Statistik Aufgabensammlung

Sommersemester 2015

Prof. Dr. Stefan Etschberger – Hochschule Augsburg

Anmerkungen zu den Übungsaufgaben:

- Nach der Vorlesung finden Sie jeweils in der Aufgabensammlung die für die jeweilige Woche zu bearbeitenden Aufgaben; besprochen und gelöst werden die Aufgaben in der darauf folgenden Woche in den Übungsgruppen.
- Grundlagen in R sind ein wichtiger, obligatorisch zu erlernender Bestandteil des Kurses; alle in den Übungsaufgaben behandelten Lösungen in R sind prüfungsrelevant und müssen auch bei veränderter Aufgabenstellung (ohne Rechner) gelöst werden können.
- Es gibt für die Klausur keine Einschränkung auf nur eine Aufgabe mit R. Klausuraufgaben mit R könnten in der Prüfung bei verschiedenen Themen als Teilaufgabe oder als separate Aufgabe eingebaut sein. R-Teile in der Klausur können, müssen aber nicht als single choice formuliert sein.
- Es gibt *kein* vorgefertigtes „cheat-sheet“ mit den wichtigsten Funktionen in R für die Klausur; bitte schreiben Sie sich die wichtigsten Funktionen inkl. Parametern auf Ihre selbsterstellte Formelsammlung. Vorausgesetzt werden für die Klausur alle in den Lösungshinweisen der Übungsaufgaben verwendeten Funktionen.

Inhalt

Aufgaben zu R Grundlagen

Aufgabe 1

R: RStudio und erste Versuche

Installation und Kennenlernen von R und RStudio

(Sofern Sie über keinen eigenen Rechner verfügen, können Sie im Rechnerraum im W-Gebäude arbeiten; dort sind R und Rstudio installiert)

a) Installieren Sie R von `http://goo.gl/ALaUXu` (für Windows) bzw. von `http://cran.r-project.org/bin/` für andere Plattformen.

R ist das Statistikprogramm, das Daten verarbeitet und die Ergebnisse ausgibt; es ist in der Rohfassung nicht sehr komfortabel zu bedienen. Deswegen arbeiten wir in diesem Kurs mit RStudio, einer sehr komfortablen und mächtigen integrierten Entwicklungsumgebung.

b) Installieren Sie RStudio von `http://goo.gl/RX11dj`.

c) Öffnen Sie RStudio. Klicken Sie in den linken unteren Bereich des Fensters („Console“), tippen Sie

```
1 + 2
```

und schließen Sie die Eingabe mit Enter ab.

In der Kommandozeile der Konsole werden alle Anweisungen eingegeben und Textrückmeldungen des Programms ausgegeben; dazu gehören Ergebnisse, aber auch Hinweise, Warnungen und Fehlermeldungen, falls etwas nicht geklappt hat. Die Kommandzeile eignet sich auch prima als Taschenrechner. Kennt man die Bedeutung einer Funktion nicht, kann man ein Fragezeichen voranstellen und bekommt eine Erklärung (rechts im Hilfebereich).

Bei Rechenoperationen gelten die Vorrangregeln der Mathematik (Potenz vor Punkt vor Strich). Der Dezimaltrenner ist ein Punkt (kein Komma). Exponential-, Logarithmus- bzw. Quadratwurzeln berechnet man über Funktionsaufrufe, das Argument steht in runden Klammern. Groß- und Kleinschreibung macht einen Unterschied. Stellt man einer Zeile ein #-Zeichen voran, wird die Zeile von R nicht ausgeführt.

d) Geben Sie folgende Ausdrücke ein und erklären Sie jeweils das Ergebnis

```
2 + 3 * 4
(2 + 3) * 4
0.2 * 3 - 1.1
0,2 * 3
2^3^2
(2^3)^2
exp(1)
?exp
```

```
log(exp(1))
sqrt(16)
16^(1/2)
Sqrt(16)
# Das ist ein Kommentar.
```

Lösungshinweis:

```
2 + 3 * 4
## [1] 14
(2 + 3) * 4
## [1] 20
0.2 * 3 - 1.1
## [1] -0.5
# 0,2 * 3 # Fehler, ',' wird nicht als Dezimalkomma
# akzeptiert
2^3^2
## [1] 512
(2^3)^2
## [1] 64
exp(1)
## [1] 2.718282
log(exp(1))
## [1] 1
sqrt(16)
## [1] 4
16^(1/2)
## [1] 4
Sqrt(16)  # Fehler, sqrt() schreibt man mit kleinem 's'

## Error in eval(expr, envir, enclos): konnte Funktion "Sqrt" nicht finden
```

Aufgabe 2

Variablen, Zuweisungen und Funktionen

Zahlen (und andere Objekte) können in R in Variablen gespeichert werden. Dazu kann der Zuweisungsoperator = oder alternativ <- beispielsweise folgendermaßen verwendet werden:

```
x <- 3.5
x2 <- 1.5  # funktioniert genauso mit x2 = 1.5
```

Mit diesen Variablen kann dann weitergerechnet werden. In Variablennamen dürfen Buchstaben, Ziffern (nicht als erstes Zeichen), Punkte und Unterstriche (_) vorkommen. Diese Bezeichner dürfen keine Leerzeichen enthalten. Auch hier ist Groß- und Kleinschreibung zu beachten.

a) Weisen Sie der Variablen `x` den Wert 4 zu. Weisen Sie dann der Variablen `x.2` den folgenden Wert zu:

$$\sqrt{3x^2 + \ln\left(\frac{1}{e^x}\right) + 5}\,.$$

Funktionsaufrufe schreibt man in R mit einem Funktionsbezeichner, auf den direkt (keine Leerstelle!) ein Paar runder Klammern folgt. Innerhalb der runden Klammern können ein oder mehrere Argumente oder Parameter der Funktion stehen. Funktionen kann man auch verschachtelt aufrufen.
Die Funktion `ls()` gibt die in der aktuellen Sitzung definierten Objekte aus. Mit `rm(<Var>)` kann man eine Variable löschen, wenn man ihren Bezeichner anstatt `<Var>` in die runden Klammern schreibt.

