

Die Pakete ggplot und ggmap

Jan-Philipp Kolb

20 Juni 2017

Das Paket ggplot2

- Entwickelt von Hadley Wickham
- Viele Informationen unter:
- <http://ggplot2.org/>
- Den Graphiken liegt eine eigene Grammatik zu Grunde

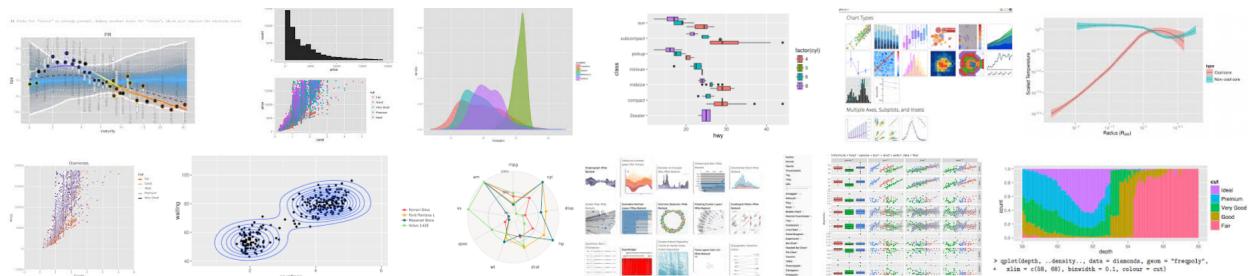


Figure 1:

Basiseinführung ggplot2

<www.r-bloggers.com/basic-introduction-to-ggplot2/>

```
install.packages("ggplot2")
```

```
library(ggplot2)
```

Der diamonds Datensatz

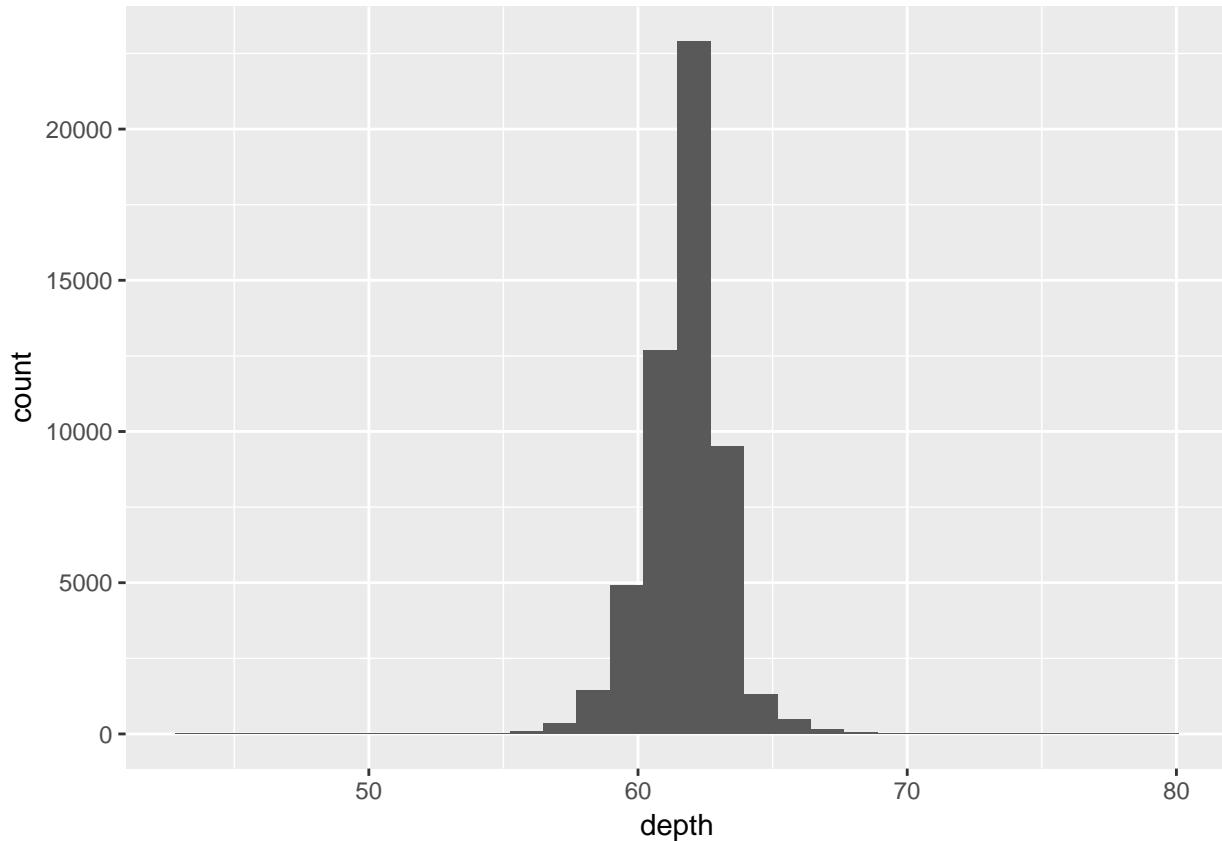
```
head(diamonds)
```

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

Wie nutzt man qplot

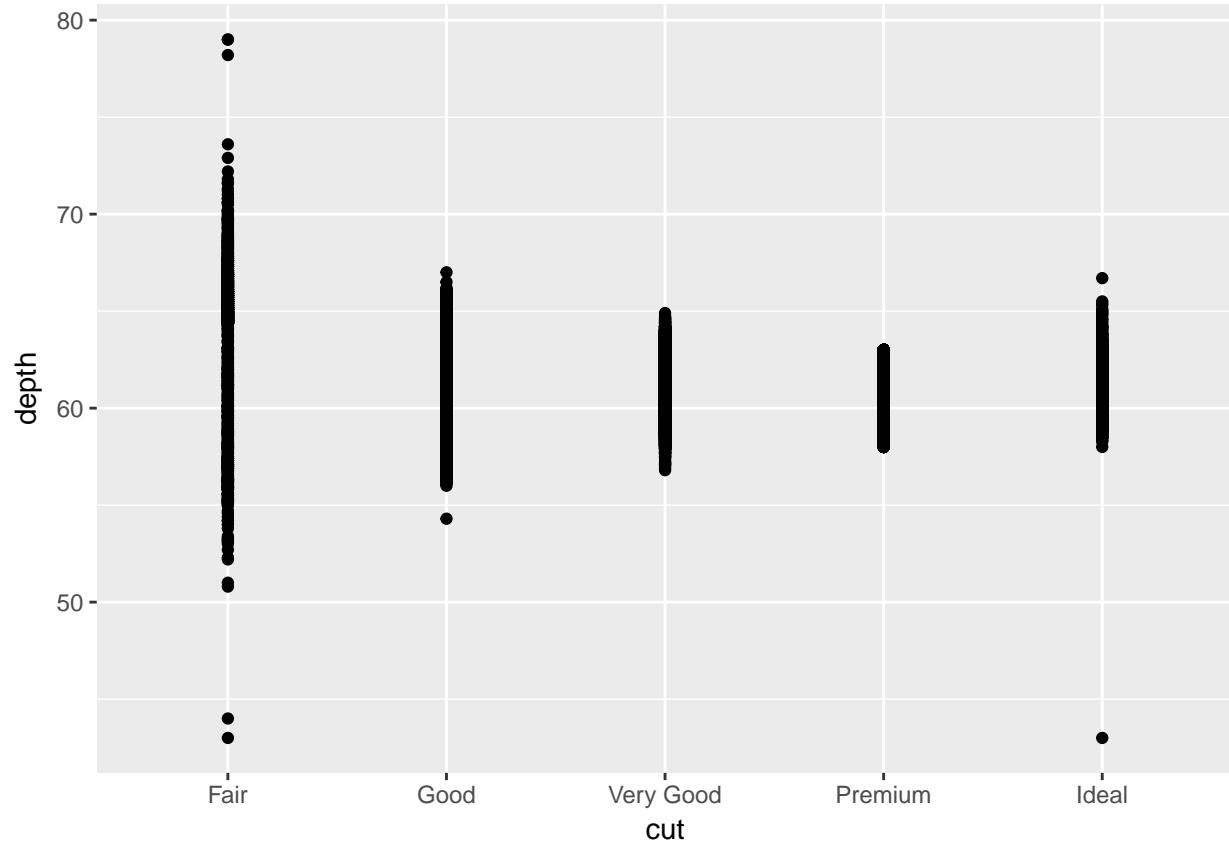
- Die Funktion `qplot` wird für schnelle Graphiken verwendet (quick plots)
- bei der Funktion `ggplot` kann man alles bis ins Detail kontrollieren

```
# histogram  
qplot(depth, data=diamonds)
```



