Datenbereinigung in R

Referent: Stefan Thieß
Betreuer: Dr. Julian Kunkel
Proseminar Programmieren in R

Gliederung

Einführung

- TidyR-Paket
- Typenumwandlung
- StrinR-Paket
- Fehlende Werte
- Extreme Werte
- Data-Warehouse

Einführung

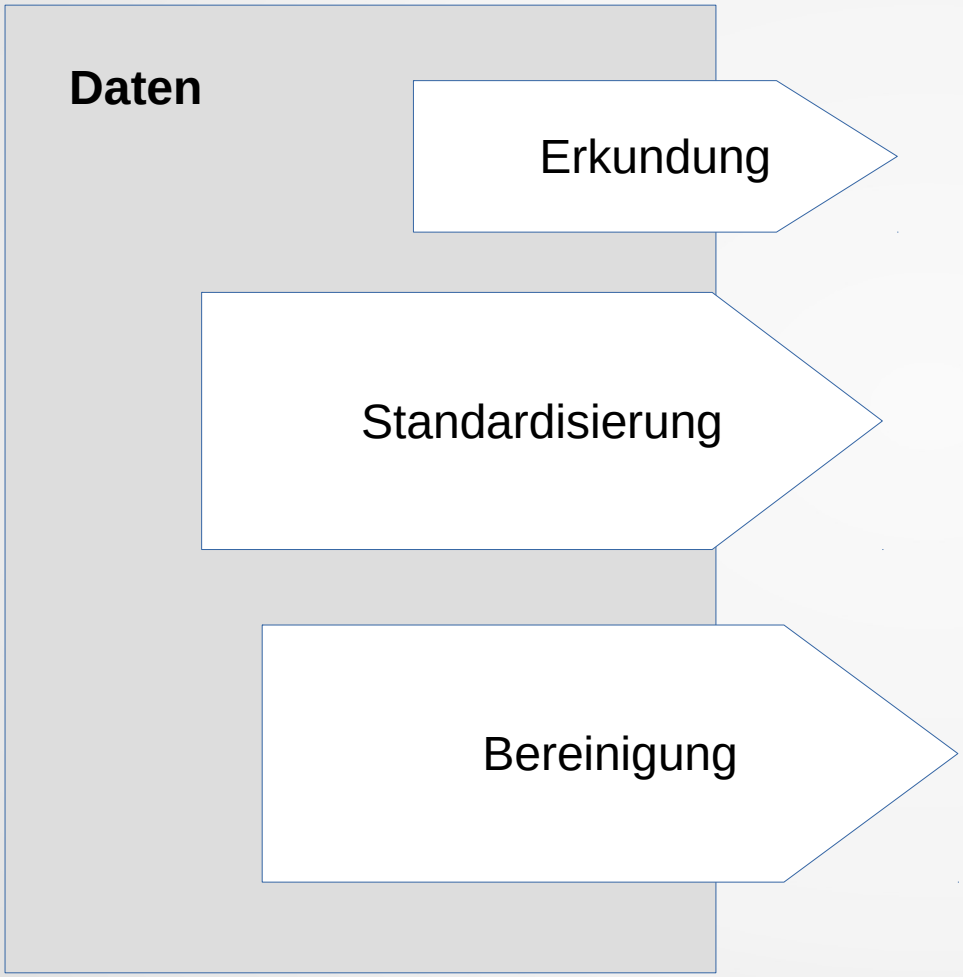
Datenbereinigung ist:

- Auffinden von Fehlern
- Erkennen von Widersprüchen
- Ersetzen von Daten

Ziel ist es:

- Die Datenqualität zu erhöhen
- Die Datenanalyse zu verbessern
- Entscheidungsgrundlagen verbessern

