

Große Daten analysieren mit dplyr

OK, heute nur **kleine** große Daten

Sebastian Sauer

Was machen wir heute? Und warum?

WAS:

- Vertraut machen mit einem schönem **Werkzeug** zur **praktischen Datenanalyse** (**dplyr**)
- Einüben von **explorativer** ("praktischer") **Analyse** von **ziemlich großen Daten**

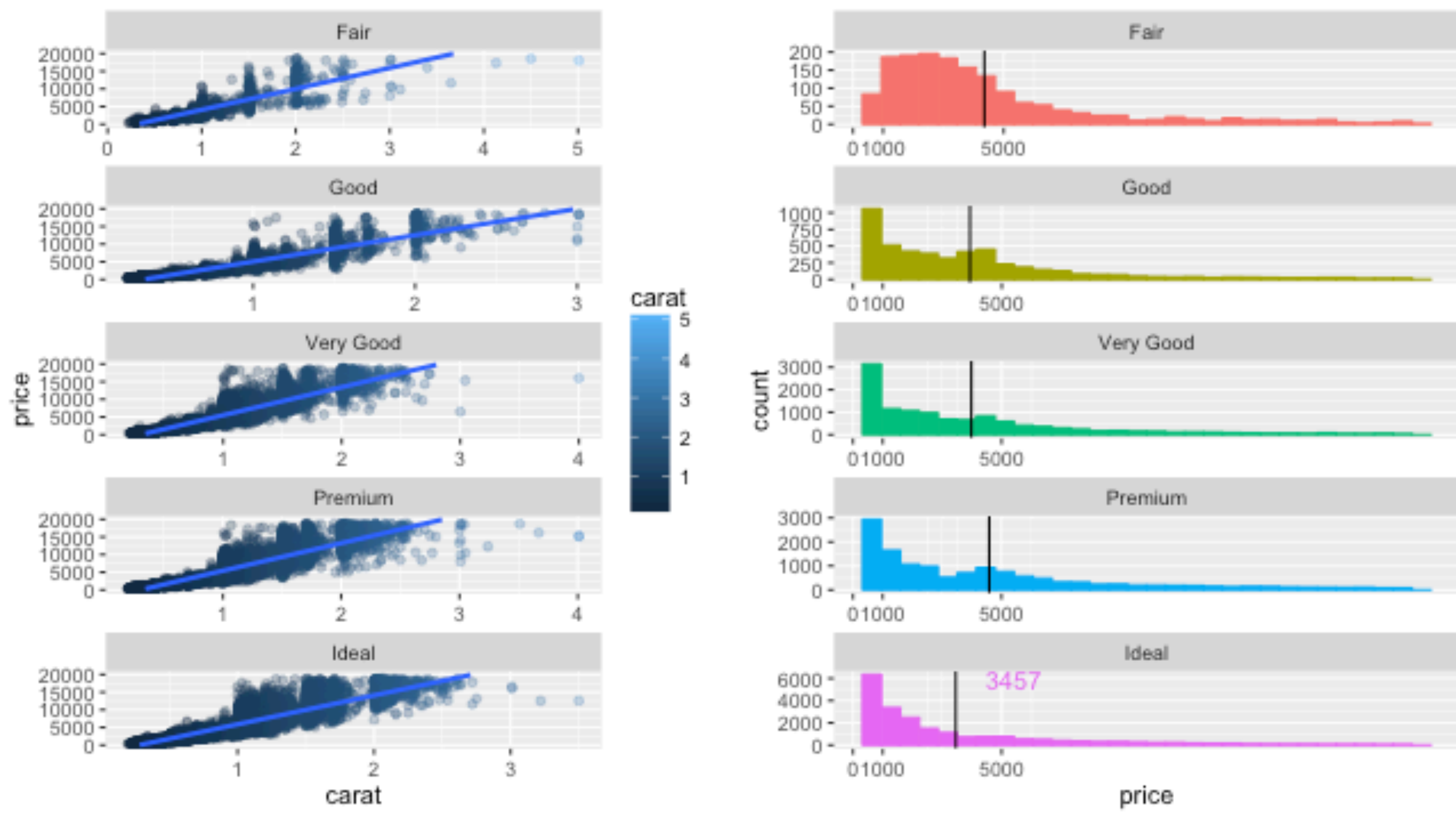
WARUM:

- Datenanalyse in Wissenschaft und Praxis besteht zum **großen Teil** im **Aufbereiten** und **Explorieren** des Datensatzes
- Die **Größe** von Datensätzen **steigt** schnell
- Wer den Schritt vom Gelegenheitsspieler zum **Routinetäter** gehen will, braucht **Profi-Werkzeug**

