1 Replacement

```
testdat_runif(1000000)
```

Schleife:

```
for(i in 1:1000000)
if(testdat[i]<0.1)testdat[i]==0
```

vektorisiert:

```
testdat[testdat<0.1]_0
```

2 Mittelwert

Schleife:

```
summe_0
for(i in 1:100000)
summe_summe+testdat[i]
summe/100000
```

vektorisiert:

```
sum(testdat)/100000
```

2.1 R-Funktion

```
mean(testdat)
```

3

4 Zeilen- und Spaltensummen einer Matrix

```
testdat_matrix(runif(1000000),nrow=1000)
```

5 Zeilensummen

Schleife:

```
row.sum_rep(NA,nrow(testdat))
for(i in 1:nrow(testdat))
row.sum[i]_sum(testdat[i,])
```

apply:

```
apply(testdat,1,sum)
```

vektorisiert:

```
testdat%*%rep(1,ncol(testdat))
```

oder

```
crossprod(t(testdat),rep(1,ncol(testdat)))
```

6 Spaltensummen

apply:

```
apply(testdat,2,sum)
```

vektorisiert:

```
t(testdat)%*%rep(1,nrow(testdat))
```

oder

```
crossprod(testdat,rep(1,nrow(testdat)))
```

7 Zeilenweiser t-Test

Erzeugung von Testdaten

(zwei Matrizen mit 100000 Zeilen und 5 Spalten)

Schleife über Spalten:

```
testdat1_matrix(NA,nrow=500000,ncol=5)
testdat2_matrix(NA,nrow=500000,ncol=5)
for(i in 1:5)
{testdat1[,i]_runif(100000)
testdat1[,i]_runif(100000)}
```

Schleife über Zeilen:

testdat1_matrix(NA,nrow=500000,ncol=5)

```
testdat2_matrix(NA,nrow=500000,ncol=5)
for(i in 1:100000)
{testdat1[i,]_runif(5)
testdat2[i,]_runif(5)}
```

vektorisiert:

```
testdat1_matrix(runif(500000),ncol=5)
testdat2_matrix(runif(500000),ncol=5)
```

Schleife ohne vorherige Spezifizierung der Matrixdimension:

```
testdat_NULL
for(i in 1:100000)
testdat_rbind(testdat,runif(5))
```

zeilenweise t-Tests

Schleife:

```
pwerte_rep(NA,nrow(testdat1))
for(i in 1:100000)
pwerte[i]_t.test(testdat1[i,],testdat2[i,])$p.value
```

Konstruktion einer Funktion zum Aufruf mit apply:

```
system.time({
gruppen_c(rep(1,5),rep(2,5))
ttest.apply_function(datenvektor,gruppenvektor)
t.test(datenvektor[gruppenvektor==1],datenvektor[gruppenvektor==2])$p.value
```

pwerte_apply(cbind(testdat1,testdat2),1,ttest.apply,gruppen)

```
Funktion für die Vektorisierung der zeilenweisen t-Tests
hier: t-Test für den Vergleich zweier Gruppen mit gleicher Varianz
ttest.vektorisiert_function(dat1,dat2)
{
#---dat1 und dat2 müssen Matrizen mit der gleichen Anzahl Zeilen sein
#---Mittelwerte: Zeilensumme dividiert durch Anzahl Spalten
n1_ncol(dat1)
n2_ncol(dat2)
means1_as.vector((dat1%*%rep(1,n1))/n1)
means2_as.vector((dat2%*%rep(1,n2))/n2)
#---die Mittelwerte müssen von einer Matrix mit 1 Spalte in Vektor verwandelt
#---werden
#---Bereinigung der Matrizen um die Zeilenmittelwerte
abw1_dat1-outer(means1,rep(1,n1))
abw2_dat2-outer(means2,rep(1,n2))
#---Zeilenvarianzen
vars1_as.vector((abw1^2)%*%rep(1,n1))
vars2_as.vector((abw2^2)%*%rep(1,n2))
#---t-test vektorisiert
testgroesse_(means1-means2)/sqrt((vars1+vars2)*(1/n1+1/n2)/(n1+n2-2))
pwert_pt(abs(testgroesse),df=n1+n2-2,lower.tail=FALSE)*2
pwert
}
ttest.vektorisiert(testdat1,testdat2)
```

Andere Beispiele für outer

```
outer(month.abb, 2001:2003, paste)
```

Problem von Benno Pütz:

```
attach(frame)
namen_paste(spot,array)
index_function(spot.aktuell,array.aktuell)
(1:nrow(frame))[namen==paste(spot.aktuell,array.aktuell)]
indexmatrix_outer(unique(spot),unique(array),index)
dimnames(indexmatrix)_list(unique(spot),unique(as.character(array)))
```

8 Einige Faustregeln

Schleifen:

Wenn für alle Elemente eines Vektors, Zeilen oder Spalten einer Matrix bzw. Teilraum eines Arrays der gleiche Vorgang gemacht werden soll, braucht man i.a. keine Schleife.

Man braucht i.a. dann eine Schleife, wenn eine Iteration vom Ergebnis der vorhergehenden abhängt.

Eine Schleife sollte nur die Anweisungen umfassen, die unbedingt hineingehören.

Die Dimensionen eines Objekts, das in einer Schleife berechnet wird, sollten vorher spezifiziert werden. Wenn man nicht weiß, wie groß das Objekt wird (Ende der Schleife durch Abbruchkriterium), definiere man das Objekt genügend groß und schneide später ab.

apply:

apply ist oft nicht schneller als eine for-Schleife, nur eleganter.

Vektorisierung:

Am schnellsten sind vektorisierten Berechnungen. Hierfür stehen folgende R-Funktionen zur Verfügung:

- Arithmetik auf Vektoren bzw. Matrizen:
  - elementweise Operationen: +, -, *, /
  - Multiplikation mit einem Skalar: *
  - Multiplikation von Vektor mit Vektor, Vektor mit Matrix oder
    Matrix mit Matrix: %*%, crossprod
- Transposition: t()
- weitere Matrixfunktionen: diag(), solve()
- outer() mit beliebiger Verknüpfungsfunktion

Falls es mehrere Alternativen für die Vektorisierung gibt, nehme man die, bei der keine Transposition nötig ist.

allgemein:

Bei benannten Objekten (z.B. Zeilen- und Spaltennamen) dauern alle Berechnungen länger (mehr Speicher benötigt).

Vor großen Berechnungen lohnt es sich, eine neue R-Sitzung zu beginnen, dann ist noch kein Speicherplatz belegt.

Bei kurzen Schleifen mit kompliziertem Inhalt lohnt es sich nicht, die Vektorisierung auszutüfteln.