Programmieren mit Statistischer Software **4. Block**
Grün **SS 2017**

4 Konstrukte

Aufgabe 1:

Die **Fibonacci-Folge** ist eine unendliche Folge von Zahlen (den Fibonacci-Zahlen), bei der sich die jeweils folgende Zahl durch Addition der beiden vorherigen Zahlen

$$f_n = f_{n-1} + f_{n-2}$$

mit den Anfangswerten $f_0 = 0$ und $f_1 = 1$ ergibt. Also, $0, 1, 1, 2, 3, 5, 8, 13, \ldots$. Schreiben Sie eine `for`-Schleife, welche in einem Vektor `fib` die ersten 30 Fibonacci-Zahlen abspeichert.

Aufgabe 2:

Im Folgenden wollen wir den Datensatz `swiss`, der ebenfalls im Paket **datasets** enthalten ist, verwenden.

- Mithilfe der Funktion `mean` kann auch ein getrimmter Mittelwert berechnet werden. Erzeugen Sie einen Vektor `tmean`, in dem Sie die getrimmten Mittelwerte für alle Variablen abspeichern. Verwenden Sie einen Wert von 0.1 für `trim`.
- Analog kann man im Datensatz auch die extremen Beobachtungen durch `NA` ersetzen und die getrimmten Mittelwerte dann mithilfe `colMeans` bestimmen. Schreiben Sie Code, der diese Vorgehensweise implementiert.

Aufgabe 3:

Schreiben Sie Code, der für eine gegebene Zahl `n` alle Primzahlen, die kleiner gleich `n` sind, bestimmt und in einem Vektor zusammenfasst.

Verwenden Sie dafür den Sieb-Algorithmus von Eratosthenes, der wie folgt funktioniert:

1. Starten Sie mit einem Vektor, der die Zahlen von 2 bis `n` enthält und das aktuelle Sieb darstellt.
2. Iterieren Sie durch die Werte von 2 bis `n`:
   - Überprüfen Sie, ob die Zahl im aktuellen Sieb enthalten ist, und brechen Sie den Schritt ab, falls dies nicht der Fall ist.
   - Entfernen Sie alle Vielfachen, aber nicht die Zahl selbst, aus dem aktuellen Sieb und erstellen Sie damit ein neues Sieb.
3. Diejenigen Zahlen, die am Ende im Sieb enthalten sind, stellen die Primzahlen dar.

Bestimmen Sie die Primzahlen und deren Anzahl für $n = 200$.

Aufgabe 4:

Primzahlenpaare sind Paare von Primzahlen (x, y), die einen Abstand von 2 haben, d.h. $y = x + 2$ gilt. Bestimmen Sie eine Liste, die alle Primzahlenpaare unter 200 enthält.

Hinweis: Falls Sie keinen Vektor mit Primzahlen als Lösung aus dem vorigen Beispiel haben, ziehen Sie einfach zufällig 50 Zahlen aus den Zahlen von 1 bis 200 und verwenden Sie diesen Vektor.

Aufgabe 5:

Ein einfacher Pseudozufallszahlengenerator für augenscheinlich gleichverteilte Daten würde folgendermaßen vorgehen:

Für gegebenene Startwerte `start1` und `start2` und Primzahl `prim` werden n Werte y_i, $i = 1, \ldots, n$ bestimmt durch:

$$\begin{aligned} y_1 &= \texttt{start1}, \\ y_2 &= \texttt{start2}, \\ y_i &= (y_{i-2} + y_{i-1}) \,\%\%\, \texttt{prim}, \end{aligned}$$

für $i = 3, \ldots, n$ und %% bestimmt den Rest bei der Division von `prim` durch $(y_{i-2} + y_{i-1})$.

Diese Werte werden dann skaliert durch:

$$x_i = \frac{y_i}{\texttt{prim} - 1},$$

für $i = 1, \ldots, n$, sodass gilt $x_i \in [0, 1]$.

Berechnen Sie die Primzahl durch:

```
> primes <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31)
> prim <- prod(primes) + 1
```

Setzen Sie $n = 10000$ und bestimmen Sie für den Startwert $\texttt{start1} = 10$ und $\texttt{start2} = 100$ die Vektoren x und y noch obigem Bildungsgesetz.

Überprüfen Sie mit einfachen Tests, ob die Werte in x unabhängig identisch verteilte Realisationen einer Gleichverteilung sein können.