Einführung



- Verstehen

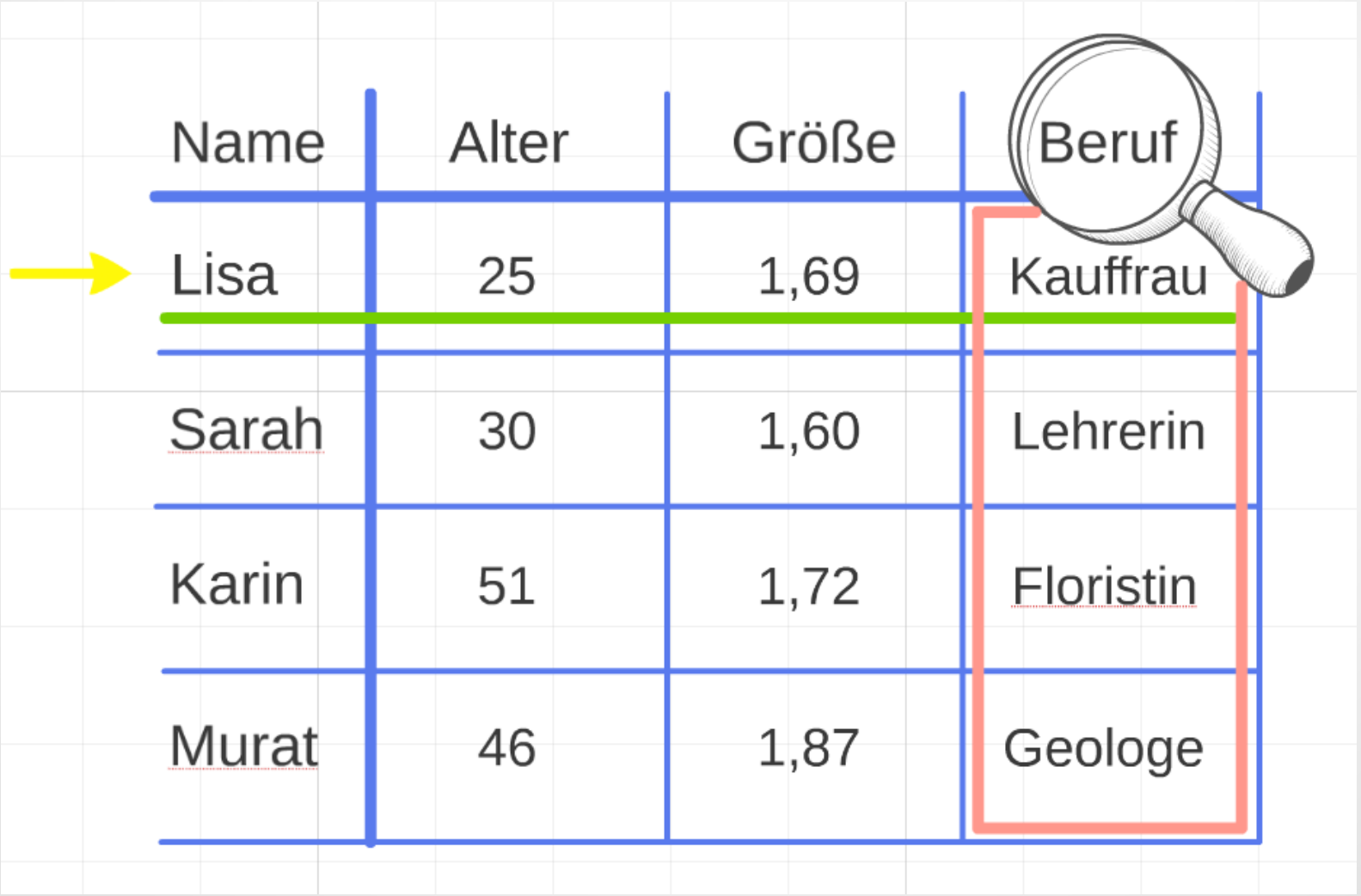
- Strukturieren
- Normieren

- Entfernen
- Ersetzen
- Ableiten
- Auftrennen

Abbildung:[AP1]

Das TidyR-Paket

Principel of tidy data by Handley Wickham



Name	Alter	Größe	Beruf
Lisa	25	1,69	Kauffrau
Sarah	30	1,60	Lehrerin
Karin	51	1,72	Floristin
Murat	46	1,87	Geologe

Das TidyR-Paket

Tidy data:

- Ein Objekt pro Reihe
- Eine Variable pro Spalte
- Ein Objekttyp pro Tabelle

Das TidyR-Paket

Dirty data:

- Spaltennamen sind Werte
- Variablen sind im Wertebereich
- mehrere Variablen sind in einer Spalte
- mehrere Objekte sind in einer Tabelle

Das TidyR-Paket

TidyR:

