Einführung in R

Stephan Goerigk

2022-10-13

Contents

Über dieses Skript

Liebe Studierende,

dieses Skript soll Sie in die grundlegenden Analysewerkzeuge in R einführen, von der grundlegenden Kodierung und Analyse bis hin zur Datenverarbeitung, dem Plotten und der statistischen Inferenz.

Wenn R Ihre erste Programmiersprache ist, ist das völlig in Ordnung. Wir gehen alles Schritt für Schritt gemeinsam durch. Die Techniken in diesem Buch sind zwar auf die meisten Datenanalyseprobleme anwendbar, da wir jedoch aus der Psychologie kommen, werde ich den Kurs auf die Lösung von Analyseproblemen ausrichten, die in der psychologischen Forschung häufig auftreten.

Ich wünsche Ihnen Viel Erfolg und Spaß!

Chapter 1

Warum ist R so gut?

1.1 Open Source

R ist zu 100 % kostenlos und verfügt daher über eine große Unterstützergemeinschaft. Im Gegensatz zu SPSS, Matlab, Excel und JMP ist R völlig kostenlos und wird es auch immer bleiben. Das spart nicht nur Geld - es bedeutet auch, dass eine riesige Gemeinschaft von R-Programmierern ständig neue R-Funktionen und -Pakete in einer Geschwindigkeit entwickelt und verbreitet, die alle anderen Pakete in den Schatten stellt. Die Größe der R-Programmiergemeinschaft ist atemberaubend. Wenn Sie jemals eine Frage dazu haben, wie man etwas in R implementiert, wird eine schnelle Google-Suche Sie praktisch jedes Mal zur Antwort führen.

1.2 Vielseitigkeit

R ist unglaublich vielseitig. Sie können R für alles verwenden, von der Berechnung einfacher zusammenfassender Statistiken über die Durchführung komplexer Simulationen bis hin zur Erstellung großartiger Diagramme. Wenn Sie sich eine analytische Aufgabe vorstellen können, können Sie sie mit ziemlicher Sicherheit in R implementieren.

1.3 R Markdown

Mit RStudio, einem Programm, das Sie beim Schreiben von R-Code unterstützt, können Sie mit RMarkdown einfach und nahtlos R-Code, Analysen, Diagramme und geschriebenen Text zu eleganten Dokumenten an einem Ort kombinieren.

Tatsächlich habe ich dieses gesamte Skript (Text, Formatierung, Diagramme, Code… ja, alles) in RStudio mit R Markdown geschrieben Mit RStudio müssen Sie sich nicht mehr mit zwei oder drei Programmen herumschlagen, z. B. Excel, Word und SPSS, wo Sie die Hälfte Ihrer Zeit mit dem Kopieren, Einfügen und Formatieren von Daten, Bildern und Tests verbringen, sondern können alles an einem Ort erledigen, so dass nichts mehr falsch gelesen, getippt oder vergessen wird.

1.4 Transparenz

In R durchgeführte Analysen sind transparent, leicht weiterzugeben und reproduzierbar. Wenn Sie einen SPSS-Benutzer fragen, wie er eine bestimmte Analyse durchgeführt hat, wird er sich ggf. nicht daran erinnern, was er vor Monaten oder Jahren tatsächlich getan hat. Wenn Sie einen R-Anwender (der gute Programmiertechniken verwendet) fragen, wie er eine Analyse durchgeführt hat, sollte er Ihnen immer den genauen Code zeigen können, den er verwendet hat. Das bedeutet natürlich nicht, dass er die richtige Analyse verwendet oder sie korrekt interpretiert hat, aber mit dem gesamten Originalcode sollten etwaige Probleme völlig transparent sein! Dies ist eine Grundvoraussetzung für offene, replizierbare Forschung.

Chapter 2

R Materialien

2.1 Cheat Sheets

In diesem Skript werden Sie viele neue Funktionen kennenlernen. Wäre es nicht schön, wenn jemand ein Wörterbuch mit vielen gängigen R-Funktionen erstellen würde? Ja, das wäre es, und zum Glück haben einige freundliche R-Programmierer genau das getan. Im Folgenden finden Sie eine Tabelle mit einigen der Funktionen, die ich empfehle. Ich empfehle Ihnen dringend, diese auszudrucken und die Funktionen zu markieren, wenn Sie sie lernen!

Link zum Base R Cheat Sheet

Link zu den R Studio Cheat Sheets

Insbesondere die Cheat Sheets zu ggplot2 und dplyr kann ich Ihnen nur wärmstens ans Herz legen

2.2 Hilfe und Inspiration online

Eine Google Suche nach einem spezifischen R Problem bringt Sie (fast) immer an Ihr Ziel. Häufig findet man gute Antworten in den github Hilfsmatierialien einzelner Pakete, auf den Community Seiten von R Studio und in den Foren von stackoverflow.

2.3 Andere Bücher

Die Inhalte dieser Bücher sind nicht prüfunsrelevant.

Es gibt viele, viele ausgezeichnete Bücher über R. Hier sind zwei, die ich empfehlen kann (von denen eines sogar umsonst ist):

Discovering Statistics with R von Field, Miles and Field

R for Data Science von Garrett Grolemund and Hadley Wickham

Chapter 3

Installation

Um R benutzen zu können müssen wir zwei Softwarepakete herunterladen:

- **R**
- **RStudio**

R ist die Programmiersprache, mit der wir arbeiten. R-Studio ist eine Benutzeroberfläche, die uns das Programmieren mit R ungemein erleichtert.

