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.

Lösung

```
ue_daten3 <- read.table("../Datensätze/ue_datensatz.txt", # Dateiname
                        header = TRUE,                    # Erste Zeile enthält Variablennamen
                        sep = ",",                        # Spalten durch Kommata getrennt
                        na.strings = "-999",              # Kodierung fehlender Werte
                        dec = ".")                        # default
```

(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.

Lösung

```
summary(ue_daten3)
```

Nach der Inspektion sollten Sie folgende Fehler entdeckt haben:

- Wert -3 auf der Variable `bf19`
- Wert 14 auf der Variable `uni4`

```
ue_daten3[ue_daten3$bf19 == -3, "bf19"] <- NA
ue_daten3[ue_daten3$uni4 == 14, "uni4"] <- NA
```

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

Lösung

```
nrow(ue_daten3)
```

```
[1] 85
```

```
ncol(ue_daten3)
```

```
[1] 61
```

```
# alternativ
# str(ue_daten3)
```

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

Lösung

```
ue_daten3 <- na.omit(ue_daten3)
```

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

Lösung

Viele Wege führen nach Rom:

```
sub_stim <- ue_daten3[, c("stim1", "stim2", "stim3")]
sub_stim <- ue_daten3[, c(1, 2, 3)]
sub_stim <- ue_daten3[, 1:3]

str(sub_stim)
```

```
'data.frame':   81 obs. of  3 variables:
 $ stim1: int  4 4 4 3 2 4 4 2 4 2 ...
 $ stim2: int  2 2 4 2 1 3 3 2 3 2 ...
 $ stim3: int  4 4 3 2 3 2 3 3 3 4 ...
```

(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.

Lösung

```
sub_stim$stim_MW <- rowMeans(sub_stim[,c("stim1","stim2","stim3")])
sub_stim$stim_MWa <- (sub_stim$stim1 +  sub_stim$stim2 + sub_stim$stim3)/3 # Alternatve Möglichkeit, ge
sub_stim[55,4]
```

```
[1] 3.333333
```

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

Lösung

```
load("../Datensätze/erstis.RData")
```

```
erstis$gruppe <- factor(erstis$gruppe, levels = 1:4,
                        labels = c("Kurs A", "Kurs B", "Kurs C", "Kurs D"))
erstis$job <- factor(erstis$job, levels = c(1,2),
                     labels = c("Nebenjob", "Kein Nebenjob"))
erstis$wohnort.alt <- factor(erstis$wohnort.alt, levels = 1:4,
                             labels = c("alte BL", "neue BL", "Berlin", "Ausland"))

str(erstis[, c("gruppe", "job", "wohnort.alt")])
```

```
'data.frame':   191 obs. of  3 variables:
 $ gruppe     : Factor w/ 4 levels "Kurs A","Kurs B",..: 2 3 2 2 4 4 3 2 3 4 ...
 $ job        : Factor w/ 2 levels "Nebenjob","Kein Nebenjob": 2 1 1 2 2 1 NA NA NA 2 ...
 $ wohnort.alt: Factor w/ 4 levels "alte BL","neue BL",..: 2 3 4 3 3 3 NA NA NA 3 ...
```

(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.

Lösung

```
str(erstis[, "code"])
```

```
 int [1:191] 1 2 3 4 5 6 7 8 9 10 ...
```

```
summary(erstis$code)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0    48.5    96.0    96.0   143.5   191.0
```

Die Variable `code` ist ursprünglich als integer (Zahlenvektor) eingelesen. Sie hat einen Wertebereich von 1 bis 191 und stellt den Probandencode dar. Da der jeweilige Zahlenwert, der einer Person zugeordnet wird, keine Bedeutung hat, könnte man die Variable auch als Faktor umkodieren.

```
erstis[, "code"] <- factor(erstis[, "code"])
```

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`).

Lösung

```
library(car)

erstis$stim3_r <- recode(erstis$stim3, recodes = "1=5; 2=4; 4=2; 5=1")
erstis$stim5_r <- recode(erstis$stim5, recodes = "1=5; 2=4; 4=2; 5=1")
erstis$stim7_r <- recode(erstis$stim7, recodes = "1=5; 2=4; 4=2; 5=1")
erstis$stim9_r <- recode(erstis$stim9, recodes = "1=5; 2=4; 4=2; 5=1")
```

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!

Lösung

```
set_wm <- c("stim2", "stim5_r", "stim7_r", "stim10")
erstis$wm <- rowMeans(erstis[, set_wm], na.rm = TRUE)
summary(erstis$wm)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  1.000   2.500   3.000   3.082   3.500   4.500       2
```

Zusatzaufgabe 1: Logische Bedingungen - `if` condition

(Diese Aufgabe ist nicht prüfungsrelevant)

(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.

Lösung

```
zahl <- -5
if (zahl > 4) {
  print("Die Zahl ist größer als 4!")
}
```

```
zahl <- 3.99
if (zahl > 4) {
  print("Die Zahl ist größer als 4!")
}
```

```
zahl <- 100
if (zahl > 4) {
  print("Die Zahl ist größer als 4!")
}
```

```
[1] "Die Zahl ist größer als 4!"
```

(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
```

Lösung

```
if (zahl %% 2 == 0) {
  print("Die Zahl ist gerade!")
}
```

```
[1] "Die Zahl ist gerade!"
```

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

(Diese Aufgabe ist nicht prüfungsrelevant)

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.

Lösung

```
zahl <- 4
if (zahl > 4) {
  print("Die Zahl ist größer als 4!")
} else {
  print("Die Zahl ist kleiner gleich 4!")
}
```

```
[1] "Die Zahl ist kleiner gleich 4!"
```

```r
zahl <- 3
if (zahl %% 2 == 0) {
  print("Die Zahl ist gerade!")
} else {
  print("Die Zahl ist ungerade!")
}
```

```
[1] "Die Zahl ist ungerade!"
```