotheken sowie Literatur zu R. Außerdem sei hier auf das Skript „Datenanalyse und Graphik mit R“ von Dr. Bernhard Klar verwiesen, aus welchem auch diese Einführung größtenteils entnommen ist. Sie finden das vollständige Skript auch auf der Homepage zum Praktikum.

Start von R
Starten Sie R.
Nach dem Start von R erscheint eine graphische Oberfläche mit einigen Menüs und der R-Console, in welche die Befehle direkt eingegeben werden können.

R als Taschenrechner
In R sind eine Vielzahl von mathematischen Funktionen eingebaut. Schauen Sie was bei den folgenden Eingaben nach drücken der Entertaste passiert (da am Ende jedes R-Befehls Enter gedrückt werden muss, wird dies von nun an nicht mehr speziell erwähnt):

```
> 2+2
> 5-3
> 2*3*4*5
> 2+3*4
> 2^5
> sqrt(2)        #Quadratwurzel
> log(2)
> sin(pi/2)
> sin(90)
```

Bemerkung: Durch `>` wird jeweils angezeigt, dass die Berechnung zu Ende ist und ein weiteres Kommando eingegeben werden kann. Die `[1]` vor der Ausgabe besagt, dass das erste angeforderte Element folgt. Alles was nach dem Zeichen `#` folgt, wird von R als Kommentar gewertet.

Vektoren und Matrizen

Erstellen Sie jetzt einen Vektor x mit den Einträgen 2,3,5,2,7,1. Dies geschieht mit Hilfe des Zuweisungsoperators „ <- “ durch die Eingabe:

```
> x <- c(2,3,5,2,7,1)
```

Das `c` in `c(2,3,5,2,7,1)` steht für „combine“ also „zusammenfassen“.
Was passiert jetzt bei der Eingabe von

```
> x
```

Überprüfen Sie, ob R zwischen Groß- und Kleinschreibung unterscheidet.

Achtung: Sie sollten keine Vektoren mit Namen, die R-Befehle sind kreieren. Beispielsweise sind die Namen `c, t, T, F` und `max` „verboten“.

Erstellen Sie jetzt einen weiteren Vektor mit Namen `y` und den Einträgen 1,2,3,4,5,6.
Anstatt mit `y <- c(1,2,3,4,5,6)`, geht dies auch einfacher mit

```
> y <- 1:6
```

oder

```
> y <- seq(1,6)
```

Was passiert bei den folgenden Eingaben?

```
> seq(1,6,0.1)
> rep(y,3)
> length(x)
> sum(y)
> x+y
> x-y
> x*y
> c(x,y)
```

Neben den bisher kennen gelernten Vektoren, die vom Datentyp numeric waren, gibt es noch Vektoren vom Typ logical und vom Typ character.

Was wird bei der Eingabe des folgenden Vektors ausgegeben?

```
> c(T,F,F,T,F,T,T)
```

Was wird bei der Eingabe des folgenden String-Vektors (das heißt eines Vektors, dessen Elemente alle vom Typ character sind) ausgegeben?

```
> c(“Karlsruhe“,“Heidelberg“,“Freiburg“,“Stuttgart“)
```

Achtung: Die Anführungszeichen können nicht einfach in R kopiert werden, sondern müssen dort nochmals neu eingegeben werden.

Um auf Teilmengen von Vektoren zuzugreifen, gibt es zwei gebräuchliche Arten:
1.) Sie können direkt die Stelle der Elemente angeben, auf die zugegriffen werden soll. Dies geschieht durch die Angabe der Elemente, auf die zugegriffen werden soll, in eckigen Klammern.
Was wird bei den folgenden Eingaben ausgegeben?

```
> x[4]
> x[c(1,5)]
> x[2:5]
```

2.) Sie können auch einen Vektor von logischen Werten angeben. Dann werden diejenigen Elemente extrahiert, welche den Wert `T` (`TRUE`) haben. Dabei können die Relationen `<`, `<=`, `>`, `>=`, `==` und `!=` verwendet werden.