3.1 Installation von R

Um R zu installieren, klicken Sie auf den, Ihrem Betriebssystem entsprechenden, Link und befolgen Sie die Anleitungen:

Operating System	Link
Windows	http://cran.r-project.org/bin/windows/base/
Mac	http://cran.r-project.org/bin/macosx/

Nach dieser Installation haben Sie bereits die volle Funktionalität des Programms. Sie werden jedoch feststellen, dass beinahe alle R-Nutzer RStudio zum programmieren nutzen, da dieses eine leichter nutzbare Oberfläche hat. Tatsächlich müssen Sie nach der Installation von RStudio das R Basisprogramm nie wieder öffnen.

3.2 Installation von RStudio

Bitte installieren Sie dann RStudio - das Programm, über welches wir auf R zugreifen und mit dem wir unsere Skripte schreiben.

Um RStudio zu installieren, klicken Sie auf diesen Link und befolgen Sie die Anleitungen: http://www.rstudio.com/products/rstudio/download/

Chapter 4

Programmaufbau

4.1 Die vier RStudio Fenster

Wenn Sie RStudio öffnen, sehen Sie vier Fenster, wie in der folgenden Abbildung dargestellt:

Wenn Sie mögen, können Sie die Reihenfolge der Fenster in den RStudio Einstellungen verändern. Sie können die Fenster auch verstecken (Minimieren/Maximieren Symbol an der oberen rechten Ecke jedes Fensters) oder ihre Größe verändern, indem die sie ihre Grenzbalken anklicken und verschieben.

Lassen Sie uns jetzt schauen, was genau die Funktion jedes der Fenster ist:

4.1.1 Source - Ihr Schreibblock für Code

Im Source Fenster erstellen und bearbeiten Sie "R-Skripte" - Ihre Codesammlungen. Keine Sorge, R-Skripte sind nur Textdateien mit der Erweiterung ".R". Wenn Sie RStudio öffnen, wird automatisch ein neues unbenanntes Skript gestartet. Bevor Sie mit der Eingabe eines unbenannten R-Skripts beginnen, sollten Sie die Datei immer unter einem neuen Dateinamen speichern (z.B. "Mein_RScript.R"). Wenn Ihr Computer während der Arbeit abstürzt, steht Ihr Code in R zur Verfügung, wenn Sie RStudio erneut öffnen.

Sie werden feststellen, dass R beim Schreiben des Skripts den Code während der Eingabe nicht tatsächlich ausführt. Damit R Ihren Code tatsächlich ausführt, müssen Sie den Code zunächst an die Konsole "senden" (wir werden im nächsten Abschnitt darüber sprechen).

Es gibt viele Möglichkeiten, Ihren Code aus dem Skript an die Konsole zu senden. Die langsamste Methode ist das Kopieren und Einfügen. Schneller geht es, wenn

Sie den Code, den Sie auswerten möchten, markieren und auf die Schaltfläche "Run" oben rechts in der Quelle klicken. Alternativ können Sie auch die Tastenkombination "Command + Return" auf dem Mac oder "Control + Enter" auf dem PC verwenden, um den gesamten markierten Code an die Konsole zu senden.

4.1.2 Konsole - Das Herzstück von R

Die Konsole ist das Herzstück von R. Hier führt R den Code aus. Am Anfang der Konsole sehen Sie das Zeichen ">". Dies ist eine Eingabeaufforderung (sog. "Prompt"), die Ihnen mitteilt, dass R bereit für neuen Code ist. Sie können direkt nach dem Prompt `>` Code in die Konsole eingeben und erhalten sofort eine Antwort. Wenn Sie zum Beispiel 2+2 in die Konsole eingeben und die Eingabetaste drücken, werden Sie sehen, dass R sofort eine Ausgabe von 4 liefert.

```
2+2
```

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

Versuchen Sie, 2+2 zu berechnen, indem Sie den Code direkt in die Konsole eingeben - und dann Enter drücken. Sie sollten das Ergebnis `[1] 4` sehen. Machen Sie sich keine Gedanken über die `[1]`, dazu kommen wir später.

Geben Sie denselben Code in das Skript ein und senden Sie ihn an die Konsole, indem Sie den Code markieren und auf die Schaltfläche "Run" in der oberen rechten Ecke des Quelltextfensters klicken. Alternativ können Sie auch die Tastenkombination "Command + Return" auf dem Mac oder "Control + Enter" unter Windows verwenden.

Tipp: Wie Sie sehen, können Sie Code entweder über das Skript oder durch direkte Eingabe in die Konsole ausführen. In 99% der Fälle sollten Sie jedoch das Skript und nicht die Konsole verwenden. Der Grund dafür ist ganz einfach: Wenn Sie den Code in die Konsole eingeben, wird er nicht gespeichert (obwohl Sie in Ihrem Befehlsverlauf nachsehen können). Und wenn Sie beim Eingeben von Code in die Konsole einen Fehler machen, müssen Sie alles noch einmal von vorne eingeben. Stattdessen ist es besser, den gesamten Code in das Skript zu schreiben. Wenn Sie bereit sind, einen Code auszuführen, können Sie ihn mit "Run" an die Konsole senden.