b) Überlegen Sie was folgende Zeilen ausgeben und führen Sie diese dann in R aus, um Ihr Ergebnis zu überprüfen.

```
x
x.2
X
x + x.2
x.Produkt <- x * x.2
x.Produkt
x.Produkt <- x.Produkt * x
ls()
rm(x)
x
ls()
```

Außer Zahlen kann R auch mit Zeichenketten umgehen. Diese können in Objekten gespeichert werden, indem man die Zeichenkette in Anführungsstriche setzt. Zeichenketten, die Zahlen beinhalten werden nicht als Zahlen interpretiert. Man kann mit ihnen also nicht rechnen.

c) Welche Ausgabe bewirken folgende Zeilen? Überlegen Sie, bevor Sie die Eingabe in R ausprobieren.

```
tubby.1 <- "Tinky-Winky"
tubby.2 <- "Dipsy"
Zahl <- 10
keine.Zahl <- "10"
Zahl + 1
keine.Zahl + 1
```

Tricks zur Ein- und Ausgabe:

- *Ist eine Eingabe in einer Zeile nicht vollständig, kann R das mit einem „+“-Zeichen anzeigen; die Eingabe kann dann vervollständigt werden.*
- *Sofortige Hilfe bei der Eingabe einer Funktion erhält man, wenn man nach Eingabe der ersten Buchstaben des Funktionsbezeichners die Tabulator-Taste betätigt. Die möglichen Funktionen werden dann zur Auswahl aufgelistet und können dann ausgewählt werden.*
- *Mit der ↑-Taste auf der Tastatur kann der letzte (oder bei zweimaligem Drücken der vorletzte usw.) Befehl wieder sichtbar gemacht und dann nochmals ausgeführt oder verändert werden.*
- *Im RStudio-Fenster finden Sie (meistens rechts oben) einen Reiter History. Auch dort werden alle eingegebenen Befehle abgespeichert.*
- *Im Reiter Environment werden alle Objekte der aktuellen Sitzung aufgelistet.*

d) Probieren Sie die angesprochenen Tricks zur Ein- und Ausgabe aus.

Lösungshinweis:

```
x <- 4
x.2 <- sqrt(3 * x^2 + log(1/exp(x)) + 5)
x.2
```

```
## [1] 7
```

```
X
```

```
## Error in eval(expr, envir, enclos): Objekt 'X' nicht gefunden
```

```
x + x.2
```

```
## [1] 11
```

```
x.Produkt <- x * x.2
x.Produkt
```

```
## [1] 28
```

```
x.Produkt <- x.Produkt * x
rm(x)
x  # Fehler: x gibt's ja nicht mehr, kann deswegen auch nicht ausgegeben werden
```

```
## Error in eval(expr, envir, enclos): Objekt 'x' nicht gefunden
```

```
tubby.1 <- "Tinky-Winky"
tubby.2 <- "Dipsy"
Zahl <- 10
keine.Zahl <- "10"
Zahl + 1  # Ergebnis: 11
```

```
## [1] 11
```

```
keine.Zahl + 1  # Fehler: das geht nicht...
```

```
## Error in keine.Zahl + 1: nicht-numerisches Argument für binären Operator
```

Aufgabe 3

Daten: Vektoren

Eine Urliste von Daten eines Merkmals wird in R durch einen Vektor repräsentiert. Zur Erzeugung eines Vektors dient die Funktion `c()`. die Einträge der Urliste werden dann zum Beispiel als Argumente von `c()` durch Kommata getrennt angegeben. Als Ausprägungen sind Zahlen oder Zeichenketten möglich. R versucht dann durch die Art der Argumente automatisch zu entscheiden, ob es sich um ein nominales oder ein metrisches Markmal handelt.

a) Legen Sie eine Urliste für das Merkmal `x` an, das die Werte 1, 4, 2, 1.5 enthält. Geben Sie `x` aus. Legen Sie ein weiteres Merkmal `Geschlecht` mit den Werten `Mann`, `Frau`, `Frau`, `Frau` an. Geben Sie auch `Geschlecht` aus. Das dritte Merkmal `z` soll die Werte `1, 2, 1, "1"` enthalten. Ist `z` für R nominal oder metrisch? Überprüfen Sie Ihre Entscheidung.

Vektoren aufeinanderfolgender ganzer Zahlen werden mit dem Doppelpunkt-Operator gebildet. `2:5` steht zum Beispiel für den Vektor mit den Zahlen 2, 3, 4, 5. Mit der Funktion `seq()` kann man genauer Vektoren als Folgen von Zahlen erzeugen. `seq(from=2, to=3, by=0.2)` erzeugt zum Beispiel den Vektor (2, 2.2, 2.4, 2.6, 2.8, 3). Mit `rep()` werden Werte oder ganze Vektoren vervielfacht als Vektor ausgegeben. Zum Beispiel ergibt `rep(c(1,2), 3)` den Vektor (1,2,1,2,1,2). Die Hilfe-Seiten (Aufruf über `?seq` bzw. `?rep`) erklären die Details.

b) Erzeugen Sie folgende Vektoren in R:

```
## [1] 5 6 7 8 9
##  [1] 10  9  8  7  6  5  4  3  2  1
## [1] -0.10 -0.08 -0.06 -0.04 -0.02  0.00
## [1] 10000 12500 15000 17500 20000
##  [1] -3 -2 -1  0  1  2 -3 -2 -1  0  1  2 -3 -2 -1  0  1
## [18]  2
##  [1]  5.0  6.0  7.0  8.0  9.0 10.0 10.1 10.2 10.3 10.4
## [11] 10.5
```

Rechenoperationen können zwischen (numerischen) Vektoren elementweise ausgeführt werden. Hat ein Vektor weniger Elemente als ein anderer, werden die Elemente vom Beginn des kürzeren Vektors einfach solange wiederholt, bis die Länge der beiden Vektoren gleich ist. Die Länge eines Vektors kann mir der Funktion `length()` ausgelesen werden. Die Summe aller Elemente eines Vektors wird mit `sum()` errechnet. Beispielsweise ergibt mit `x=1:5` und `y = c(10.1, 10.5)` die Summe `x+y` den Vektor `(11.1, 12.5, 13.1, 14.5, 15.1)`. Analog funktioniert `-, *, /`.

