Datenanalyse mit R

3 Daten und Datentypen

Tobias Wiß, Carmen Walenta und Felix Wohlgemuth

19.03.2020

Was haben wir letzte Woche gelernt?

Variablen sind Objekte in denen einzelne Werte oder auch eine Reihe von Werten gespeichert werden können

- Die Zuteilung eines Wertes zu einer Variable geschieht mit `<-`
- Bei mehreren Werten müssen diese mit `c()` verbunden werden, um dann einer Variable zugeteilt zu werden, zB: `x <- c(1,2,3,4)`
- Das ist ein Vektor mit der Länge 4

```
x <- c(1,2,3,4)

length(x)
```

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

Was haben wir letzte Woche gelernt?

- Vektoren können in Rechnungen wie einzelne Werte verwendet werden. zB: `x * 2` hier werden alle Werte in x mit 2 multipliziert

```
x
```

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

```
x * 2
```

```
## [1] 2 4 6 8
```

(Wenn Sie den Inhalt eines Vektors wissen möchten, dann geben Sie einfach den Namen des Vektors in die Konsole ein oder führen Sie den Namen im Skript aus)

Was haben wir letzte Woche gelernt?

- Wenn zwei Vektoren gleicher länger multipliziert oder summiert werden, wird der erste Wert mit dem ersten Wert verechnet, der zweite Wert mit dem zweiten Wert usw.

```
x <- c(1,2,3,4)
y <- c(10,9,8,7)

x * y
```

```
## [1] 10 18 24 28
```

```
2 * (x - y)
```

```
## [1] -18 -14 -10  -6
```

Was haben wir letzte Woche gelernt?

- Vektoren unterschiedlicher Länge können nicht einfach so verechnet werden
- Bei unterschiedlicher Länger verwendet R nachdem alle Werte des kürzeren Vektors verrechnet sind wieder den ersten Wert, dann den zweiten Wert usw.

```
z <- c(5,6,12)
q <- 12
r <- c(23, 34, 33, 2, 6, 9)
x
```

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

```
x + z
```

```
## Warning in x + z: longer object length is not a multiple of shorter object
## length
```

```
## [1]  6  8 15  9
```

Was haben wir letzte Woche gelernt?

Außnahmen:

- Vektoren mit der Länge 1
- wenn die Länge des einen Vektors ein Vielfaches der Länge des anderen Vektors ist

```
z + q
```

```
## [1] 17 18 24
```

```
z + r
```

```
## [1] 28 40 45  7 12 21
```

Was haben wir letzte Woche gelernt?

Funktionen fassen mehrere Schritte einer Berechnung zusammen

- Funktionen sind meist Teile eines Paketes
- zB berechnet die `sum()` Funktion die Summe eines Vektors, dazu geben sie den Namen des Vektors zwischen den Klammern an

```
sum(r)
```

```
## [1] 107
```

- `length()` gibt die die Länge eines Vektor aus

```
length(x)
```

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

Was haben wir letzte Woche gelernt?

- Mit der Funktion `help()` wird die Erklärung der Funktion in RStudio angezeigt
- In der Erklärung wird auch die Syntax der Funktion erklärt und welche Grundeinstellung die Funktion hat

```
help(sum)
```

Description: sum returns the sum of all the values present in its arguments
Usage: sum(..., na.rm = FALSE)
Arguments
... numeric or complex or logical vectors
na.rm logical Should missing values (including NaN) be removed?

- `sum()` hat die Grundeinstellung, dass fehlende Werte (NA) bei der Summenberechnung nicht entfernt werden
- Sie können die Einstellung bei Verwendung der Funktion ändern zB `sum(r, na.rm = TRUE)` da aber in `r` keine fehlende Werte sind, hat das kein Effekt

Was haben wir letzte Woche gelernt?

Pakete sind Bündel von Funktionen (teilweise inklusive Daten) und erweitern die Grundfunktionen von R. Sie werden von den Nutzer_innen erstellt und auf CRAN kostenfrei veröffentlicht.