4.1.3 Environment/History - Das Gedächtnis von R

In dem Tab "Environtment" dieses Bereichs werden die Namen aller Datenobjekte (wie Vektoren, Matrizen und Datenrahmen) angezeigt, die Sie in Ihrer aktuellen R-Session definiert haben. Sie können auch Informationen wie die Anzahl der Spalten und Zeilen in Datensätzen sehen. Der Tab enthält auch einige anklickbare Aktionen wie

Datensatz importieren", wodurch eine grafische Benutzeroberfläche (GUI) für wichtige Daten in R geöffnet wird.

Der Tab "Historiy" dieses Bereichs zeigt Ihnen einfach eine Sammlung aller Befehle an den Sie zuvor in der Konsole ausgewertet haben. Wenn man mit Skripten arbeitet, schaut man sich diese, allerdings relativ selten an.

Wenn Sie sich mit R besser auskennen, werden Sie das Fenster Environment/History vielleicht nützlich finden. Aber für den Moment können Sie es einfach ignorieren. Wenn Sie Ihren Bildschirm entrümpeln wollen, können Sie das Fenster auch einfach minimieren, indem Sie auf die Schaltfläche Minimieren oben rechts im Fenster klicken.

4.1.4 Files/Plots/Packages/Help/Viewer - Interaktion von R mit Dateien

Die Tabs Files/Plots/Packages/Help/Viewer zeigen Ihnen viele hilfreiche Informationen. Schauen wir uns die einzelnen Registerkarten im Detail an:

1. Files - Der Tab "Files" gibt Ihnen Zugriff auf das Dateiverzeichnis auf Ihrer Festplatte. Dateien, die Sie in Ihrem R Projekt benutzen liegen in der Regel in einem von Ihnen definierten Arbeitsverzeichnis. Wir werden in Kürze ausführlicher über Arbeitsverzeichnisse sprechen.

2. Plots - Das Plots-Panel zeigt (keine große Überraschung) alle Ihre Plots an.

3. Pakete - Zeigt eine Liste aller auf Ihrer Festplatte installierten R-Pakete an und gibt an, ob sie derzeit geladen sind oder nicht. Pakete, die in der aktuellen Sitzung geladen sind, sind markiert, während Pakete, die installiert, aber noch nicht geladen sind, nicht markiert sind. Auf die Pakete gehen wir im nächsten Abschnitt näher ein.

4. Hilfe - Hilfemenü für R-Funktionen. Sie können entweder den Namen einer Funktion in das Suchfenster eingeben oder den Code `?function.name` verwenden, um nach einer Funktion mit dem Namen `function.name` zu suchen:

```
?hist   # Wie funktioniert die Histogrammfunktion?
?t.test # Wie funktioniert der t-Test?
```

4.2 R Packages

Wenn Sie R zum ersten Mal herunterladen und installieren, installieren Sie die Base R Software. Base R enthält die meisten Funktionen, die Sie täglich

verwenden werden, wie mean() und hist(). Allerdings werden hier nur Funktionen angezeigt, die von den ursprünglichen Autoren der Sprache R geschrieben wurden. Wenn Sie auf Daten und Code zugreifen möchten, die von anderen Personen geschrieben wurden, müssen Sie diese als "Package" installieren. Ein R-Package ist einfach ein Bündel von Funktionen (also bereits geschriebener Code), die in einem übersichtlichen Paket gespeichert sind.

Ein Paket ist wie eine Glühbirne. Um es nutzen zu können, müssen Sie es zunächst in Ihr Haus (d.h. auf Ihren Computer) bestellen, indem Sie es installieren. Wenn Sie ein Paket einmal installiert haben, brauchen Sie es nie wieder zu installieren. Jedes Mal, wenn Sie das Paket tatsächlich verwenden wollen, müssen Sie es jedoch einschalten, indem Sie es laden. Und so geht's:

4.2.1 R Packages installieren

Ein Paket zu installieren bedeutet einfach, den Paketcode auf Ihren Computer herunterzuladen. Die gängigste Methode ist das Herunterladen aus dem Comprehensive R Archive Network (CRAN).

Um ein neues R-Paket von CRAN zu installieren, können Sie einfach den Code install.packages("name") ausführen, wobei "name" der Name des Pakets ist.

Um zum Beispiel das Paket ggplot2 herunterzuladen, welches wir oft zum Erstellen von Graphen verwenden, geben Sie ein:

```
# install.packages("ggplot2")
```

4.2.2 R Packages laden

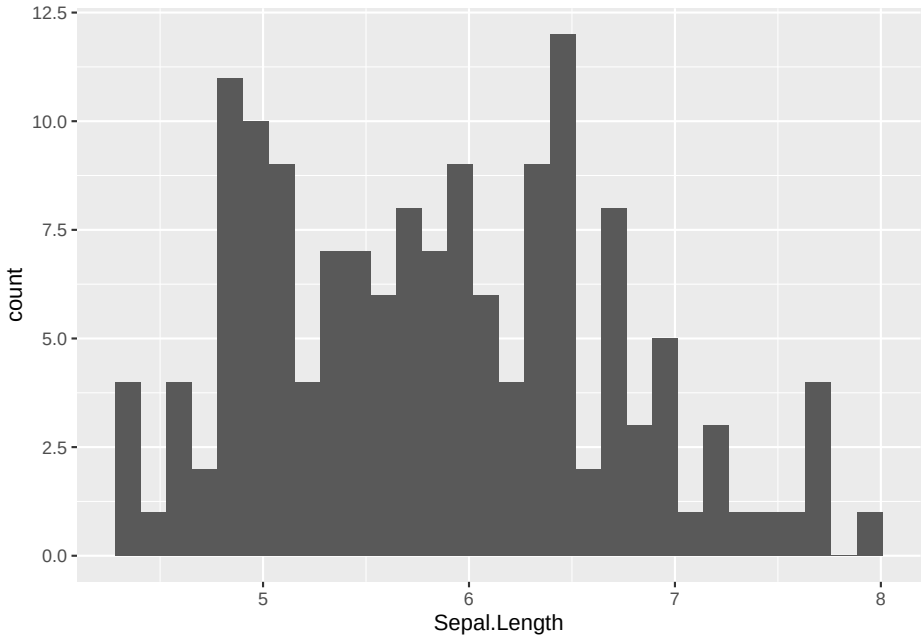
Sobald Sie ein Paket installiert haben, befindet es sich auf Ihrem Computer. Aber nur weil es auf Ihrem Computer ist, bedeutet das nicht, dass R bereit ist, es zu benutzen. Wenn Sie etwas, wie eine Funktion oder einen Datensatz, aus einem Paket verwenden wollen, müssen Sie *immer* zuerst das Paket in Ihrer R-Sitzung *laden*. Genau wie bei einer Glühbirne müssen Sie sie einschalten, um sie zu benutzen!