Schauen Sie, was man bei den folgenden Eingaben erhält:

```
> z <- x>2
> z
> x[z]
> x[x>2]
> x[x!=2]
```

Was erhalten Sie für

```
> which(x>2)
> y.neu <- y[-3]
> y.neu
```

Mit Hilfe von Vektoren können auch Matrizen gebildet werden. Dies geschieht mittels der Befehle `cbind` und `rbind` (`cbind` steht dabei für column-bind und `rbind` für row-bind).
Erstellen Sie zunächst die folgenden Matrizen:

```
> mat1 <- cbind(x,y)
> mat2 <- rbind(x,y,x+y)
```

Überprüfen Sie, ob die beiden Matrizen tatsächlich, die von Ihnen erwartete Form haben:

```
> mat1
> mat2
```

Sie können mit Hilfe des Kommandos `matrix()` Matrizen auch direkt eingeben, also anstatt `cbind(x,y)` beispielsweise

```
> matrix(c(2,3,5,2,7,1, 1,2,3,4,5,6),nrow=6,ncol=2)
```

Schauen Sie nun, was bei den nächsten Eingaben ausgegeben wird:

```
> mat2[2,3]
> mat2[2,]
> mat1[,1]
> t(mat2)
```

Das Rechnen mit Matrizen folgt denselben Regeln wie jenes mit Vektoren; es wird also elementenweise gerechnet. Wenn Sie das Matrizenprodukt und nicht das elementenweise Produkt wollen, verwenden Sie `%*%`:

```
> mat2%*%mat1
```

Data Frames

Die zweite grundlegende Datenstruktur in R sind, neben den Vektoren, so genannte Data Frames. Ein Data Frame ist eine Verallgemeinerung einer Matrix, bei der die verschiedenen Spalten unterschiedliche Datentypen haben können. Jedoch müssen alle Elemente einer Spalte vom gleichen Typ sein. Ein Data Frame kann Zeilen- und Spaltennamen besitzen.

Laden Sie das Data Frame `hills` aus dem Paket `MASS` und verschaffen Sie sich einen Überblick über seine Einträge:

```
> library(MASS); data(hills)
> help(hills)
> hills
> summary(hills)
```

Greifen Sie nun die dritte Spalte des `hills`-Datensatzes heraus und speichern sie diese in einem Vektor mit dem Namen `Zeit`. Dafür können Sie einen der folgenden drei Befehle verwenden:

```
> Zeit <- hills$time
> Zeit <- hills[,3]
> Zeit <- hills[,"time"]
```

Hinweis: Eine Liste aller Datensätze, die in den geladenen Paketen vorhanden sind, erhält man durch

```
> data()
```

Die folgende Tabelle enthält Werte, die mit einem Gummiband ermittelt wurden. Dabei gibt das Merkmal Weite die Entfernung (in cm) an, um die sich das Gummiband nach Dehnung um eine bestimmte Strecke (in mm) bewegte, nachdem es losgelassen wurde.

Dehnung (mm)	46	54	48	50	44	42	52
Weite (cm)	148	182	173	166	109	141	166

Geben Sie diese Daten als Data Frame mit Namen `gummi` ein. Dies geschieht mit dem Befehl `data.frame()`:

```
> gummi <- data.frame(dehnung=c(46,54,48,50,44,42,52),weite=c(148,182,173,166,109,141,166))
> gummi
```

R-Objekte

Alle Elemente, die in R erzeugt wurden, auch Funktionen, auf die wir später noch kurz eingehen werden, existieren als Objekte im so genannten Workspace.
Schauen Sie sich die Liste aller Objekte im Workspace an:

```
> ls()
```

Löschen Sie nun das Objekt z und überprüfen Sie, ob es tatsächlich gelöscht wurde:

```
> rm(z)
> ls()
```

Am Ende Ihrer R-Sitzung können Sie alle innerhalb dieser Sitzung erzeugten Objekte löschen.