c) Gegeben sind die Vektoren

```
x <- 4:2
y <- seq(from = 0.1, to = 0.5, by = 0.1)
```

Erklären Sie, was folgende Ausdrücke ergeben und überprüfen Sie Ihr Ergebnis in R:

```
x + y
x * y
x^3 + 1
2 * x - 3 * y
n <- length(x + y)
sum(x + y)/n
```

Teile oder einzelne Elemente eines Vektors können mit der Angabe der Indexwerte in eckigen Klammern ausgegeben werden. Auch Bedingungen mit Vergleichsoperatoren (z.B. < für kleiner als *oder == für ist gleich) sind möglich in eckigen Klammern. Verknüpfungen zwischen Vergleichen sind mit logisch UND (&) beziehungsweise ODER (|) möglich.*

d) Gegeben sind die Vektoren

```
x <- seq(from = 0, to = 100, by = 2)
y <- 100:1
```

Schreiben Sie die Ergebnisse folgender Ausdrücke auf und überprüfen Sie anschließend Ihr Ergebnis in R:

```
x[3]
y[c(1, 3, 10)]
x[1:4]
x[x > 91]
x[x > 20 & x <= 30]
y[y == 5 | y > 95 | y < 3]
```

Anmerkung: Die Ausgabe von Relationen wie $x < y$ auf Vektoren in R sind Vektoren mit den Ausprägungen `TRUE` beziehungsweise `FALSE`. Diese sogenannten logischen Vektoren können zur Indizierung von Vektoren verwendet werden; Elemente mit einem Index von `TRUE` werden ausgewählt, die mit Wert `FALSE` werden übergangen.

e) Was ergeben folgende Zeilen in R:

```
x <- seq(from = 0.2, to = 2, by = 0.3)
y <- -3:3
x < y
x^2 < x
Index <- x^2 < x
x[Index]
y[Index]
```

Lösungshinweis:

a)
```
x <- c(1, 4, 2, 1.5)  # Anlegen eines metrischen Merkmals x
# mit Ausprägungen für 4 Objekte
x  # Ausgabe

## [1] 1.0 4.0 2.0 1.5

Geschlecht <- c("Mann", "Frau", "Frau", "Frau")
Geschlecht  # nominales Merkmal, auch von 4 Objekten

## [1] "Mann" "Frau" "Frau" "Frau"

z <- c(1, 2, 1, "1")  # z ist für R nominal, da der letzte Wert
# als Zeichenkette eingegeben wurde
```

```
z
```

```
## [1] "1" "2" "1" "1"
```

b)
```
5:9
```

```
## [1] 5 6 7 8 9
```

```
10:1
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

```
seq(from = -0.1, to = 0, by = 0.02)
```

```
## [1] -0.10 -0.08 -0.06 -0.04 -0.02  0.00
```

```
seq(from = 10000, to = 20000, length.out = 5)
```

```
## [1] 10000 12500 15000 17500 20000
```

```
rep(-3:2, 3)
```

```
##  [1] -3 -2 -1  0  1  2 -3 -2 -1  0  1  2 -3 -2 -1  0  1
## [18]  2
```

```
c(5:10, seq(from = 10.1, by = 0.1, to = 10.5))
```

```
##  [1]  5.0  6.0  7.0  8.0  9.0 10.0 10.1 10.2 10.3 10.4
## [11] 10.5
```

c)
```
x <- 4:2
y <- seq(from = 0.1, to = 0.5, by = 0.1)
```

```
x + y
```

```
## [1] 4.1 3.2 2.3 4.4 3.5
```

```
x * y
```

```
## [1] 0.4 0.6 0.6 1.6 1.5
```

```
x^3 + 1
```

```
## [1] 65 28  9
```

```
2 * x - 3 * y
```

```
## [1] 7.7 5.4 3.1 6.8 4.5
```

```
n <- length(x + y)
sum(x + y)/n
```

```
## [1] 3.5
```

d)
```
x <- seq(from = 0, to = 100, by = 2)
y <- 100:1
```

```
x[3]
```

```
## [1] 4
```

```
y[c(1, 3, 10)]
```

```
## [1] 100  98  91
```

```
x[1:4]
```

```
## [1] 0 2 4 6
```

```
x[x > 91]
```

```
## [1]  92  94  96  98 100
```

```
x[x > 20 & x <= 30]
```

```
## [1] 22 24 26 28 30
```

```
y[y == 5 | y > 95 | y < 3]
```

```
## [1] 100  99  98  97  96   5   2   1
```

e)
```
x <- seq(from = 0.2, to = 2, by = 0.3)
y <- -3:3
x < y
```

```
## [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
```

```
x^2 < x
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

```
Index <- x^2 < x
x[Index]
```

```
## [1] 0.2 0.5 0.8
```

```
y[Index]
```

```
## [1] -3 -2 -1
```

Aufgabe 4

Daten mit mehreren Merkmalen als Data Frames

Urlisten mit mehr als einem Merkmal kann man sinnvoll in einer Tabelle abbilden. Die Zeilen entsprechen den Objekten, die Spalten den Merkmalen. In R heißen solche Datenmatrizen `Data Frames`; diese können mit der Funktion `data.frame()` gebildet werden. Ein Beispiel: Es soll ein Data Frame erstellt werden, der aus 6 Studenten, deren Alter, den Matrikelnummern, einer Klausurnote und der Information, ob diese Klausur bestanden wurde besteht:

```
namen <- c("Arno", "Bert", "Carl", "Doro", "Edda", "Fred")
alter <- c(19, 21, 20, 22, 20, 27)
matrnr <- c(101010, 101007, 2e+05, 123456, 654321, 111111)
note <- c(1, 5, 2.3, 2.7, 1.3, 4)
bestanden <- (note < 4.3)
Studenten <- data.frame(Name = namen, Alter = alter, Matrikelnummer = matrnr,
    Note = note, Bestanden = bestanden)
Studenten
```

```
##   Name Alter Matrikelnummer Note Bestanden
## 1 Arno    19         101010  1.0      TRUE
## 2 Bert    21         101007  5.0     FALSE
## 3 Carl    20         200000  2.3      TRUE
## 4 Doro    22         123456  2.7      TRUE
## 5 Edda    20         654321  1.3      TRUE
## 6 Fred    27         111111  4.0      TRUE
```