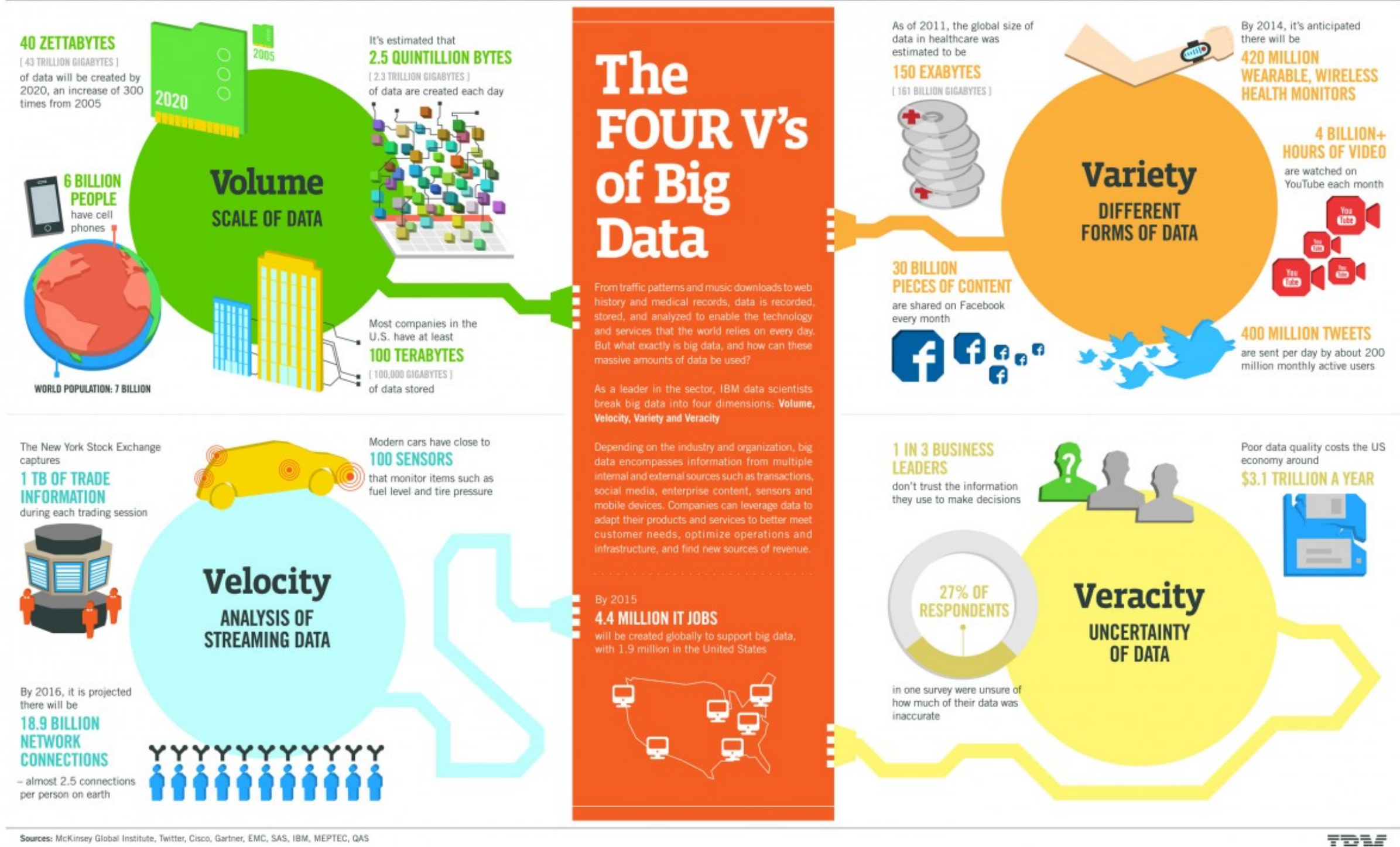
Excel oder R oder ...?

.	Excel	R	SQL
schon bekannt	X		
kleine Daten (<105 Zeilen)	X	X	
kleine große Daten (<10 GB)		X	
große Daten (>10 GB)			X
automatisierbar		X	X
transparent		X	X
moderne Statistik		X	
schöne Diagramme		X	
Interaktive Applets		X	
Open Code		X	(X)

R: Die neuesten Tools, elegante Diagramme



Big Data?



Anatomie der Datenanalyse


Mit einer handvoll Verben lassen sich die meisten Aufgaben der Datenanalyse erfassen:

- Zeilen filtern -- **filter**
- Spalten wählen -- **select**
- Sortieren -- **arrange**
- Zusammenfassen -- **summarise**
- Verändern -- **mutate**
- Gruppieren -- **group_by**

Cheatsheet


Data Wrangling with dplyr and tidy

Cheat Sheet




Tidy Data - A foundation for wrangling in R

In a tidy dataset:




Each **variable** is saved in its own **column**

&



Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



$M * A = P$

Syntax - Helpful conventions for wrangling

dplyr::tbl_df(iris)

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen.