Um ein Paket zu laden, verwenden Sie die Funktion `library()`. Nachdem wir zum Beispiel das Paket `ggplot2` installiert haben, können wir es mit `library("ggplot2")` laden:

```
#   Laden des "ggplot2" Pakekts, damit wir es benutzen können!
#   Pakete müssen zu Beginn jeder R Session neu geladen werden!
library("ggplot2")
```

Jetzt, wo Sie das Paket `ggplot2` geladen haben, können Sie jede seiner Funktionen benutzen (hier die Funktion ggplot, um einen Graph zu erstellen)!

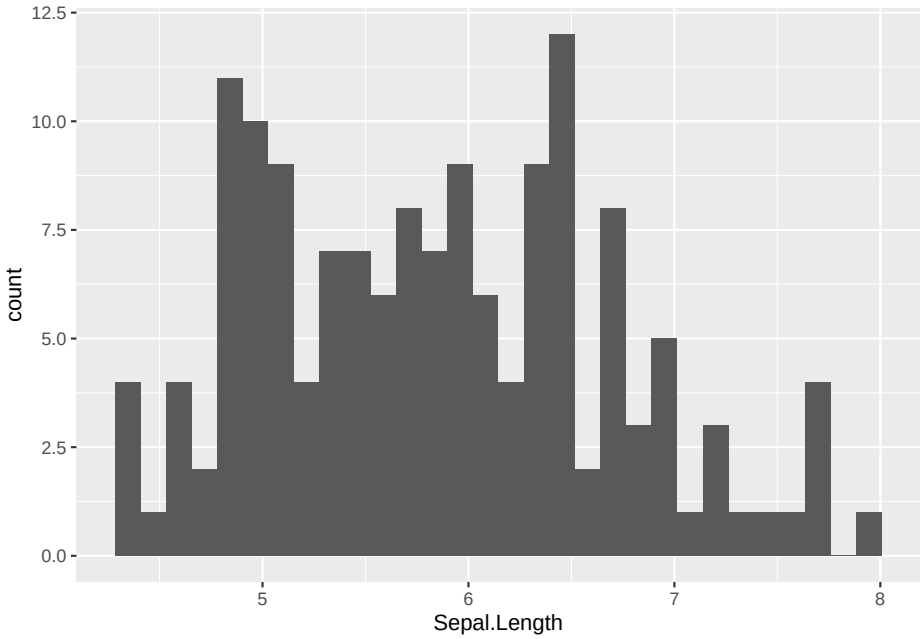
```
ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_histogram()
```



Pakete müssen zu Beginn jeder R Session neu geladen werden. Deswegen schreiben wir in der Regel ganz an den Anfang unseres Skripts gleich mehrere Zeilen, mit `library()` Befehlen für alle R Pakete, die wir für unsere Analyse benötigen werden.

In R gibt es eine Möglichkeit, ein Paket vorübergehend zu laden, ohne die Funktion `library()` zu verwenden. Um dies zu tun, können Sie einfach die Notation `package::funktion` verwenden. Diese Notation sagt R einfach, dass es das Paket nur für diesen einen Codeabschnitt laden soll. Zum Beispiel könnte ich die Funktion `ggplot` aus dem Paket `ggplot2` wie folgt verwenden:

```
ggplot2::ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_histogram()
```



Ein Vorteil der Notation “package::function” ist, dass für jeden, der den Code liest, sofort klar ist, welches Paket die Funktion enthält. Ein Nachteil ist jedoch, dass Sie, wenn Sie eine Funktion aus einem Paket häufig verwenden, gezwungen sind, den Paketnamen ständig neu einzugeben. Sie können jede Methode verwenden, die für Sie sinnvoll ist.

Chapter 5

Datenformate

5.1 Skalar

Der einfachste Objekttyp in R ist der **Skalar**. Ein Skalar Objekt ist einfach nur ein einzelner Wert, z.B. eine Zahl oder ein Wort.