Um auf einzelne Merkmale bzw. Objekte eines Data Frames zuzugreifen, kann der Zeilen- und Spaltenindex durch ein Komma getrennt in eckigen Klammern angegeben werden. Dabei können auch wie bei Vektoren mehrere Zeilen- bzw. Spaltenindexwerte angegeben werden, bzw. über Auswahloperatoren gebildet werden. Um auf ein Merkmal zuzugreifen, kann der Bezeichner des Merkmals mit einem $-Zeichen an den Bezeichner des dataframes angehängt werden.

a) Was gibt R jeweils nach folgenden Zeilen aus? Überprüfen Sie Ihre Antwort in R.

```
Studenten[1, 3]
Studenten[1:3, c(1, 4)]
Studenten[2, ]
Studenten[, 4]
Studenten$Note
Studenten[, 3:5]
Studenten[note < 4, ]
```

b) Erzeugen Sie einen Data Frame der folgenden 7 Kinder gemäß der Datentabelle

Alter	Geschlecht	Taschengeld (in Euro)	besitzt Fahrrad
6	männlich	12	ja
7	männlich	18	ja
6	weiblich	14	nein
7	weiblich	20	ja
8	männlich	26	ja
7	weiblich	20	ja
8	weiblich	20	nein

c) Generieren Sie in R einen Data Frame des Alters und des Taschengeldes aller Kinder mit höchstens 7 Jahren.

d) Berechnen Sie in R die Summe des Taschengelds aller Fahrradbesitzer.

Lösungshinweis:

a)

```
Studenten[1, 3]  # vom 1. Student aus der Liste den Wert des 3. Merkmals (M.Nr.)
## [1] 101010
Studenten[1:3, c(1, 4)]  # Studenten 1, 2, und 3, davon 1. und 4. Merkmal
##   Name Note
## 1 Arno  1.0
## 2 Bert  5.0
## 3 Carl  2.3
Studenten[2, ]  # vom 2. Student alle Merkmale
##   Name Alter Matrikelnummer Note Bestanden
## 2 Bert    21         101007    5     FALSE
Studenten[, 4]  # alle Studenten, nur das 4. Merkmal
## [1] 1.0 5.0 2.3 2.7 1.3 4.0
Studenten$Note  # dito
## [1] 1.0 5.0 2.3 2.7 1.3 4.0
Studenten[, 3:5]  # Alle Studenten, Merkmale 3, 4, 5
##   Matrikelnummer Note Bestanden
## 1         101010  1.0      TRUE
## 2         101007  5.0     FALSE
## 3         200000  2.3      TRUE
## 4         123456  2.7      TRUE
## 5         654321  1.3      TRUE
## 6         111111  4.0      TRUE
Studenten[note < 2.5, ]  # Alle Studenten mit Note besser 2.5
```

```
##   Name Alter Matrikelnummer Note Bestanden
## 1 Arno    19         101010  1.0      TRUE
## 3 Carl    20         200000  2.3      TRUE
## 5 Edda    20         654321  1.3      TRUE
```

b)
```
Kinder <- data.frame(Alter = c(6, 7, 6, 7, 8, 7, 8), Geschlecht = c("m",
    "m", "w", "w", "m", "w", "w"), Taschengeld = c(12, 18,
    14, 20, 26, 20, 20), Fahrrad = c("ja", "ja", "nein",
    "ja", "ja", "ja", "nein"))
```

c)
```
Kinder[Kinder$Alter <= 7, c("Alter", "Taschengeld")]
```

```
##   Alter Taschengeld
## 1     6          12
## 2     7          18
## 3     6          14
## 4     7          20
## 6     7          20
```

d)
```
sum(Kinder[Kinder$Fahrrad == "ja", "Taschengeld"])
```

```
## [1] 96
```

Aufgabe 5

Datentypen und Umgang mit Data Frames

R kann zwischen metrischen, ordinalen und nominalen Merkmalen unterscheiden. Das Skalenniveau eines Merkmals kann über die Funktion `str()` abgefragt werden. Um explizit ein bestimmtes Skalenniveau abzufragen, können die Funktionen `is.numeric()` für metrisches, `is.ordered()` für ordinales sowie `is.factor()` für nominale Merkmale verwendet werden.
Metrische Merkmale werden typischerweise automatisch von R erkannt, wenn gültige Zahlen eingelesen werden. Falls metrische Ausprägungen als Zeichenketten eingegeben wurden, können sie mittels `as.numeric()` umgewandelt werden.
Nominale Merkmale können mit der Funktion `factor()` angelegt, bzw. mit `as.factor()` umgewandelt werden.
Ordinale Merkmale sind in R auch factors, haben aber zusätzlich eine Rangfolge der Ausprägungen hinterlegt. Zum Anlegen eines solchen Merkmals wird `ordered()`, zum umwandeln `as.ordered()` verwendet.

a) Lesen Sie die Hilfe-Seiten zu `factor()` und und entscheiden Sie dann, was die folgenden Zeilen in R bewirken. Überprüfen Sie Ihre Lösung mit R.

```
Merkmal1 <- c(1, 2, 3, 2, 1, 2, 3)
Merkmal2 <- c("2.1", 1, 3, "-3e2", 2000, "-0.2", "-.3")
Merkmal3 <- c("gut", "gut", "katastrophal", "mittel", "katastrophal",
    "gut", "mittel")
is.numeric(Merkmal1)
is.numeric(Merkmal2)
Merkmal2 <- as.numeric(Merkmal2)
Merkmal2
is.numeric(Merkmal2)
is.factor(Merkmal3)
Merkmal3 <- factor(Merkmal3)
is.factor(Merkmal3)
Merkmal3
Merkmal3 <- ordered(Merkmal3)
Merkmal3
Merkmal3 <- ordered(Merkmal3, levels = c("katastrophal",
    "mittel", "gut"))
Merkmal3
```

b) Gegeben ist der folgende Data Frame „`Studenten`“:

```
namen <- c("Arno", "Bert", "Carl", "Doro", "Edda", "Franz")
geschlecht <- c(0, 0, 0, 1, 1, 0)
alter <- c(19, 21, 20, 22, 20, 27)
note <- c("sehr gut", "durchgefallen", "gut", "gut", "sehr gut",
    "gut")
Studenten <- data.frame(Name = namen, Alter = alter, Geschlecht = geschlecht,
    Note = note)
```

Kodieren Sie in R den Datentyp der jeweiligen Merkmale richtig (metrisch, nominal, ordinal).

c) Transformieren Sie die Ausprägungen des Merkmals `Geschlecht` von den Originalwerten `0, 1` in die Werte `Mann` bzw. `Frau`. Benutzen Sie dazu den `labels` Parameter der Funktion `factor()`.

