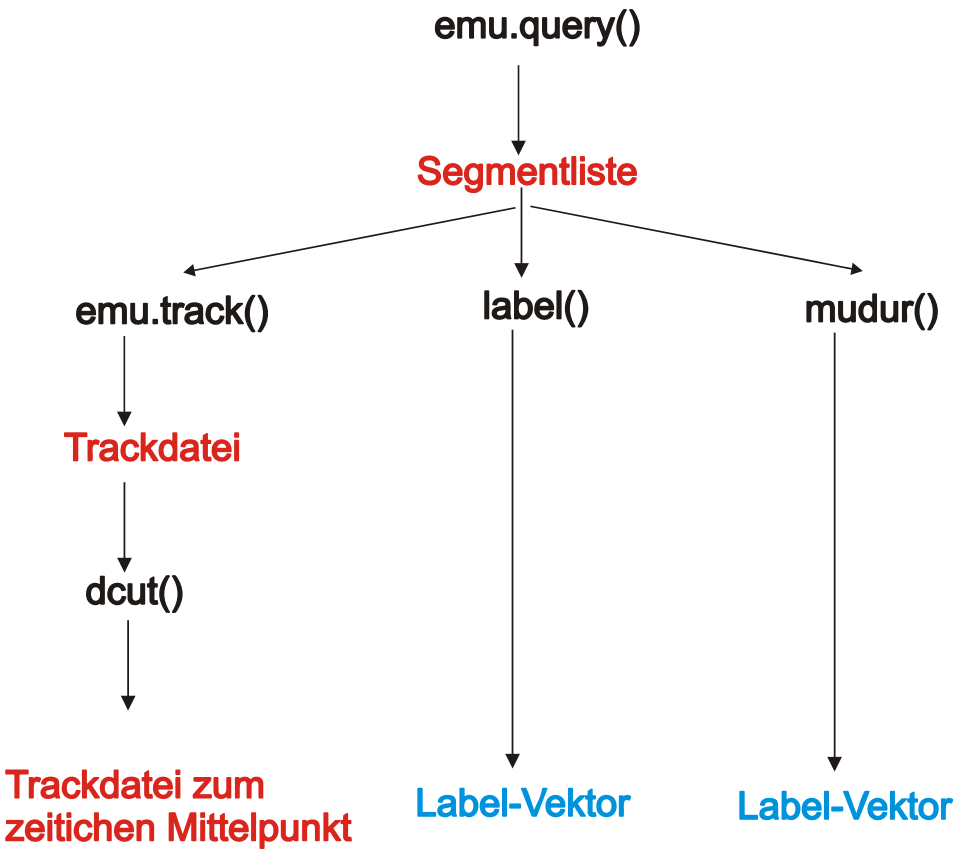
# 0. Signale und Etikettierungen abfragen




Wie eine Matrix

Wie ein Vektor

# 1. Vektoren

```
vec = c(10, -5, 30, 1)
a = bridge[,1]
vec2 = c("IPSK", 20, "Phonetik")
```

# wieviele Elemente?
```
length(a)
```

# Element 4
```
a[4]
```

# Elemente 4 und 8
```
a[c(4,8)]
```

# Alle Elemente außer Element 2
```
a[-2]
```

# letzter Element
```
a[length(a)]
```

# Elemente 1-5
```
a[1:5]
```

# Elemente 5-1
```
a[5:1]
```

# Der Vektor in der umgekehrten Reihenfolge
```
a[length(a):1]
```

# oder
```
rev(a)
```

<u>Arithmetische Funktionen</u>

| | |
|---|---|
| `*, /, +, -` | Mal, Dividiert, Plus, Minus |

# 20 von einem Vektor abziehen
```
a  - 20
```

# Zwei Vektoren derselben Länge miteinander multiplizieren
```
x = c(10, 20, 30)
y = c(1, 4, -5)
x * y
```
# Die erste Spalte von `bridge` dividiert durch die zweite
```
bridge[,1]/bridge[,2]
```

<u>Einige numerische Funktionen</u>

| | |
|---|---|
| `length()`: | wieviele Elemente in einem Vektor? |
| `mean()`, `median()`, `sd()`: | Durschschnitt, Zentralwert, Standardabweichung |
| `log()`: | Logarithmus |