Hier sind einige Beispiele für numerische Skalar Objekte:

```
# Examples of numeric scalars
a <- 100
b <- 3 / 100
c <- (a + b) / b
```

Skalare müssen nicht numerisch sein, sondern auch Worte. Wortobjekte heißen in R **characters** (aka strings). In R schreibt man characters immer in Anführungszeichen `""`. Hier sind einige Beispiele für character Skalare:

```
# Beispiele für character Skalare
d <- "Psychologe"
e <- "Zigarre"
f <- "Haben Psychologen wirklich alle Bärte und rauchen Zigarre?"
```

Wie Sie sich vermutlich vorstellen können, behandelt R numerische und character Skalare unterschiedlich. Zum Beispiel lassen sich mir numerischen Skalaren grundlegende arithmetische Operationen durchfühlen (Addition, Subtraktion, Multiplikation…) – das funktioniert mit character Skalaren nicht. Wenn Sie dennoch probieren numerische Operationen auf character Skalare anzuwenden, bekommen Sie eine Fehlermeldung, so wie diese:

```
a = "1"
b = "2"
a + b
```

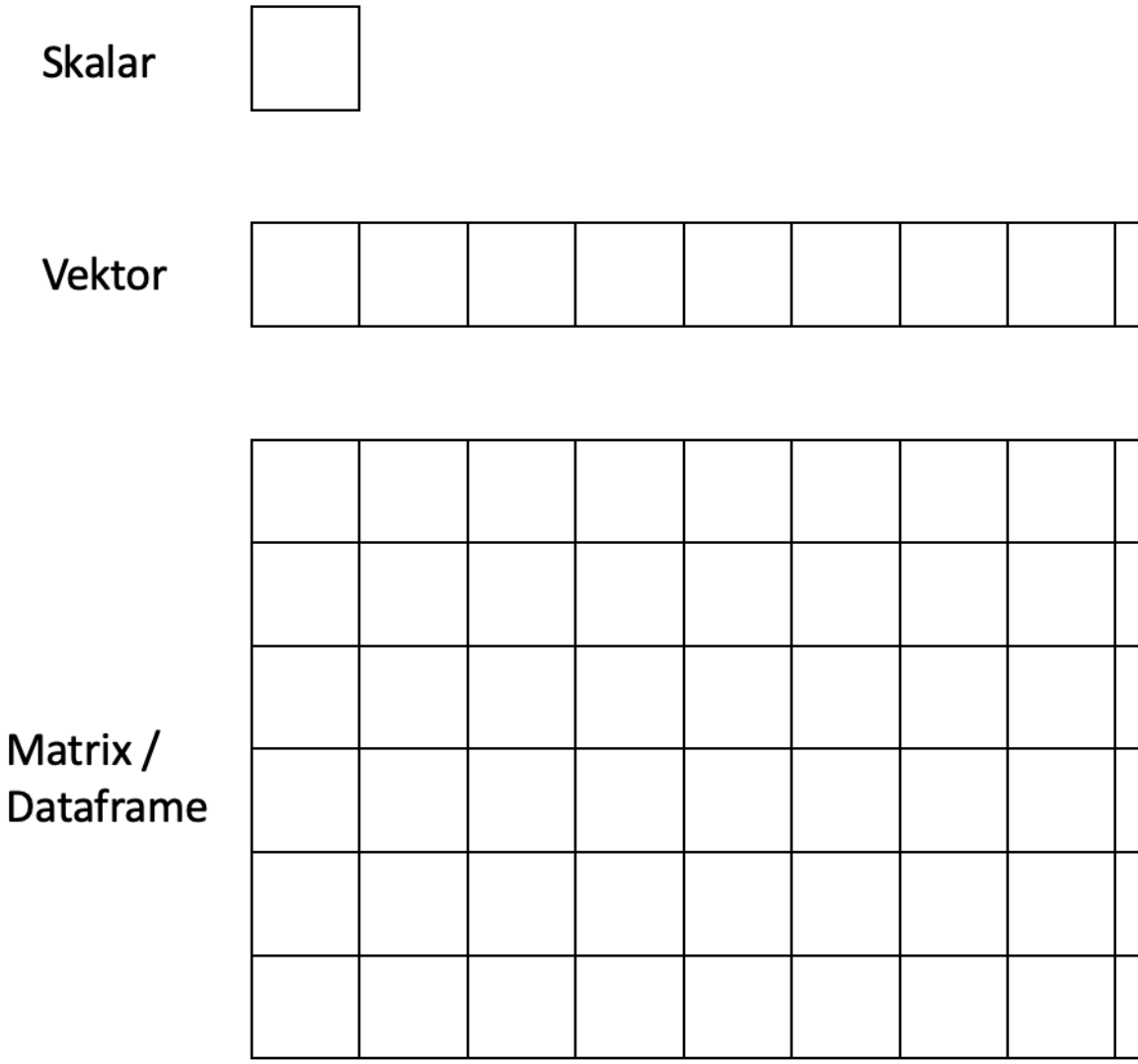


Figure 5.1: Skalar, Vektor, Matrix

"Fehler in a + b : nicht-numerisches Argument für binären Operator"

5.2 Vektor

Machen wir weiter mit `Vektoren`. Ein Vektor Objekt ist einfach eine Kombination mehrerer Skalare in einem einzelnen Objekt (z.B. eine Zahlen- oder Wortreihe). Zum Beispiel könnten die Zahlen von 1-10 in einen Vektor mit der Länge 10 kombiniert werden. Oder die Buchstaben des Alphabets könnten in einen Vektor mit der Länge 26 gespeichert werden. Genau wie Skalare, können Vektoren numerisch oder characters sein (Aber nicht beides auf einmal!)

Die einfachste Art einen Vektor zu erstellen ist mit der `c()` Funktion. Das c steht für "concatenate", was auf Englisch so viel heißt wie "zusammenbringen". Die `c()` Funktion nimmt mehrere Skalare als Input und erstellt einen Vektor, der diese Objekte enthält.