- Pakete können von CRAN mit der Funktion `install.packages("PAKETNAME")` heruntergeladen und installiert werden
- Installieren müssen Sie das Paket nur einmal
- Mit `library(PAKTENAME)` wird das Paket zur Verwendung geladen
- Zur Verwendung der Funktionen eines Paketes müssen Sie das Paket am Anfang des Skriptes laden

Falls Sie noch Fragen haben, nutzen Sie das Forum auf moodle und unterstützen Sie Ihre Kolleg*innen mit Ihrem Wissen!

Forum für R & RStudio Fragen

Hier können Sie alle Fragen, die Sie zu R und RStudio haben, stellen und auch Probleme diskutieren. Wir werden auf Ihre Fragen antworten. Bitte unterstützen Sie auch Ihre Kolleg*innen mit Ihrem Wissen. Falls Sie die Lösung für ein Problem haben, dann antworten Sie einfach unter der Frage ihrer Kolleg*in.

Objektklassen in R

R kann mit einer Vielzahl von verschiedenen Objektklassen arbeiten

- **reele Zahlen (numeric)**

```
menue_1 <- 5.80
class(menue_1)
```

```
## [1] "numeric"
```

R ist es egal ob der Vektor einen Wert oder mehrere Werte hat

```
mensa_preis <- c(4.30, 5.80, 2.90, 8.10)
class(mensa_preis)
```

```
## [1] "numeric"
```

Objektklassen in R

- **ganze Zahlen** (integer)

```
# ganze Zahlen werden R mit L dargestellt
ganze_zahlen <- c(-8L, 35L, -44L)
class(ganze_zahlen)
```

```
## [1] "integer"
```

Objektklassen in R

Mit beiden numerischen Objektklassen kann R rechnen

```
mean(mensa_preis)
```

```
## [1] 5.275
```

```
mean(ganze_zahlen)
```

```
## [1] -5.666667
```

Objektklassen in R

Aber es gibt nicht nur numerische Objektklassen

- Variablen können auch Wörter oder **Texte** (character) enthalten

```
# Text wird in R mit " " dargestellt
teacher <- c("Tobias Wiß", "Carmen Walenta", "Felix Wohlgemuth")
class(teacher)
```

```
## [1] "character"
```

Objektklassen in R

- R kann aber auch mit den **logischen** Ausdrücken `TRUE` und `FALSE` arbeiten

```
# sechs mal gewürfelt mit zwei Würfeln
# hab ich bei dem Wurf einen Pasch gewürfelt?
pasch <- c(TRUE, FALSE, TRUE, TRUE, FALSE, TRUE)
class(pasch)
```

```
## [1] "logical"
```

Objektklassen in R

Auch mit nicht-numerischen Werten kann man in R rechnen

```
# Gibt es eine/n Dozent*in mit dem Namen Felix Wohlgemuth?
teacher == "Felix Wohlgemuth"
```

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

```
# Wie lang sind die Namen der Lehrenden?
nchar(teacher)
```

```
## [1] 10 14 16
```

```
# Leerzeichen sind auch Zeichen
```

Objektklassen in R

```
# wie of habe ich einen Pasch gewürfelt?
sum(pasch)
```

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

```
# bei welchen Würfen hab ich einen Pasch gewürfelt
which(pasch == TRUE)
```

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

```
# bei welchen Würfen hab ich keinen Pasch gewürfelt
which(pasch == FALSE)
```

```
## [1] 2 5
```

Einschub

In R gibt es eine Reihe von logischen Operatoren wie ==

Operatoren	Beschreibung
==	gleich
!=	ungleich
<	kleiner
>	größer
<=	kleiner gleich
>=	größer gleich

Zur Verknüpfung von logischen Tests können Sie | für *oder* und & für *und* verwenden

Objektklassen in R

Bisher hatten wir nur Vektoren mit einer Sorte von Werten, aber in Vektoren können auch Werte unterschiedlicher Klassen gespeichert werden.

```
# ein Vektor der den Namen des Kurses, die LVA-Nummer
# und die Anzahl der Studierende enthält
kurs_fampol <- c("VU Politikfelder - Familienpolitik", 229216, 35)
class(kurs_fampol)
```

```
## [1] "character"
```

Die Klasse des ganzen Vektors ist "character" also ein Text, obwohl die einzelne Elemente eigentlich "character" oder "numeric" sind.

```
class("VU Politikfelder - Familienpolitik")
```

```
## [1] "character"
```

```
class(c(229216, 35))
```

```
## [1] "numeric"
```

Datentypen in R

Bisher haben wir mit dem Datentyp "Vektor" mit unterschiedlichen Längen und Werten unterschiedlicher Klassen gearbeitet.