- Von Hadley Wickham
- Verändert Tabellen
- Funktionen
  - gather , seperate, spread, unite

TidyR-Paket::gather

Problem: Spaltennamen sind Werte

```
> New3
   Name Kauffrau Lehrerin Floristin
1  Lisa        1        0         0
2 Sarah        0        1         0
3 Karin        0        0         1
> tidyr::gather(New3,Beruf,Wahr,-Name)
   Name      Beruf Wahr
1  Lisa  Kauffrau    1
2 Sarah  Kauffrau    0
3 Karin  Kauffrau    0
4  Lisa  Lehrerin    0
5 Sarah  Lehrerin    1
6 Karin  Lehrerin    0
7  Lisa Floristin    0
8 Sarah Floristin    0
9 Karin Floristin    1
```

Syntax: gather(data, key, value, …)

Semantik:
data: Name des Datensatzes
key: Spaltenname für die Werte in der Spalte
value: Spaltenname für die Werte
…: -data: Spaltennamen, die bleiben sollen

TidyR-Paket::spread

Problem: Variablen sind Werte

```
> Pre5
   Name Kleidungsgegenstände Anzahl
1  Lisa                Jacken     12
2  Lisa                 Hosen      8
3  Lisa                Schuhe     27
4 Sarah                Jacken     24
5 Sarah                 Hosen     13
6 Sarah                Schuhe     21
> tidyr::spread(Pre5,Kleidungsgegenstände,Anzahl)
   Name Hosen Jacken Schuhe
1  Lisa     8     12     27
2 Sarah    13     24     21
```

Syntax: spread(data, key, value)

Semantik:
data: Name des Datensatzes
key: Name der Spalte, deren Wert zum Spaltenname werden soll
value: Name der Spalte ,die zum Wert werden soll

TidyR-Paket::seperate

Problem: mehrere Variablen sind in einer Spalte

```
> Pre4
   Name Alter.Größe     Beruf
1  Lisa       25/169  Kauffrau
2 Sarah       30/160  Lehrerin
3 Karin       51/172 Floristin
> tidyr::separate(Pre4,Alter.Größe,
 c("Alter", "Größe"), sep = "/")
   Name Alter Größe     Beruf
1  Lisa    25   169  Kauffrau
2 Sarah    30   160  Lehrerin
3 Karin    51   172 Floristin
```

Syntax:
seperate (data ,col ,into)

Semantik:
data = Name des Datensatzes
col = Spaltenname, welcher geteilt werden soll
into = c("Spaltennamen in die aufgeteilt werden soll“)

TidyR-Paket::unite

```
> unite
   Name Alter age Größe
1  Lisa    NA  25   169
2 Sarah    30  NA   160
3 Karin    51  NA   172
> tidyr::unite(unite,Neu_Alter,Alter,age)
   Name Neu_Alter Größe
1  Lisa     NA_25   169
2 Sarah     30_NA   160
3 Karin     51_NA   172
```

Syntax:
unite (data,col,...)

Semantik:
data = Name des Datensatzes
col = Spaltenname der neuen Spalte
... = Namen der Spalten die zusammengefasst werden sollen

TidyR-Paket:: Tidy data

Problem: mehrere Beobachtungseinheiten sind in einer Tabelle

```
> Pre6
   Name Alter Größe Tier_Name   Tierart Größe_Tier
1  Lisa    25   169     Bello      Hund         40
2 Sarah    30   160     Felix     Katze         20
3 Karin    51   172  Flauschi Kaninchen         10
```

```
> Pre62<-Pre6[1:3]
> Pre61<-Pre6[4:6]
```

```
> Pre62
   Name Alter Größe
1  Lisa    25   169
2 Sarah    30   160
3 Karin    51   172
```

```
> Pre61
  Tier_Name   Tierart Größe_Tier
1     Bello      Hund         40
2     Felix     Katze         20
3  Flauschi Kaninchen         10
```

Typenumwandlung

Typen in R:

- character:"....."
- numeric: 23.44, 120, NaN, Inf
- integer: 4L,1123L
- factor:factor("Hello"), factor(8)
- logical: TRUE, FALSE