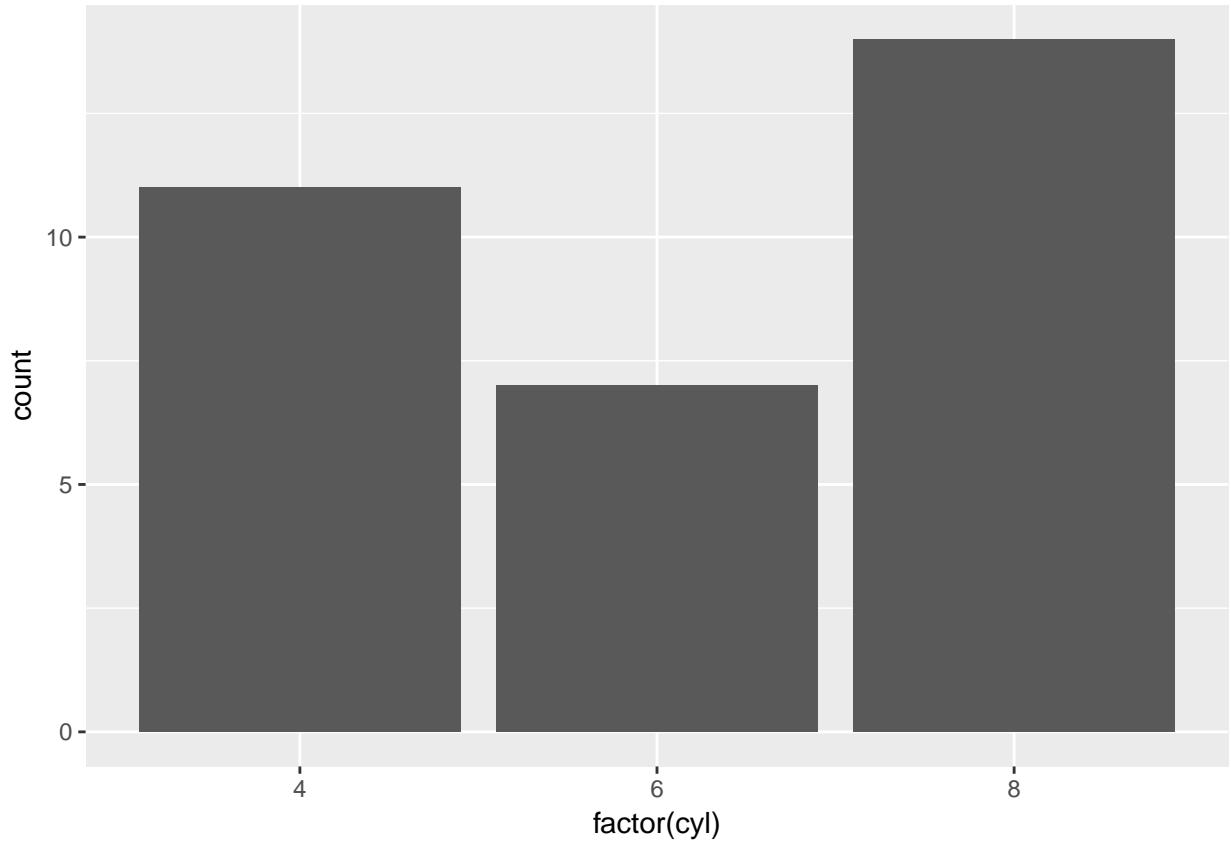
Ein Balkendiagramm

```
qplot(cut, depth, data=diamonds)
```



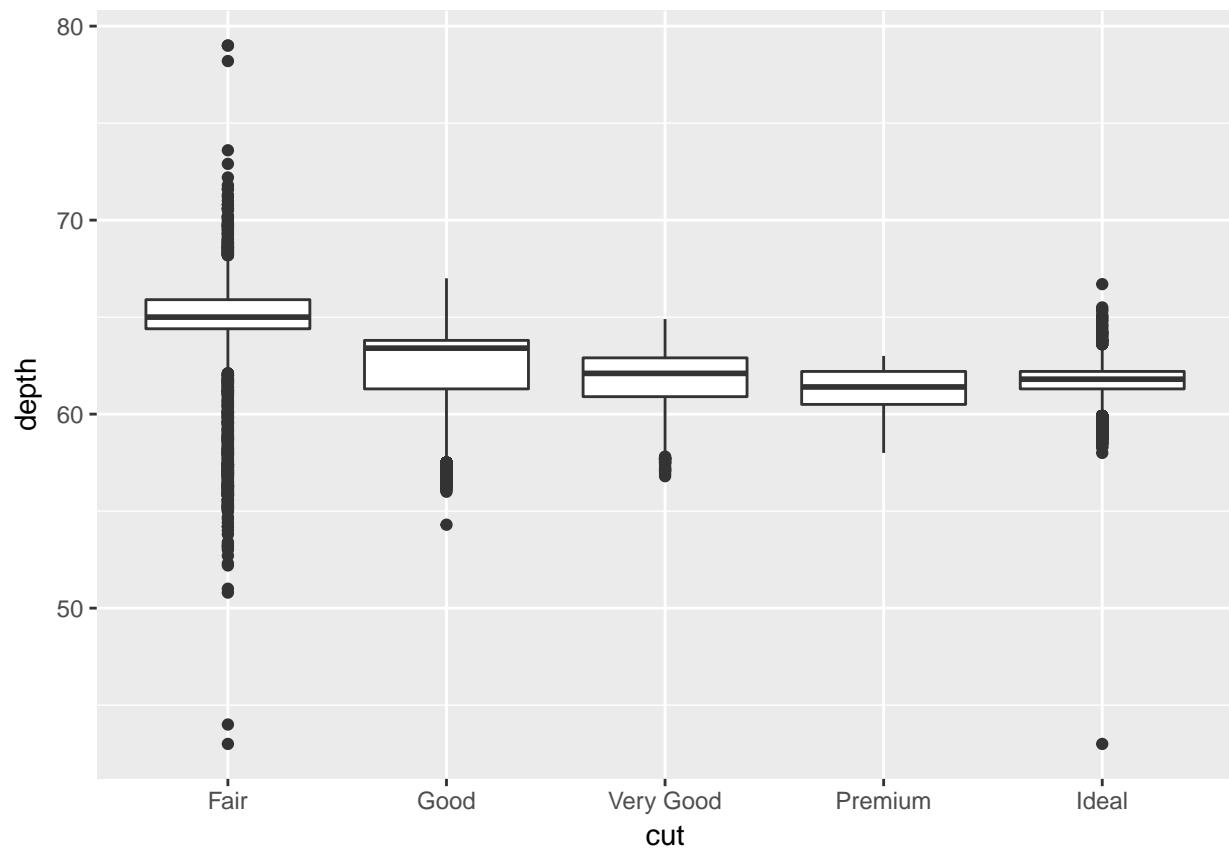
Ein weiteres Balkendiagramm

```
qplot(factor(cyl), data=mtcars,geom="bar")
```



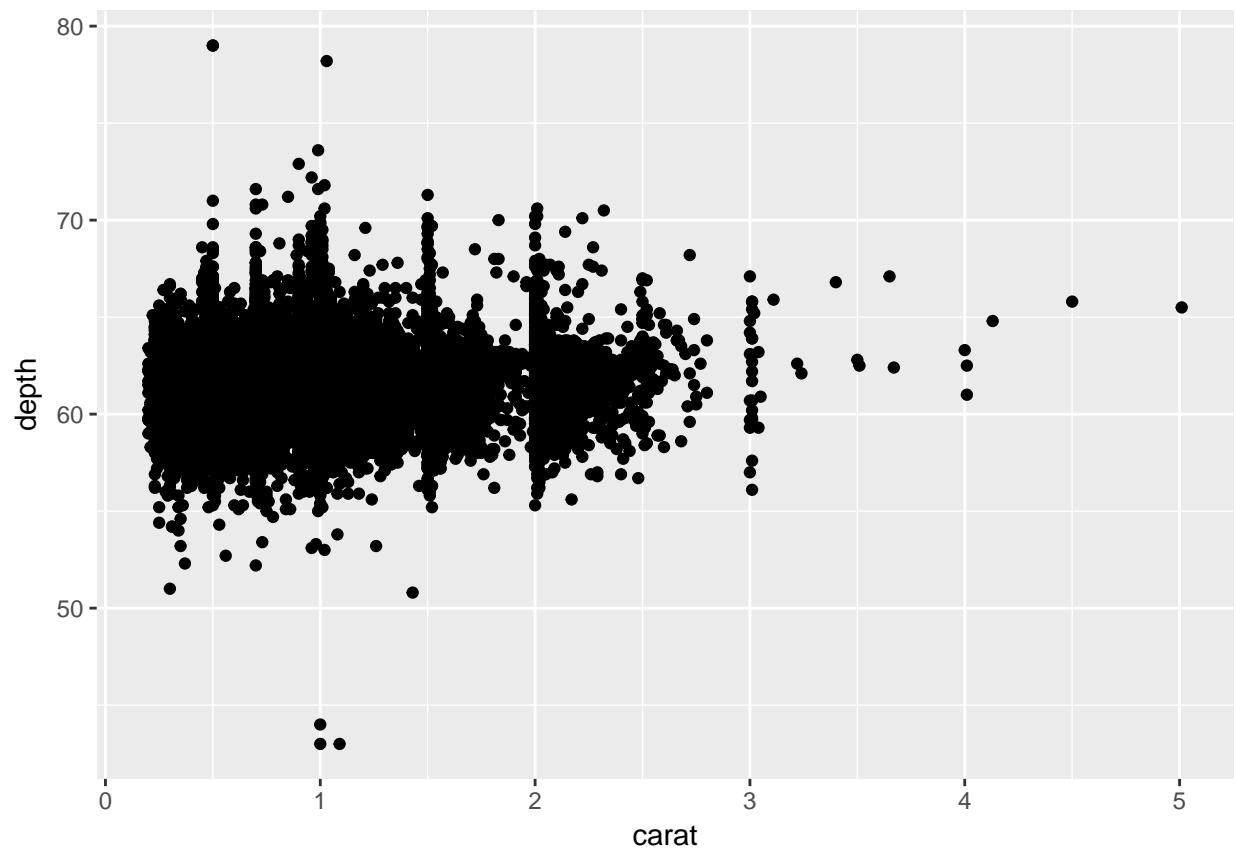
Boxplot

```
qplot(data=diamonds,x=cut,y=depth,geom="boxplot")
```