Es gibt in R aber noch weitere Datentypen:

- Vektoren
- Matrizen
- Arrays
- Dataframes
- Listen

Wir werden nur mit **Vektoren** und **Dataframes** arbeiten.

Datentypen in R

Ein **Vektor** ist eigentlich eine geordnete Folge von Werten und kann entweder aus einer Sorte von Klassen oder gemischten Klassen von Werten bestehen.

```
# der Preis des menue_1 ist ein numerischer Vektor mit einem Wert
# also der Länge 1
str(menue_1)
```

```
##  num 5.8
```

```
# die Aufzeichnung der Mensapreise ist ein numerischer Vektor
# mit mehreren Werten der Länge 4
str(mensa_preis)
```

```
##  num [1:4] 4.3 5.8 2.9 8.1
```

Datentypen in R

```
# meine Aufzeichnung über meine Paschwürfe ist ein Vektor mit
# logischen Werten und der Länge 6
str(pasch)
```

```
##  logi [1:6] TRUE FALSE TRUE TRUE FALSE TRUE
```

```
# da Vektoren geordnet sind, kann ich die Position
# der Päsche abfragen
which(pasch == TRUE)
```

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

Sortenreine Vektoren (atomare Vektoren) sind die Basis der meisten Datentypen in R.

Datentypen in R

Faktoren sind Vektoren die nur vorab festgelegte Elemente speichern. Für R sind Faktoren reine Vektoren mit ganzen Zahlen und jeder ganzen Zahl ist ein Text zugeordnet. Da ganze Zahlen gereiht werden können, werden Faktoren für ordinal skalierte Variablen verwendet.

```
# War mein Tee in den letzten 7 Tagen zu heiß, richtig oder zu kalt?
tea <- factor(c("hot", "hot", "cold", "right", "hot", "right", "hot"),
              levels = c("hot", "right", "cold"))
attributes(tea)
```

```
## $levels
## [1] "hot"   "right" "cold"
##
## $class
## [1] "factor"
```

Sie können den Inhalt innerhalb der Klammern () einer Funktion auf mehrere Zeilen aufteilen. Vergessen Sie nicht die Klammern zu schließen

Datentypen in R

Für R ist "tea" ein Vektor mit den gereihten levels "hot", "right" und "cold". Dem level "hot" ist die Zahl 1, dem level "right" die Zahl 2 und dem level "cold" die Zahl 3 zugeordnet.

```
attributes(tea)
```

```
## $levels
## [1] "hot"   "right" "cold"
##
## $class
## [1] "factor"
```

Bei der Umwandlung des Faktor-Vektors in einen numerischen Vektor verschwinden die levels und die ganzen Zahlen bleiben übrig.

```
as.numeric(tea)
```

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

Datentypen in R

Vektoren sind eindimensional, d.h. die Werte eines Vektors sind in Reihe geordnet, wie zB Ländernamen.

```
country <- c("Frankreich", "Belgien", "Finnland", "Dänemark", "Italie
country
```

```
## [1] "Frankreich" "Belgien"    "Finnland"   "Dänemark"   "Italien"
## [6] "Österreich"
```

Oder sechs mal hintereinander das gleiche Jahr 2018.

```
year <- rep(2018, times = 6)
year
```

```
## [1] 2018 2018 2018 2018 2018 2018
```

Um die Jahreszahl 2018 jedem Land zuzuordnen, brauchen wir eine zweite Dimension.

Datentypen in R

Dataframes sind vergleichbar mit einer excel-Tabelle bei der jede Spalte eine eigene Variable ist. In R können Vektoren gleicher Länge zu einem Dataframe verbunden werden, zB die Ländernamen mit den Jahreszahlen.

```
socx <- data.frame(country, year)
socx
```

```
##      country year
## 1 Frankreich 2018
## 2    Belgien 2018
## 3   Finnland 2018
## 4   Dänemark 2018
## 5    Italien 2018
## 6 Österreich 2018
```

R verknüpft die Werte der zwei Vektoren basierend auf ihrer Position innerhalb ihres Vektors.