Typenumwandlung

```
> Pre7
   Name Besitzen_ein_Auto
1  Lisa                 1
2 Sarah                 0
3 Karin                 1
>  Pre7$Besitzen_ein_Auto<-as.logical
(Pre7$Besitzen_ein_Auto)
> Pre7
   Name Besitzen_ein_Auto
1  Lisa              TRUE
2 Sarah             FALSE
3 Karin              TRUE
```

Typenumwandlung: lubridate

Das Paket lubridate

- Standardisiert Kalenderdaten und Uhrzeiten

```
> lubridate::ymd_hms("2016/7/13 10.30.09")
[1] "2016-07-13 10:30:09 UTC"
```

- ymd_hms = Jahr,Monat,Tag_Stunde,Minute,Sekunde
- sind frei kombinierbar

Typenumwandlung: lubridate

```
> PreTime
   Name Alter      Geburtstag
1  Lisa    25        21.05.90
2 Sarah    30          7/3/84
3 Karin    51 16. April 1969
> PreTime$Geburtstag<-lubridate::dmy(PreTime$Geburtstag)
> PreTime
   Name Alter Geburtstag
1  Lisa    25 1990-05-21
2 Sarah    30 1984-03-07
3 Karin    51 1969-04-16
```

Stringr-Paket

- Beinhaltet einfache Funktionen zur Veränderung von Strings
- Von Hadley Wickham programmiert

Funktionen:

- str_trim
- str_pad
- str_detect
- str_replace

Stringr-Paket

Anforderungen an Firmen-ID:

- beginnt mit „NEW“
- ist insgesamt 7 Zeichen lang
- alles hintereinander geschrieben

```
> str(StringR)
'data.frame':   3 obs. of  3 variables:
 $ Name      : chr  "Lisa" "Sarah" "Karin"
 $ Firmen.ID: chr  "oLd135    " "NEW23   85" "    new5821"
 $ Beruf     : chr  "Kauffrau" "Lehrerin" "Floristin"
```

Stringr-Paket

Firmen-ID's enthalten mehrere Fehler:

- Buchstaben sind klein
- Leerzeilen sind vor ,hinter und mitten in der Firmen-ID
- Teilweise sind Firmen-ID nicht 7 Zeichen lang

```
$ Firmen.ID: chr  "oLd135    " "NEW23    85" "    new5821"
```

Stringr-Paket

trim-Funktion:

- Entfernt alle Leerzeichen vor dem ersten und nach dem letzten Zeichen
- Syntax: str_trim(String, side = c("both", "left", "right"))

```
$ Firmen.ID: chr  "oLd135    " "NEW23    85" "    new5821"
```

```
> StringR$Firmen.ID<-stringr::str_trim(StringR$Firmen.ID)
```

```
$ Firmen.ID: chr  "oLd135" "NEW23    85" "new5821"
```

Stringr-Paket

replace_all-Funktion:

- Ersetzt Zeichen durch andere Zeichen
- Syntax: str_replace_all(string, pattern, replacement)

```
$ Firmen.ID: chr  "oLd135" "NEW23   85" "new5821"
```

```
> StringR$Firmen.ID<-stringr::str_replace_all
(StringR$Firmen.ID, pattern=" ", repl="")
```

```
$ Firmen.ID: chr  "oLd135" "NEW2385" "new5821"
```

Stringr-Paket

str_pad-Funktion:

- Fügt Zeichen an
- Syntax: str_pad(string, width, side = c("left", "right", "both"), pad = " ")

```
$ Firmen.ID: chr  "oLd135" "NEW2385" "new5821"
```

```
> StringR$Firmen.ID<-stringr::str_pad(StringR$Firmen.ID
, width = 7, side = "right", pad = "0")
```

```
$ Firmen.ID: chr  "oLd1350" "NEW2385" "new5821"
```

Stringr-Paket

toupper-Funktion:

- Macht alle Buchstaben groß
- Syntax: toupper(x)

```
$ Firmen.ID: chr  "oLd1350" "NEW2385" "new5821"
```

```
> StringR$Firmen.ID<-toupper(StringR$Firmen.ID)
```

```
$ Firmen.ID: chr  "OLD1350" "NEW2385" "NEW5821"
```

Stringr-Paket

replace_all-Funktion:

- „Old“ wird durch „NEW“ ersetzt

```
$ Firmen.ID: chr  "OLD1350" "NEW2385" "NEW5821"
```

```
> StringR$Firmen.ID<-stringr::str_replace_all(StringR
$Firmen.ID, pattern = "OLD", replacement = "NEW")
```

```
    Name Firmen.ID      Beruf
1   Lisa    NEW1350  Kauffrau
2 Sarah    NEW2385  Lehrerin
3 Karin    NEW5821 Floristin
```

Fehlende Werte

Die Attribute eines Datensatzes sind mit Werten belegt, die semantisch vom Wert NULL abweichen.