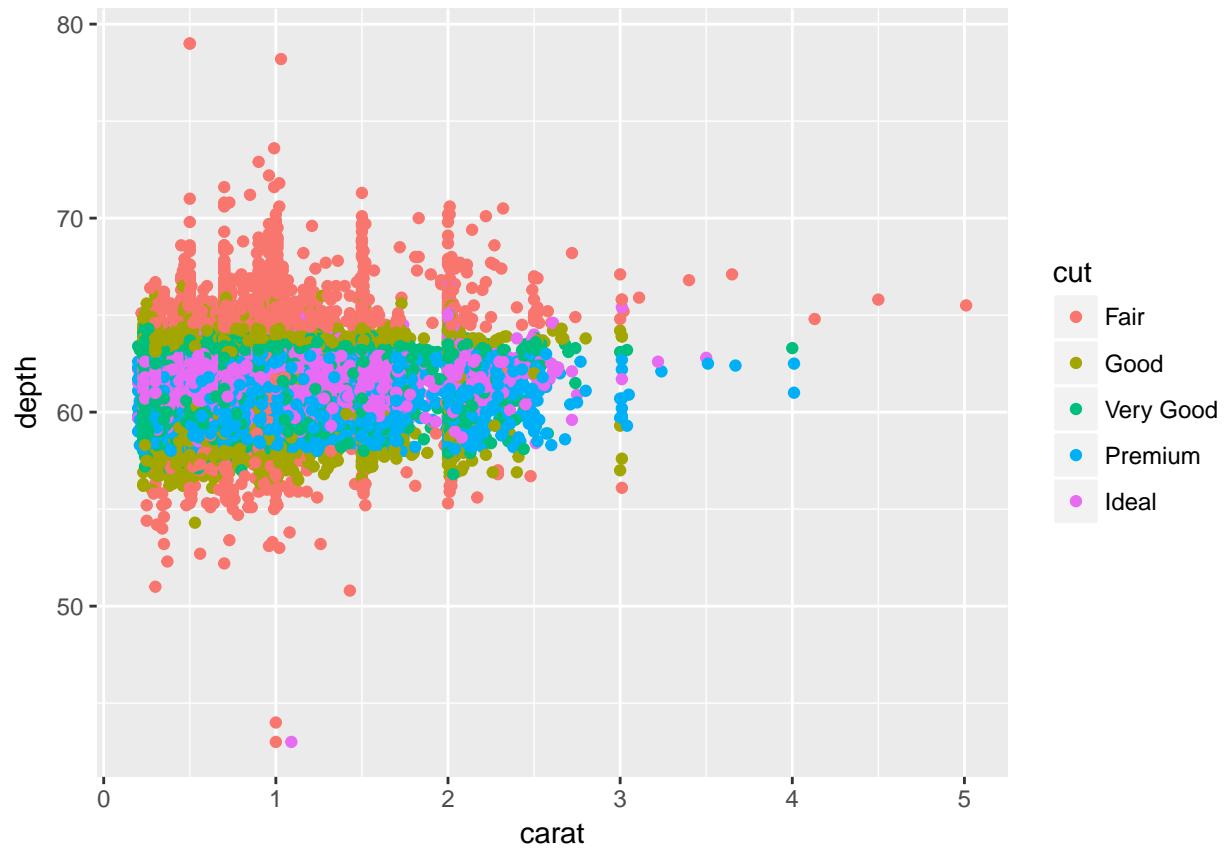
Scatterplot

```
# scatterplot
qplot(carat, depth, data=diamonds)
```



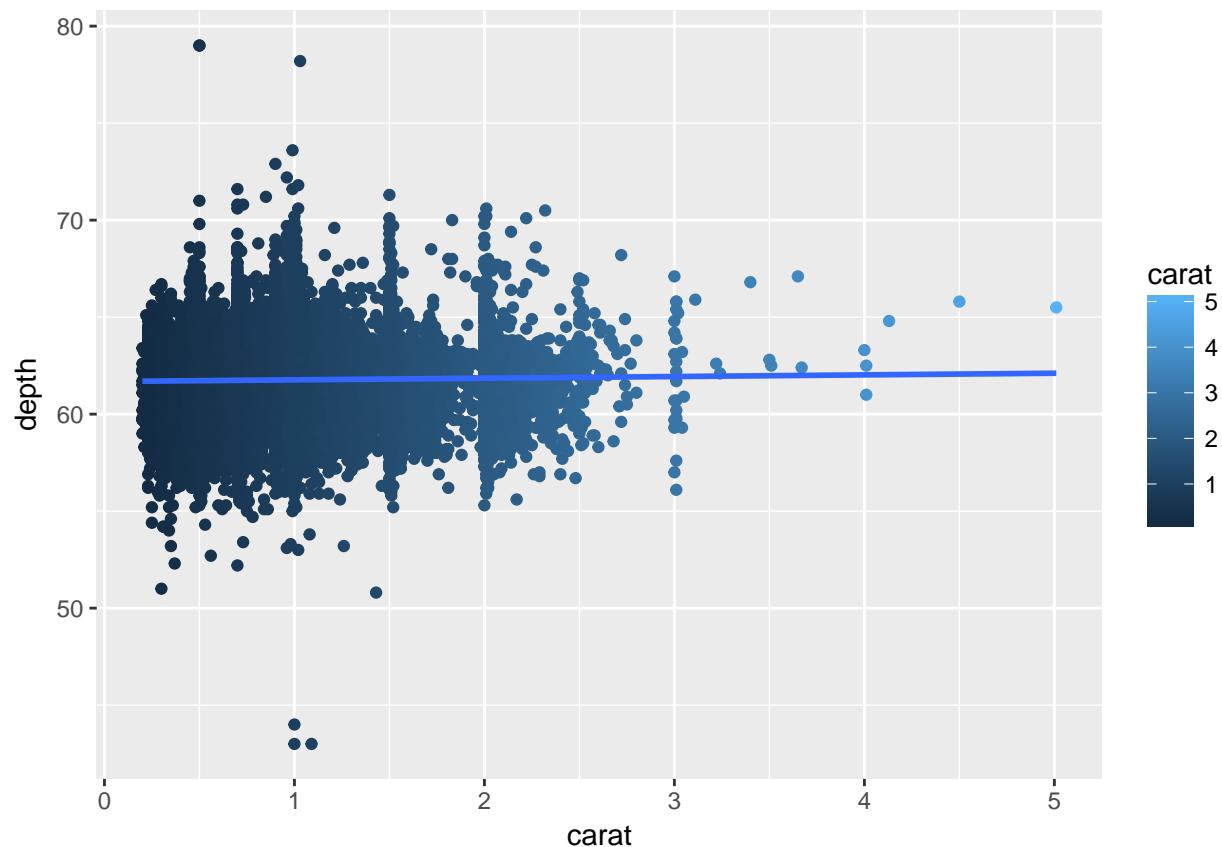
Farbe hinzufügen:

```
qplot(carat, depth, data=diamonds, color=cut)
```



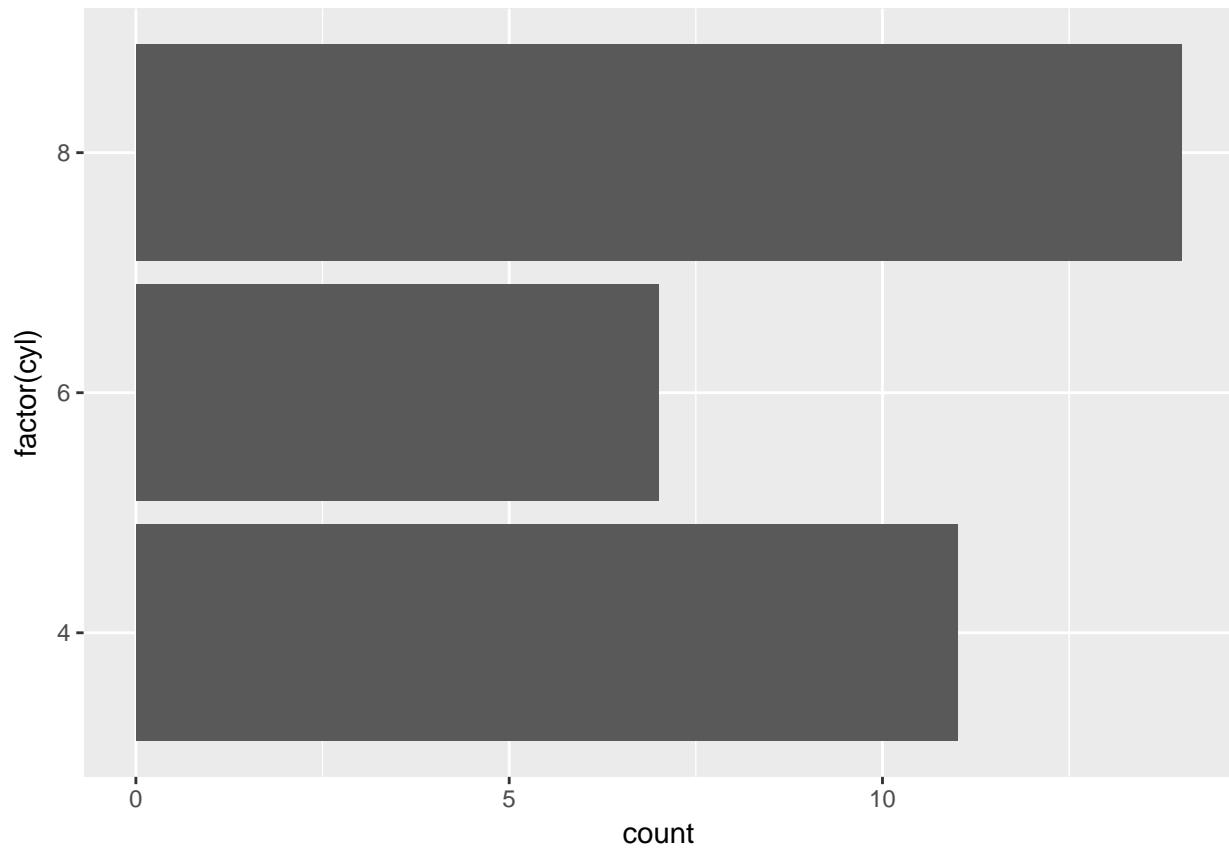
Trendlinie hinzufügen

```
myGG<-qplot(data=diamonds,x=carat,y=depth,color=cut)
myGG + stat_smooth(method="lm")
```



