R versus SQL

Christoph Dalitz

Hochschule Niederrhein
Fachbereich Elektrotechnik & Informatik

18. November 2013

Hochschule Niederrhein
University of Applied Sciences
Elektrotechnik und Informatik
Faculty of Electrical Engineering and Computer Science

R versus SQL

R als Turing-vollständige Sprache mit vielen ready-to-use Bibliotheken ist natürlich viel mächtiger als *SQL*. Allerdings ist SQL deskriptiv und intuitiv, während viele Abfragen in R nicht offensichtlich sind.

Diese Folien beschreiben ein paar typische SQL-Abfragen und wie sie in R realisiert werden können.

Sample data

data1

label	*value*	*color*
lb1	0.34	green
lb2	0.56	blue
lb1	NA	blue
lb5	1.23	green
lb1	1.11	NA

data2

color	*value*
green	0.50
blue	0.75
red	0.25

Hochschule Niederrhein
University of Applied Sciences
Elektrotechnik und Informatik
Faculty of Electrical Engineering and Computer Science

Metadaten

Spaltennamen und -typen

SQL-Befehl:

```
\d data1        (PostgreSQL)
.schema data1   (SQLite)
```

Entsprechung in R:

```
colnames(data1)   # nur Spaltennamen
str(data1)        # auch Datentypen
```

Anzahl Tupel

SQL-Befehl:

```
SELECT COUNT(*) FROM data1;
```

Entsprechung in R:

```
nrow(data1)
```


Hochschule Niederrhein
University of Applied Sciences
Elektrotechnik und Informatik
Faculty of Electrical Engineering and Computer Science

Einfache Abfragen

Abfragen ohne Bedingung

SQL-Befehle:

1. `SELECT * FROM data1;`
2. `SELECT * FROM data1 LIMIT 5;`
3. `SELECT label FROM data1;`
4. `SELECT label, color FROM data1;`
5. `SELECT * FROM data1 ORDER BY label, color;`

Entsprechungen in R:

1. `data1`
2. `head(data1, n=5)`
3. `data1$label` oder `subset(data1, select=c('label'))`
4. `data1[,c("label","color")]`
   `cbind(label=data1$label, color=data1$color)`
5. `data1[order(data1$label, data1$color), ]`

Hochschule Niederrhein
University of Applied Sciences
Elektrotechnik und Informatik
Faculty of Electrical Engineering and Computer Science

Einfache Abfragen

Abfragen mit Bedingungen

SQL-Befehle:

1. `SELECT * FROM data1 WHERE label = 'lb1';`
2. `SELECT label, value FROM data1 WHERE value > 1.0;`
3. `SELECT * FROM data1 WHERE color IS NOT NULL;`

Entsprechungen in R:

1. ```
   data1[data1$label == 'lb1', ]
   subset(data1, label == 'lb1')
   ```
2. `subset(data1, value > 1.0, select=label:value)`
3. ```
   data1[!is.na(data1$color),]
   subset(data1, !is.na(color))
   na.omit(data1)  # alle Spalten nicht NA
   ```


Hochschule Niederrhein
University of Applied Sciences
Elektrotechnik und Informatik
Faculty of Electrical Engineering and Computer Science

Joins

Inner und Outer Join

SQL-Befehle:

1. `SELECT * FROM data1 JOIN data2`
   `ON data1.color = data2.color;`
2. `SELECT * FROM data1 RIGHT OUTER JOIN data2`
   `ON data1.color = data2.color;`

Entsprechungen in R:

1. `merge(data1, data2, by.x='color', by.y='color')`
   `merge(data1, data2, by='color')`
2. `merge(data1, data2, by='color', all.y=TRUE)`

Achtung: bei *merge* matchen auch *NA* Werte,
wenn nicht gewollt: Parameter *incomparables=NA*

Hochschule Niederrhein
University of Applied Sciences
Elektrotechnik und Informatik
Faculty of Electrical Engineering and Computer Science

Subqueries

Exists

SQL-Befehl:

```
SELECT color FROM data2 WHERE NOT EXISTS
   (SELECT 1 FROM data1
    WHERE data1.color = data2.color);
```

Entsprechung in R:

```
col1 <- unique(na.omit(data1$color))
subset(data2, !(color %in% col1), select=color)
```

Vergleich

SQL-Befehl:

```
SELECT * FROM data1 WHERE value =
   (SELECT MAX(value) FROM data1)
```

Entsprechung in R:

```
maxval <- max(data1$value, na.rm=TRUE)
subset(data2, value == maxval)
```

Gruppierung

SQL-Befehl:

```
SELECT label, MEAN(val) FROM data1
        GROUP BY label;
```

Entsprechung in R:

```
tapply(data1$value, data1$label,
        FUN=mean, na.rm=TRUE)
aggregate(data1$value, by=list(data1$label),
        FUN=mean, na.rm=TRUE)
```

Lösung durch explizite Schleife:

```
result <- data.frame()
for (x in unique(as.vector(data1$label))) {
  m <- mean(subset(data1, label==x, select=value), na.rm=TRUE)
  result <- rbind(result, data.frame(label=x,mean=m))
}
result
```