Fehlende Werte

Fehlende Werte:

- In R als NA
- #N/A in Excel
- "." in SAS
- als Leerzeichen in weiteren Programmen

Fehlende Werte

Suche fehlender Daten:

- is.na(data)
- any(is.na(data)
- sum(is.na(data))
- summary(data)

Fehlende Werte

is.na-Funktion:

- Gibt einen Wahrheitswert wieder
- In Form des Datentyps der Datei
- Syntax: is.na(data)

```
> mis
    Name Alter Größe
1   Lisa    25   169
2 Sarah    NA    NA
3 Karin    NA   172
```

```
> is.na(mis)
      Name Alter Größe
[1,] FALSE FALSE FALSE
[2,] FALSE  TRUE  TRUE
[3,] FALSE  TRUE FALSE
```

Fehlende Werte

Abwandlungen der is.na-Funktionen:

any(is.na(data)):

- Gibt an, ob irgendein NA vorliegt

```
> any(is.na(mis))
[1] TRUE
```

sum(is.na(data)):

- Summiert die NA's im Datensatz

```
> sum(is.na(mis))
[1] 3
```

Fehlende Werte

complete.cases-Funktion

- Gibt eine Wahrheitswert wieder
- Syntax: complete.cases(data)

```
> complete.cases(mis)
[1]  TRUE FALSE FALSE
```

```
> mis
    Name Alter Größe
1   Lisa    25   169
2 Sarah    NA    NA
3 Karin    NA   172
```

Fehlende Werte

na.omit-Funktion:

- Entfernt alle Reihen mit NA's
- Syntax: na.omit(data)

```
> na.omit(mis)
  Name Alter Größe
1 Lisa    25   169
```

```
> mis
    Name Alter Größe
1   Lisa    25   169
2  Sarah    NA    NA
3  Karin    NA   172
```

Fehlende Werte

Spalten mit fehlenden Werten entfernen:

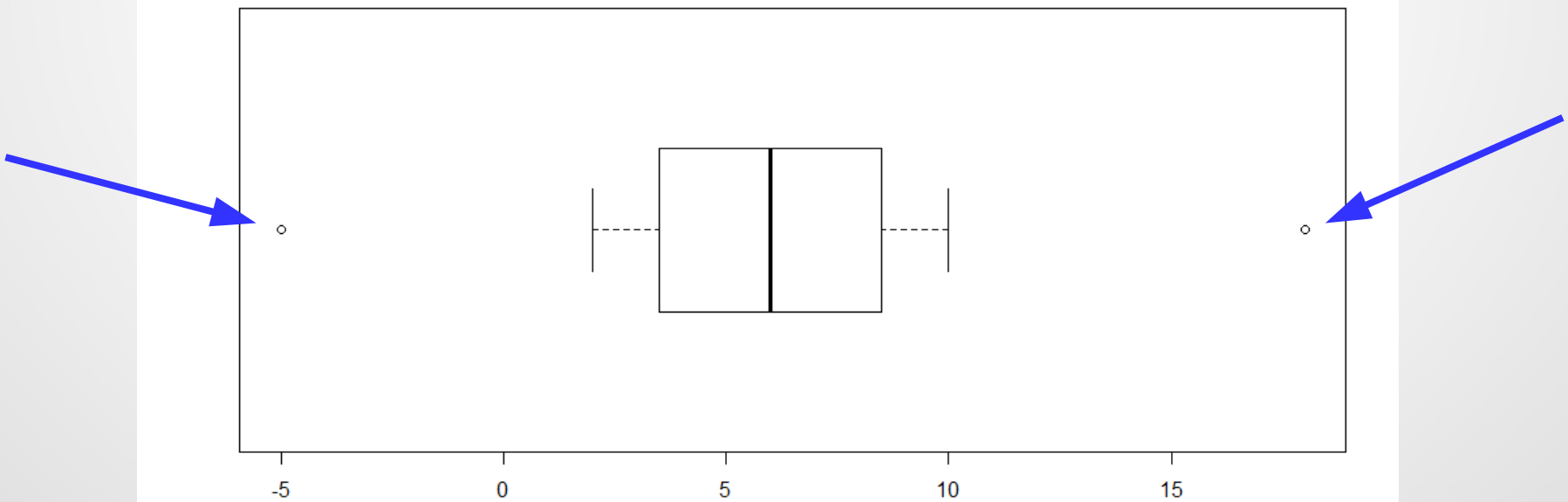
```
> mis2[,complete.cases(mis)]
[1] "Lisa"  "Sarah" "Karin"
```

```
> mis
   Name Alter Größe
1  Lisa    25   169
2 Sarah    NA    NA
3 Karin    NA   172
```