Graphik mit R

Machen sie sich jetzt mit den Graphikfunktionen von R vertraut.
Zeichnen Sie dazu zunächst ein Histogramm der `dist`-Werte im `hills`-Datensatz:

```
> hist(hills$dist)
```

Erstellen Sie nun ein Streudiagramm (auch „Scatterplot“ oder „Punktewolke“ genannt) der `time`-Werte gegen die `climb`-Werte. Ein solches Diagramm ist auf verschiedene Arten erhältlich. Überprüfen Sie, ob die folgenden Befehle alle die gleiche Graphik liefern. Ändern Sie hierzu zunächst durch den Aufruf von `par()` die Standardeinstellung der Graphik-Parameter, sodass anstatt einer Graphik drei pro Seite ausgegeben werden:

```
> par(mfrow=c(3,1))  # Anzahl der Graphiken unter- bzw. nebeneinander
> plot(hills$climb,hills$time)
> plot(time~climb, data=hills)  # Formelschreibweise
> plot(hills$time~hills$climb)
```

Stellen Sie nun wieder eine Graphik pro Seite ein:

```
> par(mfrow=c(1,1))
```

Was passiert bei der nächsten Eingabe?

```
> plot(time~climb)
```

Mit dem `attach`-Befehl findet R die Merkmale `time` und `climb` auch ohne explizite Angabe:

```
> attach(hills)
> plot(time~climb)
> detach(hills)
> plot(time~climb)
```

Fügen Sie nun Ihrer Graphik den Titel „Streudiagramm“ hinzu:
```
> title(“Streudiagramm“)
```
Durch das Anklicken des Print-Buttons können Sie das Streudiagramm ausdrucken.

Was erhalten Sie bei der Eingabe
```
> pairs(hills)
```

Machen Sie sich mit Hilfe des nächsten Aufrufes ein Bild über die, weit über das bisher Gesehene hinausgehenden, Graphikfähigkeiten von R.
```
> demo(graphics)
```

Hinweis: Eine Liste aller zur Verfügung stehenden Demos erhalten sie mit dem Befehl `demo()`.

(Lineare) Regression mit R
Zeichnen Sie nun das Streudiagramm der `time`-Werte gegen die `dist`-Werte des `hills`-Datensatzes. Welchen Zusammenhang zwischen diesen beiden Größen beobachten Sie?
Passen Sie eine Regressionsgerade der Form $\texttt{time} = \beta_1 + \beta_2 \cdot \texttt{dist}$ an die Daten an. Bestimmen Sie dafür die geschätzten Regressionskoeffizienten $\widehat{\beta}_1$ und $\widehat{\beta}_2$, welche man mit der Kleinste-Quadrate Methode erhält, und zeichnen Sie die Regressionsgerade in das Streudiagramm ein:
```
> plot(hills$time~hills$dist)
> lm.hills <- lm(time~dist,data=hills)
> lm.hills
> lm.hills$coefficients
> abline(lm.hills)
```

Betrachten Sie den Regressionsoutput, welchen Sie mit Hilfe des folgenden Kommandos erhalten.
```
> summary(lm.hills)
```
Dieser Output wird im weiteren Verlauf des Praktikums noch detaillierter besprochen.
Versuchen Sie das obige Vorgehen bei den `hills`-Daten auf die `gummi`-Daten zu übertragen.

Zufallsvariablen und Verteilungen
Erzeugen Sie zehn im Intervall $[0, 1]$ gleichverteilte Zufallszahlen mit Hilfe des Befehls
```
> runif(10,0,1)
```
Eine entsprechende Funktion gibt es für viele Verteilungen; zum Beispiel werden durch `rnorm(N,mu,sigma)` `N` (stochastisch unabhängige) Realisierungen von normalverteilten Zufallsvariablen mit Erwartungswert `mu` und Standardabweichung(!) `sigma` erzeugt. Generieren Sie mit Hilfe dieser Funktion nun fünf $N(2, 9)-$verteilte Zufallszahlen.
```
> rnorm(5,2,3)
```

Bemerkung: Da sich die im Computer erzeugten Folgen von Zufallszahlen zwar in vieler Hinsicht wie zufällige Zahlenfolgen verhalten, sie aber rein deterministisch erzeugt werden, nennt man die Zahlen Pseudozufallszahlen.

