Programmieren mit Statistischer Software **7. Block**
Grün **SS 2017**

7 Mehr zu Funktionen in R

Aufgabe 1:

Schreiben Sie eine Funktion `lokation`, die folgendermaßen definiert ist:

```
> lokation <- function(x, na.rm = TRUE, type = c("mean", "median"), ...) {}
```

- Falls `type = "mean"`, soll der Mittelwert ausgerechnet werden. Falls `type = "median"`, der Median. Für alle anderen Werte von `type` soll ein Fehler ausgegeben werden. Verwenden Sie die Funktion `match.arg()`.
- Das Argument `na.rm` soll an die jeweilige Funktion zum Berechnen der Lokation (Mittelwert, Median) weitergegeben werden.
- Das Argument `...` soll ebenfalls an die Funktion (falls möglich) weitergegeben werden.

Wenden Sie die Funktion folgendermaßen an:

```
> lokation(faithful$waiting)
> lokation(faithful$waiting, type = "median")
> is.na(faithful$waiting[3]) <- TRUE
> lokation(faithful$waiting, na.rm = FALSE, type = "median")
```

Machen Sie auch einen Aufruf, wo Sie das `...` Argument benützen.

Aufgabe 2:

Schreiben Sie eine Funktion `streuung`, die folgendermaßen definiert ist:

```
> streuung <- function(x, type = 1, na.rm = FALSE, ...) {}
```

- In der LV Methodenlehre I wurden verschieden Streuungsmaße definiert (siehe C. Duller, Einführung in die Statistik mit Excel und SPSS).
- Das Argument `type` soll für die Werte von 1 bis k, wobei k die Anzahl verschiedener Streuungsmaße ist, jeweils ein anderes Streuungsmaß berechnen.
- Falls `na.rm = TRUE` sollen `NA`s entfernt werden, ansonsten wird `NA` zurückgegeben, falls `NA`s im Datensatz vorhanden sind.
- Verwenden Sie gegebenenfalls das `...` Argument, um Argumente an die Funktionen zum Bestimmen der Streuungsmaße weiterzugeben.

Wenden Sie die Funktion auf einem beliebigen Datensatz an, um alle verschiedenen Streuungsmaße dafür zu bestimmen.

Aufgabe 3:

Schreiben Sie eine Funktion `moment`, die folgendermaßen definiert ist:

```
> moment <- function(x, k = 1, shift = FALSE, a = mean(x)) {}
```

- Die Momente wurden in der LV Methodenlehre I definiert (siehe C. Duller, Einführung in die Statistik mit Excel und SPSS).
- Für `shift = TRUE` soll die Funktion die Momente der Ordnung `k` in Bezug auf den Punkt `a` berechnen.
- Für `shift = FALSE` soll die Funktion die gewöhnlichen Momente der Ordnung `k` berechnen.

Wenden Sie die Funktion folgendermaßen an:

```
> moment(faithful$waiting, 3, shift = FALSE)
> moment(faithful$waiting, 3, shift = TRUE)
```

Aufgabe 4:

Das **Heron-Verfahren** ist ein Rechenverfahren zur Berechnung einer Näherung der Quadratwurzel einer Zahl. Es ist ein Spezialfall des Newton-Verfahrens. Die Iterationsvorschrift lautet:

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2}$$

Wobei a die Zahl ist, deren Quadratwurzel bestimmt werden soll. x_0 kann beliebig, aber ungleich Null, festgesetzt werden. n muss eine nichtnegative ganze Zahl sein. Überprüfen Sie diese Bedingungen und geben Sie gegebenfalls einen Fehler aus.

Schreiben Sie eine Funktion

```
> heron <- function(a, n) { ... }
```

welche die Näherung der Quadratwurzel von `a` nach `n` Iterationsschritten zurückliefert. Und berechnen Sie dann die folgenden Quadratwurzeln:

```
> h37 <- heron(37, 10)
> h9 <- heron(9, 4)
```

Aufgabe 5:

Schreiben Sie eine Funktion, die eine Funktion zur Berechnung der Werte der empirischen Verteilungsfunktion (siehe C. Duller, Einführung in die Statistik mit Excel und SPSS), die bzgl. der Daten `x` bestimmt wurde, zurückgeben soll:

```
> F <- function(x, na.rm = TRUE, type = c("stetig", "diskret"),
+   breaks = quantile(x, seq(0, 1, length = min(sqrt(length(x)), 10)))) { ... }
```

mit folgenden Argumenten:

- `x`: Vektor der Daten.
- `na.rm`: Logischer Wert, ob fehlende Werte in `x` vor der Bestimmung der empirischen Verteilugsfunktion entfernt werden sollen oder nicht.
- `type`: Zeichenkette, die angibt, ob die Verteilungsfunktion für eine stetige oder diskrete Variable bestimmt werden soll, d.h., die Verteilungsfunktion ist eine Treppenfunktion oder eine stückweise lineare Funktion.
- `breaks`: Numerischer Vektor, der angibt, wie die Intervallgrenzen sind, um aus der Variablen `x` ein intervallskaliertes Merkmal zu konstruieren (siehe `?cut`), falls `type = "stetig"`.

Das bedeutet, retourniert soll eine Funktion der Form werden

```
> function(x) { ... }
```

wobei `x` nur ein numerischer Vektor der Länge 1 sein darf.

Falls der Vektor `x` `NA`s enthält und `na.rm = FALSE` ist, dann soll eine Funktion zurückgegeben werden, die immer `NA` zurückgibt.

Wenden Sie `ECDF` folgendermaßen an:

```
> set.seed(1234)
> x <- rnorm(200)
> foo <- ECDF(x)
> Vectorize(foo)(c(-2, 0, 2))
> x <- rbinom(1000, 0.4, size = 5)
> foo <- ECDF(x, type = "diskret")
> Vectorize(foo)(c(0:5, 4.5))
> ECDF(c(x, NA), na.rm = FALSE)(1)
```

Hinweis: Zum Erzeugen der intervallskalierten Variable kann man `cut()` verwenden, wobei man dessen Argument `breaks` gleich dem `breaks` Argument von `ECDF` setzt und `include.lowest = TRUE` verwendet.