In einem Data Frame kann die Anzahl der Objekte mit `nrow()`, die Anzahl der Merkmale mit `ncol()` ausgegeben werden. Dient ein Merkmal lediglich als eindeutiger Bezeichner für die Objekte des Data Frames (z.B. Name oder Personalnummer) werden diese Bezeichner den Objekten üblicherweise mit der Funktion `row.names()` über die Zuweisung `row.names() <- bezeichner` zugewiesen. Möchte man bestimmte Merkmale eines Data Frames löschen, kann man ihnen `NULL` zuweisen. Sortiert wird ein Data Frame nach den Ausprägungen eines bestimmten Merkmals über die Funktion `order()`. Umgedreht wird die Reihenfolge im Aufruf von `order()` mit dem Parameter `decreasing=TRUE`.

d) Wie ist das Ergebnis folgender Anweisungen? Überprüfen Sie Ihr Ergebnis mit R.

```
nrow(Studenten)
ncol(Studenten)
row.names(Studenten) <- Studenten$Name
Studenten
Studenten$Name <- NULL
Studenten
Studenten[order(Studenten$Alter), ]
Studenten[order(Studenten$Note), ]
Studenten[order(Studenten$Note, decreasing = TRUE), ]
```

Weitere Objekte werden einem Data Frame mit `rbind()` hinzugefügt, mit `cbind()` erweitert man Data Frames um weitere Merkmale.

e) Erweitern Sie den Datensatz `Studenten` um die 26-jährige Studentin Gerlinde, die in der Prüfung sehr gut abgeschnitten hat.

Lösungshinweis:

a)
```
Merkmal1 <- c(1, 2, 3, 2, 1, 2, 3)
Merkmal2 <- c("2.1", 1, 3, "-3e2", 2000, "-0.2", "-.3")
Merkmal3 <- c("gut", "gut", "katastrophal", "mittel", "katastrophal",
    "gut", "mittel")
is.numeric(Merkmal1)

## [1] TRUE

is.numeric(Merkmal2)

## [1] FALSE

Merkmal2 <- as.numeric(Merkmal2)
Merkmal2

## [1]    2.1    1.0    3.0 -300.0 2000.0   -0.2   -0.3
```

```r
is.numeric(Merkmal2)
```

```
## [1] TRUE
```

```r
is.factor(Merkmal3)
```

```
## [1] FALSE
```

```r
Merkmal3 <- factor(Merkmal3)
is.factor(Merkmal3)
```

```
## [1] TRUE
```

```r
Merkmal3
```

```
## [1] gut          gut          katastrophal mittel
## [5] katastrophal gut          mittel
## Levels: gut katastrophal mittel
```

```r
Merkmal3 <- ordered(Merkmal3)
Merkmal3
```

```
## [1] gut          gut          katastrophal mittel
## [5] katastrophal gut          mittel
## Levels: gut < katastrophal < mittel
```

```r
Merkmal3 <- ordered(Merkmal3, levels = c("katastrophal",
    "mittel", "gut"))
Merkmal3
```

```
## [1] gut          gut          katastrophal mittel
## [5] katastrophal gut          mittel
## Levels: katastrophal < mittel < gut
```

b)

```r
str(Studenten)  # Name und Alter sind OK, Geschlecht und Note nicht
```

```
## 'data.frame': 6 obs. of  4 variables:
##  $ Name      : Factor w/ 6 levels "Arno","Bert",..: 1 2 3 4 5 6
##  $ Alter     : num  19 21 20 22 20 27
##  $ Geschlecht: num  0 0 0 1 1 0
##  $ Note      : Factor w/ 3 levels "durchgefallen",..: 3 1 2 2 3 2
```

```r
Studenten$Geschlecht <- factor(Studenten$Geschlecht)
Studenten$Note <- ordered(Studenten$Note, levels = c("durchgefallen",
    "gut", "sehr gut"))
Studenten$Note  # jetzt stimmt die Reihenfolge
```

```
## [1] sehr gut      durchgefallen gut
## [4] gut           sehr gut      gut
## Levels: durchgefallen < gut < sehr gut
```

```r
str(Studenten)
```

```
## 'data.frame': 6 obs. of  4 variables:
##  $ Name      : Factor w/ 6 levels "Arno","Bert",..: 1 2 3 4 5 6
##  $ Alter     : num  19 21 20 22 20 27
##  $ Geschlecht: Factor w/ 2 levels "0","1": 1 1 1 2 2 1
##  $ Note      : Ord.factor w/ 3 levels "durchgefallen"<..: 3 1 2 2 3 2
```

c)
```
Studenten
```
```
##    Name Alter Geschlecht          Note
## 1  Arno    19          0      sehr gut
## 2  Bert    21          0 durchgefallen
## 3  Carl    20          0           gut
## 4  Doro    22          1           gut
## 5  Edda    20          1      sehr gut
## 6 Franz    27          0           gut
```
```
Studenten$Geschlecht <- factor(Studenten$Geschlecht, labels = c("Mann",
    "Frau"))
Studenten
```
```
##    Name Alter Geschlecht          Note
## 1  Arno    19       Mann      sehr gut
## 2  Bert    21       Mann durchgefallen
## 3  Carl    20       Mann           gut
## 4  Doro    22       Frau           gut
## 5  Edda    20       Frau      sehr gut
## 6 Franz    27       Mann           gut
```