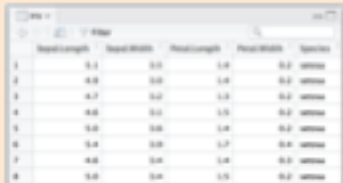
```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length
1           5.1           3.5           1.4
2           4.9           3.0           1.4
3           4.7           3.2           1.3
4           4.6           3.1           1.5
5           5.0           3.0           1.4
#> # A tibble: 150 x 5
#>   Sepal.Length Sepal.Width Petal.Length
#>   <dbl>         <dbl>         <dbl>
```

dplyr::glimpse(iris)


Information dense summary of tbl data.

utils::View(iris)

View data set in spreadsheet-like display (note capital V).




Reshaping Data - Change the layout of a data set




tidyr::gather(cases, "year", "n", 2:4)

Gather columns into rows.




tidyr::spread(pollution, size, amount)

Spread rows into columns.



tidyr::separate(storms, date, c("y", "m", "d"))

Separate one column into several.



tidyr::unite(data, col, ..., sep)

Unite several columns into one.

dplyr::data_frame(a = 1:3, b = 4:6)

Combine vectors into data frame (optimized).

dplyr::arrange(mtcars, mpg)

Order rows by values of a column (low to high).


dplyr::arrange(mtcars, desc(mpg))

Order rows by values of a column (high to low).

dplyr::rename(tbl, y = year)

Rename the columns of a data frame.

Subset Observations (Rows)



dplyr::filter(iris, Sepal.Length > 7)

Extract rows that meet logical criteria.

dplyr::distinct(iris)

Remove duplicate rows.

dplyr::sample_frac(iris, 0.5, replace = TRUE)

Randomly select fraction of rows.

dplyr::sample_n(iris, 10, replace = TRUE)

Randomly select n rows.

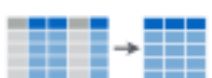
dplyr::slice(iris, 10:15)

Select rows by position.

dplyr::top_n(storms, 2, date)

Select and order top n entries (by group if grouped data).

Subset Variables (Columns)



dplyr::select(iris, Sepal.Width, Petal.Length, Species)

Select columns by name or helper function.

Helper functions for select - ?select

```
select(iris, contains("l"))
#> # A tibble: 150 x 3
#>   Sepal.Length Sepal.Width Petal.Length
#>   <dbl>         <dbl>         <dbl>
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           3.0           1.4
#> # A tibble: 150 x 3
#>   Sepal.Length Sepal.Width Petal.Length
#>   <dbl>         <dbl>         <dbl>
```

select(iris, contains("l"))

Select columns whose name contains a character string.

select(iris, ends_with("Length"))

Select columns whose name ends with a character string.

select(iris, everything())

Select every column.

select(iris, matches("l"))

Select columns whose name matches a regular expression.

select(iris, num_range("x", 1:5))

Select columns named c("x1", "x2", "x3", "x4", "x5").

select(iris, one_of("Species", "Genus"))

Select columns whose names are in a group of names.

select(iris, starts_with("Sepal"))

Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)

Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, ~Species)

Select all columns except Species.

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

<	Less than	<=	Not equal to
>	Greater than	>=	Group membership
==	Equal to	is.na	Is NA
!=	Not equal to	is.na	Is not NA
&	Logical AND	&&	Boolean operators
	Logical OR		Boolean operators
!	Logical NOT	!	Boolean operators

RStudio is a trademark of RStudio, Inc. | © 2016 RStudio | info@rstudio.com | 844-444-1222 | rstudio.com

data-cards/crystal-github/rstudio/CRANv1 for data sets

Learn more with books.greentopkage.com/c/dplyr/ | dplyr 0.4.0 | tidy 0.2.0 | Updated: 2016-08-01

Diese Software brauchen wir

- [R](#)

```
# packages müssen einmalig installiert sein, bevor Sie sie laden können
# update.packages() # zur Sicherheit auf den neuesten Stand kommen
# install.packages(c("dplyr", "ggplot2", "nycflights13"))

library(dplyr)
library(ggplot2)
library(nycflights13)
data(flights) # lädt Datensatz
?flights # Beschreibung des Datensatzes
```

- Installieren Sie [RStudio](#).
- Alternativ kann man den Datensatz `flights` auch [hier](#) herunterladen.
- Excel-Freaks: Verfolgt die Analyse parallel in Excel mit!

glimpse(flights)

```
## Observations: 336,776
## Variables: 19
## $ year      (int) 2013, 2013, 2013, 2013, 2013, 20...
## $ month     (int) 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ day       (int) 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ dep_time  (int) 517, 533, 542, 544, 554, 554, 55...
## $ sched_dep_time (int) 515, 529, 540, 545, 600, 558, 60...
## $ dep_delay (dbl) 2, 4, 2, -1, -6, -4, -5, -3, -3,...
## $ arr_time  (int) 830, 850, 923, 1004, 812, 740, 9...
## $ sched_arr_time (int) 819, 830, 850, 1022, 837, 728, 8...
## $ arr_delay (dbl) 11, 20, 33, -18, -25, 12, 19, -1...
## $ carrier   (chr) "UA", "UA", "AA", "B6", "DL", "U...
## $ flight    (int) 1545, 1714, 1141, 725, 461, 1696...
## $ tailnum   (chr) "N14228", "N24211", "N619AA", "N...
## $ origin    (chr) "EWR", "LGA", "JFK", "JFK", "LGA...
## $ dest      (chr) "IAH", "IAH", "MIA", "BQN", "ATL...
## $ air_time  (dbl) 227, 227, 160, 183, 116, 150, 15...
## $ distance  (dbl) 1400, 1416, 1089, 1576, 762, 719...
## $ hour      (dbl) 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6,...
## $ minute    (dbl) 15, 29, 40, 45, 0, 58, 0, 0, 0, ...
## $ time_hour (time) 2013-01-01 05:00:00, 2013-01-01...
```