Die Verteilungsfunktion $F(t) = P(X \leq t)$ einer Zufallsvariable X an der Stelle t können Sie berechnen, indem Sie bei der Funktion für die Zufallszahlenerzeugung das `r` (für random) durch ein `p` (für probability) und die Anzahl N durch t ersetzen.
Berechnen Sie den Wert der Verteilungsfunktion einer Standardnormalverteilung an der Stelle 0:
```
> pnorm(0,0,1)
```
Analog erhalten Sie die Dichte bzw. das Quantil an der Stelle an der Stelle t, indem Sie das `r` durch ein `d` (für density) bzw. durch ein `q` (für quantile) ersetzen.

Bestimmen Sie damit den Wert der Dichte einer Standardnormalverteilung an der Stelle 0 und überprüfen Sie Ihr Ergebnis, indem Sie diesen Wert „von Hand“ berechnen:

```
> dnorm(0,0,1)
> 1/sqrt(2*pi)
```

Berechnen Sie auch noch das Quantil der Standardnormalverteilung an der Stelle 0.5:

```
> qnorm(0.5,0,1)
```

Bei diskreten Zufallsvariablen X erhalten Sie durch das Voranstellen von `d` die Wahrscheinlichkeit $P(X = k)$. Bestimmen Sie für $X \sim Bin(3, 0.5)$ und $k = 0, \ldots, 3$ jeweils diese Wahrscheinlichkeit:

```
> dbinom(0:3,3,0.5)
```

Die folgende Tabelle gibt Ihnen einen Überblick über die Verteilungen in R.

Verteilung	**Zufallszahlen**	**Verteilungsfunktion**
$Bin(n,p)$	`rbinom(N, n, p)`	`pbinom(t, n, p)`
$Hyp(n,r,s)$	`rhyper(N, r, s, n)`	`phyper(t, r, s, n)`
$Po(\lambda)$	`rpois(N, λ)`	`ppois(t, λ)`
$G(p)$	`rgeom(N, p)`	`pgeom(t, p)`
$U(a,b)$	`runif(N, a, b)`	`punif(t, a, b)`
$Exp(\lambda)$	`rexp(N, λ)`	`pexp(t, λ)`
$N(\mu,\sigma^2)$	`rnorm(N, μ, σ)`	`pnorm(t, μ, σ)`
$LN(\mu,\sigma^2)$	`rlnorm(N, μ, σ)`	`plnorm(t, μ, σ)`

In den Standard-Paketen von R sind außerdem vorhanden:
Beta-Verteilung (`beta`), Cauchy-Verteilung (`chauchy`), Chi-Quadrat-Verteilung (`chisq`), F-Verteilung (`f`), Gamma-Verteilung (`gamma`), logistische Verteilung (`logis`), negative Binomialverteilung (`nbinom`), Student'sche t-Verteilung (`t`), Tukey-Verteilung (`tukey`), Weibull-Verteilung (`weibull`), Wilcoxon-Verteilung (`wilcox`).

Viele weitere Verteilungen finden Sie in anderen Paketen, insbesondere im Paket `SuppDists`.

Schleifen

Wandeln Sie die Temperatur in Grad Celsius in Grad Fahrenheit um. Verwenden Sie hierzu zunächst eine `for`-Schleife:

```
> for(celsius in 25:30){print(c(celsius,9/5*celsius+32))}
```

Da die allermeisten Funktionen in R Vektoren als Argumente zulassen, können Schleifen in R oft vermieden werden.
Lösen Sie die obige Aufgabe nun ohne Verwendung einer Schleife.

```
> celsius <- 25:30
> print(9/5*celsius+32)
```

Weitere Befehle zur Konstruktion von Schleifen sind `while`, `repeat`, `break`.

Wichtige eingebaute Funktionen im Überblick

Im Folgenden sind einige (teilweise schon bekannte) wichtige Funktionen, welche in R eingebaut sind, aufgeführt:

```
print()      # ein einzelnes R Objekt ausgeben
cat()        # mehrere R Objekt nacheinander ausgeben
summary()    # Zusammenfassung eines Objektes ausgeben
             # Details haengen vom Objekt ab
```

```
length()     # Anzahl der Elemente in einem Vektor
             # Anzahl der Merkmale in einem Data Frame
mean()       # Mittelwert
sum()        # Summe
prod()       # Produkt
var()        # Varianz
sd()         # Standardabweichung
median()     # Median
IQR()        # Quartilsabstand
quantile()   # Gibt Minimum, unteres Quartil, Median,
             # oberes Quartil und Maximum aus
min()        # Minimum
max()        # Maximum
sort()       # ordnet die Elemente der Groesse nach
rank()       # bestimmt die Raenge der Elemente
```

Wenden Sie diese Funktionen (soweit möglich) auf die beiden Vektoren `x` und `y` von vorher an.