d)
```
nrow(Studenten)
```
```
## [1] 6
```
```
ncol(Studenten)
```
```
## [1] 4
```
```
row.names(Studenten) <- Studenten$Name
Studenten
```
```
##        Name Alter Geschlecht          Note
## Arno   Arno    19       Mann      sehr gut
## Bert   Bert    21       Mann durchgefallen
## Carl   Carl    20       Mann           gut
## Doro   Doro    22       Frau           gut
## Edda   Edda    20       Frau      sehr gut
## Franz Franz    27       Mann           gut
```
```
Studenten$Name <- NULL
Studenten
```
```
##       Alter Geschlecht          Note
## Arno     19       Mann      sehr gut
## Bert     21       Mann durchgefallen
## Carl     20       Mann           gut
## Doro     22       Frau           gut
## Edda     20       Frau      sehr gut
## Franz    27       Mann           gut
```
```
Studenten[order(Studenten$Alter), ]
```
```
##       Alter Geschlecht          Note
## Arno     19       Mann      sehr gut
## Carl     20       Mann           gut
## Edda     20       Frau      sehr gut
```

```
## Bert     21       Mann durchgefallen
## Doro     22       Frau           gut
## Franz    27       Mann           gut
```

```
Studenten[order(Studenten$Note), ]
```

```
##       Alter Geschlecht          Note
## Bert     21       Mann durchgefallen
## Carl     20       Mann           gut
## Doro     22       Frau           gut
## Franz    27       Mann           gut
## Arno     19       Mann      sehr gut
## Edda     20       Frau      sehr gut
```

```
Studenten[order(Studenten$Note, decreasing = TRUE), ]
```

```
##       Alter Geschlecht          Note
## Arno     19       Mann      sehr gut
## Edda     20       Frau      sehr gut
## Carl     20       Mann           gut
## Doro     22       Frau           gut
## Franz    27       Mann           gut
## Bert     21       Mann durchgefallen
```

e)
```
Studenten <- rbind(Studenten, data.frame(Alter = 26, Geschlecht = "Frau",
    Note = "sehr gut", row.names = "Gerlinde"))
Studenten
```

```
##          Alter Geschlecht          Note
## Arno        19       Mann      sehr gut
## Bert        21       Mann durchgefallen
## Carl        20       Mann           gut
## Doro        22       Frau           gut
## Edda        20       Frau      sehr gut
## Franz       27       Mann           gut
## Gerlinde    26       Frau      sehr gut
```

Aufgabe 6

Einlesen von Daten aus Textdateien

Wenn Daten als Tabelle in einer Textdatei vorliegen können sie leicht in R eingelesen werden; dazu ist es gut, wenn in der Tabelle die Zeilen den Objekten und die Spalten den Merkmalen entsprechen. Über die graphische Benutzeroberfläche in RStudio funktioniert das Importieren einfach über den Knopf „`Import Dataset`“ im Reiter `Environment`. Dabei kann eine lokale Datei ausgewählt werden oder eine Datei, die über das Netzwerk via URL erreichbar ist. Klickt man auf `Import` werden Die Daten als Data Frame unter dem angegebenen Bezeichner zum Beispiel mit der Funktion `read.csv()` eingelesen.

a) Importieren Sie mit der graphischen Benutzeroberfläche von RStudio die Umfragedaten aus der Vorlesung von der Adresse `http://goo.gl/Mg6kmj` und speichern Sie den Data Frame unter der Bezeichnung „`Umfrage`“.

b) Importieren Sie die Daten nochmals, diesmal aber indem Sie eine lokale Kopie der csv-Datei auf Ihrer Festplatte anlegen und das Einlesen über die Funktion `read.csv()` bewerkstelligen.

Die Funktion `head()` zeigt die ersten Objekte eines Data Frames an, mit `summary()` bekommt man einen Überblick über die Verteilung der Ausprägungen in den einzelnen Merkmalen.

c) Verschaffen Sie sich einen Überblick über die Daten, indem Sie sich die Struktur des Data Frames mit `str()`, die ersten Objekte mit `head()` und die Verteilung der Ausprägungen in den einzelnen Merkmalen mit `summary()` ansehen.

Lösungshinweis:

a)
```
# Klicken auf Import Dataset, Angabe der URL, Ändern des
# Bezeichners Ändern des Dezimaltrenners auf ','
Umfrage <- read.csv("http://goo.gl/Mg6kmj", sep = ";", dec = ",")
```

b)
```
Umfrage <- read.csv("c:/Verzeichnis/Umfrage_HSA_2015_03.csv",
    sep = ";", dec = ",")
```