Wenn man`c()` benutzt, muss man immer ein **Komma** zwischen die Objekte setzen (Skalare oder Vektoren), die man kombinieren möchte.

Lassen Sie uns die `c()` Funktion nutzen um einen Vektor zu erstellen der `a` heißt und die Zahlen von 1 bis 7 enthält

Let's use the `c()` function to create a vector called `a` containing the integers from 1 to 5.

```
a = c(1, 2, 3, 4, 5, 6, 7)
# Das Ergebnis ausgeben
a
```

```
## [1] 1 2 3 4 5 6 7
```

Sie können auch character Vektoren erstellen, indem Sie die `c()` auf einzelne character Skalare Funktion anwenden:

```
char.vec = c("Freud", "Wundt", "Bandura", "Watson", "Jung")
# Das Ergebnis ausgeben
char.vec
```

```
## [1] "Freud"   "Wundt"   "Bandura" "Watson"  "Jung"
```

5.2.1 Vektor Typen

Vektoren sind ein zentrales Element von R. Ein Vektor kann Zahlen, Buchstaben oder logische Werte enthalten, aber niemals eine Kombination

Der Vektor ist die Entsprechung der **Variable** und die Skalare, aus denen der Vektor besteht sind die **Merkmalsausprägungen** der Variable.

5.2.2 Faktor Variablen

Wir haben bereits gelernt, wie man einen Vektor aus character Objekten erstellt. Manchmal brauchen wir in R jedoch Variablen, die nicht nur Worte enthalten, sondern dem Programm mitteilen, dass es sich um feste Gruppen oder **Kategorien** handelt. Es geht also nicht nur um eine "Sammlung" von Worten (z.B. Nachnamen von Probanden), sondern um festgelegte Analyseeinheiten. Solche Variablen heißen in R `factor`.

In einer factor Variable ist jeder Kategorie eine Zahl zugeordnet (z.B. 1 = männlich, 2 = weiblich).

Um Faktor Variablen zu erstellen, machen wie einen Vorgang, den man **Kodieren** nennt und das geht so:

Wir haben einen Vektor mit Codes 1 und 2 für männlich und weiblich vorliegen:

```
geschlecht = c(1, 2, 2, 1, 2)
# Das Ergebnis ausgeben
geschlecht
```

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

In dieser Form erkennt R diesen Vektor als numerische Variable. Um Sie in einen Faktor umzuwandeln, definieren wir die Zahlen (1 und 2) als `levels` des Faktors und geben dann jedem level einen Namen (`labels`):

```
geschlecht = factor(geschlecht, levels = c(1,2), labels = c("männlich", "weiblich"))
# Das Ergebnis ausgeben
geschlecht
```

```
## [1] männlich weiblich weiblich männlich weiblich
## Levels: männlich weiblich
```

Das Ergebnis ist eine codierte Faktorvariable. Wenn wir Sie uns ausgeben lassen erhalten wir unter den Merkmalsausprägungen eine Liste mit den einzelnen Kategorien (levels) des Faktors.

R wird uns für Faktoren alle Ergebnisse nach der **Reihenfolge** der levels anzeigen. Wenn wir keine Faktorvariable haben, sondern eine character Variable funktioniert die Reihenfolge immer alphabetisch.

5.2.3 Vektor Indizierung

Manchmal möchten wir wieder einen einzelnen Skalar auswählen, der als Teil von einem Vektor gespeichert ist. Diese **Auswahl** eines Einzelelements nennt man **Indizierung**. Die Auswahl eines kleineren Objekts aus einem größeren Objekt funktioniert in R immer mit `[]`.

Benötigen wir aus einem Vektor z.B. genau den 3. Skalar, schreiben wir einfach eine 3 in eckige Klammern hinter den Vektor.

```
char.vec = c("Freud", "Wundt", "Bandura", "Watson", "Jung")
# Das Ergebnis ausgeben
char.vec[5]
```

```
## [1] "Jung"
```

5.3 Matrizen und Dataframes

In der Psychologie beobachten wir für unsere Studien fast immer mehr als eine Variable. Wir könnten diese alle in einzelnen Vektoren speichern und uns die Objektnamen merken. Z.B.

```
Name = c("Max", "Maja", "Mia", "Moritz", "Markus")
Alter = c(20, 31, 25, 34, 51)
Diagnose = c("Depression", "Zwangsstörung", "Depression", "Soziale Phobie", "Depression")
```

5.3.1 Erstellen von Datenmatrizen

Praktischer ist es, die einzelnen Vektoren in Tabellenform zu speichern, der **Datenmatrix**. In R heißen Datenmatrizen `data.frame`. Wir können die Vektoren folgendermaßen zu einem data.frame kombinieren:

```
df = data.frame(Name, Alter, Diagnose)
# Das Ergebnis ausgeben
df
```

```
##     Name Alter       Diagnose
## 1    Max    20     Depression
## 2   Maja    31  Zwangsstörung
## 3    Mia    25     Depression
## 4 Moritz    34 Soziale Phobie
## 5 Markus    51     Depression
```

Wie in jeder Datenmatrix entsprechen die **Zeilen** den einzelnen Personen (Fällen) und die **Spalten** den Variablen.

