Wintersemester 2006/2007 Abgabe am 30. November 2006
Prof. Dr. Stefan Posch, Dipl.-Bioinf. Jan Grau

9. Übung „Angewandte Bioinformatik mit Perl und R“

1. Sie haben Daten aus einer Erhebung, bei der für eine Reihe von Studierenden Geburtsjahr, Schuhgröße und Gewicht erfragt wird.

   (a) Speichern Sie diese Daten in einer Matrix, wobei jede Zeile die Daten einer Person enthalten soll. Füllen Sie diese Matrix mit den Daten von mindestens 10 echten (oder gefälschten) Personen.

   (b) Benennen Sie die Zeilen und Spalten mit sinnvollen Namen.

   (c) Jetzt ist Ihnen eingefallen, dass Sie auch das Geschlecht der Personen benötigen. Legen Sie hierfür einen Vektor an, der `"w"` bzw. `m"` für die jeweilige Person enthält. Geben Sie auch den Elementen dieses Vektors Namen, und zwar natürlich dieselben wie in der Datenmatrix. Um sicher zu gehen, dass diese tatsächlich übereinstimmen, benutzen Sie bei der Benennung exakt die Namen aus der Datenmatrix.

   (d) Selektieren Sie aus der Datenmatrix die Einträge von Studentinnen und Studenten, wobei jeweils eine Matrix derselben Form entstehen soll.

   Erzeugen Sie nun eine Matrix, die nur Gewicht und Schuhgröße enthält.

   (e) Berechnen Sie (möglichst einfach natürlich) das mittlere Gewicht aller Studierenden, und dann nach Geschlecht getrennt.

   Was berechnen die Ausdrücke: `mean(daten[geschlecht == m"])` und
   `mean(daten[geschlecht == m"],"gewicht")`

2. (a) Legen Sie nun eine Liste an, die die Datenmatrix und den Vektor enthält, wobei die Komponenten natürlich wiederum geeigent benannt werden sollen.

   (b) Weisen Sie dieser Liste ein weiteres Element zu, welches eine (kurze) Beschreibung der Erhebung enthält. Falls Sie es nicht schon gemacht haben, geben Sie auch diesem Element einen sinnvollen Namen.

   (c) Erzeugen Sie eine Unterliste, die wieder nur die Datenmatrix und den Vektor mit den Geschlechtern enthält. Verzichten Sie hierbei auf Ihr Wissen, in welcher Reihenfolge die Elemente in der Liste stehen.

   (d) Nun fällt Ihnen auf, dass am Ende der Beschreibung ein Satz fehlt. Erweitern Sie die Beschreibung entsprechend. Hierbei sollen Sie den Anfangsteil, der ja bereits getippt ist, wieder benutzen.

3. (a) Erzeugen Sie einen *data frame* der die Datenmatrix und das Geschlecht enthält.

   (b) Selektieren Sie aus diesem *data frame* dieselbem Matrizen wie in Aufgabe 1d

   (c) Hängen Sie nun an die Bezeichner der Zeilen Ihres *data frame* vorlaufende Nummer (beginnend ab 1) an. Diese Zuweisung soll für beliebige *data frames* funktionieren, schreiben Sie also am besten eine kleine Funktion hierfür.

4. (a) Lesen Sie die Daten aus mtcars.txt ein. (Achtung: Die Einträge sind durch Tabulatoren (`"\t"`) getrennt.) Da `mtcars` auch eine in R verfügbarer Datensatz ist, erhalten Sie mit `? mtcars` nähre Informationen.

   (b) Berechnen Sie den mittleren Verbrauch in miles/gallon alle Autotypen.

   (c) Berechnen Sie dann den mittleren Verbrauch in Abhängigkeit von der Anzahl der Zylinder.

   Schreiben Sie nun eine Funktion hierzu, die als Argumente das `data frame`, die zwei Spaltenbezeichner und den geforderten Wert für die zweite Spalte aufweist.

   Falls nicht schon ohnehin der Fall, sollen Ihre Funktionen auch für permutierte Spalten im Datensatz funktionieren.

5. Schreiben Sie eine Funktion zur Berechung der Fakultät.

   Berechnen Sie für die Zahlen 1 bis 50 (ohne Rücksicht auf Effizienz) die Fakultät in einem Vektor. (Hinweis: benutzen Sie die Funktion `sapply`

6. Wir betrachten ein Zeitsignal, welches in einem numerischen Vektor gegeben ist. Wir wollen nun ein "sliding window" (d.h. ein Fenster fester Breite) über das Signal wandern lassen und aus den Funktionswerten innerhalb des Fensters einen neuen Funktionswert bestimmen, z.B. den Mittelwert.

   Schreiben Sie zunächst eine Funktion, die in dem Vektor `f` in einem Fenster der Breite $2 \cdot b + 1$ symmetrisch um den Index i den Mittelwert berechnet und als Funktionswert zurückgibt.

   Wenden Sie Ihre Funktion auf verschiedene Fenster folgendes Signals an:

```
g <- function (sd) sqrt(2*pi)*sd*dnorm( (-3*sd):(3*sd), sd=sd)
f <- rep(50,100)
sd = 7
i = 25
a = 20
f[(i-3*sd):(i+3*sd)] <- f[(i-3*sd):(i+3*sd)] + a*g(sd)

sd = 5
i = 65
a = 11.5
f[(i-3*sd):(i+3*sd)] <- f[(i-3*sd):(i+3*sd)] + a*g(sd)
```

   Schreiben Sie nun eine Funktion, die eine neuen Vektor zurück liefert, der gerade diese lokalen Mittelungen enthält. Schreiben Sie zwei Versionen, eine unter Nutzung von `sapply`, eine unter Nutzung einer for-Schleife.