c)
```
options(width = 70)
str(Umfrage)
```
```
## 'data.frame': 205 obs. of  14 variables:
##  $ Alter      : int  21 20 19 20 20 24 20 27 23 21 ...
##  $ Groesse    : int  173 164 172 168 169 185 170 165 184 178 ...
##  $ Geschlecht : Factor w/ 2 levels "Frau","Mann": 1 1 1 1 1 2 1 1 2 2 ...
##  $ AlterV     : int  54 57 49 45 43 54 49 53 52 55 ...
##  $ AlterM     : int  51 57 58 49 42 52 53 53 48 55 ...
```

```
##  $ GroesseV   : int  187 172 193 185 182 179 182 175 182 180 ...
##  $ GroesseM   : int  170 166 162 164 164 163 172 165 175 168 ...
##  $ Geschwister: int  1 0 3 3 5 2 2 1 2 1 ...
##  $ Farbe      : Factor w/ 6 levels "blau","gelb",..: 6 6 4 4 6 4 3 6 4 6 ...
##  $ AusgKomm   : num  156 450 240 35.8 450 250 100 300 450 1300 ...
##  $ AnzSchuhe  : int  17 22 15 15 22 8 20 10 3 7 ...
##  $ AusgSchuhe : int  50 500 400 100 450 90 250 200 300 200 ...
##  $ NoteMathe  : num  5 1.7 2.3 2 2 4 NA 4 2.7 2.7 ...
##  $ MatheZufr  : Factor w/ 5 levels "","geht so","nicht",..: 3 4 4 4 4 2 3 3 5 5 ...
```

```
head(Umfrage)
```

```
##   Alter Groesse Geschlecht AlterV AlterM GroesseV GroesseM
## 1    21     173       Frau     54     51      187      170
## 2    20     164       Frau     57     57      172      166
## 3    19     172       Frau     49     58      193      162
## 4    20     168       Frau     45     49      185      164
## 5    20     169       Frau     43     42      182      164
## 6    24     185       Mann     54     52      179      163
##   Geschwister   Farbe AusgKomm AnzSchuhe AusgSchuhe NoteMathe
## 1           1   weiss    156.0        17         50       5.0
## 2           0   weiss    450.0        22        500       1.7
## 3           3 schwarz    240.0        15        400       2.3
## 4           3 schwarz     35.8        15        100       2.0
## 5           5   weiss    450.0        22        450       2.0
## 6           2 schwarz    250.0         8         90       4.0
##   MatheZufr
## 1     nicht
## 2      sehr
## 3      sehr
## 4      sehr
## 5      sehr
## 6   geht so
```

```
summary(Umfrage)
```

```
##      Alter          Groesse      Geschlecht     AlterV
##  Min.   :18.00   Min.   :156.0   Frau:134   Min.   :38.00
##  1st Qu.:20.00   1st Qu.:166.0   Mann: 71   1st Qu.:50.00
##  Median :21.00   Median :170.0              Median :54.00
##  Mean   :22.22   Mean   :172.6              Mean   :53.95
##  3rd Qu.:23.00   3rd Qu.:179.0              3rd Qu.:57.00
##  Max.   :36.00   Max.   :197.0              Max.   :77.00
##
##      AlterM        GroesseV        GroesseM      Geschwister
##  Min.   :37.0   Min.   :160.0   Min.   :150.0   Min.   :0.000
##  1st Qu.:48.0   1st Qu.:175.0   1st Qu.:163.0   1st Qu.:1.000
##  Median :51.0   Median :180.0   Median :166.0   Median :1.000
##  Mean   :51.5   Mean   :179.7   Mean   :166.8   Mean   :1.473
##  3rd Qu.:54.0   3rd Qu.:183.0   3rd Qu.:170.0   3rd Qu.:2.000
##  Max.   :68.0   Max.   :202.0   Max.   :192.0   Max.   :9.000
##
##      Farbe        AusgKomm         AnzSchuhe       AusgSchuhe
##  blau   :11   Min.   :  30.0   Min.   : 2.00   Min.   :   0.0
```

```
##  gelb   : 4   1st Qu.: 250.0   1st Qu.:10.00   1st Qu.: 150.0
##  rot    :13   Median : 360.0   Median :20.00   Median : 250.0
##  schwarz:97   Mean   : 429.4   Mean   :21.58   Mean   : 296.6
##  silber :17   3rd Qu.: 570.0   3rd Qu.:30.00   3rd Qu.: 400.0
##  weiss  :63   Max.   :1868.0   Max.   :80.00   Max.   :2000.0
##
##    NoteMathe         MatheZufr
##  Min.   :1.000            :21
##  1st Qu.:2.500   geht so  :47
##  Median :3.300   nicht    :68
##  Mean   :3.272   sehr     :26
##  3rd Qu.:4.000   zufrieden:43
##  Max.   :5.000
##  NA's   :34
```

Aufgabe 7

R-Skripten: Führen eines Logbuches der eigenen Analysen

In vielen Fällen besteht eine statistische Untersuchung aus mehr als einem Schritt. Meistens werden Daten eingelesen, bereinigt, aufbereitet, verdichtet, graphisch dargestellt usw. Um diesen Ablauf zu dokumentieren kann man eine Textdatei mit der Endung `.R`, ein sogenanntes `R`-Skript erstellen und alle Kommandos dort ablegen, mit Kommentaren dokumentieren und für spätere Wiederverwendung abspeichern.

a) Legen Sie eine `.R`-Datei mit dem Bezeichner `SS2015-Statistik-Uebung.R` an (in RStudio über `File -> New -> R-Script`) und schreiben Sie in diese Datei in die ersten Zeilen als Kommentar (`#-Zeichen voranstellen`) Ihren Namen, das Datum sowie eine Anmerkung, dass diese Datei alle `R`-Lösungen der Übungsaufgaben enthält.

b) Fügen Sie für jede bis hierher bearbeitete Aufgabe nach einem entsprechenden Kommentar den jeweiligen R-Code in diese Datei ein und schreiben Sie zu möglichst vielen Zeilen einen Kommentar, in dem Sie eine Anmerkung schreiben was die Zeile bewirkt.

Um eine Zeile aus einem `R`-Skript in `R` auszuführen, kann der Cursor in die entsprechende Zeile platziert werden; durch die Tastenkombination `Strg-Enter` (auf englischsprachigen Tastaturen `Ctrl-Enter`) wird die Zeile in die Console kopiert und ausgeführt; danach springt der Cursor in die nächste Zeile des Skripts. Wiederholt man das mehrmals, werden der Reihe nach alle Zeilen ausgeführt (Kommentarzeilen werden übergangen). Möchte man mehr als eine Zeile ausführen, kann man den entsprechenden Teil des Skripts mit der Maus markieren und mit `Strg-Enter` ausführen.

c) Führen Sie die Befehle der ersten R-Aufgabe zunächst zeilenweise aus und beobachten Sie die Ein- und Ausgaben in der Console, danach markieren Sie die komplette Aufgabe und wiederholen die Ausführung.

Lösungshinweis:

a)
```
# ------------------------------------------------------ 12.3.2015,
# Max Maier R-Skript zur Statistik Übung im SS 2015
# ------------------------------------------------------
```

b)
```
# ------------------------------------------------------ 12.3.2015,
# Max Maier R-Skript zur Statistik Übung im SS 2015
# ------------------------------------------------------
# Aufgabe 1
2 + 3 * 4  # hier gilt Punkt vor Strich
(2 + 3) * 4  # Klammer zuerst
0.2 * 3 - 1.1
# 0,2 * 3 # Fehler, ',' wird nicht als Dezimalkomma akzeptiert
```

```
2^3^2  # entspricht 2^(3^2)
(2^3)^2
exp(1)  # das ist e^1
log(exp(1))  # log() entspricht dem ln; e^x und ln heben sich auf
sqrt(16)  # Quadratwurzel
16^(1/2)  # auch QW
# Sqrt(16) # Fehler, sqrt() schreibt man mit kleinem 's'
# Aufgabe 2 ...
```

