Einführung in R: Erste Schritte

Georg Hoffmann

2022-07-04

Inhalt

Kursvoraussetzungen

Dieser Einführungskurs ist Teil einer Skriptenreihe der Trillium Akademie. Weitere Informationen zum Kurs können Interessierte per Mail unter *georg.hoffmann@trillium.de* erfragen.

Zur Durchführung benötigt man einen PC/Laptop mit Zugang zum Internet, auf dem die beiden kostenlosen Programme *R* und *RStudio* installiert werden können. Bei Computern, die in ein Institutionsnetzwerk eingebunden sind, sind in der Regel Administratorrechte erforderlich (Anfrage bei der IT-Abteilung).

Alle Beispiele in diesem Skript beziehen sich auf das Betriebssystem *Windows* ab Version 10 mit deutscher Tastatur. Für andere Betriebssysteme (*macOS, Linux*) und andere Tastaturen gelten Besonderheiten, die man bei Bedarf im Internet recherchieren kann. Hier sind drei Beispiele:

- Vor der Installation von *R* auf einem Apple-Rechner muss man das (ebenfalls kostenlose) Hilfspaket XQuartz installieren.
- Die Taste *Strg* für “Steuerung” heißt auf der englischen Tastatur *Ctrl* für “Control”.
- Auf der Apple-Tastatur gibt es keine Tilde (~); für dieses in *R* wichtige Zeichen drückt man die Tastenkombination *Option (Alt)* + *n* (Merkhilfe: n wie in Señorita), gefolgt von einem Leerzeichen.

Allen, die wenig IT-Erfahrung haben, wird die Installation von *R* und *RStudio* auf einem Windows-Rechner empfohlen.

Lektion 1

Hintergrund und Installation

Das Softwarewerkzeug *R* ist seit 1995 unter einer General Public License verfügbar. Mit der Gründung des weltweiten Servernetzes Comprehensive R Archive Network CRAN im Jahr 1998 und der gemeinnützigen Stiftung R Foundation im Jahr 2002 begann der Siegeszug dieser Software und Programmiersprache, insbesondere in der Welt der Statistik, der Bioinformatik und den Data Sciences. Heute rangiert *R* konstant unter den zwanzig am häufigsten eingesetzten Programmiersprachen.

Die gewaltige Funktionsbibliothek von *R* wird nicht zentral durch ein Unternehmen, sondern durch Millionen von Nutzern unter der Kontrolle einer professionellen Kernmannschaft befüllt. Dies ist der wohl wichtigste Grund für die weite Verbreitung von *R*. So wie *Wikipedia* kommerzielle Enzyklopädien (*Brockhaus*, *Encyclopedia Britannica*) vom Markt verdrängt hat, stellt *R* für

professionelle Statistikprogramme wie *SPSS* und *SAS* eine ernst zu nehmende Konkurrenz dar, weil es nicht nur kostenlos, sondern auch viel umfangreicher und aktueller ist.

Wenn man in eine Suchmaschine einfach ein R eingebt, erhält man erwartungsgemäß Abermillionen von Websites, in denen ein R vorkommt. Aber auf den vordersten Rängen rangieren mit Sicherheit die Links zur Programmiersprache R und zum Download der Software unter www.r-project.org. Sofort danach folgen viele Tutorials und Youtube-Videos für Einsteiger.

Installation und Test von R

Am schnellsten gelingt die Installation unter Windows folgendermaßen:

1. Download von einem Server in Deutschland aufrufen, z. B. ftp.gwdg.de/pub/misc/cran/ in Göttingen.
2. Klick auf *Download R for Windows* und *Install R for the first time*.
3. Alle Voreinstellungen übernehmen (immer auf *Weiter* oder *OK* klicken).

Auf heutigen Rechnern empfiehlt sich die 64-Bit-Version. Ob man die deutsche oder englische Version wählt, ist Geschmackssache. Im vorliegenden Skript verwende ich die englischen Begriffe.

Wenn alles geklappt hat, kann man das Programm durch Klick auf das R-Icon starten. Ist das Icon nicht zu sehen, dann sucht man im Programmordner unter *R/bin/* nach *RGui.exe*.

Als ersten Funktionstest geben wir beim blinkenden Cursor folgende Zeile ein (ob mit oder ohne Leerzeichen, ist egal):

```
1 : 50
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
```

Nach dem Drücken auf *Enter* erhält man die Zahlen 1 bis 50 (der Doppelpunkt steht in R für *bis*). Am Zeilenanfang steht jeweils zur besseren Orientierung die laufende Nummer. Wer gleich einmal ein bisschen spielen will, erhält mit *log(1:10)* die natürlichen Logarithmen der Zahlen von 1 bis 10 und mit *boxplot(200:300)* eine sog. Kastengrafik (engl. *boxplot*).

Zum Beenden kann man einfach das x rechts oben anklicken. Die Frage nach dem Speichern beantworten wir mit Nein.

Installation und Test von RStudio

Die Benutzeroberfläche von *R* reicht für einfache Befehle aus, ist aber für größere Aufgaben zu spartanisch. Deshalb installieren wir nun auch noch die integrierte Entwicklungsumgebung RStudio, die seit 2011 verfügbar ist. Die URL heißt www.rstudio.com. Hier wählen wir die kostenlose Basisversion RStudio Desktop für unser Betriebssystem aus und akzeptieren wieder alle Voreinstellungen mit *Weiter* und *OK*.

Auch für *RStudio* findet sich nach erfolgreicher Installation ein Icon auf dem Bildschirm, mit dem man das Programm starten kann. Wenn das Icon nicht zu sehen ist, kann man im Programmordner unter *RStudio/bin/* nach *rstudio.exe* suchen. *RStudio* greift im Hintergrund auf *R* zu, ohne dass man *R* dafür eigens öffnen muss.

Die Benutzeroberfläche von *RStudio* ist deutlich reichhaltiger als die von R. Sie besteht aus einer Menüleiste (*File, Edit* usw.) und vier Fenstern. Falls nur drei Fenster zu sehen sind (ein großes links, zwei kleinere rechts), muss man das linke Fenster halb groß machen. Das linke obere Fenster heißt **Editorfenster** und trägt den vorläufigen Namen *Untitled1*. Hier kann man R-Befehle eingeben und laufen lassen.

