Programmieren mit R für Einsteiger

3. Tabellen / 3.3 Dateien einlesen

Berry Boessenkool 2022-02-25 11:40

Dateien einlesen (Daten importieren)

Generelle Syntax:

```
eine_tabelle <- read.table(file="dateiname.txt")
```

Eingelesene Daten immer prüfen!
(90% der Fehler Downstream passieren, weil das ignoriert wird):

```
daten_objekt_name <- read.table("Datei.txt")
str(daten_objekt_name)
```

```
# Häufig benötigte Argumente:
read.table("Datei.txt", header=TRUE, dec=",", sep="\t")
```

Häufige Argumente für `read.table`

```
read.table(file="Dateiname.txt",  # kann auch URL sein
header=TRUE,        # Erste Zeile als Spaltennamen lesen
dec=",",            # Komma als Dezimaltrennzeichen
sep="_",            # Unterstrich als Spaltentrenner ("\t" -> tabstop)
fill=TRUE,          # unvollständige Zeilen mit NAs am Ende füllen
skip=12,            # Erste 12 Zeilen ignorieren (zB mit Metadaten)
comment.char="%", # Zeilen ab % ignorieren (Standard: # wie in R Code)
na.strings=c(-999, "NN"), # Kennzeichnung Fehlwerte
stringsAsFactors=FALSE,    # Zeichenketten nicht in factors umwandeln
text="1,2,3",              # kleiner Beispieldatensatz im Skript
...)                       # Weitere Argumente, siehe ?read.table
```

Der `stringsAsFactors` Default ist `FALSE` seit R Version 4.0.0 (2020-04-24).

```
read.table(header=TRUE, sep=",", text="
Beispiel, Spalte
Sinn, 42
Bond, 007")
##   Beispiel Spalte
## 1     Sinn     42
## 2     Bond      7
```

Alternativen zu `read.table`

- `read.csv()`: **c**omma **s**eparated **v**alues (`read.table` mit anderen Standardwerten)
- `read.fwf()`: feste Spaltenbreiten (**f**ixed **w**idth **f**ormatted data)
- `readLines()`: Zeilen als Zeichenketten-Vektor einlesen
- `scan()`: selten benötigt (auch im Kern von read.table verwendet)

Komplexe Dateien:

- `readxl::read_excel()`: Exceldateien, siehe github.com/tidyverse/readxl
- `readBin()`: für binäre Dateien
- `raster::raster` / `rgdal::readGDAL`: Geodaten (grd, asc, tif)
- `ncdf4::nc_open()` + `ncdf4::ncvar_get`: NetCDF Dateien

Häufige Fehlermeldungen mit `read.table`

Error in scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings, : line _ did not have _ elements	`header`, `sep` oder `fill` richtig angeben.
Warning message: In read.table("_.txt", _) : incomplete final line found by readTableHeader on 'C:_.txt'	Manuell am Ende der Datei einen Zeilenumbruch einfügen (`ENTER`). line break = line feed (LF) = newline = carriage return (CR).
`str(gelesenesDataframe)` zeigt "factor" in manchen numerischen Spalten	Datei prüfen! ggf. `dec` richtig angeben

Schwere von Fehlermeldungen:
error: Funktion bricht ab, Lösung notwendig
warning: Funktion macht ggf. was falsches. Nur ignorieren, wenn Grund bekannt aber nicht lösbar.

```
# Daten mit Tausenderzeichen "3,590.18" einlesen:
df$spalte <- as.numeric( gsub(",", "", df$spalte) )
```

```
# Riskant: Kommas manuell ersetzen:
df$spalte <- as.numeric( gsub(",", ".", df$spalte) )
```

Ordner-spezifische Dateien nutzen

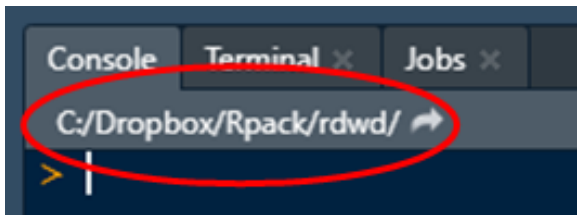
R liest und schreibt Dateien im Arbeitsverzeichnis (working directory).
Quick and dirty:

```
setwd("C:/Users/berry/Projekt_XY") # set working directory
```

Windows Nutzer: `\` ist eine Unsitte von Microsoft. Beim Kopieren eines Pfades aus dem Datei-Browser (Explorer) alle ersetzen mit `/`.

Generell **mit Rstudio Projekten arbeiten**, siehe Folie im Abschnitt 1.2.

Aktuelles Arbeitsverzeichnis zeigen mit `getwd()`



Immer relative Dateinamen nutzen, also Dateien die im WD vorliegen:

```
tabelle <- read.table("Unterordner/Datei.txt")
```

```
dir() # Verfügbare Dateinamen im WD anzeigen
```

Ordner managen

Hilfreiche Funktionen:

```r
getwd() # Aktuelles Working Directory (WD) zeigen
setwd("..") # WD eine Ebene höher setzen

dir() # Dateien / Ordner im WD auflisten
dir(recursive=TRUE) # Auch Items in Unterordner anzeigen
dir("../other_subfolder") # in anderem Ordner

file.create() # Dateien erstellen
file.rename() # Dateien umbenennen
file.remove() # Dateien löschen, siehe auch unlink()
file.copy()   # Dateien kopieren
file.exists() # Dateinamen auf Existenz prüfen

dir.create()  # Ordner erstellen
dir.exists()  # Ordnerpräsenz prüfen
```

Daten schreiben

```
write.table(x=mynewdata, file="output.txt",
            quote=FALSE, row.names=FALSE,
            fileEncoding="UTF-8")
```

Randnotiz: Daten niemals manuell ändern:

```
newtable <- edit(oldtable)
# oder auch kürzer:
fix(mytable) # behält nichtmal die alten Daten
```

Das wäre nicht reproduzierbar. Besser ein Skript schreiben:

```
mytable[265, "Temperatur"] <- 17.53 # original 175.3 vermutlich Typo
mytable[1:24, "Messwert"] <- NA # falsche Messmethode am ersten Tag
```

Zusammenfassung

Daten einlesen:

- `setwd`, `dir`, Rstudio projects
- `file.rename`, `file.exists`, `dir.create`, ...
- `read.table(file, header, dec, sep, skip)`, `write.table`
- `read.csv`, `read.fwf`, `readxl::read_excel`, `scan`
- `str`