

# Daten bündigen mit dplyr und tidyr Schummelzettel



## Syntax - hilfreiche Konventionen zur Formatierung

### dplyr::tbl\_df(iris)

Verwandelt Daten in eine „tbl“ Klasse. Eine „tbl“ (alias Tabelle) ist einfacher einzusehen als ein „data frame“ (alias Datenframe). R zeigt nur die Daten an, die auf den Bildschirm passen:

```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length
1             5.1         3.5         1.4
2             4.9         3.0         1.4
3             4.7         3.2         1.3
4             4.6         3.1         1.5
5             5.0         3.6         1.4
..          ...         ...         ...
Variables not shown: Petal.Width (dbl),
Species (fctr)
```

### dplyr::glimpse(iris)

Zusammenfassung der „tbl“ Daten.

### utils::View(iris)

Zeigt den Datensatz in Tabellenformat an (Großschreibung von V beachten).

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

### dplyr::%>%

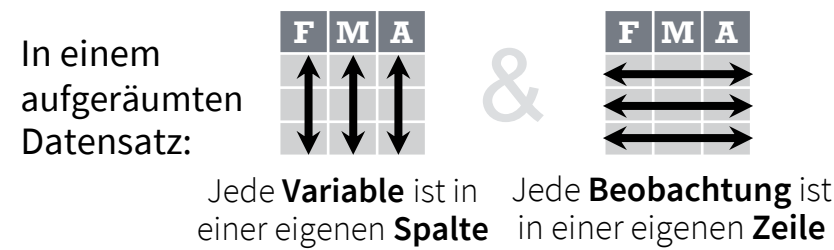
Verwendet das Objekt links vom Symbol als ersten Eingabeparameter (oder . Eingabeparameter) der Funktion auf der rechten Seite des Symbols.

**x %>% f(y)** ist identisch zu **f(x, y)**  
**y %>% f(x, ., z)** ist identisch zu **f(x, y, z)**

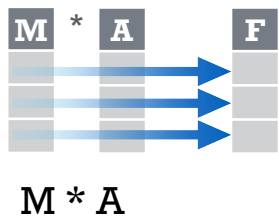
Die „Pipe“ (alias Verkettung) mit %>% macht den Code lesbarer, z. B.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

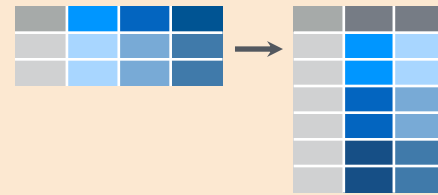
## Daten aufräumen - eine Basis der Datenmanipulation in R



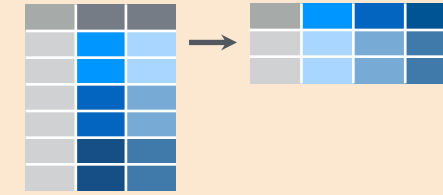
Aufgeräumte Daten ergänzen die **vektorierten Operationen** in R. Beobachtungen (Zeilen) bleiben automatisch erhalten wenn Variablen (Spalten) manipuliert werden. Kein anderes Format hat ein solch intuitives Zusammenspiel mit R.



## Daten umformen - verändert das Layout eines Datensatzes



**tidyr::gather(cases, "year", "n", 2:4)**  
Spalten als Zeilen zusammenziehen.



**tidyr::spread(pollution, size, amount)**  
Zeilen als Spalten ausbreiten.



**tidyr::separate(storms, date, c("y", "m", "d"))**  
Eine Spalte in mehrere aufteilen.



**tidyr::unite(data, col, ..., sep)**  
Mehrere Spalten zu einer vereinigen.

- dplyr::data\_frame(a = 1:3, b = 4:6)**  
Vektoren in einem „data frame“ verbinden (optimiert).
- dplyr::arrange(mtcars, mpg)**  
Zeilen anhand von Werten in einer Spalte sortieren (von klein nach groß).
- dplyr::arrange(mtcars, desc(mpg))**  
Zeilen anhand von Werten in einer Spalte sortieren (von groß nach klein).
- dplyr::rename(tb, y = year)**  
Spalten von einem „data frame“ umbenennen.