Zum Testen geben wir links oben wieder *1 : 50* ein. Mit der Tastenkombination *Strg* + *Enter* wird dieser Befehl in das linke untere Fenster, die sog. **Konsole** kopiert und ausgeführt. Wir erhalten wie schon beim obigen Test von *R* die Zahlenreihe 1 bis 50.

Als nächstes geben wir folgende drei Zeilen ein und drücken am Zeilenende nur *Enter* (ohne *Strg*):

```
a = 5

b = 10

summe = a + b
```

Man kann nun alle drei Zeilen markieren und wahlweise wieder *Strg* + *Enter* drücken oder auf *Run* (direkt über dem Editor-Fenster) klicken. Nun erscheinen im Fenster rechts oben die Namen *a, b* und *summe* der soeben erzeugten drei Variablen mit den Werten 5, 10 und 15.

Hinweis: Es gibt in R verschiedene Methoden, um einer Variablen Werte zuzuweisen; wir verwenden hier das Gleichheitszeichen, weil es besonders intuitiv ist. Weitere Möglichkeiten behandeln wir in einem späteren Skript.

Schließlich geben wir im Editorfenster eine weitere Zeile *boxplot(200 : 300)* ein und lassen sie laufen. Nun erscheint im Fenster rechts unten eine Kastengrafik. Durch Klick auf *Export* (direkt über der Grafik) kann man das Bild in verschiedenen Formaten (png, jpg, pdf usw.) speichern – oder auch in die Zwischenablage kopieren (*Copy to Clipboard*), um es beispielsweise in eine Word- oder Powerpoint-Datei einzufügen.

Bevor wir diese Lektion beenden, wollen wir unser erstes R-Programm speichern. Man wählt dazu im Menü unter *File* den Punkt *Save* oder drückt *Strg* + *s*. Die Frage nach dem Dateiformat beantworten wir mit *UTF-8*, damit deutsche Umlaute korrekt angezeigt werden. Als Dateinamen geben wir z. B. *Test* ein; das Programm wird dann als *Test.R* gespeichert, und dieser Name erscheint auch anstelle von *Untitled1* auf dem Reiter des Editorfensters. Nun kann man *RStudio* mit dem x rechts oben schließen und die Frage nach dem Speichern wieder mit Nein beantworten.

Hinweis: Wenn man nachträglich feststellt, dass Umlaute nicht korrekt dargestellt werden, weil man das Format *UTF-8* nicht eingestellt hat, kann man dies jederzeit nachholen, indem man im Menü unter *File* den Punkt *Save with Encoding* wählt.

Fragen und Aufgaben zu Lektion 1

1. Wodurch unterscheidet sich R grundsätzlich von kommerziellen Statistikpaketen wie zum Beispiel SPSS?
2. Wo findet man unter Windows die ausführbaren Programm *R* und *RStudio*, wenn man die Icons versehentlich vom Bildschirm gelöscht hat?
3. Nenne für jedes der vier Fenster von *RStudio* eine typische Verwendung.
4. Starte *RStudio* neu, indem du die gespeicherte Datei *Test.R* doppelklickst. Füge weitere Programmzeilen hinzu (zum Beispiel für das Produkt der Variablen a und b) und speichere das Programm erneut ab.

Lektion 2

Rechnen auf der R-Konsole

Für einfache Aufgaben, die nicht gespeichert werden sollen, genügt die R-Konsole, also das Fenster links unten. Diese Oberfläche erhält man auch, wenn man statt *RStudio* nur *R* öffnet, aber es empfiehlt sich, bevorzugt mit *RStudio* zu arbeiten, da diese Oberfläche mehr Komfort bietet.

Beim Start wird auf der Konsole ein Text angezeigt, den wir in aller Regel einfach löschen (*Edit>>Clear Console* oder *Strg* + *L* oder Klick auf die kleine Bürste rechts über der Konsole). Aus dem Text geht unter anderem hervor, welche R-Version installiert ist. Das kann wichtig sein,

wenn ein R-Programm eine bestimmte Version voraussetzt und gegebenenfalls ein Update benötigt. Um eine Anleitung für Updates zu erhalten, sucht man am besten im Internet nach “R Update”. Hier ist die Kurzversion für diejenigen, die mit R schon etwas vertraut sind:

```
if(!require(installr)) {install.packages("installr")}

library(installr)

updateR()
```

Die Konsole ist ein einfacher Kommandozeilen-Editor, der in erster Linie mit der Tastatur und nicht mit der Maus bedient wird. Für Ungeübte ist der Gebrauch gewöhnungsbedürftig, aber wenn man das Prinzip einmal verstanden hat, wird man die schnelle und einfache Handhabung zu schätzen wissen: Man tippt eine Befehlszeile wie z. B. *boxplot(1:100)* ein, drückt *Enter* und erhält sofort das Ergebnis.

In dieser Lektion wollen wir die Konsole für einfache Berechnungen nutzen und bei dieser Gelegenheit den Gebrauch der Rechenoperatoren von R kennenlernen. Diese sind wie in praktisch allen Programmiersprachen Plus, Minus, Stern und Schrägstrich (*Backslash*) für die Grundrechenarten, das Hütchen ^ für die Potenzrechnung und runde Klammern für die Zusammenfassung von Rechenausdrücken.Das Hütchen (*accent circonflex*) erhält man, indem man die Taste links oben unter *Esc* und anschließend die Leertaste drückt.

Hier sind einige Beispiele zum Ausprobieren:

- 2 + 5
- 2 - 5
- 2 * 5
- 2 / 5
- 2 ^ 5
- (2 + 5) ^ 2

Die Ergebnisse werden unmittelbar unter der Eingabe ausgegeben. Davor steht die Zeilennummer [1], weil es für jede Rechenoperation nur eine einzige Ausgabezeile gibt. Eine mehrzeilige Ausgabe wäre zum Beispiel:

```
(1 : 20) ^ 2 / 7

##  [1]  0.1428571  0.5714286  1.2857143  2.2857143  3.5714286  5.1428571

##  [7]  7.0000000  9.1428571 11.5714286 14.2857143 17.2857143 20.5714286

## [13] 24.1428571 28.0000000 32.1428571 36.5714286 41.2857143 46.2857143

## [19] 51.5714286 57.1428571
```

Man beachte noch einmal den Unterschied zwischen Schrägstrich und Doppelpunkt: Das Zeichen : steht in *R* nicht für *dividiert durch*, sondern für *bis*. Hier wurden die Zahlen 1 bis 20 quadriert und die Ergebnisse durch 7 dividiert. Wie man an den Ergebnissen sieht, muss man in R anstelle des deutschen Dezimalkommas immer den Dezimalpunkt verwenden.