Der Datensatz **mtcars**

- Ein Datensatz zu technischen Merkmalen von Autos aus der US-Zeitschrift *Motor Trends*.
- Wir benutzen als "Spielzeug-Datensatz" (XXS-Data)
- Der Datensatz ist in R schon enthalten.

Sehen Sie sich die Hilfe zu dem Datensatz an:

```
?mtcars
```

Zeilen filtern mit filter()

Auszug aus `mtcars`:

	mpg	cyl	hp	wt
Mazda RX4	21.0	6	110	2.620
Mazda RX4 Wag	21.0	6	110	2.875
Datsun 710	22.8	4	93	2.320
Hornet 4 Drive	21.4	6	110	3.215
Hornet Sportabout	18.7	8	175	3.440
Valiant	18.1	6	105	3.460
Duster 360	14.3	8	245	3.570
Merc 240D	24.4	4	62	3.190
Merc 230	22.8	4	95	3.150
Merc 280	19.2	6	123	3.440

Spalten gefiltert mit `cyl == 8`:

mpg	cyl	hp	wt
18.7	8	175	3.44
14.3	8	245	3.57

Beispiele für filter()

Entschlüsseln Sie diese Filter (Datensatz `flights`):

```
filter(mtcars, hp > 100)
filter(mtcars, cyl %in% c(4, 6))
filter(mtcars, gear == 3 | gear == 4)
filter(mtcars, hp > 300 & cyl == 8)
```

Übung zu filter()

Identifizieren Sie folgende Flüge:

1. von JFK nach PWM (Portland)
2. von JFK nach PWM (Portland) im Januar
3. von JFK nach PWM (Portland) im Januar mit mehr als einer Stunde Verspätung
4. von JFK nach PWM (Portland) im Januar zwischen Mitternacht und 5 Uhr
5. von JFK deren Ankunftsverspätung doppelt so groß war wie die Abflugverspätung, und die nach Atlanta geflogen sind

Lösungsideen

```
filter(flights, origin == "JFK")
```

```
filter(flights, origin == "JFK" & month == 1)
```

```
filter(flights, origin == "JFK" & month == 1 & dep_time < 500 & dest == "PWM" )
```

```
filter(flights, origin == "JFK" & month == 1 & dep_time > 500 & dest == "PWM" )
```

```
filter(flights, origin == "JFK" & arr_delay > 2 * dep_delay & month == 1, dest == "ATL")
```


Spalten wählen mit select()

```
select(mtcars, mpg, cyl, hp)
```

Auszug aus `mtcars`:

	mpg	cyl	hp	disp	wt	qsec
Mazda RX4	21.0	6	110	160	2.620	16.46
Mazda RX4 Wag	21.0	6	110	160	2.875	17.02
Datsun 710	22.8	4	93	108	2.320	18.61
Hornet 4 Drive	21.4	6	110	258	3.215	19.44
Hornet Sportabout	18.7	8	175	360	3.440	17.02
Valiant	18.1	6	105	225	3.460	20.22

Spalten ausgewählt mit `select(mtcars, mpg, cyl, hp)`:

	mpg	cyl	hp
Mazda RX4	21.0	6	110
Mazda RX4 Wag	21.0	6	110
Datsun 710	22.8	4	93
Hornet 4 Drive	21.4	6	110
Hornet Sportabout	18.7	8	175
Valiant	18.1	6	105

Übung zu select

- Lesen Sie die Hilfe zu `select()`. Auf welche Arten kann man noch Spalten (Variablen) auswählen?
- Schreiben Sie 3 Arten auf, um die Spalten mit den Verzögerungen auszuwählen.
- Stellen Sie sich vor, Sie haben einen Datensatz mit 1000 Spalten: V1 .. V1000. Inwiefern ist das mit Excel noch praktikabel?