R bezeichnet Zeilen und Spalten als **rows** und **columns**. Wollen wir z.B. wissen, wie viele Zeilen der data.frame hat, können wir `nrow()` benutzen. Für die Anzahl der Spalten nehmen wir `ncol()`:

```
nrow(df)
```

```
## [1] 5
```

```
ncol(df)
```

```
## [1] 3
```

Wenn wir die einzelnen Vektoren nicht bereits vorher definiert haben, können wir auch alles in einem Schritt machen. Das Ergebnis ist das gleiche:

```
df = data.frame("Name" = c("Max", "Maja", "Mia", "Moritz", "Markus"),
                "Alter" = c(20, 31, 25, 34, 51),
                "Diagnose" = c("Depression", "Zwangsstörung", "Depression", "Soziale Ph
                )
# Das Ergebnis ausgeben
df
```

```
##     Name Alter       Diagnose
## 1    Max    20     Depression
## 2   Maja    31  Zwangsstörung
## 3    Mia    25     Depression
## 4 Moritz    34 Soziale Phobie
## 5 Markus    51     Depression
```

Wollen wir wieder eine einzelne Variable aus dem benutzen, können wir diese über das `$` Zeichen anwählen:

```
df$Alter
```

```
## [1] 20 31 25 34 51
```

5.3.2 Indizierung

Wollen wir aus dem data.frame wieder einzelne Elemente benutzen, nutzen wir wieder die Indizierung. Auch hier brauchen wir die `[]`. Da wir im data.frame Zeilen und Spalten haben, brauchen wir eine Möglichkeit beides auszuwählen, wie ein Curser der von links nach rechts, bzw. von oben nach unten läuft.

Wir trennen dafür unsere `[]` mit einem Komma `[,]`. Alles was **links vom Komma** steht bezieht sich auf Zeilen alles **rechts vom Komma** bezieht sich auf Spalten.

Lassen Sie uns einmal die Zelle in der 1. Zeile (also die 1. Person) und der 3. Variable auswählen:

```
df[1,3]
```

```
## [1] Depression
## Levels: Depression Soziale Phobie Zwangsstörung
```

Lassen wir die Zahl vor dem Komma weg, bekommen wir alle Werte aus der Spalte:

```
df[,3]
```

```
## [1] Depression     Zwangsstörung  Depression     Soziale Phobie Depression
## Levels: Depression Soziale Phobie Zwangsstörung
```

Lassen wir die Zahl nach dem Komma weg, bekommen wir alle Werte aus der Reihe:

```
df[1,]
```

```
##   Name Alter   Diagnose
## 1  Max    20 Depression
```

Chapter 6

Daten erstellen

6.1 Manuell

Die manuelle Eingabe von Daten erfolgt über die `c()` Funktion. Mit ihrer Hilfe können wir Skalare zu Vektoren verbinden...

```
a = c(1, 2, 4, 6, 1)
```

...mehrere Vektoren aneinanderhängen...

```
a = c(1, 2, 4, 6, 1)
b = c(2, 3)
c = c(a, b)
c
```

```
## [1] 1 2 4 6 1 2 3
```

...und Vektoren gleicher Länge zu data.frames kombinieren:

```
daten = data.frame(aufmerksamkeit = c(58, 46, 29, 51),
                   gedaechtnis = c(22, 67, 22, 31))
daten
```

```
##   aufmerksamkeit gedaechtnis
## 1             58          22
## 2             46          67
## 3             29          22
## 4             51          31
```

6.2 Automatisch

Wir haben bereits die `c()` Funktion gelernt.

Die `c()` Funktion ist die einfachste Art einen Vektor zu erstellen, sie ist aber vermutlich auch die umständlichste. Stellen Sie sich zum Beispiel vor, Sie wollen einen Vektor erstellen, der alle Zahlen von 0 bis 100 enthält. Diese Zahlen wollen Sie definitiv nicht alle in die Klammer von `c()` eintippen.

Glücklicherweise hat R viele eingebaute Funktionen, um leicht automatisch numerische Vektoren zu erstellen.

Lassen Sie uns mit dreien davon starten`a:b`, `seq()`, and `rep()`:

Funktion	Beispiel	Ergebnis
`c(a, b, ...)`	`c(1, 5, 9)`	1, 5, 9
`a:b`	`1:5`	1, 2, 3, 4, 5
`seq(from, to, by, length.out)`	`seq(from = 0, to = 6, by = 2)`	0, 2, 4, 6
`rep(x, times, each, length.out)`	`rep(c(7, 8), times = 2, each = 2)`	7, 7, 8, 8, 7, 7, 8, 8

6.3 Zufällig

In R haben Sie die möglichkeit Daten anhand einer Wahrscheinlichkeitsverteilung zu simulieren.

Wollen wir beispielsweise eine normalverteilte Variable mit zufälligen Werten erstellen, können wir die `rnorm()` Funktion nutzen. Dafür müssen wir lediglich angeben, wie viele Werte wir haben wollen (`n`) und welchen Mittelwert (`mean`) und welche Standardabweichung (`sd`) die Verteilung haben soll:

```
rnorm(n = 20, mean = 0, sd = 1)
```

```
##  [1]  0.68071251  0.01081914  0.54228901  0.88462382  0.40850961 -0.71401971
##  [7]  1.00804396 -0.74217360 -0.39099623  0.63593396 -0.26811788 -0.25618546
## [13] -0.49961666 -0.61526780  0.31446141  0.21037250  1.01921704  0.70867677
## [19]  0.89245593  1.46940607
```

Chapter 7

Daten importieren und speichern

In diesem Kapitel werden wir die Grundlagen der R-Objektverwaltung behandeln. Es wird erläutert, wie Sie neue Objekte, z. B. externe Datensätze, in R laden, wie Sie die bereits vorhandenen Objekte verwalten und wie Sie Objekte aus R in externe Dateien exportieren, die Sie mit anderen Personen teilen oder für Ihre eigene zukünftige Verwendung speichern können.