Komplexere Berechnungen führt man mithilfe von Funktionen aus. Dazu tippt man den Funktionsnamen ein, gefolgt von runden Klammern, in denen steht, worauf die Funktion angewendet werden soll. Wir schreiben zum Beispiel für die Wurzel aus 10:

```
sqrt(10)

## [1] 3.162278
```

Die meisten Funktionsnamen für Rechenoperationen sind selbsterklärend:

Rechenoperation	R-Funktion	Beispiel
Wurzel	sqrt	sqrt(4)
Mittelwert	mean	mean(1:10)
Median	median	median(1:10)
natürlicher Logarithmus	log	log(10)
dekadischer Logarithmus	log10	log10(10)
Exponentialfunktion	exp	exp(1)
Sinus	sin	sin(pi/2)

Man muss das aber nicht auswendig lernen, sondern kann die gewünschte Rechenoperation im Internet suchen, indem man davor den Buchstaben r eintippt, also z. B. r wurzel ziehen. Hat man den richtigen Begriff gefunden, kann man sich zusätzlich spezifische Informationen von R ausgeben lassen, indem man auf der Konsole ein Fragezeichen und den Funktionsnamen eingibt, also z. B. *?sqrt*.

Und zuletzt kommen hier noch zwei Tricks, um sich auf der Konsole Tipparbeit zu sparen:

- Man kann den letzten eingebenen Befehl wiederholen, indem man auf die Pfeiltaste nach oben (*PageUp*) drückt. Das ist bei längeren Ausdrücken nützlich, die man modifizieren möchte (z. B. wenn man sich vertippt hat).
- In *RStudio* erhält man nach dem Eintippen der ersten drei Buchstaben eines Funktionsnamens passende Vorschläge, aus denen man mit der TAB-Taste oder Doppelklick den richtigen auswählen kann. Diese Möglichkeit sollte man grundsätzlich nutzen, um sich vor Tippfehlern zu schützen.

Fragen und Aufgaben zu Lektion 2

1. Wofür eignet sich die Konsole von *R* im Vergleich zum Editorfenster besonders gut?
2. Wieviele verschiedene Möglichkeiten gibt es, um den Inhalt der Konsole zu löschen?
3. Berechne deinen Body Mass Index nach der Formel Gewicht (in kg) dividiert durch Größe (in m) zum Quadrat.
4. Berechne die dekadischen Logarithmen der Zahlen 5 bis 10.
5. Gib die geraden Zahlen von 2 bis 20 auf der Konsole aus.
6. Warum gibt die Funktion *sqrt(2,7)* eine Fehlermeldung aus?
7. Wie heißt die R-Funktion zur Berechnung der Standardabweichung?
8. Berechne Mittelwert und Standardabweichung der Zahlenreihe 7 bis 17.

Lektion 3

Daten erzeugen und verarbeiten

Statistik ist die Lehre vom Umgang mit quantitativen Daten, also mit zählbaren oder messbaren Beobachtungen. In dieser Lektion lernen wir, wie man Daten mithilfe von R-Funktionen erzeugt und manipuliert, speichert und wieder einliest.

Am besten legt man für diese Übung ein neues R-Skript an (Menü *File* oder Tastenkombination *Strg* + *Umschalt* + *n*).

Datenreihen (Vektoren)

Die einfachste Form, quantitative Daten für statistische Auswertungen zu generieren, kennen wir bereits: Mithilfe des Doppelpunkts haben wir fortlaufende Reihen ganzer Zahlen erzeugt (z. B. 1 : 100). Diese kann man durch Rechenformeln auf vielfältige Weise weiterverarbeiten. Zum Beispiel erhält man durch Multiplikation mit 2 lauter gerade Zahlen.

Solche systematischen Zahlensequenzen kann man komfortabler und flexibler mit der Funktion *seq* erzeugen. Für die ungeraden Zahlen von 1 bis 11 geben wir beispielsweise den folgenden Code in das Editorfenster ein:

```
seq(from = 1, to = 11, by = 2)
## [1]  1  3  5  7  9 11
```

Um beliebige Daten wie zum Beispiel die Ergebnisse einer Messreihe aneinander zu reihen, gibt es in R die Funktion *c*() für *combine*.

```
Größe = c(1.74, 1.64, 1.92)
Gewicht = c(70, 75, 100)
```

Diese Zahlenreihen (sog. *Vektoren*) kann man nun komfortabel weiter verarbeiten, z. B. um den Body Mass Index zu berechnen:

```
Gewicht / Größe ^2
## [1] 23.12062 27.88519 27.12674
```

Praxistipp: Mit dem Suchbegriff *r zahlen runden* finden wir im Internet die R-Funktion *round()*, in der man die Anzahl der gewünschten Nachkommastellen mit dem Schlüsselwort *digits* angeben kann:

```
BMI = round(Gewicht / Größe ^2, digits = 1)
BMI
## [1] 23.1 27.9 27.1
```

Tabellen (Matrizen)

Neben eindimensionalen Vektoren verwendet man für statistische Auswertungen sehr häufig zweidimensionale Tabellen, sog. *Matrizen*. Als Anwendungsbeispiel fügen wir unsere drei Vektoren zu einer *Matrix* mit je 3 Spalten und Zeilen zusammen. Dafür gibt es die Funktionen *cbind* und *rbind* (*c* und *r* steht hier für *columns* bzw. *rows*).

```
tab1 = cbind(Größe, Gewicht, BMI)
tab2 = rbind(Größe, Gewicht, BMI)
tab1
##      Größe Gewicht  BMI
## [1,]  1.74      70 23.1
## [2,]  1.64      75 27.9
## [3,]  1.92     100 27.1
tab2
##          [,1]  [,2]   [,3]
## Größe    1.74  1.64   1.92
```

```
## Gewicht 70.00 75.00 100.00
## BMI     23.10 27.90  27.10
```

Im ersten Fall sind die beiden Vektoren spaltenweise, im zweiten Fall zeilenweise angeordnet.