Graphik drehen

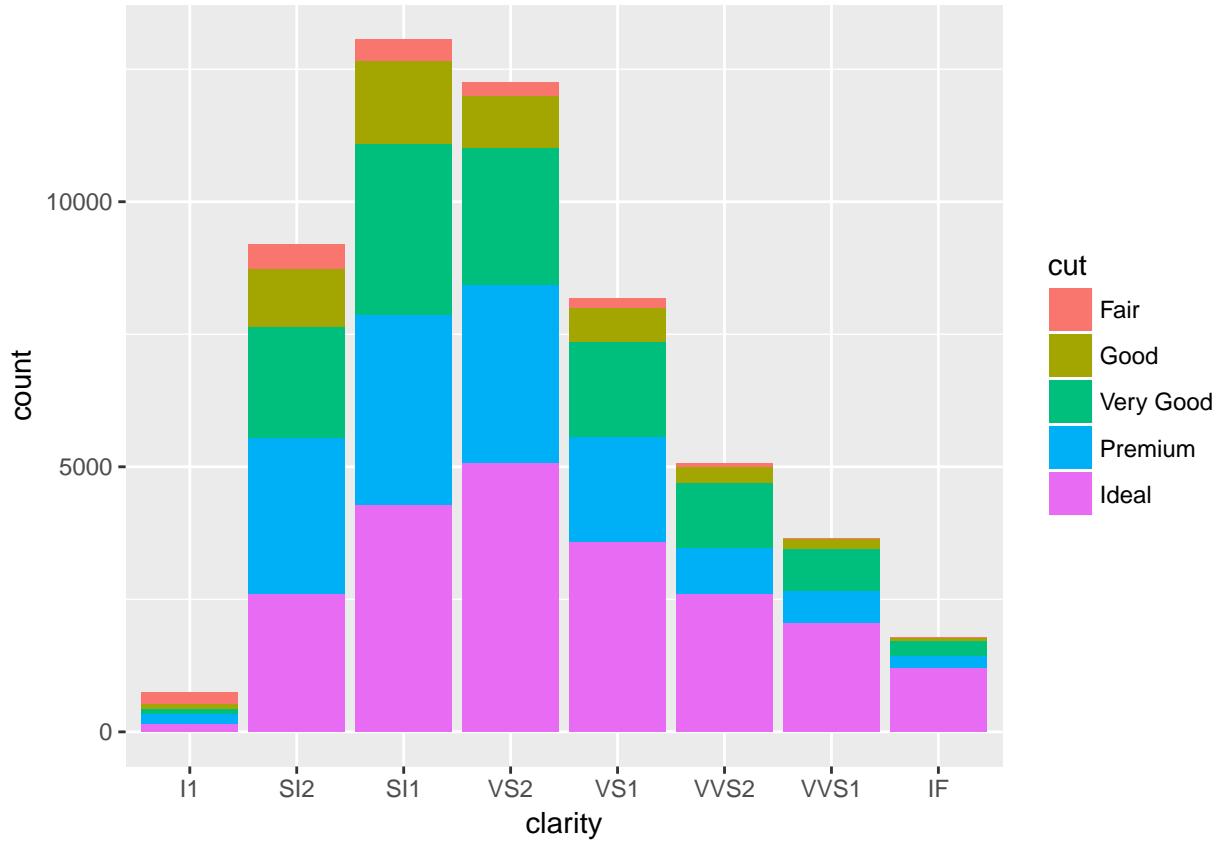
```
qplot(factor(cyl), data=mtcars, geom="bar") +  
coord_flip()
```



Wie nutzt man ggplot

- die aesthetics:

```
ggplot(diamonds, aes(clarity, fill=cut)) + geom_bar()
```



Farben selber wählen

Es wird das Paket `RColorBrewer` verwendet um die Farbpalette zu ändern

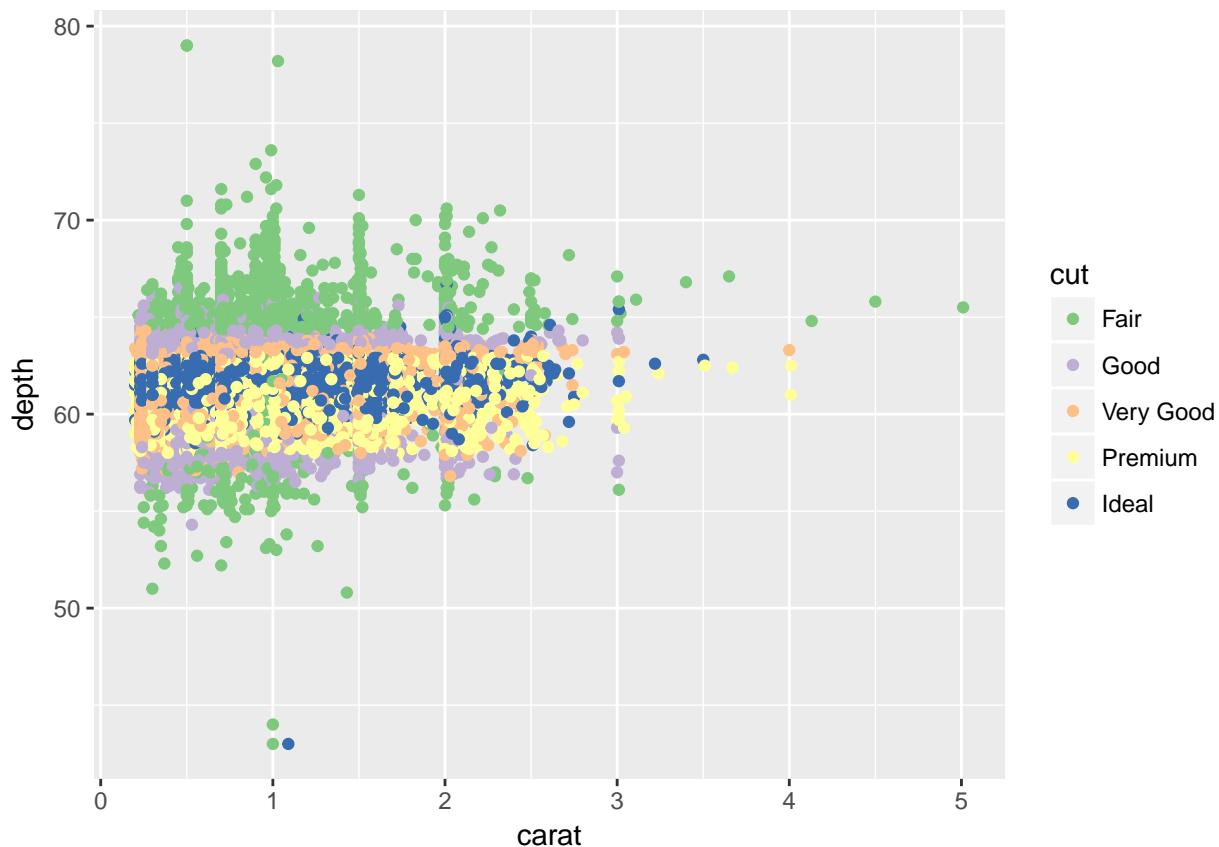
```
install.packages("RColorBrewer")

library(RColorBrewer)
myColors <- brewer.pal(5, "Accent")
names(myColors) <- levels(diamonds$cut)
colScale <- scale_colour_manual(name = "cut",
                                  values = myColors)
```

<http://stackoverflow.com/questions/6919025/>

Eine Graphik mit den gewählten Farben

```
p <- ggplot(diamonds, aes(carat, depth, colour = cut)) +
  geom_point()
p + colScale
```



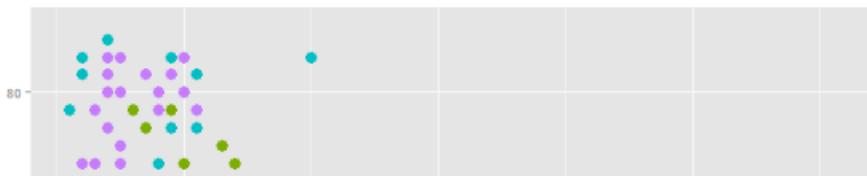
Speichern mit ggsave

```
ggsave("Graphik.jpg")
```

Links

- Warum man ggplot2 für einfache Grafiken nutzen sollte
- Einführung in ggplot2

Introduction to ggplot2



- ggplot2 Basics

- Noam Ross - Quick Introduction to ggplot2
- Plugin ggplot2

Why I use ggplot2

February 12, 2016

By David Robinson

 Like 585

 Share

 Share

69

(This article was first published on [Variance Explained](#), and kindly contributed to [R-bloggers](#))

590
SHARES

 Share

 Tweet

Figure 2:

Gliederung

Arten von räumlichen Daten:

- Straßenkarten
- Satelliten Bilder
- Physische Daten und Karten
- Abstrakte Karten
- ...

Das R-paket ggmap wird im folgenden genutzt um verschiedene Kartentypen darzustellen.

Mit qmap kann man eine schnelle Karte erzeugen.

Straßenkarten

- Straßenkarte werden sehr häufig verwendet.
- Diese Karten zeigen Haupt- und Nebenstraßen (abhängig vom Detail)
- oft sind auch weitere Informationen enthalten. Wie beispielsweise Flughäfen, Städte, Campingplätze oder andere Orte von Interesse
- Beispiel einer Straßenkarte für Mannheim.

Installieren des Paketes

- Zur Erstellung der Karten brauchen wir das Paket ggmap:

```
devtools::install_github("dkahle/ggmap")
devtools::install_github("hadley/ggplot2")
install.packages("ggmap")
```

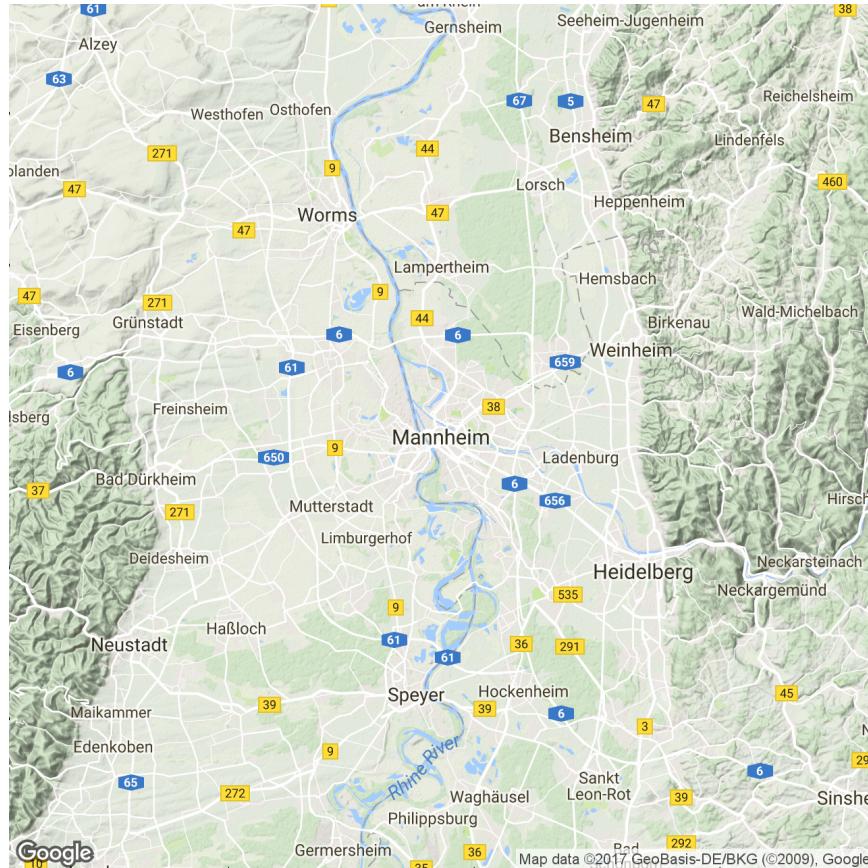
Paket ggmap - Hallo Welt

- Um das Paket zu laden verwenden wir den Befehl `library`

```
library(ggmap)
```

