Daten einlesen mit R

Um Datensätze mit R zu analysieren, müssen diese zuerst eingelesen werden. Je nach **Struktur des Datenfiles** kommen verschiedene Einlese-Befehle zur Anwendung. Es können u.a. Textfiles (.txt, .dat), Datenfiles (.csv) und Excel-Files eingelesen werden.

Import von Daten aus Excel

Excel-Dateien können im Prinzip direkt eingelesen werden. Wir empfehlen jedoch, das Excelfile abzuspeichern als .csv-File (Menu *File - Save as - File Type: CSV*) und dieses mit `read.csv()` einzulesen. Warum?

- Grafiken, Rahmen, Formatierungen verunmöglichen das Einlesen.
- Sie müssen entscheiden, welche Daten aus dem ganzen Excelfile tatsächlich ins Datenfile gehören. Sämtliche Daten müssen in **einem** File stehen, nicht auf mehrere Tabs verteilt.
- Das .csv-File enthält die nackten Daten, die mit jedem Editor (Notepad) angezeigt werden können. So sehen sie genau, was nachher ins R eingelesen wird und können allfällige (Excel-) Fehler (!) noch erkennen.
- Ein .csv-File kann problemlos verschickt werden und bereitet auf keinem Betriebssystem Schwierigkeiten.

Pfadbezeichnungen bei Windows

Windows benutzt bei Pfadangaben den Backslash "\". Beim Einfügen eines Pfades mittels Copy&Paste in einen R-Befehl wie `read.csv()` müssen die Backslashes "\" durch Forwardslashes "/" oder durch Doppelbackslashes "\\" ersetzt werden. Andernfalls gibt R eine Fehlermeldung aus. Pfadbezeichnungen mit Leerzeichen hingegen sind unproblematisch.

Beispiele:

Windows:	R:
C:\WBL\Daten.csv	`read.csv("C:/WBL/Daten.csv", header=T)`
C:\WBL\Daten.csv	`read.csv("C:\\WBL\\Daten.csv", header=T)`
C:\My Documents\Daten Nov07.txt	`read.table("C:/My Documents/Daten Nov07.txt")`

Case Sensitivity

Sämtliche Befehle, Funktionen, Argumente, Pfadangaben in R sind **"case sensitive"**! Es spielt eine Rolle, ob die Buchstaben gross oder klein geschrieben sind. Es ist also Vorsicht geboten beim Abtippen von langen Pfadangaben! (http://stat.ethz.ch/**T**eaching/**D**atasets/**WBL**/**g**ross.dat)

Übersicht

In der folgenden Übersicht finden Sie die gebräuchlichsten R-Befehle zum Einlesen von Daten. Unten werden zu jedem Befehl Details und Beispiele angegeben. Ausführliche Informationen zu den einzelnen Befehlen entnehmen Sie der R-Hilfe mit `?read.csv`.

R-Befehl	geeignet für	wichtige Argumente
`read.table()`	.txt .dat	`header=TRUE/FALSE, sep= , dec= , fill=TRUE`
`read.csv()`	.csv	`header=TRUE/FALSE, sep= , dec=`
`scan()`	.txt .dat ...	`file= , what= , header=T/F, sep= , skip= , fill=T/F`

- `read.table()`

  Für die meisten „kleineren" Datenfiles kann `read.table()` verwendet werden. Für sehr grosse Datenfiles ist `read.table()` zu langsam, weshalb besser `scan()` benützt wird.
  Falls die Bezeichnungen der Variablen in der ersten Datenzeile stehen, wählen Sie `header=TRUE`, ansonsten `header=FALSE`. Das Trennzeichen zwischen den Variablen (Datenfile in einem Editor anschauen!) muss mit dem Argument `sep="..."` angegeben werden. Das Trennzeichen "Tabulator" wählen Sie mit `sep="\t"`. (Space und Tabulator als Trennzeichen werden auch ohne Angabe von `sep` erkannt.) Mit `dec=","` kann das Dezimaltrennzeichen eingegeben werden, falls es nicht "." ist.
  Beim Abspeichern von Excel-Tabellen als .csv-Files kommt es vor, dass einzelne Zeilen am Ende ein (leeres) Feld zu wenig haben, was zu einer Fehlermeldung beim Einlesen führt. Dieser Fehler wird mit dem Argument `fill=TRUE` verhindert.

