Vorbereitung

Bitte führen Sie zur Vorbereitung folgende Schritte aus:

1. Starten Sie RStudio.
2. Löschen Sie den Workspace.
3. Setzen Sie das Arbeitsverzeichnis: Session » Set Working Directory » Choose Directory.
4. Öffnen Sie ein R-Skript.
5. Nachdem Sie die Aufgaben bearbeitet haben, speichern Sie das Skript unter einem geeigneten Namen ab.

Aufgabe 1 - Öffnen von Datenformaten in R

Achten Sie auf die Benutzung des richtigen Befehls und der richtigen Argumente!

(i) Rufen Sie die Hilfeseite zur Funktion `read.csv()` auf und überfliegen Sie die **Details-**Sektion. Es wird nicht von Ihnen verlangt, dass Sie die Inhalte sofort und vollständig verstehen. Es geht lediglich um ein erstes Vertrautmachen mit den Hilfeseiten.
Gibt es einen Unterschied zwischen der `read.csv()`- und `read.table()`-Funktion? Wenn ja, was unterscheidet diese Funktionen?

Lösung:

```
?read.csv
```

Auf der Hilfeseite können Sie folgenden Absatz finden:

> `read.csv` and `read.csv2` are identical to `read.table` except for the defaults. They are intended for reading 'comma separated value' files ('`.csv`') or (`read.csv2`) the variant used in countries that use a comma as decimal point and a semicolon as field separator. Similarly, `read.delim` and `read.delim2` are for reading delimited files, defaulting to the TAB character for the delimiter. Notice that `header = TRUE` and `fill = TRUE` in these variants, and that the comment character is disabled.

Aus diesem Absatz können Sie entnehmen, dass `read.csv` und `read.table` bis auf die *Defaults* der Argumente identisch sind. Defaults sind die Grundwerte, die die Argumente einer Funktion haben, bevor sie vom Nutzer verändert werden.

Beispiel aus der Vorlesung:

```
log(2)
```

```
[1] 0.6931472
```

Standardmäßig hat `log` e als Basis.

Den genauen Unterschied zwischen den Defaults der `read.table` und `read.csv` können Sie der **Usage-**Sektion entnehmen.

(ii) Laden Sie mit R die Datei `test.csv` (Moodle) und speichern Sie den Inhalt in einem Objekt `testdaten`.

Lösung:

```
testdaten <- read.table("Dateipfad",
                        header = TRUE,
                        sep = ";",
                        na.strings = "-999")
```

oder

```
testdaten <- read.csv2("Dateipfad",
                       na.strings = "-999")
```

Hier sind die Defaults bereits so gesetzt, dass lediglich ein eintippen des Datenorts und der Kodierung der fehlenden Werte notwendig ist.

(iii) Lassen Sie sich den Inhalt des Datensatzes angeben.

Lösung:

```
testdaten
```

Aufgabe 2 - Installieren und Laden eines Pakets

(i) Prüfen Sie, ob das Paket `psych` bereits installiert ist. Wenn nicht, installieren Sie es bitte, indem Sie den notwendigen Befehl verwenden. Falls das Paket schon installiert ist, aktualisieren Sie es, sodass Sie die neuste Version haben.

Lösung:

```
install.packages("psych")
```

Falls bereits installiert:

```
update.packages()
```

(ii) Aktivieren Sie das Paket in R

Lösung:

```
library(psych)
```

By the way: Sie können auch Hilfeseiten zu Paketen aufrufen!

```
?psych
```

Aufgabe 3 - Speichern von Objekten

(Diese Aufgabe ist nicht prüfungsrelevant)

(i) Speichern Sie den Datensatz `testdaten` in einem anderen Format als dem ursprünglichen csv-Format.

Lösung:

```
save(testdaten, file = "testdaten.RData")
```

By the way: Alternativ können Sie auch mithilfe der `write.table`-Funktion Datensätze in anderen Formaten als dem R-eigenen Dateiformat abspeichern. Als kleine Zusatzübung erkunden Sie diese Funktion über die Hilfe-Seite.

Zusatzaufgabe: Eigene Funktionen in R (für Fortgeschrittene)

Lösung

```
mal3 <- function(x) { # x ist unser Argument
  x <- x * 3          # Inhalt der Funktion
  return(x)           # Rückgabewert
}
```

Nun können wir die Funktion mit dem Funktionsnamen aufrufen. Hier wurde 3 als Argument für `x` übergeben:

```
mal3(x = 3)
```

```
[1] 9
```

Wie sonst auch können wir der Funktion ein Objekt übergeben.

```
zahl <- 10
mal3(zahl)
```

```
[1] 30
```