Extreme Werte

Outliers sind extreme Wert , der von dem Großteil der anderen Werte abweicht.

```
> num = c(-5,2,3, 4,5, 6,7, 8,9, 10, 18)
> boxplot(num, horizontal = TRUE)
```



Extreme Werte

Offensichtliche Fehler:

Formen:

- Werte sind nicht plausibel

  z.B. Alter einer Person beträgt 243 Jahre
- Werte machen keinen Sinn

  z.B. Alter einer Person beträgt -5 Jahre
- Werte liegen in verschiedenen Einheiten vor

  z.B. Größe, Gewicht, Geschwindigkeit

Extreme Werte

Offensichtliche Fehler finden mit editrules:

- Möglichkeit zur Erstellung von Bedingungen
- Bedingung können in Objekte gespeichert werden
- Bedingungen können im Editor gemacht werden

Extreme Werte

Offensichtliche Fehler finden mit editset:

Syntax:

- editset(c(“Bedingung“))

```
> (R<-editset(c("Alter>=0", "Alter<=150"
, "Größe>=40","Größe<= 275")))

Edit set:
num1 : 0 <= Alter
num2 : Alter <= 150
num3 : 40 <= Größe
num4 : Größe <= 275
```

Extreme Werte

Offensichtliche Fehler finden mit violatedEdits:

Syntax: violatedEdits(editrule-Variable ,data)

```
> editR
   Name Alter Größe
1  Lisa     25   350
2 Sarah    180    25
3 Karin    -51   172
> violatedEdits(R, editR)
      edit
record  num1  num2  num3  num4
     1 FALSE FALSE FALSE  TRUE
     2 FALSE  TRUE  TRUE FALSE
     3  TRUE FALSE FALSE FALSE
```

Extreme Werte

```
> editR
   Name Alter Größe
1  Lisa    25   350
2 Sarah   180    25
3 Karin   -51   172
> violatedEdits(R, editR)
      edit
record  num1  num2  num3  num4
     1 FALSE FALSE FALSE  TRUE
     2 FALSE  TRUE  TRUE FALSE
     3  TRUE FALSE FALSE FALSE
> R

Edit set:
num1 : 0 <= Alter
num2 : Alter <= 150
num3 : 40 <= Größe
num4 : Größe <= 275
```

Vollständigkeit

Fehlende Daten ersetzen mit:

