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

(i) Lesen Sie die Datei `ue_datensatz.txt` (siehe Moodle) mit Hilfe des `read.table()`-Befehls in R als Objekt `ue_daten3` ein. Achten Sie auf die Verwendung der richtigen Argumente.

(ii) Überprüfen Sie den Datensatz mithilfe von `summary()`. Finden Sie die zwei Eingabefehler im Datensatz und ersetzen Sie diese durch fehlende Werte (`NA`). Nutzen Sie das Kodierschema auf Moodle (im Datensätze-Ordner), um die Eingabefehler erkennen zu können.

(iii) Lassen Sie sich die Anzahl der Personen und Variablen ausgeben.

(iv) Entfernen Sie alle Personen mit fehlenden Werten aus dem Datensatz.

(v) Nutzen Sie die Indexierung, um ein Objekt zu erstellen, das für alle Personen nur die ersten drei Stimmungsvariablen (`stim1`, `stim2`, `stim3`) enthält.

(vi) Erstellen Sie eine neue Variable in dem neuen Objekt, die für jede Person den Mittelwert der ersten drei Stimmungsvariablen enthält. Lassen Sie sich den Mittelwert für die 55. Person im Datensatz ausgeben.

Aufgabe 2 - Faktoren

(i) Laden Sie den `erstis.RData` Datensatz in R.

(ii) Konvertieren Sie im Datensatz die Variablen `gruppe`, `job` und `wohnort.alt` in Faktoren. Verwenden Sie die folgenden Labels:

- `gruppe`: 1 = Kurs A, 2 = Kurs B, 3 = Kurs C, 4 = Kurs D
- `job`: 1 = Nebenjob, 2 = Kein Nebenjob
- `wohnort.alt`: 1 = alte BL, 2 = neue BL, 3 = Berlin, 4 = Ausland

(iii) Schauen Sie sich die Variable `code` an. Welche Funktion hat diese Variable? Welches Variablenniveau sollte die Variable Ihrem Verständnis nach haben? Konvertieren Sie gegegebenfalls.

Aufgabe 3 - Umkodieren

Polen Sie die folgenden Stimmungsitems im Datensatz so um, dass bei allen Items ein höherer Wert einer stärker ausgeprägten ruhigen bzw. wachen Stimmung entspricht:

`stim3`, `stim5`, `stim7`, `stim9`

Bilden Sie die Namen der neuen Variablen, indem Sie an den alten Namen ein "_r" anhängen (Bsp.: `stim3_r`).

Aufgabe 4 - Skalenbildung

Bilden Sie eine neue Variable im Datensatz, die die Skalenmittelwerte für die Items der wachen Stimmung darstellt. Nennen Sie diese Variable `wm` (wach-müde) und mitteln Sie dazu alle Items der Skala. Zu dieser Skala gehören das zweite, fünfte, siebte und zehnte Stimmungsitem. Wählen Sie einen liberalen Umgang mit fehlenden Werten (Berechnung des Skalenwertes auch bei Personen mit fehlenden Werten). **Achtung**: Vergessen Sie nicht, die in Aufgabe 2 erstellten umgepolten Items zu verwenden!

Zusatzaufgabe 1: Logische Bedingungen - `if` condition

(Diese Aufgabe ist nicht prüfungsrelevant)

Beim Datenmanagement ist es nicht selten der Fall, dass man einen Befehl oder eine Funktion nur unter einer bestimmten Bedingung ausführen möchte. Das könnte die Erstellung eines Teildatensatzes oder die Überprüfung der Gültigkeit von Werten einer Variable sein. Mit Hilfe von sogenannten `wenn`-Bedingungen (`if` conditions) lassen sich solche Probleme lösen.

In R kann man eine `wenn`-Bedingung mit der `if ()`-Funktion erzeugen. Die Bedingung wird hier wie bei einer regulären Funktion in die Klammern geschrieben. Allerdings mit einem Leerzeichen zwischen `if` und den Klammern `()`! Die Bedingung kann `TRUE` (wahr) oder `FALSE` (falsch) sein. Im Falle, dass die Bedingung falsch ist, passiert nichts, da R die Konsequenz der `if`-Bedingung nur ausführt, wenn diese auch wahr ist. Im Falle einer wahren Bedingung führt R die Konsequenz aus, welche in geschwungenen Klammern steht.

```
if (BEDINGUNG) { # in runden Klammern; kann TRUE oder FALSE sein kann
  KONSEQUENZ     # in geschwungenen Klammern
}
```

Beispiel: (Falls Ihnen nicht klar ist, wie das Ergebnis zustande kommt, kopieren Sie den Code in Ihr eigenes Skript und führen Sie diesen selbst Schritt für Schritt in RStudio aus)

```
x <- 5
z <- 10
if (x == 100) {
  z <- z*x
}
z
```

```
[1] 10
```

(i) Schreiben Sie eine `if` condition, welche überprüft, ob eine beliebige Zahl größer als 4 ist. Ist die Zahl größer als 4, soll folgende Nachricht in der Konsole ausgegeben werden: Die Zahl ist größer als 4! Nutzen Sie dafür die Funktion `print()`. (`?print` könnte hilfreich sein). Überprüfen Sie die Richtigkeit Ihres Codes, indem Sie nacheinander $-5, 3.99, 100$ in die Bedingung einsetzen.

(ii) Schreiben Sie eine `if` condition, welche überprüft, ob eine beliebige Zahl gerade ist. Lassen Sie sich eine geeignete Nachricht in der Konsole ausgeben, falls die Zahl gerade sein sollte. Überprüfen Sie die Richtigkeit Ihres Codes, indem Sie nacheinander $-2, 6, \pi$ in die Bedinung einsetzen.
Tipp: Die Modulooperation `%%` könnte hier hilfreich sein. Mit der Modulooperation berechnet man den Rest einer Division zweier Zahlen. (siehe die Beispiele im Modulokapitel auf Wikipedia: Wikipedia - Modulo)

Beispiel:

```
9 %% 3
```

```
[1] 0
```

```
10 %% 3
```

```
[1] 1
```

Zusatzaufgabe 2: Logische Bedingungen - `if-else` condition

(Diese Aufgabe ist nicht prüfungsrelevant)

Mit der einfachen `wenn`-Bedingung wird ein Befehl oder ein ganzer Codeblock nur dann ausgeführt, wenn die Bedingung zutrifft. Falls auch Code ausgeführt werden soll, wenn die Bedingung nicht zutrifft, muss die `wenn-dann`-Bedingung (`if-else` condition) verwendet werden.

```
if (BEDINGUNG) {
  KONSEQUENZ # wenn BEDINGUNG wahr
} else {
  KONSEQUENZ # wenn BEDINGUNG falsch
}
```

Ergänzen Sie die in Zusatzaufgabe 1 erstellten `if` conditions, um ein `else`. Überlegen Sie sich eine passende Nachricht (via `print()`) für die Ausgabe in der Konsole, wenn die Konsequenz der `else`-Bedingung zutrifft.