Lösungsideen

```
select(flights, arr_delay, dep_delay)
```

```
select(flights, arr_delay:dep_delay)
```

```
select(flights, contains("delay"))
```

```
select(flights, ends_with("delay"))
```

```
select(flights, c(6, 9))
```

```
auswahl <- c("dep_delay", "arr_delay")
```

```
select(flights, one_of(auswahl))
```

Zeilen sortieren mit arrange()

`arrange(mtcars, cyl)`

Auszug aus `mtcars`:

	mpg	cyl	hp	disp	wt	qsec
Mazda RX4	21.0	6	110	160	2.620	16.46
Mazda RX4	21.0	6	110	160	2.875	17.02
Wag						
Datsun 710	22.8	4	93	108	2.320	18.61
Hornet 4 Drive	21.4	6	110	258	3.215	19.44
Hornet Sportabout	18.7	8	175	360	3.440	17.02
Valiant	18.1	6	105	225	3.460	20.22

Zeilen *aufsteigend* sortiert nach `cyl` und nach `hp`:

mpg	cyl	hp
22.8	4	93
18.1	6	105
21.0	6	110
21.0	6	110
21.4	6	110
18.7	8	175

Zeilen absteigend sortieren mit arrange(desc())

`select(mtcars, arrange(desc(cyl)))` ("descending": engl. für absteigend)

Auszug aus `mtcars`:

	mpg	cyl	hp	disp	wt	qsec
Mazda RX4	21.0	6	110	160	2.620	16.46
Mazda RX4	21.0	6	110	160	2.875	17.02
Wag						
Datsun 710	22.8	4	93	108	2.320	18.61
Hornet 4 Drive	21.4	6	110	258	3.215	19.44
Hornet Sportabout	18.7	8	175	360	3.440	17.02
Valiant	18.1	6	105	225	3.460	20.22

Zeilen **absteigend** sortiert nach `cyl`:

mpg	cyl	hp
18.7	8	175
21.0	6	110
21.0	6	110
21.4	6	110
18.1	6	105
22.8	4	93

Übung zu arrange()

1. Ordnen Sie die Flüge nach Datum und Uhrzeit.
2. Welche Flüge hatten die größte Verspätung?
3. Welche Flüge holten die meiste Verspätung während des Fluges auf?
4. Welche Airlines hatten die größte Verspätung? Hm.

Lösungsideen

```
arrange(flights, month, day, sched_dep_time)

flights2 <- select(flights, dep_delay, arr_delay, tailnum, flight, dest)

arrange(flights2, desc(dep_delay))

arrange(flights2, desc(dep_delay - arr_delay))
```

Variablen (und ihre Werte) verändern

```
mutate(flights, wt_kg = wt / 1000 * 2, wt_per_ps = wt_kg / hp)
```

Auszug aus `mtcars`:

	mpg	cyl	hp	disp	wt	qsec
Mazda RX4	21.0	6	110	160	2.620	16.46
Mazda RX4	21.0	6	110	160	2.875	17.02
Wag						
Datsun 710	22.8	4	93	108	2.320	18.61
Hornet 4 Drive	21.4	6	110	258	3.215	19.44
Hornet Sportabout	18.7	8	175	360	3.440	17.02
Valiant	18.1	6	105	225	3.460	20.22

Neue Spalte `wt_per_ps`: Gewicht (`wt`) pro PS (`hp`):

wt	mpg	cyl	hp	wt_kg	wt_per_ps
2.620	21.0	6	110	1310.0	11.909091
2.875	21.0	6	110	1437.5	13.068182
2.320	22.8	4	93	1160.0	12.473118
3.215	21.4	6	110	1607.5	14.613636
3.440	18.7	8	175	1720.0	9.828571
3.460	18.1	6	105	1730.0	16.476191

Übung zu mutate()

1. Berechnen Sie die Geschwindigkeit (mph) jedes Fluges. Welche Flüge flogen am schnellsten?
2. Erzeugen Sie eine neue Variable, die angibt, wieviel Zeit ein Flug verloren oder aufgeholt hat.
3. Berechnen Sie die Flugdistanz in km.

Lösungsideen

```
mutate(flights, speed = distance / air_time)
arrange(flights, speed)
mutate(flights, delay = dep_delay - arr_delay)
mutate(flights, dist_km = distance / 1.6)
```

Zusammenfassen mit summarise()

```
summarise(flights, hp_mean = mean(hp))
```

	mpg	cyl	hp	disp
Mazda RX4	21.0	6	110	160
Mazda RX4 Wag	21.0	6	110	160
Datsun 710	22.8	4	93	108
Hornet 4 Drive	21.4	6	110	258
Hornet Sportabout	18.7	8	175	360
Valiant	18.1	6	105	225

Zusammenfassung der Spalte `hp` in einen einzigen Wert (Mittelwert):

hp_mean
117.1667

Gruppieren plus zusammenfassen mit summarise()

```
mtcars_by_cyl = group_by(mtcars, cyl) summarise(mtcars_by_cyl, p_cyl_mean = mean(hp, na.rm = TRUE))
```

Gruppieren nach `cyl` (und in einem data.frame ausgeben):