Anstelle von Zahlen kann man mit der Funktion *c*() auch Wörter zu Vektoren kombinieren. Wichtig ist hierbei, dass man die Textinhalte (sog. *strings*) in Anführungszeichen schreibt. So können wir beispielsweise die Tabelle *tab1* mit Zeilennamen versehen:

```
rownames(tab1) = c("Lukas", "Anna", "Markus")
tab1
##        Größe Gewicht  BMI
## Lukas   1.74      70 23.1
## Anna    1.64      75 27.9
## Markus  1.92     100 27.1
```

Wie man sieht, wurden die Spaltennamen automatisch aus den Variablennamen übernommen. Mit der Funktion *colnames*() kann man die Spaltennamen auch gezielt festlegen bzw. überschreiben.

```
colnames(tab1) = c("Größe (m)", "Gewicht (kg)", "Body Mass Index")
tab1
##        Größe (m) Gewicht (kg) Body Mass Index
## Lukas       1.74           70            23.1
## Anna        1.64           75            27.9
## Markus      1.92          100            27.1
```

Daten speichern

Zum Abschluss dieser Lektion speichern wir die Tabelle ab, um sie zu einem späteren Zeitpunkt mit *R* (oder einem anderen Programm wie etwa *Excel*) wieder einlesen zu können. Bevor wir das tun, sollten wir das “Arbeitsverzeichnis” (*working directory*) festlegen, in das wir die Datei speichern wollen, um sie leichter wiederzufinden. Das geht am sichersten über den Menüpunkt *Session >> Set Working Directory >> Choose Directory*. Hier wählt man das gewünschte Verzeichnis aus und klickt auf *Open*. Auf der Konsole wird die Funktion *setwd* (für *set working directory*) mit dem korrekten Pfad angezeigt und ausgeführt. Mit dem Befehl *getwd()* kann man sich davon überzeugen, dass es geklappt hat.

Praxistipp: Diese *getwd*-Zeile kann man auch aus der Konsole an den Anfang des Programms im Editorfenster kopieren, sodass bei erneutem Programmaufruf immer gleich das richtige Verzeichnis eingestellt wird.

Zum Speichern von Daten gibt es in R verschiedene Befehle, je nachdem in welchem Format die Speicherung erfolgen soll. Für eine Übersicht geben wir auf der Konsole *?write.table* ein. Das universellste Format zum Datenaustausch zwischen verschiedenen Programm ist *.csv* (*character-separated variables*), ein Textformat, das standardmäßig als Spaltentrennzeichen das Komma verwendet. In Deutschland, wo das Komma bereits für das Dezimalkomma vergeben ist, empfiehlt sich das Format *csv2* mit einem Semikolon als Spaltentrenner.

Wir wählen hier also den Befehl *write.csv2* in folgender Form:

```
write.csv2(tab1, file = "BMI.csv")
```

Hinweis: Wer eine internationale Ländereinstellung verwendet, lässt einfach die 2 weg: *write.csv*.

Wenn dieser Befehl erfolgreich ausgeführt wurde, sollte sich im gewünschten Arbeitsverzeichnis eine Datei namens *BMI.csv* befinden, die man über die rechte Maustaste mit verschiedenen Programmen wie zum Beispiel *Excel* öffnen und weiterbearbeiten kann.

Wenn man nun *RStudio* schließt und zu einem späteren Zeitpunkt wieder öffnet, kann man diese Datei erneut einlesen und mit R-Befehlen weiterverarbeiten. Die einfachste unter vielen Varianten zum Lesen von csv-Dateien ist der Befehl *tab1 = read.csv2(file.choose())*. Damit öffnet sich das Browserfenster von Windows; notfalls muss man es mit einem Klick auf das Browser-Icon in der Task-Leiste in voller Größe anzeigen. Man kann dann das csv-File in die Variable *tab1* einlesen. Statt *tab1* ist natürlich auch jeder andere Variablenname (z. B. *x* oder *dat*) zulässig.

Hinweis: Anstelle von *file.choose()* kann man auch wie oben erläutert das richtige Arbeitsverzeichnis mit *setwd()* bzw. dem Menüpunkt *Session>>Set Working Directory>>Choose Directory* einstellen und den gewünschten Dateinamen direkt in Anführungszeichen eingeben (hier also *"BMI.csv"*). Das erspart das Öffnen des Browsers, aber man muss dafür auch sicher sein, den richtigen Dateinamen anzugeben, sonst erhält man eine Fehlermeldung (*cannot open the connection*).

Auch diese Lektion speichern wir mit *Strg* + *s* unter einem geeigneten Namen.