Datentypen in R

Somit haben wir unsere erste Tabelle in R. Sechs Ländernamen und sechs mal das Jahr 2018. Interessant wird es, wenn wir zu den Ländern die öffentlichen Ausgaben für Sozialpolitik hinzufügen.
(Quelle: http://www.oecd.org/social/expenditure.htm)

```
public_socx <- c(31.2, 28.9, 28.7, 28.0, 27.9, 26.9)
socx <- data.frame(country, year, public_socx)
socx
```

```
##       country year public_socx
## 1 Frankreich 2018        31.2
## 2    Belgien 2018        28.9
## 3   Finnland 2018        28.7
## 4   Dänemark 2018        28.0
## 5    Italien 2018        27.9
## 6 Österreich 2018        26.9
```

Wie auch schon vorher ist die Reihenfolge der Werte des "public_socx" Vektors wichtig. Würden die Werte in einer anderen Reihenfolge sein, wird dem Land die falschen Sozialausgaben zugeordnet.

Datentypen in R

Jede Spalte in einem Dataframe ist wie ein eigener Vektor und hat deshalb auch eine eigene Objektklasse.

```
str(socx)
```

```
## 'data.frame':    6 obs. of  3 variables:
##  $ country    : Factor w/ 6 levels "Belgien","Dänemark",..: 4 1 3 2 5 6
##  $ year       : num  2018 2018 2018 2018 2018 ...
##  $ public_socx: num  31.2 28.9 28.7 28 27.9 26.9
```

Datentypen in R

Ein Faktor-Vektor für die Ländernamen? Hier wäre die Objektklasse "character" praktischer, da Ländernamen nicht ordinal skaliert sind. Mit `as.character()` können Vektoren und Spalten eines Dataframes umgewandelt werden.

```
# Entweder bevor Sie das Dataframe erstellen
country <- as.character(country)
socx <- data.frame(country, year, public_socx)

# Oder im schon existierenden Dataframe mit dem $Variablennamen
socx$country <- as.character(socx$country)

str(socx)
```

```
## 'data.frame':    6 obs. of  3 variables:
##  $ country    : chr  "Frankreich" "Belgien" "Finnland" "Dänemark" ...
##  $ year       : num  2018 2018 2018 2018 2018 ...
##  $ public_socx: num  31.2 28.9 28.7 28 27.9 26.9
```

Datentypen in R

Mit `rbind()` können Sie einem Dataframe neue Zeilen hinzufügen. In unserem Fall sind das die Sozialausgaben von Schweden im Jahr 2018 und die Sozialausgaben von Deutschland im Jahr 2018.

```
socx <- rbind(socx,
      c("Schweden", 2018, 26.1),
      c("Deutschland", 2018, 25.1))
socx
```

```
##       country year public_socx
## 1  Frankreich 2018        31.2
## 2     Belgien 2018        28.9
## 3    Finnland 2018        28.7
## 4    Dänemark 2018          28
## 5     Italien 2018        27.9
## 6  Österreich 2018        26.9
## 7    Schweden 2018        26.1
## 8 Deutschland 2018        25.1
```

Datentypen in R

Leider hat R beim Hinzufügen von Schweden und Deutschland die Variablen `year` und `public_socx` zu "character" geändert.

```
str(socx)
```

```
## 'data.frame':    8 obs. of  3 variables:
##  $ country    : chr  "Frankreich" "Belgien" "Finnland" "Dänemark" ...
##  $ year       : chr  "2018" "2018" "2018" "2018" ...
##  $ public_socx: chr  "31.2" "28.9" "28.7" "28" ...
```

Das kann mit `as.numeric()` rückgängig gemacht werden.

```
socx$year <- as.numeric(socx$year)
socx$public_socx <- as.numeric(socx$public_socx)
```

Daten indizieren

Bei der Umwandlung der `country` Variable mussten wir R sagen auf welche Variable im Dataframe `socx` sich die Funktion `as.character()` bezieht und in welcher Variable die Umwandlung gespeichert werden soll.