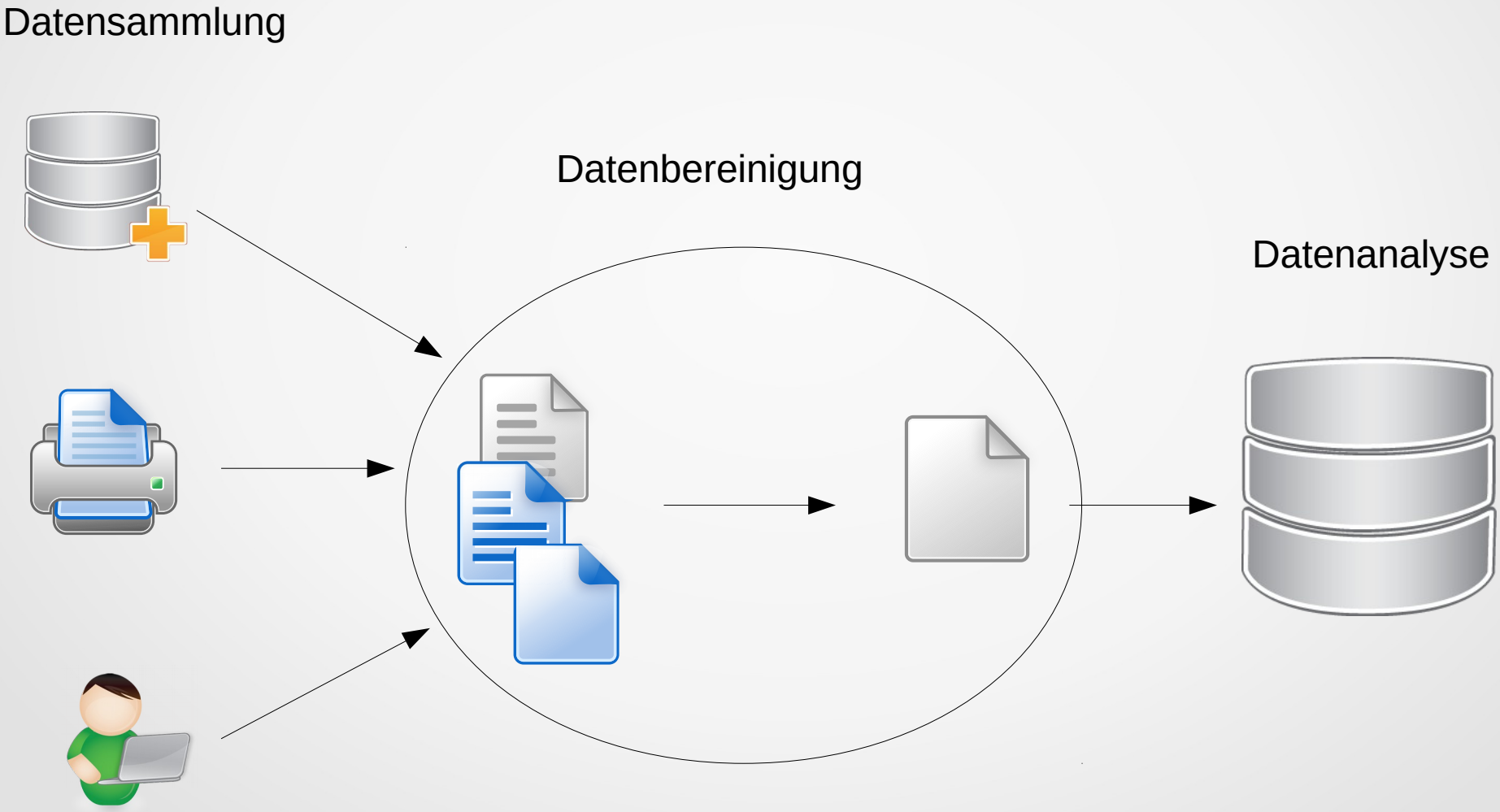
```
> data
  Firmenname Einnahmen Ausgaben Gewinn
1     FirmaA       1000       NA    600
2     FirmaB         NA      400    900
3     FirmaC        550      300     NA
```

```
> Cor<-editset(expression(Einnahmen - Ausgaben == Gewinn))
```

```
> data_new<-deduImpute(Cor, data)
```

```
> data_new$corrected
  Firmenname Einnahmen Ausgaben Gewinn
1     FirmaA       1000      400    600
2     FirmaB       1300      400    900
3     FirmaC        550      300    250
```