Fragen und Aufgaben zu Lektion 3

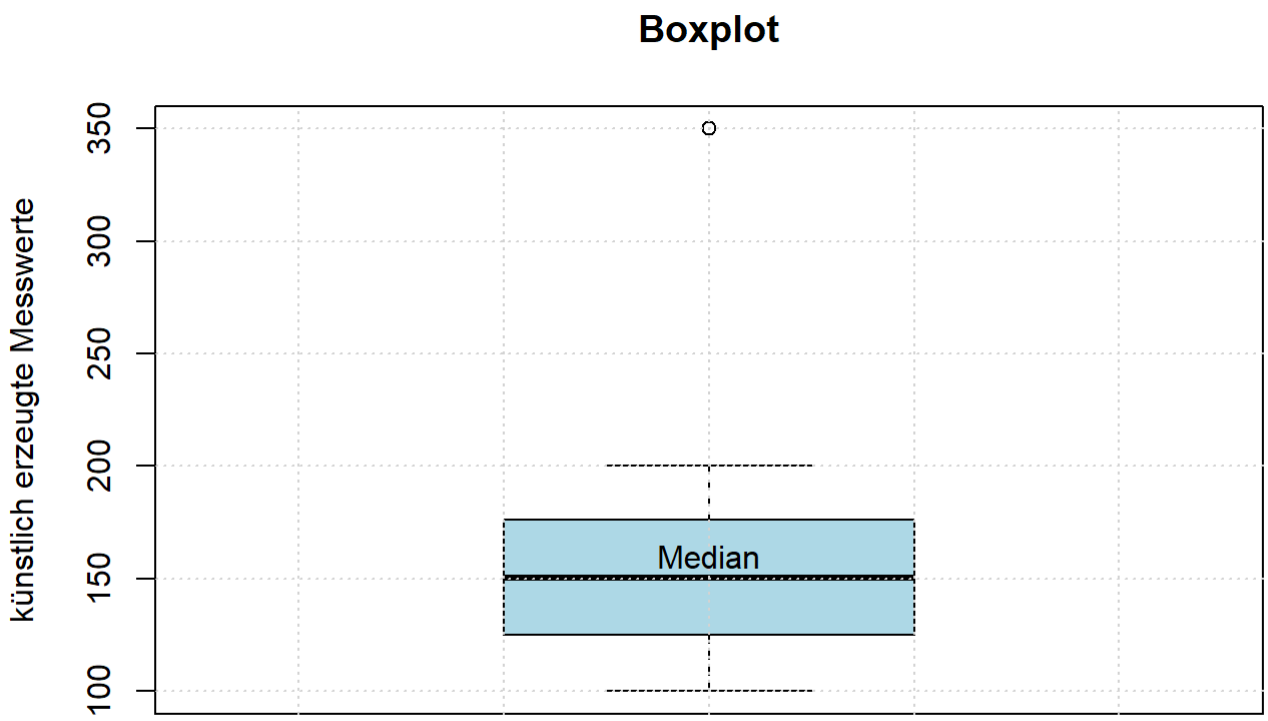
1. Was muss man bei der Eingabe von Strings (im Gegensatz zu Zahlenwerten) besonders beachten?
2. Wofür steht der Name *csv* und wofür wird dieses Dateiformat bevorzugt verwendet?
3. Erzeuge mit der Funktion *seq* eine Zahlensequenz 70, 80, … 130 und weise sie der Variablen *x1* zu.
4. Dividiere alle Elemente von *x1* durch 18 und weise das Ergebnis der Variablen *x2* zu. Freaks können das Ergebnis auch noch auf zwei Nachkommastellen runden.
5. Füge beide Vektoren mit der Funktion *cbind* zu einer Tabelle namens *BZ* zusammen.
6. Erzeuge mit der Funktion *c* (für *combine*) einen Vektor aus den beiden Begriffen *mg/dl* und *mmol/l* und verwende diese mithilfe von *colnames(BZ)* als Spaltennamen für die Tabelle *BZ*.
7. Verwende die Zahlenreihe 1 bis 7 mithilfe von *rownames(BZ)* als Zeilennamen der Tabelle.
8. Speichere die Tabelle *BZ* unter dem Namen *Blutzucker.csv* ab und öffne die gespeicherte Datei mit *Excel*.
9. Zwei Aufgaben für Excel-Fans: Füge in *Excel* weitere Zeilen mit Werten hinzu und speichere die Tabelle wieder im csv-Format ab (*Speichern unter >> Dateityp CSV*).
10. Lies diese Tabelle erneut mit *R* ein und zeige sie auf der Konsole an.

Lektion 4

Grafiken erzeugen und formatieren

Grafische Darstellungen gehören zu den besonderen Stärken von *R*. Wie wir bereits gesehen haben, genügt häufig eine einzige, kurze Befehlszeile wie z. B. *boxplot(1:100)*, um eine instruktive Abbildung zu erzeugen, Mit wenigen zusätzlichen Angaben kann man die Achsen beschriften, Farben festlegen, Texte einfügen usw. Wir erstellen ein neues R-Skript und tippen in das Editorfenster eine erweiterte Boxplot-Funktion ein:

```
boxplot(x = c(100 : 200, 350), main = "Boxplot", ylab = "künstlich erzeugte Messwer
te", col = "lightblue")

grid()

text(x = 1, y = 160, "Median")
```



Diese kleine Grafik demonstriert auf einen Blick eine ganze Reihe wichtiger Aussagen zu R:

- Den Grafiktyp legt man mit einem geeigneten (häufig selbsterklärenden) Funktionsnamen wie z. B. *boxplot* fest. Andere Funktionsnamen sind *plot* (xy-Diagramm), *barplot* (Balkendiagramm), *hist* (Histogramm) usw.
- In der Klammer wird angegeben, aus welchen Werten die Grafik erzeugt werden soll.
- Dahinter folgen – durch Kommas abgetrennt – weitere Parameter der Grafikfunktion, z. B. *main* für die Überschrift oder *col* für die Farbe.
- Weitere Grafikelemente wie etwa Texte oder Gitternetzlinien können in separaten Zeilen hinzugefügt werden. Der Text “Median” wurde hier zum Beispiel in der Position x = 1 und y = 160 eingefügt.

Ganz nebenbei sei an dieser Stelle angemerkt, dass der **Boxplot** zu den leistungsfähigsten Grafiken der *deskriptiven Statistik* gehört, um sich einen schnellen Überlick über einen Datensatz zu verschaffen. Er besteht aus einer “Box”, die die zentralen 50 % aller Werte enthält (in unserem Fall also die Zahlen zwischen 125 und 175). Im Inneren der Box befindet sich ein dicker Strich, der die untere Hälfte der Daten von der oberen trennt; dieser Grenzwert heißt *Median*. Schließlich gibt es noch zwei “Antennen” (engl. *Whiskers* für *Schnurrbarthaare*), die den gesamten Wertebereich umfassen. Etwaige Ausreißer, die nicht zur Verteilung der Werte passen, werden als separate Punkte dargestellt; hier ist dies der Wert 350, den wir den Daten mit der Funktion *c*() hinzugefügt haben.

Die enorme Vielfalt der grafischen Möglichkeiten von R würde den Rahmen dieser kurzen Einführung sprengen. Selbstverständlich gibt es für jeden Grafiktyp spezifische Parameter, mit denen man das Erscheinungsbild beeinflussen kann. Zusätzlich zu den Grafikfunktionen der Basisversion von *R* kann man große Zusatzpakete wie etwa *ggplot2* laden, die hochprofessionelle Darstellungen ermöglichen. Wer sich hier Überblick verschaffen möchte, sucht am besten im Internet nach Begriffen wie *Grafiken mit R* oder *Grafiken mit ggplot*.