Zusammenfassen der Spalte `hp` bei jeder Gruppe in einen einzigen Wert:

	mpg	cyl	hp	disp
Mazda RX4	21.0	6	110	160
Mazda RX4 Wag	21.0	6	110	160
Datsun 710	22.8	4	93	108
Hornet 4 Drive	21.4	6	110	258
Hornet Sportabout	18.7	8	175	360
Valiant	18.1	6	105	225

cyl	hp_cyl_mean
4	93.00
6	108.75
8	175.00

Funktionen zum Zusammenfassen

- `min()`, `max()`, `median()`, `quantile()`
- `mean()`, `sd()`, `sum()`
- `n()`, `n_distinct()`
- Jede Funktion, die eine *Spalte* als *Input* nimmt und einen *einzelnen Wert* ausgibt

Übung zu `summarise()` nach `group_by()`

1. Berechnen Sie die mittlere Verspätung pro Flughafen!
2. Ermitteln Sie pro Monat den Flug mit der größten Verspätung!
3. Geben Sie die Airlines mit der geringsten mittleren Verspätung an!

Lösungsideen

1. Berechnen Sie die mittlere Verspätung pro Flughafen!

```
f2 <- group_by(flights, origin)
f3 <- mutate(f2, delay = dep_delay - arr_delay)
summarise(f3, delay_mean = mean(delay, na.rm = TRUE))
```

2. Ermitteln Sie pro Monat den Flug mit der größten Verspätung!

```
f2 <- group_by(flights, month)
f3 <- mutate(f2, delay = dep_delay - arr_delay)
summarise(f3, delay_max = max(delay, na.rm = T))
```

3. Geben Sie die Airlines mit der geringsten mittleren Verspätung an!

```
f2 <- group_by(flights, carrier)
f3 <- mutate(f2, delay = dep_delay - arr_delay)
f4 <- filter(f3, !is.na(delay))
f5 <- summarise(f4, delay_min = mean(delay))
arrange(f5, delay_min)
```

Verschachtelte Syntax ist schwer zu lesen

```
hourly_delay <- filter(  
  summarise(  
    group_by(  
      filter(  
flights,  
        !is.na(dep_delay)  
      ),  
date, hour ),  
    delay = mean(dep_delay),  
n = n() ),  
n > 10 )
```

Die Pfeife %>%

- [Das ist keine Pfeife](#) {magrittr}
- `x %>% f(y)` ist dasselbe wie `f(x, y)`

```
hourly_delay <- flights %>%  
  filter(!is.na(dep_delay)) %>%  
  group_by(date, hour) %>%  
  summarise(delay = mean(dep_delay), n = n()) %>%  
  filter(n > 10)
```

- Tipp: %>% kann man lesen als "*und dann*"

Übung zur Pfeife

1. Was sind die oberen 10% der Airlines bei der Verspätung?
2. Berechnen Sie die mittlere Verspätung aller Flüge mit deutlicher Verspätung (> 1 Stunde)!

Lösungsideen

Was sind die oberen 10% der Airlines bei der Verspätung?

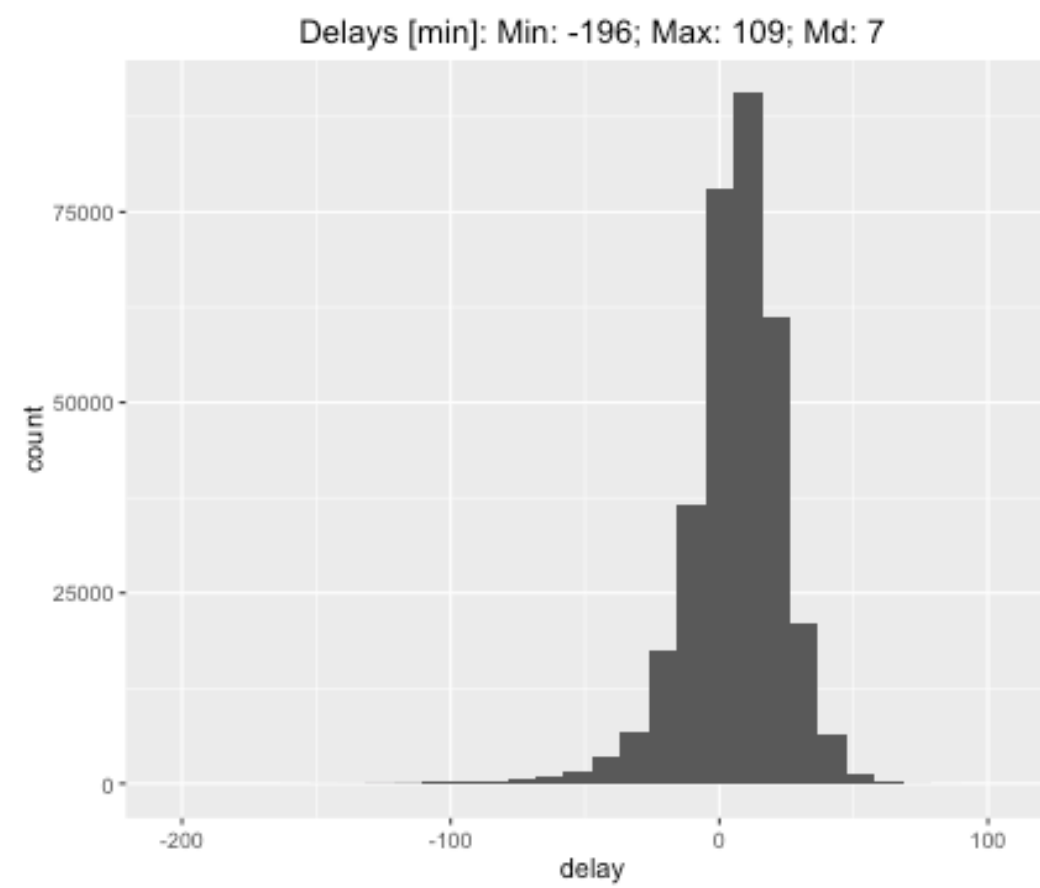
```
flights %>%
  group_by(carrier) %>% na.omit() %>%
  mutate(delay = dep_delay - arr_delay) %>%
  summarise(delay_mean = mean(delay, na.rm = TRUE)) %>%
  filter(delay_mean < quantile(delay_mean, .1)) %>%
  # oder: filter(ntile(delay_mean, 10) == 1) %>%
  arrange(delay_mean)
```

