Data Wrangling - Datenaufbereitung

Contents

1 Data Wrangling - Datenaufbereitung

Datenaufbereitung in R mit dem package `dplyr`.

1.1 Grammatik der Datenaufbereitung

Hadley Wickham, einer der Autoren von `dplyr` hat fünf Verben für die Arbeit mit Daten in einem `dataframe` identifiziert:

- `select()` Teilmenge an Spalten (d. h. Features, Variablen) auswählen
- `filter()` Teilmenge an Zeilen (d. h. Beobachtungen) auswählen
- `mutate()` Spalten hinzufügen oder modifizieren
- `arrange()` sortieren von Zeilen (d. h. Beobachtunen)
- `summarize()` aggregieren von Spalten (über Zeilen hinweg) z. B. gruppieren nach bestimmten Kriterien

Jeder Befehl hat einen `dataframe` als erstes (Input-)Argument und einen `dataframe` als Rückgabewert.

- die Befehle können zusammen (verschachtelt) verwendet werden
- zum Aufbereiten einer **einzelnen Datentabelle**.

Lade das package `dplyr` damit `select()`, `filter()`, etc. verwenden kann.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Es kann auch `tidyverse` geladen werden, denn es enthält `dplyr`.

1.2 Beispieldaten

Als Beispielsdaten dienen die folgenden Noten von drei Studenten

```
df <- data.frame(
  student = rep(c("Anna","Beate","Chris"),3),
  semester= rep(seq(from=1, to=6, by=2)  ,3),
  note    = c(1,2,1,4,5,1,3,3,3)
  )
df
```

```
##   student semester note
## 1    Anna        1    1
## 2   Beate        3    2
## 3   Chris        5    1
## 4    Anna        1    4
## 5   Beate        3    5
## 6   Chris        5    1
## 7    Anna        1    3
## 8   Beate        3    3
## 9   Chris        5    3
```

```
str(df)
```

```
## 'data.frame':    9 obs. of  3 variables:
##  $ student : chr  "Anna" "Beate" "Chris" "Anna" ...
##  $ semester: num  1 3 5 1 3 5 1 3 5
##  $ note    : num  1 2 1 4 5 1 3 3 3
```

Ein `dataframe` ist nötig aber `tibble` ist nicht nötig.

1.2.1 `select()` - Auswählen von Spalten (Variablen oder Features)

```r
select(df, student, semester)
```

```
##   student semester
## 1    Anna        1
## 2   Beate        3
## 3   Chris        5
## 4    Anna        1
## 5   Beate        3
## 6   Chris        5
## 7    Anna        1
## 8   Beate        3
## 9   Chris        5
```

Mit Befehl `unique()` aus dem package `base` jeden Studenten(-Namen) nur einmal anzeigen. Das package `base` wird beim starten von `R` automatisch geladen.

```r
unique(select(df, student, semester))
```

```
##   student semester
## 1    Anna        1
## 2   Beate        3
## 3   Chris        5
```

1.2.2 `filter()` - Auswählen von Zeilen (Wertebereich einschränken)

```r
filter(df, note==1)
```

```
##   student semester note
## 1    Anna        1    1
## 2   Chris        5    1
## 3   Chris        5    1
```

```r
filter(df, note<=2)
```

```
##   student semester note
## 1    Anna        1    1
## 2   Beate        3    2
## 3   Chris        5    1
## 4   Chris        5    1
```

```r
filter(df, note==1 & student != "Anna")
```

```
##   student semester note
## 1   Chris        5    1
## 2   Chris        5    1
```

```
unique(
  select(
    filter(df, note==1 & student != "Anna"), student
  )
)
```

```
##   student
## 1   Chris
```

```
filter(df, note>1 & student %in% c("Anna", "Chris") )
```

```
##   student semester note
## 1    Anna        1    4
## 2    Anna        1    3
## 3   Chris        5    3
```

Die Variable `student` wird für die Überprüfung von `student=="Anna"` benötigt und daher `filter()` vor `select()` ausgeführt.