Ich möchte hier am Beispiel eines einfachen xy-Diagramms noch einmal die Grundfunktionen demonstrieren, die in den meisten Grafikanwendungen von *R* zur Anwendung kommen. Dazu erzeugen wir zwei Datensätze x und y, die aus Zufallszahlen bestehen. Dafür setzen wir die Funktionen *rnorm* und *runif* ein; das *r* steht hier für *random* (Zufall) und der Rest des Namens für den Verteilungstyp (Normal- bzw. Gleichverteilung).

```
x = runif(n = 200, min = 20, max = 80)

y = rnorm(n = 200, mean = 2.4, sd = 0.1)
```

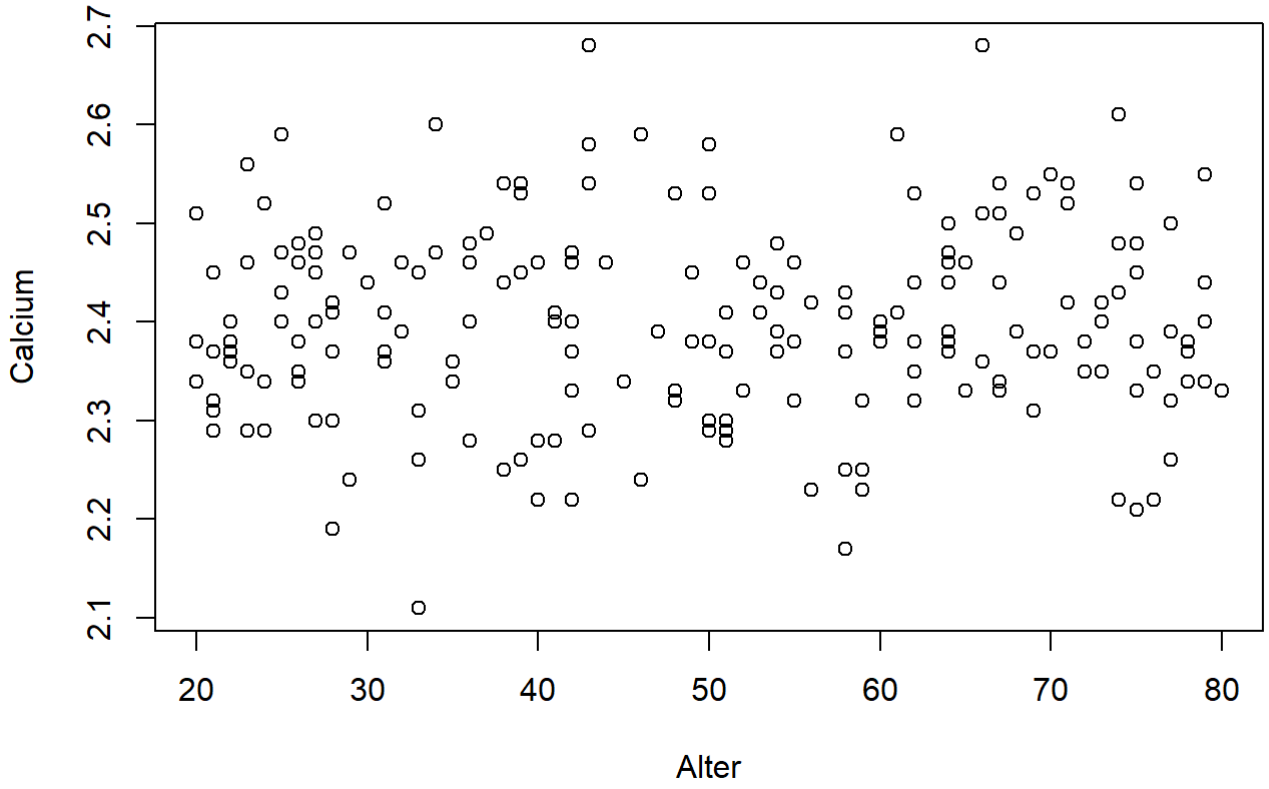
Wer sich für die Details dieser Funktionen interessiert, kann auf der Konsole *?runif()* und *?rnorm()* eingeben. Dort sieht man auch gleich, welche weiteren Funktionen es zu den beiden Verteilungstypen sonst noch gibt; so kann man mit *dnorm* zum Beispiel die berühmte Gauß'sche Glockenkurve konstruieren, wobei das *d* für die *Dichte* steht.

Der Vektor x soll hier für ein gleichmäßig verteiltes Alter von gesunden Probanden stehen und der Vektor y für normalverteilte Calciumwerte mit einem Referenzintervall von etwa 2.2 bis 2.6 mmol/l. Wir runden also die beiden soeben erzeugten Vektoren auf 0 bzw. 2 Nachkommastellen und geben ihnen sprechende Namen.