c)

```
# ------------------------------------------------------- 12.3.2015,
# Max Maier R-Skript zur Statistik Übung im SS 2015
# -------------------------------------------------------
# Aufgabe 1
2 + 3 * 4  # hier gilt Punkt vor Strich
```

```
## [1] 14
```

```
(2 + 3) * 4  # Klammer zuerst
```

```
## [1] 20
```

```
0.2 * 3 - 1.1
```

```
## [1] -0.5
```

```
# 0,2 * 3 # Fehler, ',' wird nicht als Dezimalkomma akzeptiert
2^3^2  # entspricht 2^(3^2)
```

```
## [1] 512
```

```
(2^3)^2
```

```
## [1] 64
```

```
exp(1)  # das ist e^1
```

```
## [1] 2.718282
```

```
log(exp(1))  # log() entspricht dem ln; e^x und ln heben sich auf
```

```
## [1] 1
```

```
sqrt(16)  # Quadratwurzel
```

```
## [1] 4
```

```
16^(1/2)  # auch QW
```

```
## [1] 4
```

```
# Sqrt(16) # Fehler, sqrt() schreibt man mit kleinem 's'
# Aufgabe 2 ...
```

Aufgabe 8

Häufigkeiten in R und Umgang mit fehlenden Werten

Häufigkeitsauszählungen können in R mit `table()` erstellt werden. `cumsum()` bildet kumulierte Summen. `mean()` berechnet das arithmetische Mittel, `median()` den Median

a) Lesen Sie von `http://goo.gl/dZkICg` die Daten der Vorlesungsumfrage ein und bilden Sie jeweils eine Tabelle der absoluten und relativen sowie der absoluten kumulierten und relativen kumulierten Häufigkeiten des Merkmals Alter.

b) Wandeln Sie das Merkmal `MatheZufr` (Antwort auf „Waren Sie zufrieden mit Ihrer Leistung in der Matheklausur) in ein ordinales Merkmal mit sinnvoller Reihenfolge um.

c) Berechnen Sie den Median und das arithmetische Mittel aller metrischen Merkmale der eingelesenen Daten.

Ausprägungen fehlender Werte werden in R mit `NA` (Not Available) dargestellt, Objekte mit fehlenden Werten können mittels `na.omit()` gelöscht werden. Die Funktion `sort()` gibt einen metrisch oder ordinal skalierten Vektor in aufsteigender Reihenfolge zurück.

d) Für ordinale Merkmale ist der Median zwar definiert, mit der eingebauten Funktion in `R` erhält man aber eine Fehlermeldung. Lösen Sie das Problem und berechnen Sie den Median aller (vorhandenen) Ausprägungen des Merkmals `MatheZufr`.

Lösungshinweis:

a)
```
Umfrage <- read.csv("http://goo.gl/Mg6kmj", sep = ";", dec = ",")
T <- table(Umfrage$Alter)  # absolute Häufigkeiten
n <- length(Umfrage$Alter)  # Anzahl der Objekte
T
```
```
##
## 18 19 20 21 22 23 24 25 26 27 28 29 31 32 33 34 36
## 10 27 39 29 26 23 14  6  7  6  7  4  1  3  1  1  1
```
```
cumsum(T)  # kumuliert
```
```
##  18  19  20  21  22  23  24  25  26  27  28  29  31  32  33  34  36
##  10  37  76 105 131 154 168 174 181 187 194 198 199 202 203 204 205
```
```
T/n  # relative Häufigkeiten
```
```
##
##          18          19          20          21          22
## 0.048780488 0.131707317 0.190243902 0.141463415 0.126829268
##          23          24          25          26          27
## 0.112195122 0.068292683 0.029268293 0.034146341 0.029268293
##          28          29          31          32          33
```

```
## 0.034146341 0.019512195 0.004878049 0.014634146 0.004878049
##          34          36
## 0.004878049 0.004878049
```

```
round(T/n, 3)  # auf drei Kommastellen gerundet
```

```
##
##    18    19    20    21    22    23    24    25    26    27    28
## 0.049 0.132 0.190 0.141 0.127 0.112 0.068 0.029 0.034 0.029 0.034
##    29    31    32    33    34    36
## 0.020 0.005 0.015 0.005 0.005 0.005
```

```
round(cumsum(T)/n, 3)  # kumulierte rel. Häufigkeiten
```

```
##    18    19    20    21    22    23    24    25    26    27    28
## 0.049 0.180 0.371 0.512 0.639 0.751 0.820 0.849 0.883 0.912 0.946
##    29    31    32    33    34    36
## 0.966 0.971 0.985 0.990 0.995 1.000
```

```
cbind(as.data.frame(T), as.data.frame(cumsum(T)), as.data.frame(T/n), as.data.frame(cumsum(T)/
```

```
##    Var1 Freq cumsum(T) Var1        Freq cumsum(T)/n
## 18   18   10         10   18 0.048780488  0.04878049
## 19   19   27         37   19 0.131707317  0.18048780
## 20   20   39         76   20 0.190243902  0.37073171
## 21   21   29        105   21 0.141463415  0.51219512
## 22   22   26        131   22 0.126829268  0.63902439
## 23   23   23        154   23 0.112195122  0.75121951
## 24   24   14        168   24 0.068292683  0.81951220
## 25   25    6        174   25 0.029268293  0.84878049
## 26   26    7        181   26 0.034146341  0.88292683
## 27   27    6        187   27 0.029268293  0.91219512
## 28   28    7        194   28 0.034146341  0.94634146
## 29   29    4        198   29 0.019512195  0.96585366
## 31   31    1        199   31 0.004878049  0.97073171
## 32   32    3        202   32 0.014634146  0.98536585
## 33   33    1        203   33 0.004878049  0.99024390
## 34   34    1        204   34 0.004878049  0.99512195
## 36   36    1        205   36 0.004878049  1.00000000
```

b)

```
Umfrage$MatheZufr <- ordered(Umfrage$MatheZufr, levels = c("nicht", "geht so",
    "zufrieden", "sehr"))
Mathe.sortiert <- sort(Umfrage$MatheZufr)
n <- length(Mathe.sortiert)
is.integer(n/2)  # FALSE, also n ungerade
```

```
## [1] FALSE
```

```
Mathe.Median <- Mathe.sortiert[(n + 1)/2]
Mathe.Median
```

```
## [1] geht so
## Levels: nicht < geht so < zufrieden < sehr
```