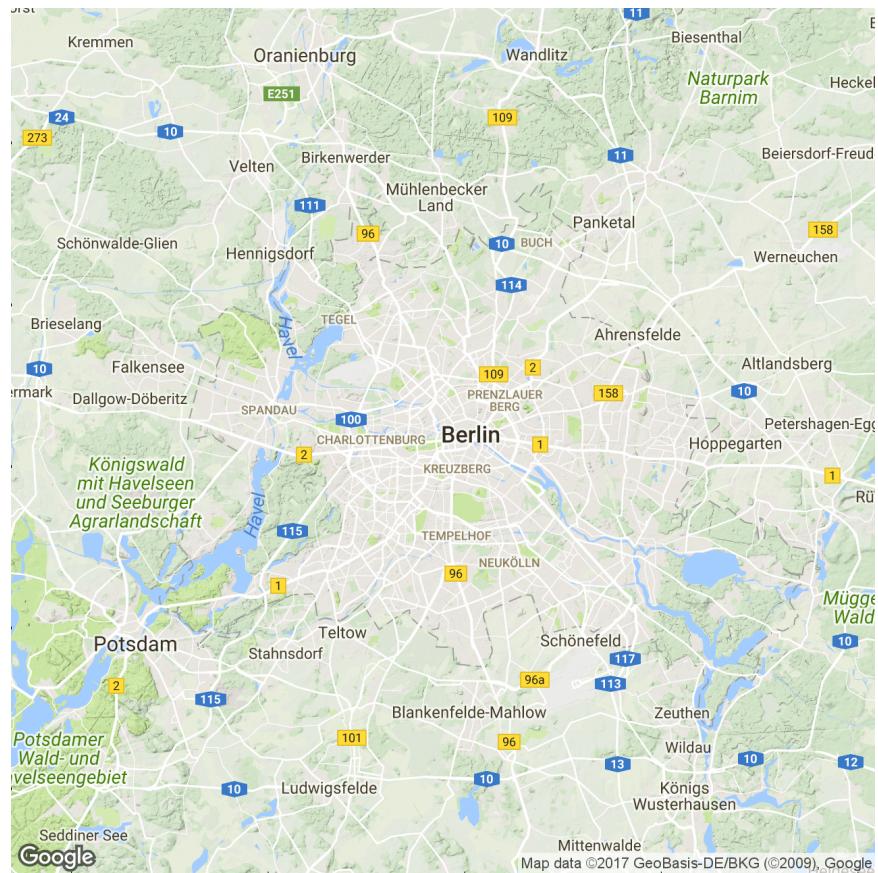
Und schon kann die erste Karte erstellt werden:

```
qmap("Mannheim")
```



Karte für eine Sehenswürdigkeit

```
BBT <- qmap("Berlin Brandenburger Tor")
BBT
```



Karte für einen ganzen Staat

```
qmap("Germany")
```

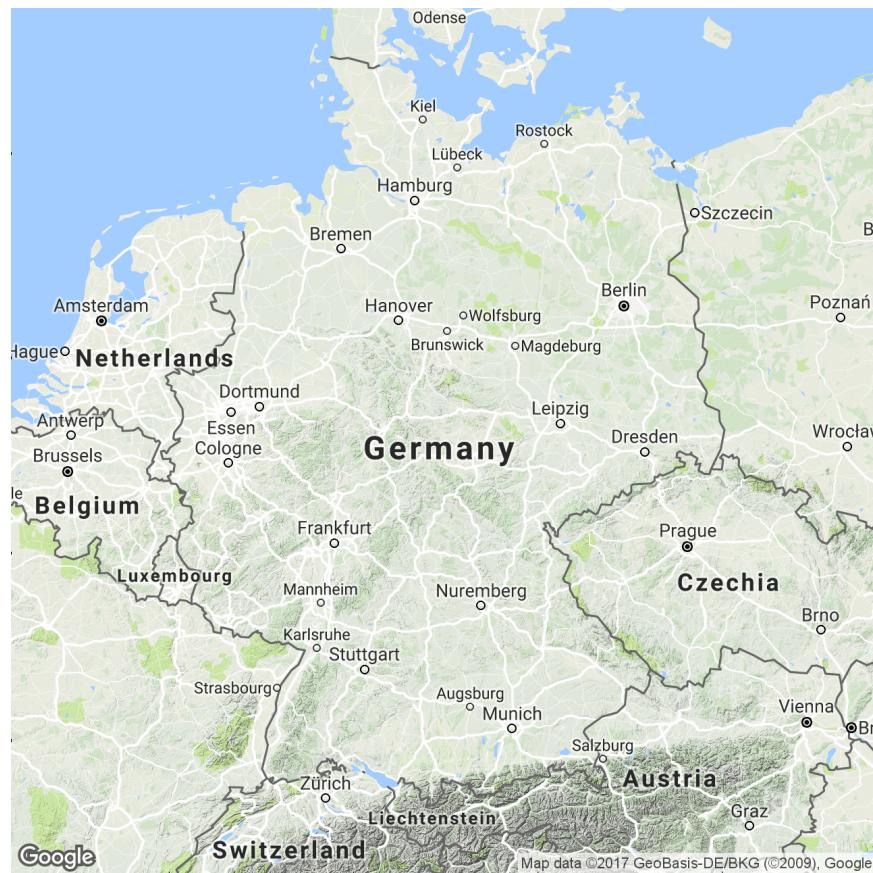


- Wir brauchen ein anderes *zoom level*

Ein anderes *zoom level*

- level 3 - Kontinent
- level 10 - Stadt
- level 21 - Gebäude

```
qmap("Germany", zoom = 6)
```



Hilfe bekommen wir mit dem Fragezeichen

```
?qmap
```

Verschiedene Abschnitte in der Hilfe:

- Description
- Usage
- Arguments
- Value
- Author(s)
- See Also
- Examples

Die Beispiele in der Hilfe

Ausschnitt aus der Hilfe Seite zum Befehl `qmap`:

Das Beispiel kann man direkt in die Konsole kopieren:

```
# qmap("baylor university")
qmap("baylor university", zoom = 14)
# und so weiter
```

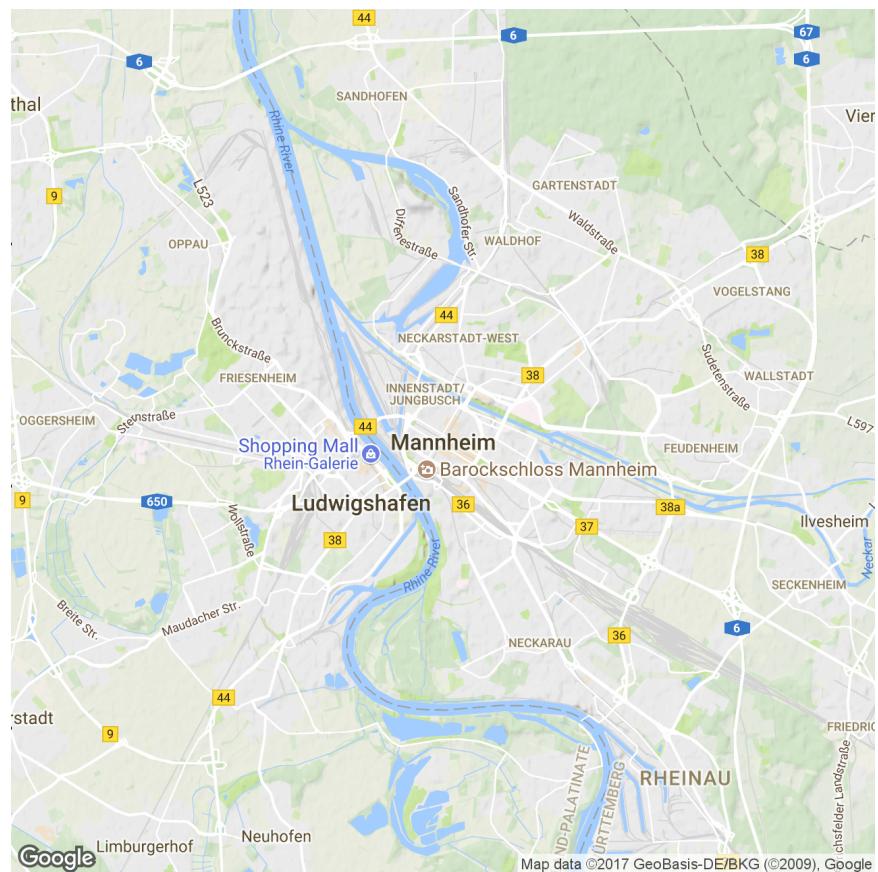
Examples

```
## Not run:  
# these examples have been excluded for checking efficiency  
  
qmap(location = "baylor university")  
qmap(location = "baylor university", zoom = 14)  
qmap(location = "baylor university", zoom = 14, source = "osm")
```