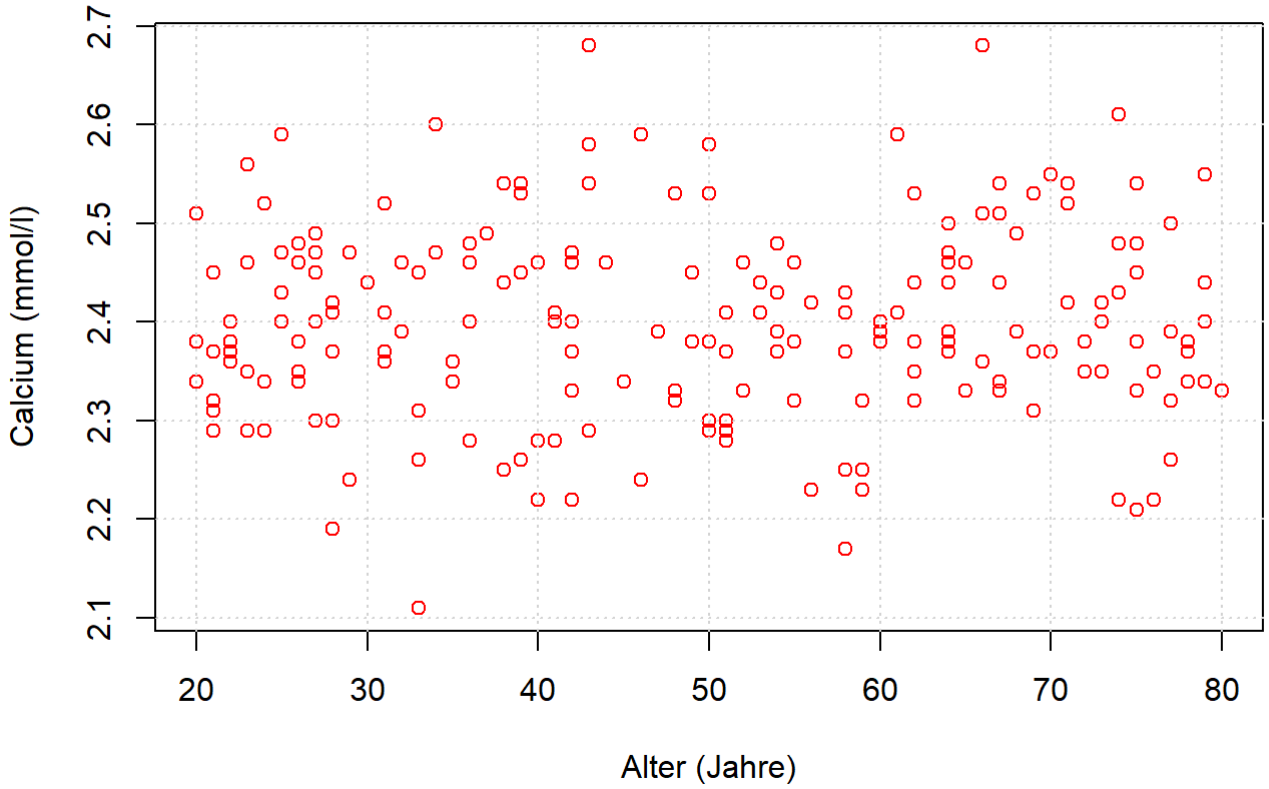
```
Alter = round(x, digits = 0)

Calcium = round(y, digits = 2)
```

Den Erfolg sehen wir im Fenster rechts oben (*Global Environment*): Es gibt nun sowohl die Variablen *x* und *y* als auch *Alter* und *Calcium*. Daraus erzeugen wir nun mit dem Befehl *plot* eine einfache xy-Grafik:

```
plot(x = Alter, y = Calcium)
```

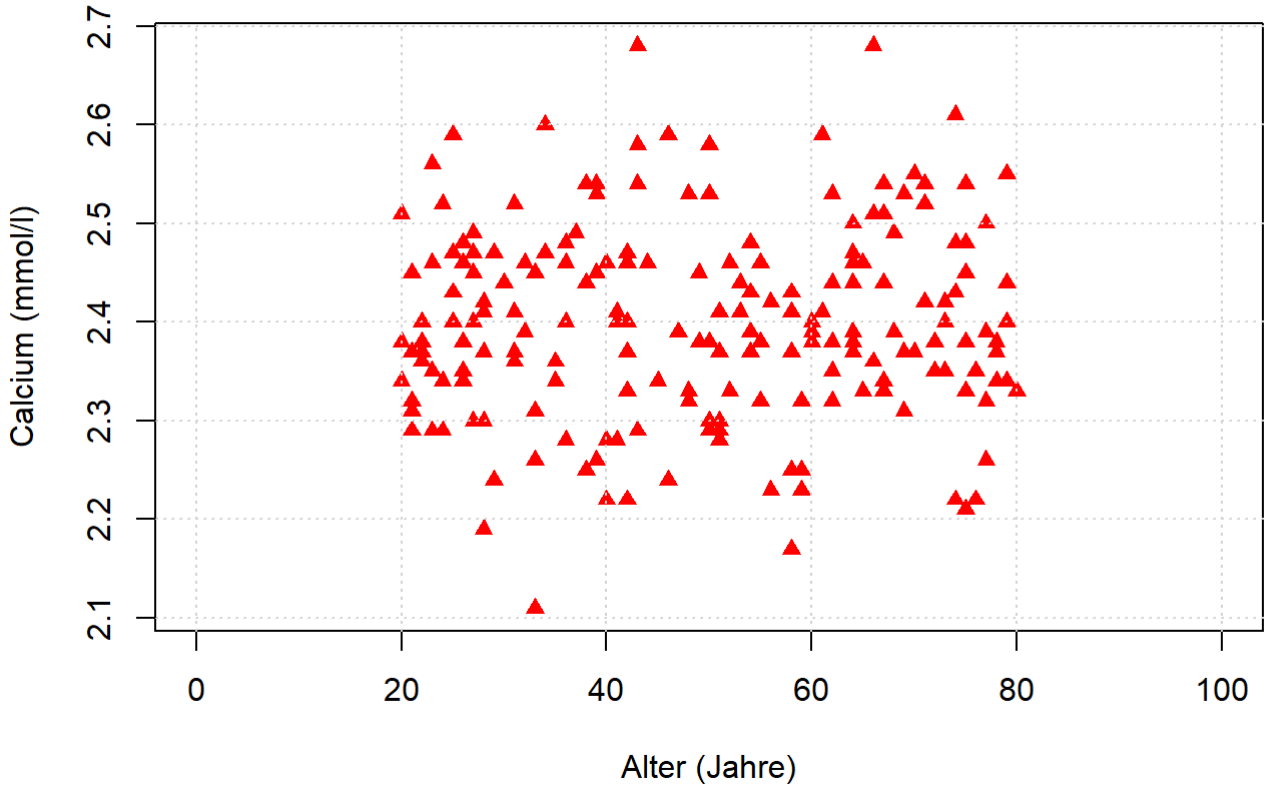


Einfacher geht’s wohl nicht. Im nächsten Schritt fügen wir analog zur obigen Boxplot-Funktion Achsenbeschriftungen “Alter (Jahre)” und “Calcium (mmol/l)” hinzu, färben die Punkte rot und fügen ein Gitternetz ein.