^ hoch

```
# Logarithmus von x
log(x)
# Logarithmus von x (Basis 10)
log(x, base=10)
# 2^3
2^3
# Wurzel 2
2^(1/2)
# oder
sqrt(2)
# Die Summe aller Elemente
sum(x)
```

# 2. Matrizen

Zugriff auf Elemente einer Matrix: [i,j]

**Vor dem Komma bezieht sich auf Reihen.**
**Nach dem Komma bezieht sich auf Spalten.**

(Sonst sind es dieselben Befehle wie beim Vektor, zB):

# Reihen 1-3, Spalte 2
```
bridge[1:3,2]
```
# Reihe 1, Spalten 1 und 3
```
bridge[1,c(1,3)]
```
Usw.

# Reihen 1-2 (aller Spalten)
```
bridge[1:2,]
```
# Spalten 1 und 3 (aller Reihen)
```
bridge[,c(1,3)]
```

Funktionen

| | |
|---|---|
| `nrow():` | Reihenanzahl |
| `ncol():` | Spaltenanzahl |
| `dim():` | Dimensionen |

**Die Anwendung von numerischen Funktionen ist wie bei Vektoren**

# 20 von der Matrix `bridge` abziehen
```
bridge – 20
```

# `bridge` mit sich selbst multiplizieren
```
bridge * bridge
```

```
# entspricht:
bridge^2
```

# Der Durschnitt aller Elemente von `bridge`
```
mean(bridge)
```

Die `apply()` Funktion wird verwendet, um Funktionen wie `mean()`, `median()`, `sd()`, `sum()`, **getrennt auf Reihen oder Spalten einer Matrix anzuwenden**

# Summe jeder Reihe
```
apply(bridge, 1, sum)
```
# Standardabweichung jeder Spalte
```
apply(bridge, 2, sd)
```

## 3. Segmentlisten

- Entstehen durch `emu.query()` – also eine Datenbankabfrage.
- bestehen aus 4 Spalten (die Etikettierung, Startzeit, Endzeit, Äußerung)
- können wie eine Matrix behandelt werden

```
vok = emu.query("seminar04", "*", "Phonetic = i: | e:]")
# die ersten 3 Vokale
vok[1:3,]
# der letzte Vokal
vok[nrow(vok),]
# Usw. Also wie bei einer Matrix
```

Einige Funktionen

```
# Zusammenfassung der Segmentliste
summary(vok)
# die Etikettierungen
label(vok)
# die Äußerungen
utt(vok)
# die Dauern
mudur(vok)
# und nrow(), ncol(), dim() wie bei einer Matrix
```

# 4. Trackdateien

- Enthalten Signalwerte vom Anfang bis zum Ende des jeweiligen Segmentes
- Entstehen durch `emu.track()`, die auf eine Segmente angewandt wird
- Können zum großen Teil wie eine Matrix behandelt werden
- `dcut()` kann darauf angewendet werden, um Signale zu einem gewissen Zeitpunkt zu bekommen

```
grund = emu.track(vok[1:5,], "F0")
```

**Vor dem Komma bezieht sich auf Segmente (Reihen).**
**Nach dem Komma bezieht sich auf Signale (Spalten)**

```
# F0-Werte vom Anfang bis zum Ende des 3en Segmentes
grund[3,]
# Eine Abbildung davon
plot(grund[3,], type="l")
# Reihen/Spalten Anzahl (die Antwort ist 5 1 weil 5 Segmente und ein Track (F0)
dim(grund)
# F1-F4 der ersten 5 Segmente
form = emu.track(vok[1:5,], "fm")
# F1-F4 vom 3en Segment und eine Abbildung davon
plot(form[3,], type="l")
# F2 und F4 der Segmente 1 und 5
form[c(1,5), c(2, 4)]
# usw.

# Anwendung von dcut()
# F0 vom dritten Segment zum Segment-Onset
dcut(grund[3,], 0, prop=T)

# F2-Werte aller Segmente
dcut(form[,2], 1, prop=T)

# F1 und F3 zum zeitlichen Mittelpunkt, aller Segmente außer dem zweiten
dcut(form[-2,c(1,3)], 0.5, prop=T)
```