Figure 3: qmap Example

Ein anderes *zoom level*

```
qmap("Mannheim", zoom = 12)
```



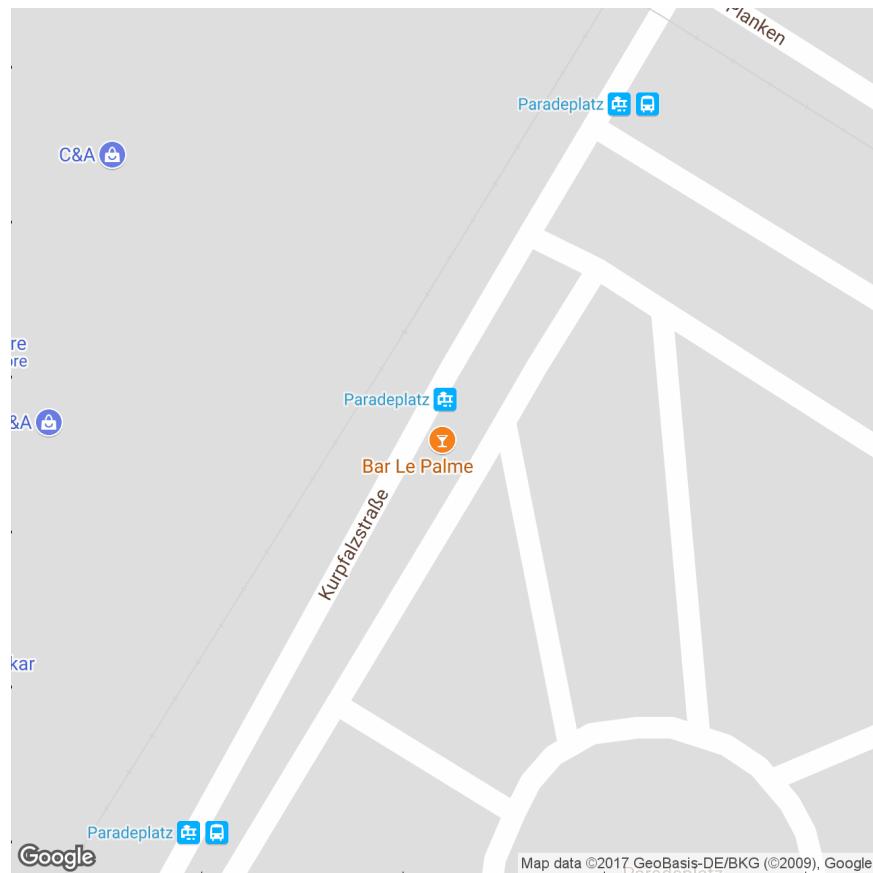
Näher rankommen

```
qmap('Mannheim', zoom = 13)
```



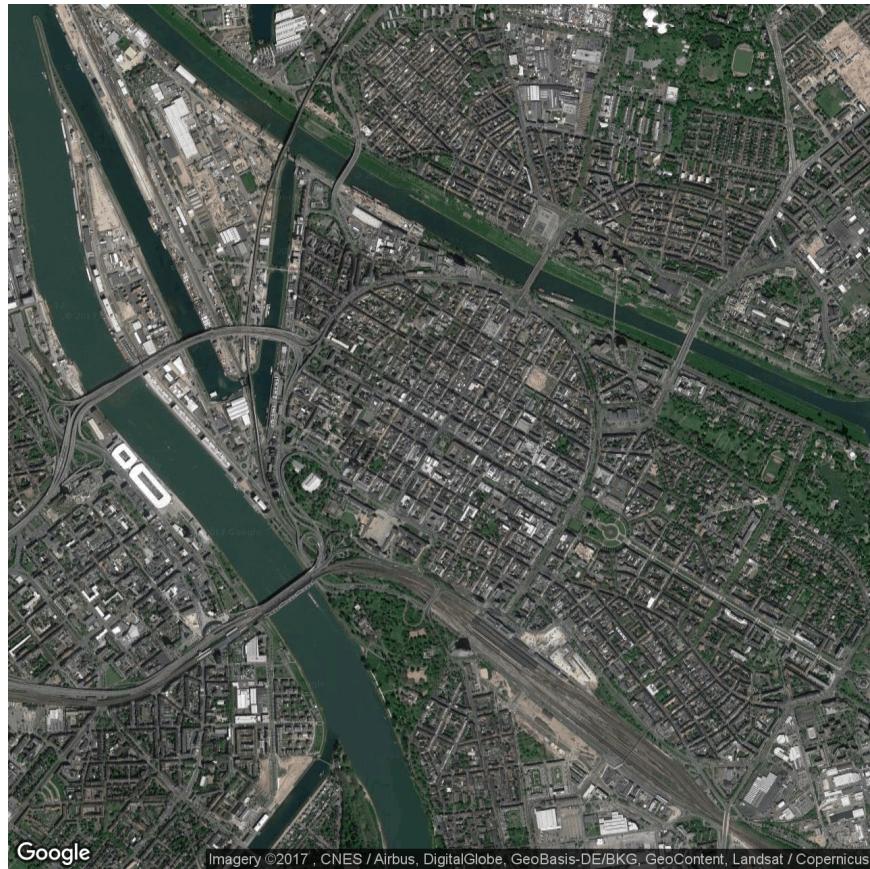
Ganz nah dran

```
qmap('Mannheim', zoom = 20)
```



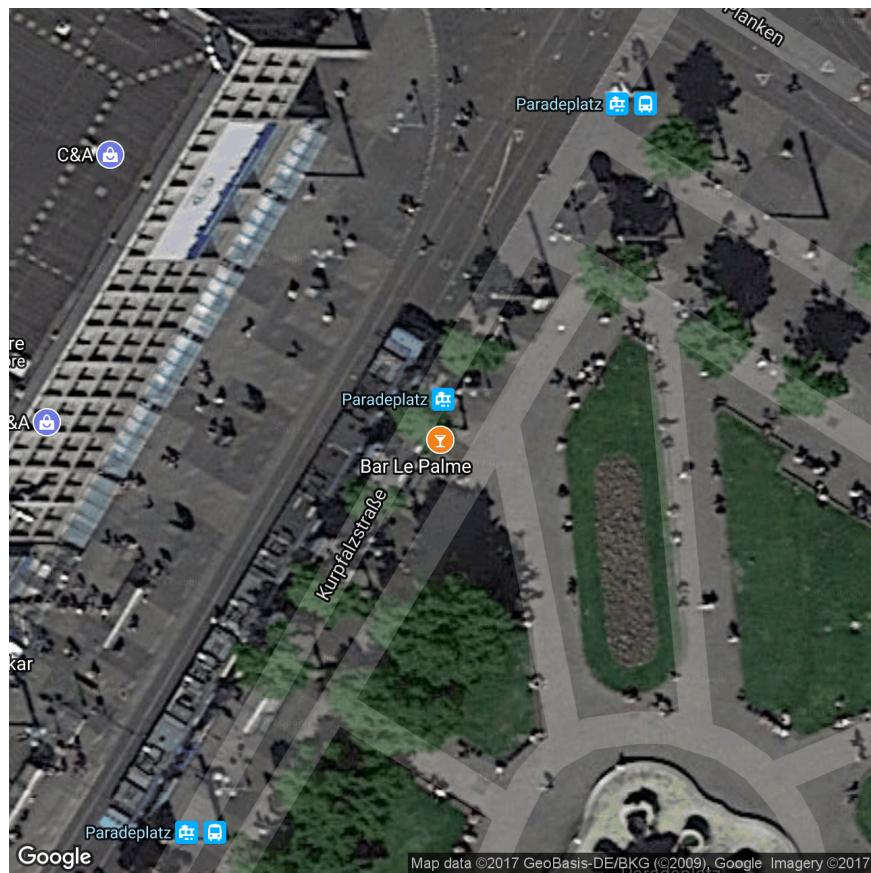
ggmap - maptype satellite

```
qmap('Mannheim', zoom = 14, maptype="satellite")
```



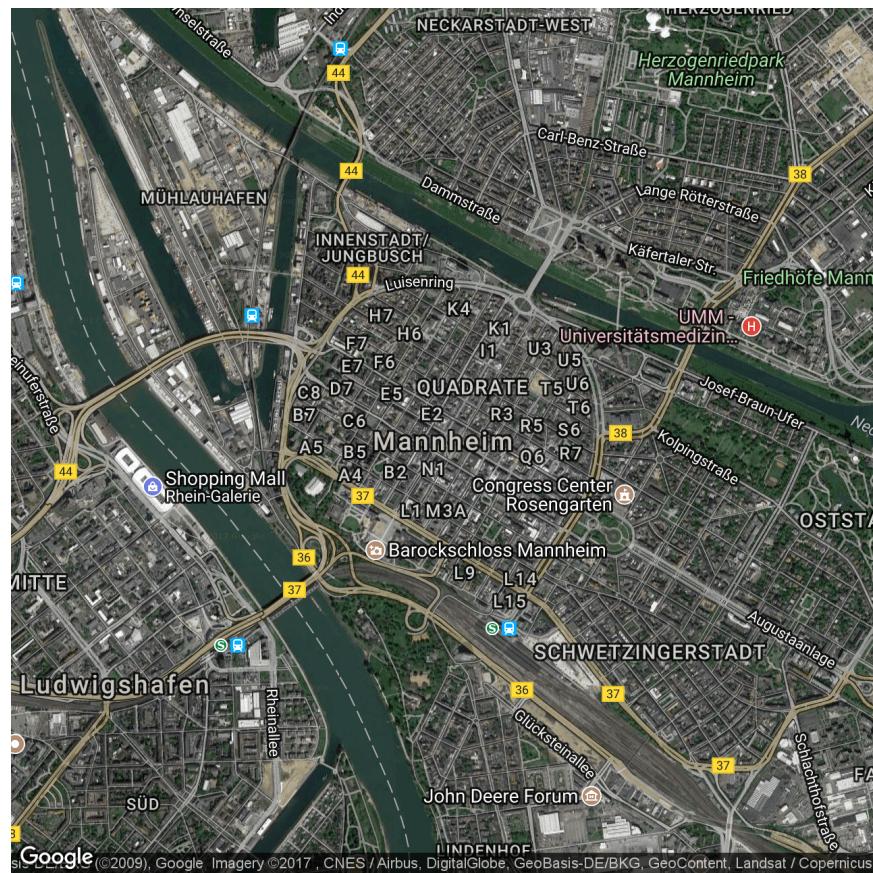
ggmap - maptype satellite zoom 20

```
qmap('Mannheim', zoom = 20, maptype="satellite")
```