Data-Warehouse und R



Data-Warehouse und R

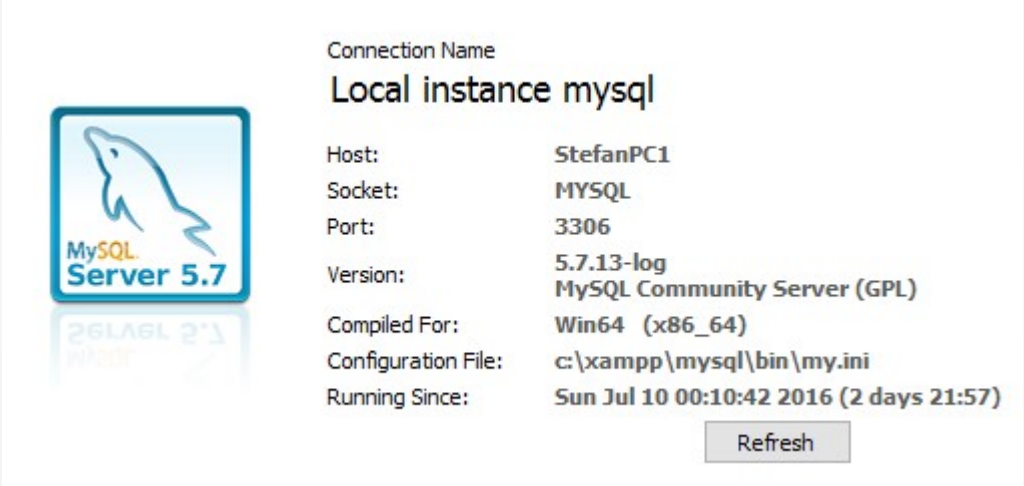
Frage: Wie gestaltet sich der Datenimport/export in R mit Datenbanken?

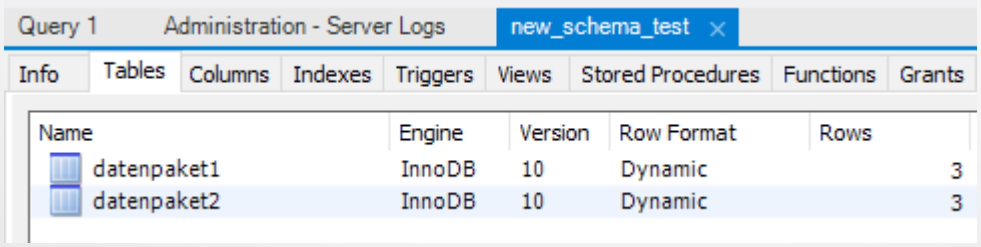
Pakete zu verbinden mit Datenbanken:

System	Paket in R
ODBC	RODBC
MySQL	RMySQL
Oracle	ROracle
PostgreSQL	RPostgreSQL
SQLite	RSQlite

Data-Warehouse und R

Beispiel: RmySQL – R mit Datenbank verbinden





Data-Warehouse und R

Beispiel: RmySQL – auf Daten zugreifen

```
> DataBase = dbConnect(MySQL(), user='root', password=
'[redacted]', dbname='new_schema_test', host='localhost'
)
> dbListTables(DataBase)
[1] "datenpaket1" "datenpaket2"
> dbReadTable(DataBase,"datenpaket1")
    Name Alter GrÃ.ÃŸe    X.Beruf
1 Thomas    28     178 FuÃŸballer
2    Dio    34     186       Koch
3  Murat    51     192  Disponent
> dbReadTable(DataBase,"datenpaket2")
   Name Alter GrÃ.ÃŸe     Beruf
1  Lisa    25     169  Kauffrau
2 Sarah    30     160  Lehrerin
3 Karin    51     172 Floristin
```

Datensätze zusammenführen

```
> DatenPaket1
    Name Alter Größe       Beruf
1 Thomas     28   178  Fußballer
2    Dio     34   186       Koch
3  Murat     51   192  Disponent
```

```
> DatenPaket2
   Name Alter Größe      Beruf
1  Lisa     25   169  Kauffrau
2 Sarah     30   160  Lehrerin
3 Karin     51   172 Floristin
```

```
> DatenPaketFinal<-rbind(DatenPaket1,DatenPaket2)
> DatenPaketFinal
    Name Alter Größe       Beruf
1 Thomas     28   178  Fußballer
2    Dio     34   186       Koch
3  Murat     51   192  Disponent
4   Lisa     25   169   Kauffrau
5  Sarah     30   160   Lehrerin
6  Karin     51   172  Floristin
```

Zusammenfassung

Datenreinigung in R:

- tidyR – verändert Tabellen
- lubridate – standardisiert Zeitdaten
- StrinR – verändert Zeichen
- is.na() - finden von fehlenden Werten
- Editset – falsche/fehlende Werte ersetzen
- RmySQL – Datenaustausch zwischen R und MySQL

Quellenverzeichnis

[AP1]“Datenqualität erfolgreich Steuern“, Hanser Verlag, Seite: 157

Weitere Quellen:

- Data Cleaning:Problems and Current Approaches, Erhard Rahm, Hong Hai Do
- Tidy data , Hadley Wickham, 2009