Funktionen

| | |
|---|---|
| `emu.track()`: | um die Trackdatei zu bekommen |
| `summary()`: | wie bei einer Segmentliste |
| `plot()`: | um eine Trackdatei abzubilden |
| `dcut()`: | um eine Trackdatei zu schneiden |
| `nrow()`, `ncol()`, `dim()` | wie bei einer Matrix |

# 5. Logische Vektoren

- Mit logischen Vektoren werden bei Vektoren Fragen gestellt, auf die man die Antwort bekommt: T (ja) oder F (nein).
- Sie entstehen durch Vergleichunsoperatoren: `==, <=, <, !=, >, >=, %in%`
- Sie sind nützlich, um auf Elemente in einem Objekt (Vektor, Matrix, Segmentliste, Trackdatei...) zuzugreifen.

**(a) Wie stellt man die Frage?**

# Ein Vektor von 8 Diphthong-Labels
```
vec = dip.l[1:8]
```

# Welche Elemente von `vec` sind `"aI"`?
```
vec == "aI"
```
# Welche Elemente von `vec` sind `"aU"` oder `"OY"`?
```
vec %in% c("aU", "OY")
```
# Welche Elemente von `vec` sind nicht `"aI"`?
```
vec != "aI"
```
# Welche Elemente der ersten Spalte von `bridge` sind größer als 20?
```
bridge[,1] > 20
```

**(b) Zugriff auf Elemente: Vektoren**

# 8 Diphthong-Labels in einem Vektor
```
vec = dip.l[1:8]
```
# die entsprechenden Dauern
```
dur = mudur(dip[1:8,])
```

# Die Dauern von `"aI"`
```
dur[vec=="aI"]
```
# Die Labels mit einer Dauer von mehr als 120 ms
```
vec[dur > 120]
```
# Die Labels mit einer Dauer zwischen 120 ms und 160 ms
```
vec[dur > 120 & dur < 160]
```

**(c) Zugriff auf Elemente: Matrizen, Segmentlisten, Trackdateien**

`bridge` is eine vorhandene Matrix. Nehmen wir an, die drei Spalten entsprechen die Tage der Woche
```
tage = c("M", "Di", "Mitt")
```
# Reihen 1-5 von Mittwoch
```
bridge[1:5,tage=="Mitt"]
```
# Die letzte Reihe von Montag und Mittwoch
```
bridge[nrow(bridge), tage %in% c("M", "Mitt")]
```

**Der Zugriff auf Elemente mit logischen Vektoren in einer Trackdatei funktioniert auf genau diesselbe Weise, zB:**

# Segmentliste
```
vok = emu.query("seminar04", "*", "Phonetic = i: | e:]")
```
# Label-Vektor
```
l.vok = label(vok)
```
# Trackdatei der Formanten F1-F4
```
f = emu.track(vok, "fm")
```
# Formanten zum zeitlichen Mittelpunkt
```
f5 = dcut(f, .5, prop=T)
```

# Welche Etikettierungen sind "e:"?
```
# temp = l.vok=="e:"
```

# Segmentliste der "e:" Vokale
```
vok[temp,]
```

# Trackdatei der "e:" Vokale
```
f[temp,]
```

# Wie oben, aber nur F1 und F2
```
f[temp,1:2]
```

# F1 und F3 der "e:" Vokale zum zeitlichen Mittelpunkt
```
f5[temp,c(1,3)]
```

Funktionen für logische Vektoren

`sum()`: wieviele Ts?

`any()`: gibt es mindestens einen T?

# 6. Zusammenfassung der Funktionen in diesem Dokument

| | siehe |
|---|---|
| any() | 5. Log. Vektoren |
| apply() | 2. Matrix |
| c() | 1. Vektoren |
| dim() | 2. Matrix |
| dcut() | 4.Trackdateien |
| emu.query() | 3. Segmentlisten |
| emu.track() | 4. Trackdateien |
| label() | 3. Segmentlisten |
| length() | 1. Vektoren |
| log() | 1. Vektoren |
| mean() | 1. Vektoren |
| median() | 1. Vektoren |
| mudur() | 3. Segmentlisten |
| ncol() | 2. Matrix |
| nrow() | 2. Matrix |
| rev() | 1. Vektoren |
| sd() | 1. Vektoren |
| sqrt() | 1. Vektoren |
| sum() | 1. Vektoren, 5. Logische Vektoren |
| utt() | 3. Vektoren |