Wenn man auf der Konsole *?plot()* eingibt, erhält man unter den weiterführenden Links *The Default Scatterplot Function* und *Generic X-Y-Plotting* eine Vielzahl weiterer Parameter, die das Aussehen der Grafik beeinflussen. Hier findet sich beispielsweise unter *Points* der wichtige Parameter *pch*, mit dem man die Form der Symbole verändern kann. *pch = 19* erzeugt ausgefüllte Punkte, *pch = 23* umrandete Rauten usw. Mit xlim und ylim kann man die Grenzen der x- und y-Achse festlegen. Trägt man beispielsweise *xlim = c(0, 100)* ein, geht die x-Achse von 0 bis 100 Jahre. Um den Code übersichtlich zu halten, kann man nach den Kommas jederzeit Zeilenumbrüche einfügen. Sie werden von *R* ebenso wie Leerzeichen im Code einfach ignoriert.

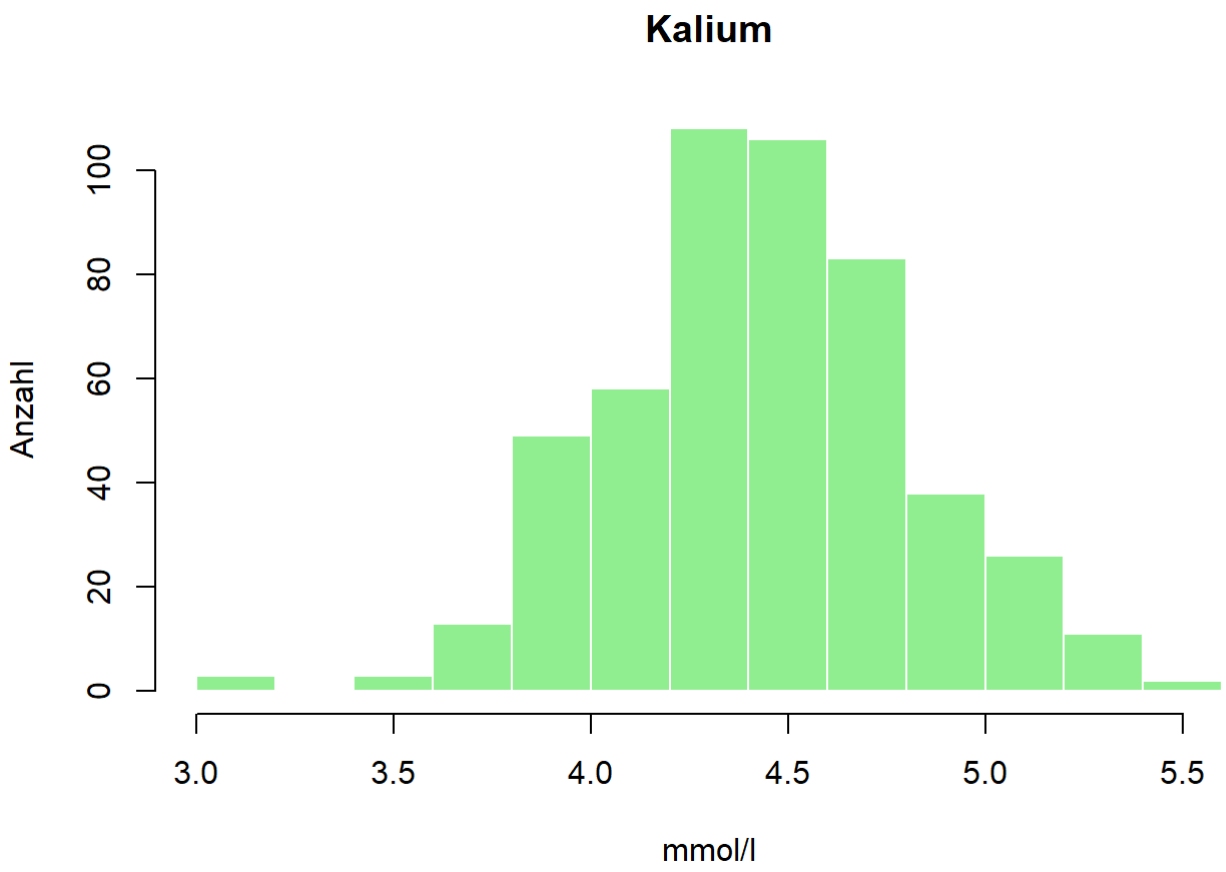
```
plot(Alter, Calcium,
     xlim = c(0, 100),
     xlab = "Alter (Jahre)", ylab = "Calcium (mmol/l)",
     col = "red", pch = 17)
grid()
```



Zum Abschluss dieser kleinen Übung und zur Wiederholung des bisher Gelernten speichern wir die Grafik als PDF-File ab, fügen die beiden Vektoren *Alter* und *Calcium* zu einer Tabelle mit einem frei gewählten Variablennamen zusammen und speichern sie als csv-File ab. Danach können wir auch diese Lektion 4 als R-File speichern und *RStudio* schließen.

Fragen und Aufgaben zu Lektion 4

1. Was bedeutet der Buchstabe *r* in den Funktionsnamen *runif* und *rnorm*?
2. Schließe das Programm der Lektion 4, rufe es erneut auf und erstelle aus den Calciumwerten anstelle des xy-Diagramms einen Boxplot. Beschrifte die Grafik und erkläre die Elemente *Box* und *Whiskers*.
3. Erzeuge nach dem Vorbild von Calcium mit der Funktion *rnorm* 500 normalverteilte Kaliumwerte (mean = 4.4, sd = 0.4) und erstelle daraus ein xy-Diagramm mit dem Alter auf der x-Achse. Beschrifte das Diagramm und füge ein Gitternetz ein.
4. Erstelle aus den Kaliumwerten ein Histogramm mit der Funktion *hist*.
5. Für Grafikfreaks: Beschrifte das Histogramm mit sinnvollen deutschen Begriffen und färbe es hellgrün. Finde mit *?hist* heraus, wie man die Farbe der Rahmenlinien verändert.


Kalium
Anzahl
0
20
40
60
80
100
3.0
3.5
4.0
4.5
5.0
5.5
mmol/l