1.2.3 `mutate()` - Variable `textnote` hinzufügen

```
df <-
mutate(df, textnote = ifelse(note==1, "sehr gut",
                             ifelse(note==2, "gut",
                                    ifelse(note==3, "befriedigend",
                                           ifelse(note==4, "genügend", "nicht genügend")
                                           )
                                    )
                             )
       )
df
```

```
##   student semester note       textnote
## 1    Anna        1    1       sehr gut
## 2   Beate        3    2            gut
## 3   Chris        5    1       sehr gut
## 4    Anna        1    4       genügend
## 5   Beate        3    5 nicht genügend
## 6   Chris        5    1       sehr gut
## 7    Anna        1    3   befriedigend
## 8   Beate        3    3   befriedigend
## 9   Chris        5    3   befriedigend
```

1.2.4 `mutate()` - Variable `textnote` verändern

Varialbe `textnote` vom Typ `<chr>` in den Typ `<fct>` ändern:

```
df <- mutate(df, textnote = as.factor(textnote))
df
```

```
##   student semester note       textnote
## 1    Anna        1    1       sehr gut
## 2   Beate        3    2            gut
## 3   Chris        5    1       sehr gut
## 4    Anna        1    4       genügend
## 5   Beate        3    5 nicht genügend
## 6   Chris        5    1       sehr gut
## 7    Anna        1    3   befriedigend
## 8   Beate        3    3   befriedigend
## 9   Chris        5    3   befriedigend
```

Rename variable `student` into Varialbe `student_name` :

```
rename(df, student_name = student)   ## NB: new_variable = Old_variable (--> old_variable vaishes)
```

```
##   student_name semester note       textnote
## 1         Anna        1    1       sehr gut
## 2        Beate        3    2            gut
## 3        Chris        5    1       sehr gut
## 4         Anna        1    4       genügend
## 5        Beate        3    5 nicht genügend
## 6        Chris        5    1       sehr gut
## 7         Anna        1    3   befriedigend
## 8        Beate        3    3   befriedigend
## 9        Chris        5    3   befriedigend
```

Mit `rename(df,`**`new_variable`**` = `**`old_variable`**`)` und die **old_variable** verschwindet

1.2.5 `arrange()` - Zeilen sortieren nach Spalteneinträgen

```
arrange(df, note, desc(semester))
```

```
##   student semester note       textnote
## 1   Chris        5    1       sehr gut
## 2   Chris        5    1       sehr gut
## 3    Anna        1    1       sehr gut
## 4   Beate        3    2            gut
## 5   Chris        5    3   befriedigend
## 6   Beate        3    3   befriedigend
## 7    Anna        1    3   befriedigend
## 8    Anna        1    4       genügend
## 9   Beate        3    5 nicht genügend
```

1.2.6 `summarize()` - Daten zusammenfassen

In amerikanischer `summarize` und englischer `summarise` Schreibweise verfügbar.

```
summarize(df,       ## dataframe is 1. Arguemnt
          N=n(),    ## always use N=n() for proper functioning
          notenschnitt= mean(note),
          best_note   = min(note),
          worst_note  = max(note)
          )
```

```
##   N notenschnitt best_note worst_note
## 1 9     2.555556         1          5
```

`N=n()`, immer verwenden damit `summarize()` einwandfrei funktioniert

1.2.7 `summarize()` und `group_by()` - Daten gruppeniert zusammenfassen

Die Daten nach Studenten(-Namen) gruppieren und zusammenfassen:

```
summarize(group_by(df, student),   ## dataframe is 1. Arguemnt
          N=n(),                   ## always use N=n()
          notenschnitt= mean(note),
          best_note   = min(note),
          worst_note  = max(note)
          ) %>% arrange(notenschnitt)  ## use %>% pipe  to ammend by another command
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 5
##   student     N notenschnitt best_note worst_note
##   <chr>   <int>        <dbl>     <dbl>      <dbl>
## 1 Chris       3         1.67         1          3
## 2 Anna        3         2.67         1          4
## 3 Beate       3         3.33         2          5
```

`group_by` macht aus dem `dataframe` einen `tibble`

Rewrite with `%>%` Pipe and select only student und notenschnitt:

```
df %>% group_by(student) %>% summarise(
          N=n(),                   ## always use N=n()
          notenschnitt= mean(note),
          best_note   = min(note),
          worst_note  = max(note)) %>%
  arrange(notenschnitt) %>%
  select(student, notenschnitt)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 2
```

```
##   student notenschnitt
##   <chr>          <dbl>
## 1 Chris           1.67
## 2 Anna            2.67
## 3 Beate           3.33
```

Die Daten nach Noten gruppieren und zusammenfassen:

```
df %>% group_by(note) %>% summarize(N=n()) %>%
  arrange(desc(N)) %>%
  select(N, note)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 5 x 2
##       N  note
##   <int> <dbl>
## 1     3     1
## 2     3     3
## 3     1     2
## 4     1     4
## 5     1     5
```