## Beobachtungen (Zeilen) filtern



- dplyr::filter(iris, Sepal.Length > 7)**  
Zeilen herausfiltern die eine Bedingung erfüllen.
- dplyr::distinct(iris)**  
Duplikate entfernen (zeilenweise).
- dplyr::sample\_frac(iris, 0.5, replace = TRUE)**  
Bruchteil der Zeilen stichprobenartig auswählen.
- dplyr::sample\_n(iris, 10, replace = TRUE)**  
n Zeilen stichprobenartig auswählen.
- dplyr::slice(iris, 10:15)**  
Zeilen anhand ihrer Position auswählen.
- dplyr::top\_n(storms, 2, date)**  
Beste n Einträge auswählen und sortieren (nach Gruppe falls die Daten gruppiert sind).

## Variablen (Spalten) filtern



- dplyr::select(iris, Sepal.Width, Petal.Length, Species)**  
Spalten anhand ihres Namens (oder mittels nachstehend angeführter Hilfsfunktionen) auswählen.

### Hilfsfunktionen für select - ?select

- select(iris, contains("."))**  
Spalten auswählen deren Name eine Zeichenkette beinhaltet.
- select(iris, ends\_with("Length"))**  
Spalten auswählen deren Name mit einer Zeichenkette ended.
- select(iris, everything())**  
Alle Spalten auswählen.
- select(iris, matches(".t."))**  
Spalten auswählen deren Name mit einem regulären Ausdruck übereinstimmt.
- select(iris, num\_range("x", 1:5))**  
Spalten mit den Name x1, x2, x3, x4 und x5 auswählen.
- select(iris, one\_of(c("Species", "Genus")))**  
Spalten auswählen deren Namen in einer List mehrerer Namen sind.
- select(iris, starts\_with("Sepal"))**  
Spalten auswählen deren Name mit einer Zeichenkette beginnt.
- select(iris, Sepal.Length:Petal.Width)**  
Alle Spalten von Sepal.Length bis Petal.Width (inklusive) auswählen.
- select(iris, -Species)**  
Alle Spalten außer Species auswählen.

### Logik in R - ?Comparison, ?base::Logic

<	kleiner als	!=	ungleich
>	größer als	%in%	Gruppenzugehörigkeit
==	gleich	is.na	ist NA
<=	kleiner gleich	!is.na	ist keinNA
>=	größer gleich	&,  , !, xor, any, all	boolesche Operatoren

## Daten zusammenfassen



`dplyr::summarise(iris, avg = mean(Sepal.Length))`

Daten in eine einzelne Zeile zusammenfassen.

`dplyr::summarise_each(iris, funs(mean))`

Zusammenfassungs-Funktion auf jede Spalte anwenden.

`dplyr::count(iris, Species, wt = Sepal.Length)`

Anzahl der Zeilen mit jedem eindeutigen Wert der Variablen (mit oder ohne Gewichtung) zählen.



Summarise verwendet **Zusammenfassungs-Funktionen**, d. h. Funktionen die einen Vektor als Eingabe haben und einen einzelnen Ausgabewert haben, z. B.

**dplyr::first**

Erster Wert eines Vektors.

**dplyr::last**

Letzter Wert eines Vektors.

**dplyr::nth**

n-ter Wert eines Vektors.

**dplyr::n**

Anzahl der Werte eines Vektors.

**dplyr::n\_distinct**

Anzahl der unterschiedlichen Werte eines Vektors.

**IQR**

Interquartilsabstand eines Vektors.

**min**

Minimalwert eines Vektors.

**max**

Maximalwert eines Vektors.

**mean**

Arithmetisches Mittel eines Vektors.

**median**

Median eines Vektors.

**var**

Varianz eines Vektors.

**sd**

Standardabweichung eines Vektors.

## Daten gruppieren

`dplyr::group_by(iris, Species)`

Daten in Zeilen gruppieren, die den selben Wert in Species haben.

`dplyr::ungroup(iris)`

Gruppierung im „data frame“ aufheben.

`iris %>% group_by(Species) %>% summarise(...)`

Separate Zusammenfassung für jede Gruppe berechnen.



## Neue Variablen erstellen



`dplyr::mutate(iris, sepal = Sepal.Length + Sepal.Width)`

Neue Spalten berechnen und hinzufügen.

`dplyr::mutate_each(iris, funs(min_rank))`

Fenster-Funktion auf jede Spalte anwenden.

`dplyr::transmute(iris, sepal = Sepal.Length + Sepal.Width)`

Neue Spalten berechnen. Ursprüngliche Spalten entfernen.