Berechnen Sie die mittlere Verspätung aller Flüge mit deutlicher Verspätung (> 1 Stunde)!

```
flights %>% na.omit() %>% mutate(delay = dep_delay - arr_delay) %>%
  filter(delay > 60) %>%
  summarise(delay_mean = mean(delay),
            n = n()) %>% # Anzahl
  arrange(delay_mean)
```

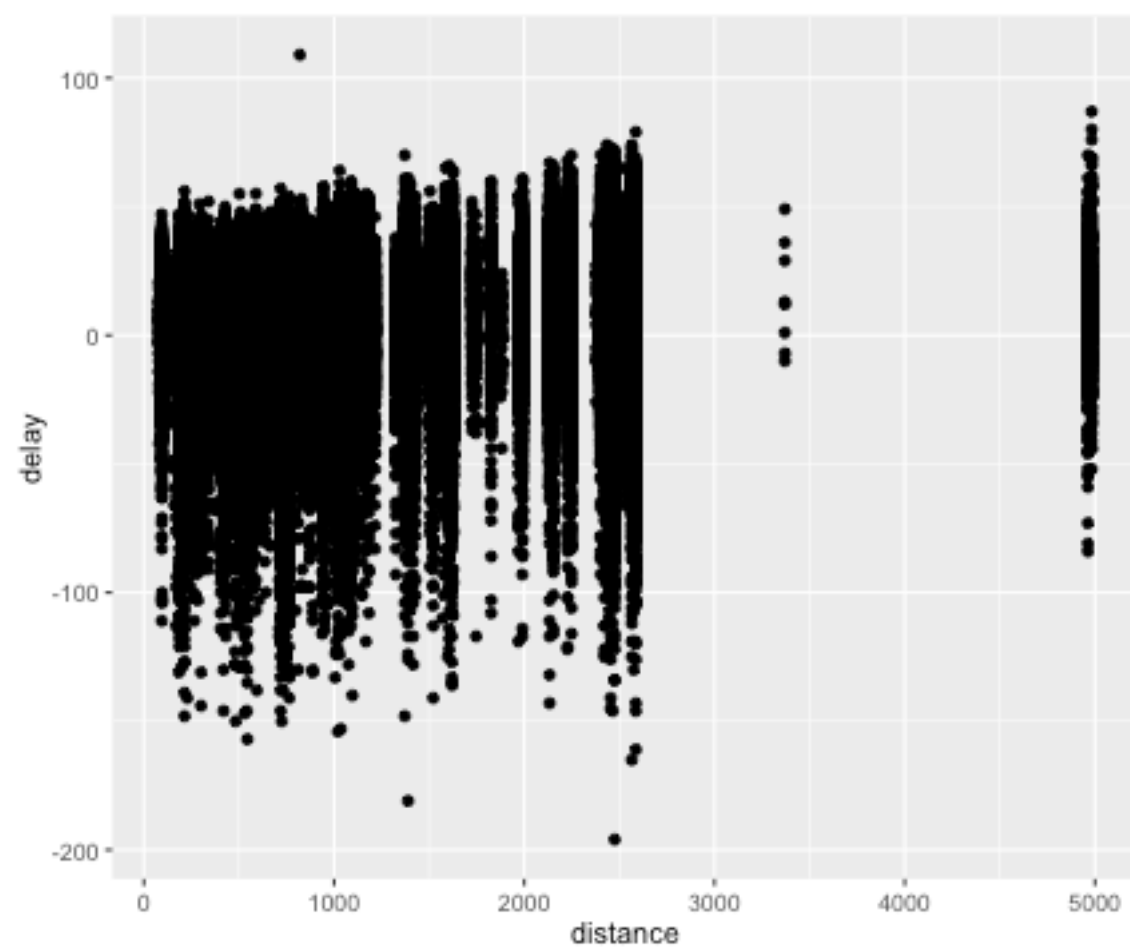
Wie sind die Verspätungen verteilt?

```
f2 <- flights %>%  
  na.omit() %>% mutate(delay = dep_delay - arr_delay)  
  
qplot(data = f2, x = delay,  
      main = paste("Delays [min]: Min: ", min(f2$delay),  
                    "; Max: ", max(f2$delay),  
                    "; Md: ", median(f2$delay), sep = " "))
```