ggmap - maptype hybrid

```
qmap("Mannheim", zoom = 14, maptype="hybrid")
```

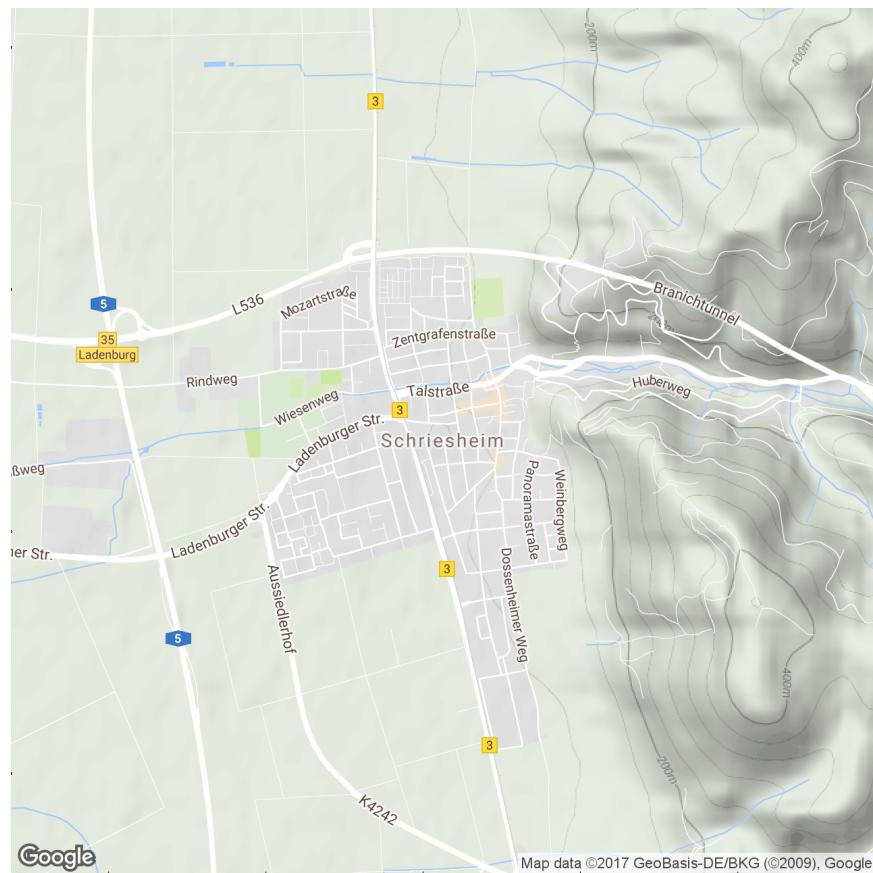


Terrain/physical maps

- Aus Physischen Karten kann man Informationen über Berge, Flüsse und Seen ablesen.
- Farben werden oft genutzt um Höhenunterschiede zu visualisieren

ggmap - terrain map

```
qmap('Schriesheim', zoom = 14, maptype="terrain")
```



Abstrahierte Karten (<http://www.designfaves.com>)



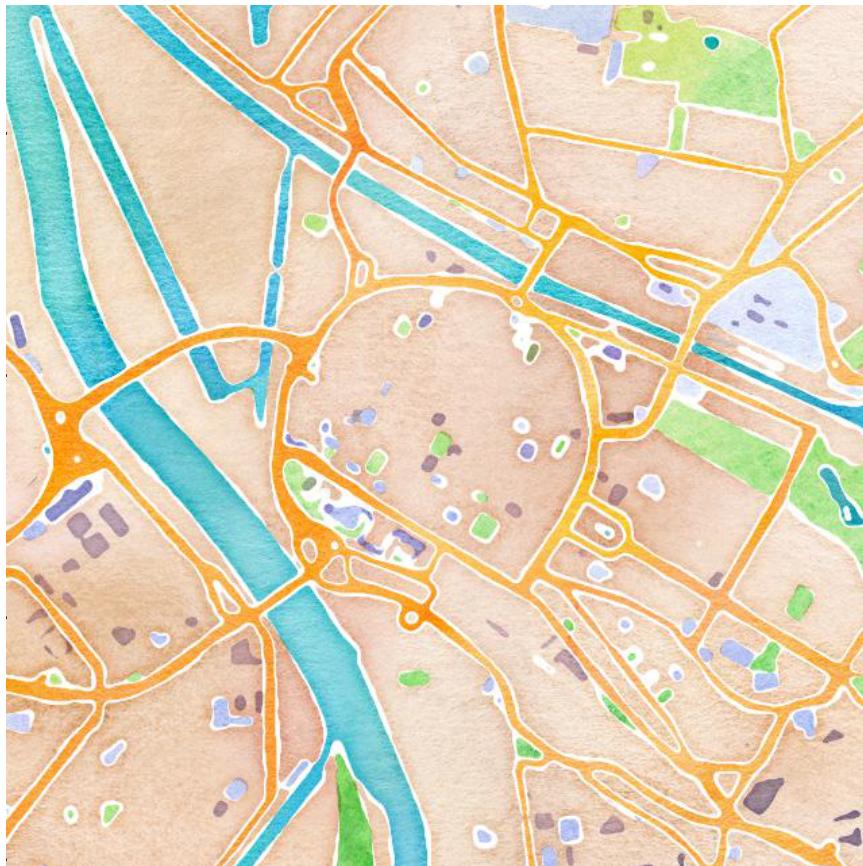
Figure 4: New York

- Abstraktion wird genutzt um nur die essentiellen Informationen einer Karte zu zeigen.
- Bsp. U-Bahn Karten - wichtig sind Richtungen und wenig Infos zur Orientierung

- Im folgenden werden Karten vorgestellt, die sich gut als Hintergrundkarten eignen.

ggmap - maptype watercolor

```
qmap('Mannheim', zoom = 14, maptype="watercolor", source="stamen")
```



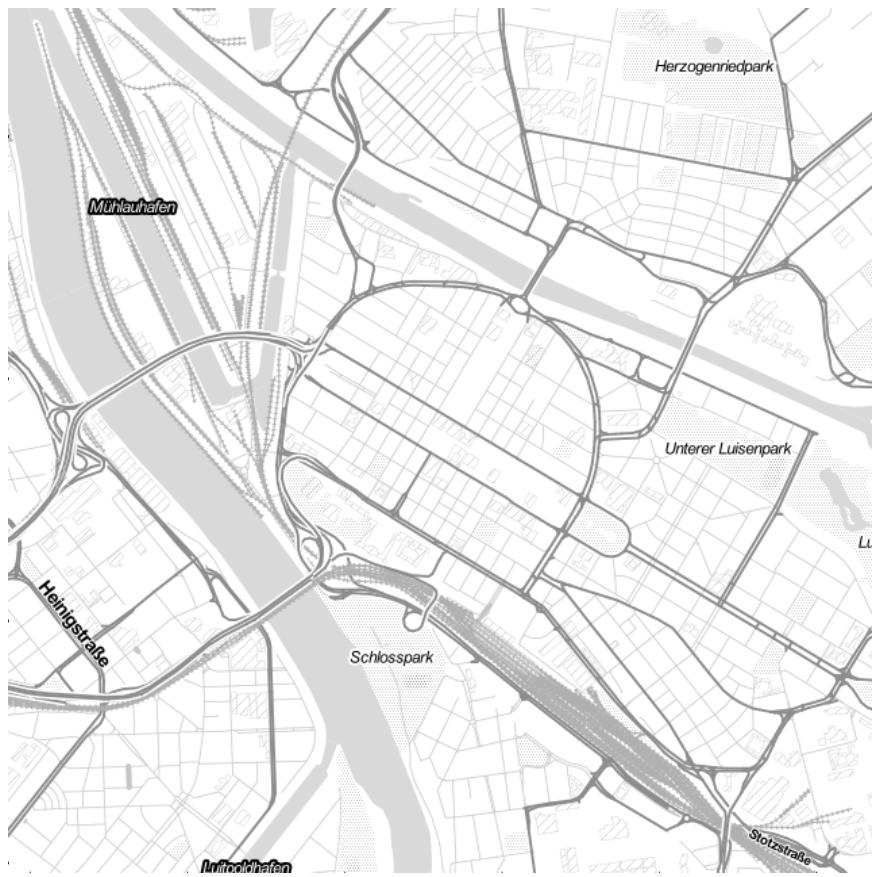
ggmap - source stamen

```
qmap('Mannheim', zoom = 14,  
      maptype="toner", source="stamen")
```



ggmap - maptype toner-lite

```
qmap('Mannheim', zoom = 14,  
      maptype="toner-lite", source="stamen")
```



ggmap - maptype toner-hybrid

```
qmap('Mannheim', zoom = 14,  
      maptype="toner-hybrid", source="stamen")
```



ggmap - maptype terrain-lines

```
qmap('Mannheim', zoom = 14,  
      maptype="terrain-lines", source="stamen")
```



Graphiken speichern

ggmap - ein Objekt erzeugen

- <- ist der Zuweisungspfeil um ein Objekt zu erzeugen
- Dieses Vorgehen macht bspw. Sinn, wenn mehrere Karten nebeneinander gebraucht werden.

```
MA_map <- qmap('Mannheim',
                 zoom = 14,
                 maptype="toner",
                 source="stamen")
```

Geokodierung

Geocoding (...) uses a description of a location, most typically a postal address or place name, to find geographic coordinates from spatial reference data ...