Das haben wir mit `socx$country` gemacht. Damit haben wir die `country` Variable indiziert.

```
socx$country
```

```
## [1] "Frankreich"  "Belgien"     "Finnland"    "Dänemark"    "Italien"
## [6] "Österreich"  "Schweden"    "Deutschland"
```

```
# $ können wir auch in Rechnungen verwenden
# dafür muss die Variable numerisch sein!
mean(socx$public_socx)
```

```
## [1] 27.85
```

Daten indizieren

Es gibt noch weitere Wege ganze Variablen oder einzelne Werte zu indizieren:

- Anhand der Position innerhalb eines Vektors mit `[]`

```
# ich möchte wissen ob am 3. Tag mein
# Tee heiß, kalt oder richtig war
tea
```

```
## [1] hot   hot   cold  right hot   right hot
## Levels: hot right cold
```

```
tea[3]
```

```
## [1] cold
## Levels: hot right cold
```

Daten indizieren

```
# Wie war der Tee am 1. und am 7.Tag?
tea[c(1,7)]
```

```
## [1] hot hot
## Levels: hot right cold
```

```
# man kann auch alle andere Tage mit - auschließen
tea[-c(2,3,4,5,6)]
```

```
## [1] hot hot
## Levels: hot right cold
```

```
# eine Reihe von Zahlen kann mit dem Start:Ende verkürzt werden
tea[-c(2:6)]
```

```
## [1] hot hot
## Levels: hot right cold
```

Daten indizieren

- Anhand von logischen Bedingungen

```
# Nur alle Beobachtungen wo mein Tee richtig war
tea[tea == "right"]
```

```
## [1] right right
## Levels: hot right cold
```

```
# das kann ich nutzen, um die Anzahl der Tage
# an denen mein Tee richtig war zu zählen
length(tea[tea == "right"])
```

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

Daten indizieren

Dataframes sind zweidimensional, deshalb brauch R die Information zur Zeilen- und Spaltenposition

```
#  um den Wert in der 7. Zeile und 3. Spalte auszuwählen
socx[7,3]
```

```
## [1] 26.1
```

```
# um die Werte in der 7. & 8. Zeile und 3. Spalte auszuwählen
socx[c(7,8), 3]
```

```
## [1] 26.1 25.1
```

```
# um die komplette 7. Zeile auszuwählen
socx[7, ]
```

```
##    country year public_socx
## 7 Schweden 2018        26.1
```

Daten indizieren

```
# um die kompette 3. Spalte auszuwählen
socx[, 3]
```

```
## [1] 31.2 28.9 28.7 28.0 27.9 26.9 26.1 25.1
```

R gibt uns die Werte der dritten Variable in unserem Dataframe. Richtig übersichtlich ist das nicht.

Am besten wählen wir die Variable mit ihrem Namen `public_socx` und `$` aus.

```
socx$public_socx
```

```
## [1] 31.2 28.9 28.7 28.0 27.9 26.9 26.1 25.1
```

Übung

Diese Woche erstellen Sie Ihr erstes Dataframe und berechnen die durchschnittlichen Sozialausgaben von 3 Ländern.

- Gehen Sie auf http://www.oecd.org/social/expenditure.htm und wählen Sie 3 Länder aus
- Erstellen Sie ein R Skript mit Ihrem # Namen und # Übung 3 am Anfang
- Erstellen Sie einen Vektor der die Namen der 3 Länder ihre Wahl beinhaltet
- Erstellen Sie einen zweiten Vektor der die Sozialausgaben der drei Länder beinhaltet
- Erstellen Sie ein Dataframe mit `data.frame()` und den Namen Ihrer zwei Vektoren
- Berechnen Sie die durschnittlichen Ausgaben mit `mean()` und in den Klammern den Variablennamen der Sozialausgaben

Laden Sie das Skript bis 25.03. 12:00 auf moodle hoch

Falls Sie noch Fragen haben, nutzen Sie das Forum auf moodle und schauen Sie in die Lektüre für diese Woche

Forum für R & RStudio Fragen

Hier können Sie alle Fragen, die Sie zu R und RStudio haben, stellen und auch Probleme diskutieren. Wir werden auf Ihre Fragen antworten. Bitte unterstützen Sie auch Ihre Kolleg*innen mit Ihrem Wissen. Falls Sie die Lösung für ein Problem haben, dann antworten Sie einfach unter der Frage ihrer Kolleg*in.