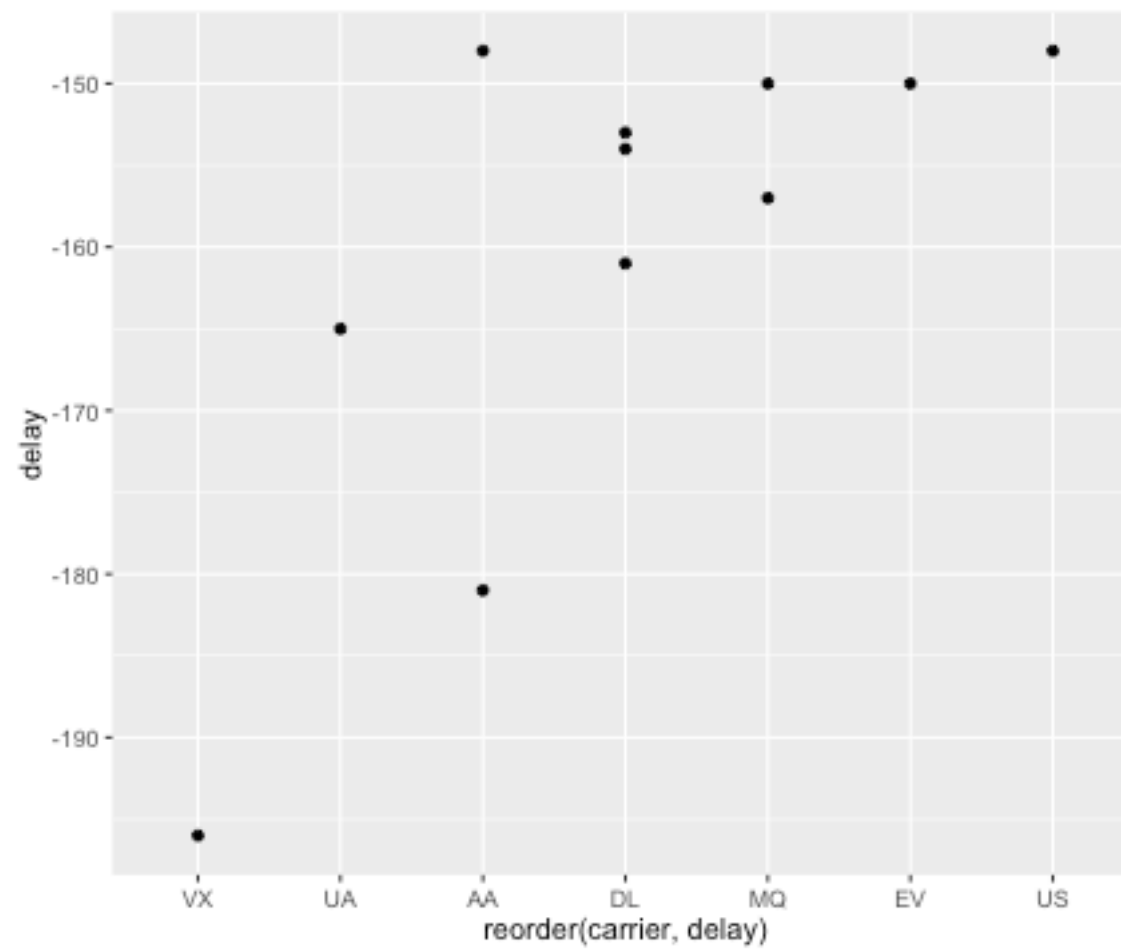
Hängen Flugzeit und Verspätung zusammen?

```
flights %>%  
  mutate(delay = dep_delay - arr_delay) %>%  
  na.omit() %>% ggplot(x = distance, y = delay, data = .)
```



Was ist die Top-10 der lahmen Airlines?

```
flights %>%
  group_by(carrier) %>% na.omit() %>% mutate(delay = dep_delay - arr_delay) %>%
  ungroup() %>% filter(min_rank(delay) < 11) %>%
  arrange(delay) %>% qplot(data = ., x = reorder(carrier, delay), y = delay,
    geom = "point")
```



Danke/Referenzen

- Danke an Hadley Wickham für `dplyr`, `ggplot2` und das Hadleyverse
- Dieser Kurs basiert auf [diesem](#) Tutorial von Hadley Wickham
- Kontakt: Sebastian Sauer, sebastian.sauer-AT-fom.de
- Die Folien (inkl. Syntax) findet sich [hier](#)