Mutate verwendet **Fenster-Funktionen**, d. h. Funktionen die einen Vektor als Eingabe und ebenfalls als Ausgabe haben, z. B.

**dplyr::lead**

Werteverschiebung um 1 nach vorne.

**dplyr::lag**

Werteverschiebung um 1 nach hinten.

**dplyr::dense\_rank**

Rangordnung ohne Lücke.

**dplyr::min\_rank**

Rangordnung. Kleinerer Rang bei Gleichstand.

**dplyr::percent\_rank**

Rangordnung skaliert auf [0, 1].

**dplyr::row\_number**

Rangordnung. Erster Wert bekommt den kleineren Rang bei Gleichstand.

**dplyr::ntile**

Vektor in n Behälter aufteilen.

**dplyr::between**

Sind Werte zwischen a und b?

**dplyr::cume\_dist**

Summenverteilung

**dplyr::cumall**

Kumulativ alle („all“)

**dplyr::cumany**

Kumulativ irgendeines („any“)

**dplyr::cummean**

Kumulativer Mittelwert („mean“)

**cumsum**

Kumulative Summe

**cummax**

Kumulatives Maximum

**cummin**

Kumulatives Minimum

**cumprod**

Kumulatives Produkt

**pmax**

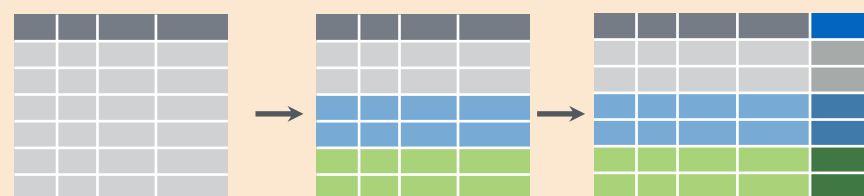
Elementweises Maximum

**pmin**

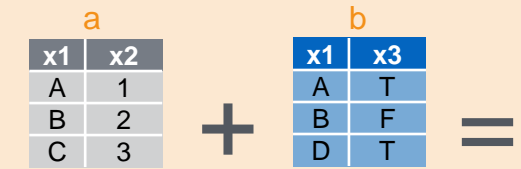
Elementweises Minimum

`iris %>% group_by(Species) %>% mutate(...)`

Neue Variablen für jede Gruppe berechnen.



## Datensätze zusammenführen



Verändernde Verbindung (engl. „join“)

**dplyr::left\_join(a, b, by = "x1")**

Alle Datensätze von a mit passenden Datensätzen von b.

**dplyr::right\_join(a, b, by = "x1")**

Alle Datensätze von b mit passenden Datensätzen von a.

**dplyr::inner\_join(a, b, by = "x1")**

Schnittmenge beider Datensätze. Beinhaltet nur Zeilen die in beiden Datensätzen vorkommen.

**dplyr::full\_join(a, b, by = "x1")**

Alle Datensätze von a und b. Ihre Schnittmenge wird zusammengefasst.

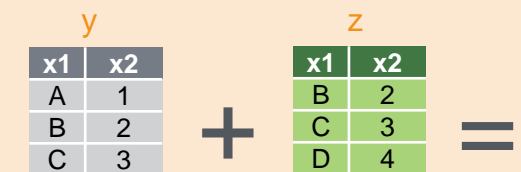
Filternde Verbindung

**dplyr::semi\_join(a, b, by = "x1")**

Alle Zeilen von a mit Übereinstimmung in b.

**dplyr::anti\_join(a, b, by = "x1")**

Alle Zeilen von a ohne Übereinstimmung in b.



Mengen-Operation

**dplyr::intersect(y, z)**

Schnittmenge, d. h. Zeilen die in beiden Datensätzen vorkommen.

**dplyr::union(y, z)**

Vereinigungsmenge, d. h. Zeilen die in einem oder beiden Datensätzen vorkommen.

**dplyr::setdiff(y, z)**

Differenzmenge/Restmenge/Komplement, d. h. Zeilen von y die nicht in z vorkommen.

Verbindung

**dplyr::bind\_cols(y, z)**  
z neben y als neue Spalten anfügen. Achtung: Zeilen anhand ihrer Position bestimmt!

**dplyr::bind\_rows(y, z)**  
z unterhalb von y als neue Zeilen anfügen.