Die Funktion `sapply()` wendet eine Funktion auf alle Spalten eines Data Frames an. Geben Sie mit Hilfe dieser Funktion die Mittelwerte sowie die Standardabweichungen der einzelnen Spalten des Datensatzes `hills` an:

```
> sapply(hills,mean)
> sapply(hills,sd)
```

Definieren von Funktionen

Es ist einfach, Funktionen in R selbst zu definieren. Die folgende Aufgabe ist ein Beispiel hierfür. Definieren Sie nun selbst eine Funktion in R mit dem Namen `c.nach.f`, welche die Temperatur in Grad Celsius nach Grad Fahrenheit umrechnet:

```
> c.nach.f <- function(celsius){9/5*celsius+32}
```

Rufen Sie die eben definierte Funktion `c.nach.f` nun für die Werte 15 und 25:30 auf.

```
> c.nach.f(15)
> c.nach.f(25:30)
```

Von einer Funktion wird immer der Wert des letzten (und in diesem Beispiel einzigen) Ausdrucks im Funktionskörper zurückgegeben. Alternativ ist es möglich mit `return()` einen Wert explizit zurückzugeben. Beachten Sie, dass der Funktionskörper durch `{ }` eingeschlossen wird.

Hilfe in R

Wenn Sie einen Befehl nicht kennen oder mehr Details zu einem Befehl wissen wollen, benutzen Sie den `help()`-Befehl.

Schauen sie sich die Erklärung zum `lm`-Befehl an:

```
> help(lm)
```

Erhalten Sie bei der folgenden Eingabe dasselbe?

```
> ?lm
```

Führen Sie nun das Beispiel am Ende der Hilfeseite mittels `example()` aus:

```
> example(lm)
```

Hinweis: Wenn Sie zu einem Begriff Befehle suchen, deren Namen Sie nicht kennen, erhalten Sie mit Hilfe des Kommandos `help.search()` eine Liste von Befehlen, die mit diesem Begriff assoziiert sind. (In Klammern wird dabei jeweils die Library aufgeführt, in welcher der entsprechende Befehl vorkommt. Diese müssen Sie unter Umständen zuerst mittels `library(Libname)` laden, bevor Sie ihre Befehle verwenden können.)

Suchen Sie den Befehl für die Varianz:
```
> help.search(“variance“)
```

Beenden von R

Beenden Sie nun Ihre R-Sitzung indem sie das File-Menü anklicken und dort auf Exit gehen. Alternativ können Sie R durch die Eingabe des quit-Kommandos beenden:
```
> q()
```
Antworten Sie mit „No“ auf die Frage „Save workspace image?“. Falls Sie mit „Yes“ antworten, so werden alle Objekte, die während Ihrer Sitzung erzeugt worden sind, gesichert.

Arbeiten mit einem .R (Script-)File

Die Befehlseingabe direkt in der R-Console ist meist nur für wenige kurze Befehle sinnvoll. Für größere statistische Analysen hat es sich als sehr sinnvoll erwiesen, die Befehle in eine Textdatei zu schreiben und sie von dort an die R-Console zu übertragen. Die Datei kann dann editiert und gespeichert werden, so dass am Ende die vollständige Datenauswertung vorliegt. Am einfachsten ist die Befehlseingabe mittels Skriptfiles.

Klicken Sie dazu nach dem öffnen von R im Datei-Menü auf Neues Skript. Dann öffnet sich ein Fenster, in dem Befehle eingegeben, editiert und gespeichert werden können. Ein bereits vorhandenes Skriptfile können Sie ebenfalls über das Datei-Menü öffnen.
Mit der Taste F5 können Sie die aktuelle Zeile bzw. den markierten Bereich ausführen lassen. Durch den Eintrag „Alles ausführen“ im Bearbeiten-Menü werden alle Befehle im Skriptfile der Reihe nach ausgeführt.