7.1 Funktionen zur ORganisation des Workspace

In diesem Kapitel werden wir einige hilfreiche Funktionen zur Verwaltung Ihres Arbeitsbereichs vorstellen:

Code	Description
`ls()`	Alle Objekte im aktuellen Arbeitsbereich anzeigen
`rm(x, y, ..)`	Entfernt die Objete `y`, `y`… aus dem Arbeitsbereich
`rm(list = ls())`	Entfernt *alle* Objekte aus dem Arbeitsbereich
`getwd()`	Zeigt das aktuelle Arbeitsverzeichnis an
`setwd(file = "dir)`	Wechselt das Arbeitsverzeichnis zu einem bestimmten Dateipfad
`list.files()`	Zeigt die Namen aller Dateien im Arbeitsverzeichnis an

Code	Description
`write.table(x, file = "mydata.txt", sep)`	writes the object `x` to a text file called `mydata.txt`. Define how the columns will be separated with `sep` (e.g.; `sep = ","` for a comma–separated file, and `sep = \t"` for a tab–separated file).
`save(x, y, .., file = "meineDaten.RData)`	Speichert Objekte `x`, `y`, … in das R Objekt `meineDaten.RData`
`save.image(file = "meineSession.RData")`	Speichert *alle* Objekte aus dem Arbeitsbereich nach `meineSession.RData`
`load(file = "meineDaten.RData")`	Läd Objekte in der Datei `meineDaten.RData`
`foreign::read.spss("Daten.sav")`	Läd den SPSS Datensatz `Daten.sav`
`read.csv("Daten.csv")`	Läd den csv Datensatz `Daten.csv`
`readxl::read_xlsx("Daten.xlsx")`	Läd den Excel Datensatz `Daten.xlsx`

Ihr Computer ist ein Labyrinth aus Ordnern und Dateien. Wenn Sie außerhalb von R eine bestimmte Datei öffnen möchten, öffnen Sie wahrscheinlich ein Explorer-Fenster, mit dem Sie die Ordner auf Ihrem Computer visuell durchsuchen können. Oder Sie wählen die zuletzt geöffneten Dateien aus oder geben den Namen der Datei in ein Suchfeld ein, um den Computer die Suche für Sie übernehmen zu lassen. Während dieses Vorgehen normalerweise für nichtprogrammierende Aufgaben funktioniert, ist es für R ein No-Go. Das Hauptproblem ist, dass Sie bei all diesen Methoden Ihre Ordner visuell durchsuchen und die Maus bewegen müssen, um Ordner und Dateien auszuwählen, die dem Gesuchten entsprechen. Wenn Sie in R programmieren, müssen Sie alle Schritte in Ihren Analysen so spezifizieren, dass sie von anderen und von Ihnen selbst leicht nachvollzogen werden können. Das bedeutet, dass Sie nicht einfach sagen können: “Finde diese eine Datei, die ich mir vor einer Woche gemailt habe” oder “Suche nach einer Datei, die so aussieht wie”MeinFoto.jpg”. Stattdessen müssen Sie in der Lage sein, R-Code zu schreiben, der R genau sagt, wo wichtige Dateien zu finden sind - entweder auf Ihrem Computer oder im Internet.

Um diese Aufgabe zu erleichtern, verwendet R Arbeitsverzeichnisse.

7.2 Arbeitsverzeichnis (Working Directory)

Das Arbeitsverzeichnis ist lediglich ein Dateipfad auf Ihrem Computer, der den Standardspeicherort aller Dateien festlegt, die Sie in R einlesen oder aus R heraus speichern. Mit anderen Worten, ein Arbeitsverzeichnis ist wie eine kleine Kiste irgendwo auf Ihrem Computer, die an ein bestimmtes Analyseprojekt gebunden ist. Wenn Sie R auffordern, einen Datensatz zu importieren wird davon ausgegangen, dass sich die Datei in Ihrem Arbeitsverzeichnis befindet.

Sie können zu jedem Zeitpunkt nur ein Arbeitsverzeichnis aktiv haben. Das aktive Arbeitsverzeichnis wird als Ihr aktuelles Arbeitsverzeichnis bezeichnet.

7.3 Working Environment

Der Arbeitsbereich (auch als Arbeitsumgebung bezeichnet) enthält alle Objekte und Funktionen, die Sie entweder in der aktuellen Sitzung definiert oder aus einer früheren Sitzung geladen haben. Als Sie RStudio zum ersten Mal starteten, war die Arbeitsumgebung leer, da Sie keine neuen Objekte oder Funktionen erstellt hatten. Wenn Sie jedoch neue Objekte und Funktionen mit dem Zuweisungsoperator = definiert haben, wurden diese neuen Objekte in Ihrer Arbeitsumgebung gespeichert. Wenn Sie RStudio nach der Definition neuer Objekte schlossen, erhielten Sie wahrscheinlich eine Meldung mit der Frage "Save workspace image…?". Damit möchte RStudio Sie fragen, ob Sie alle derzeit in Ihrem Arbeitsbereich definierten Objekte als Bilddatei auf Ihrem Computer speichern möchten.

```
# getwd()
```

7.4 Daten importieren

7.5 Daten speichern