  Beispiele:

```
------------------------------
| Name Alter Grösse Geschlecht
| Anna 19 167 w
| Paul 22 185 m
| Udo 20 179 m

> d.bsp <- read.table("Pfad/bsp_space.txt", header=TRUE)

-----------------------------------
| Name   Alter   Grösse   Geschlecht
| Anna   19      167      w
| Paul   22      185      m
| Udo    20      179      m

> d.bsp <- read.table("Pfad/bsp_tab.txt", header=TRUE, sep="\t")

## Datenfile in R anzeigen
> d.bsp
  Name Alter Grösse Geschlecht
1 Anna    19    167          w
2 Paul    22    185          m
3  Udo    20    179          m
```

- `read.csv()`

  Wie der Name sagt, ist `read.csv()` für .csv-Files ("Character Separated Values") geeignet. Als Trennzeichen werden neben Komma auch Semikolon, Doppelpunkt und Tabulator verwendet. Default-Wert ist `sep=","`.

  Beispiele:

```
------------------------------
| Name,Alter,Grösse,Geschlecht
| Anna,19,167,w
| Paul,22,185,m
| Udo,20,179,m

> d.bsp <- read.csv("Pfad/bsp_komma.csv", header=TRUE)

------------------------------
| Name;Alter;Grösse;Geschlecht
| Anna;19;167;w
| Paul;22;185;m
| Udo;20;179;m

> d.bsp <- read.csv("Pfad/bsp_semikolon.csv", header=TRUE, sep=";")

## Datenfile in R anzeigen:
> d.bsp
  Name Alter Grösse Geschlecht
1 Anna    19    167          w
2 Paul    22    185          m
3  Udo    20    179          m
```

- `scan()`

  Wenn das Datenfile nicht den oben genannten Strukturen entspricht oder für spezielle Einlesemethoden kann der flexible Befehl `scan()` verwendet werden. Die Prozedur `scan()` liegt den anderen Einlese-Befehlen zugrunde und ist schneller als diese, allerdings auch schwieriger zu bedienen.

  Beispiel 1:
  Einlesen einer Zeitreihe (eindimensionale Daten) mit `scan()`:

```
----------------------------------------
| 16.1; 16.8; 17.5; 18.0; 18.4; 18.5; 19.1

> d.temp <- scan(file="Pfad/bsp_zeitreihe.txt", what="numeric", sep=";")
Read 7 items

## Anzeige des Datenfiles in R:
> d.temp
[1] 16.1 16.8 17.5 18.0 18.4 18.5 19.1
```


Beispiel 2:
Ein Vektor (z.B. Messwerte) kann manuell in der R-Konsole eingegeben werden mit dem folgenden Befehl.

```
> d.messungen <- scan() [ENTER]
1: 2.7  9.2 [ENTER]
3: 8.4 [ENTER]
4: [ENTER]
Read 3 items

## Anzeigen der Messwerte in R:
> d.messungen
[1] "2.7" "9.2" "8.4"
```

Beispiel 3:
`scan()` bietet eine elegante Möglichkeit, um z.B. sehr viele Dateinamen für die spätere Verwendung in R einzulesen. Die Dateinamen können direkt aus einem anderen Programm (z.B. Shell) kopiert werden. Das Argument `what=""` gibt an, dass Characters statt Zahlen angegeben werden sollen.
Die **letzte Zeile** muss **eine Leerzeile** sein!

```
t.filenames <- scan(what="")
Results_Exp1_5.Juni06.xls
Results_Exp2_7.Juni06.xls
Results_Exp3_17.Juni06.xls
resultate_exp6_4.Juli06.xls
resultate_exp7_5.Juli06.xls
                              < < < < < < <  Leerzeile!
[Weitere R-Befehle]

> 1: 2: 3: 4: 5: 6: Read 5 items

## Anzeigen der Dateinamen in R:
> t.filenames
[1] "Results_Exp1_5.Juni06.xls"   "Results_Exp2_7.Juni06.xls"
[3] "Results_Exp3_17.Juni06.xls"  "resultate_exp6_4.Juli06.xls"
[5] "resultate_exp7_5.Juli06.xls"
```