Wikipedia - Geocoding

```
library(ggmap)
geocode("Mannheim",source="google")
```

	lon	lat
	8.463182	49.48608

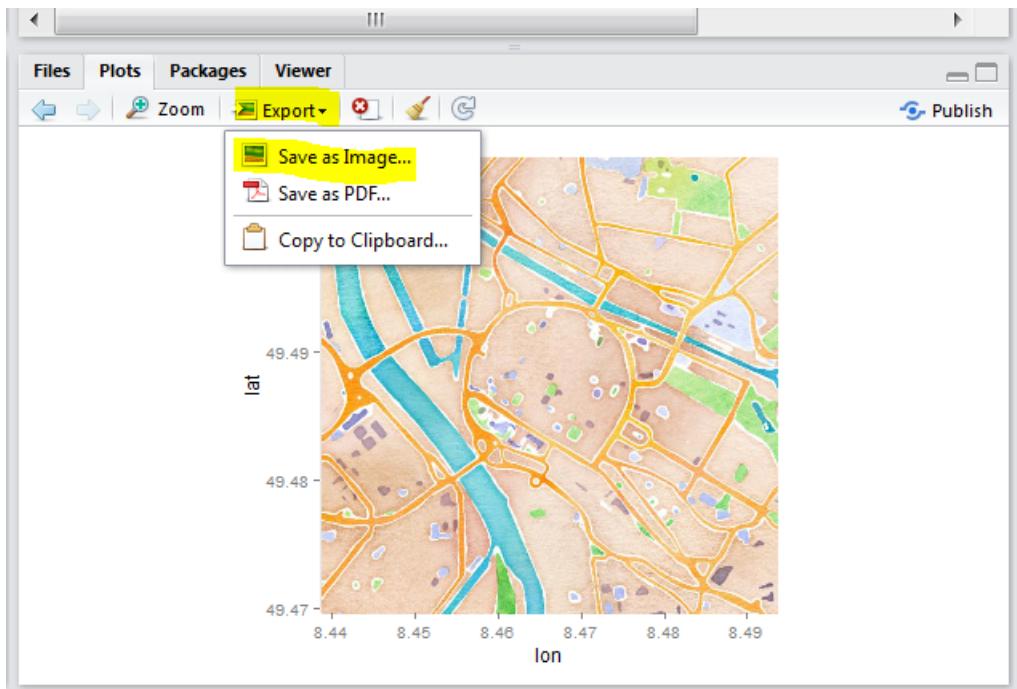


Figure 5: RstudioExport

Latitude und Longitude

<http://modernsurvivalblog.com>

Koordinaten verschiedener Orte in Deutschland

cities	lon	lat
Hamburg	9.993682	53.55108
Koeln	6.960279	50.93753
Dresden	13.737262	51.05041
Muenchen	11.581981	48.13513

Reverse Geokodierung

Reverse geocoding is the process of back (reverse) coding of a point location (latitude, longitude) to a readable address or place name. This permits the identification of nearby street addresses, places, and/or areal subdivisions such as neighbourhoods, county, state, or country.

Quelle: Wikipedia

```
revgeocode(c(48,8))
```

```
## [1] "Unnamed Road, Somalia"
```

Die Distanz zwischen zwei Punkten

```
mapdist("Q1, 4 Mannheim","B2, 1 Mannheim")
```

```
##           from      to    m   km 29 miles seconds minutes
## 1 Q1, 4 Mannheim B2, 1 Mannheim 749 0.749 0.4654286     215 3.583333
##          hours
## 1  0.05972222
```

```
mapdist("Q1, 4 Mannheim","B2, 1 Mannheim", mode="walking")
```

LATITUDE

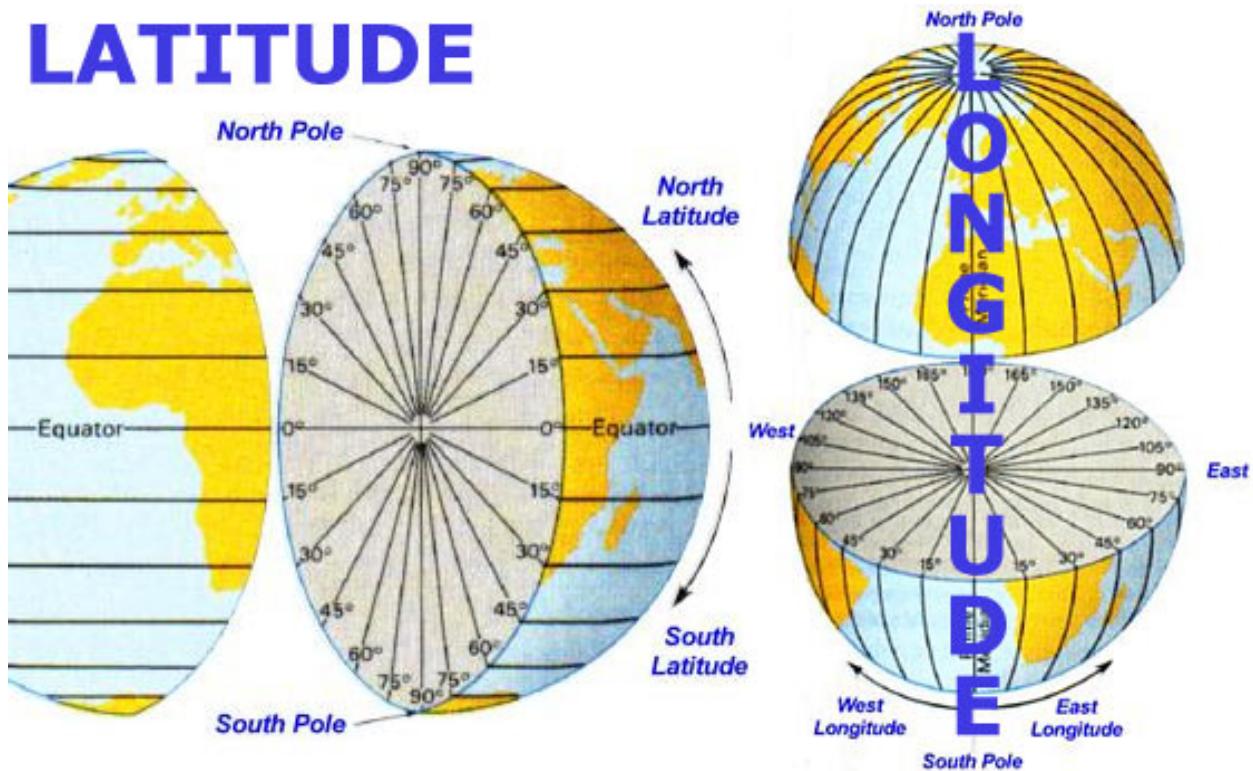


Figure 6: LatLon

Geokodierung - verschiedene Punkte von Interesse

```

POI1 <- geocode("B2, 1 Mannheim",source="google")
POI2 <- geocode("Hbf Mannheim",source="google")
POI3 <- geocode("Mannheim, Friedrichsplatz",source="google")
ListPOI <-rbind(POI1,POI2,POI3)
POI1;POI2;POI3

##           lon      lat
## 1 8.462844 49.48569
##           lon      lat
## 1 8.469879 49.47972
##           lon      lat
## 1 8.475208 49.48326

```

Punkte in der Karte

```

MA_map +
geom_point(aes(x = lon, y = lat),
data = ListPOI)

```



Punkte in der Karte

```
MA_map +  
geom_point(aes(x = lon, y = lat), col="red",  
data = ListPOI)
```



ggmap - verschiedene Farben

```
ListPOI$color <- c("A", "B", "C")
MA_map +
  geom_point(aes(x = lon, y = lat, col = color),
  data = ListPOI)
```



ggmap - größere Punkte

```
ListPOI$size <- c(10,20,30)
MA_map +
  geom_point(aes(x = lon, y = lat, col=color, size=size),
  data = ListPOI)
```



Eine Route von Google maps bekommen

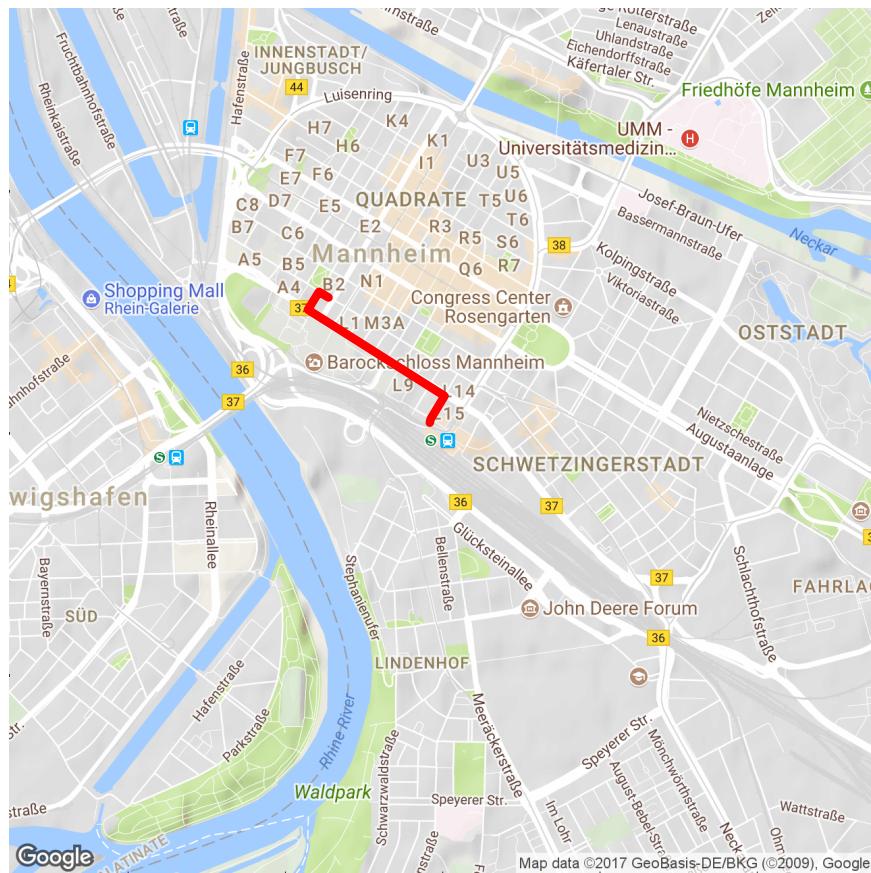
```
from <- "Mannheim Hbf"
to <- "Mannheim B2 , 1"
route_df <- route(from, to, structure = "route")
```

Mehr Information

<http://rpackages.ianhowson.com/cran/ggmap/man/route.html>

Eine Karte mit dieser Information zeichnen

```
qmap("Mannheim Hbf", zoom = 14) +
  geom_path(
    aes(x = lon, y = lat), colour = "red", size = 1.5,
    data = route_df, lineend = "round"
  )
```



Wie fügt man Punkte hinzu

- Nutzung von geom_point
- Question on stackoverflow

<http://i.stack.imgur.com>

Cheatsheet

- Cheatsheet zu data visualisation

<https://www.rstudio.com/>

Resourcen und Literatur

- Artikel von David Kahle und Hadley Wickham zur Nutzung von ggmap.
- Schnell eine Karte bekommen
- Karten machen mit R
- Problem mit der Installation von ggmap

Take Home Message

Was klar sein sollte:

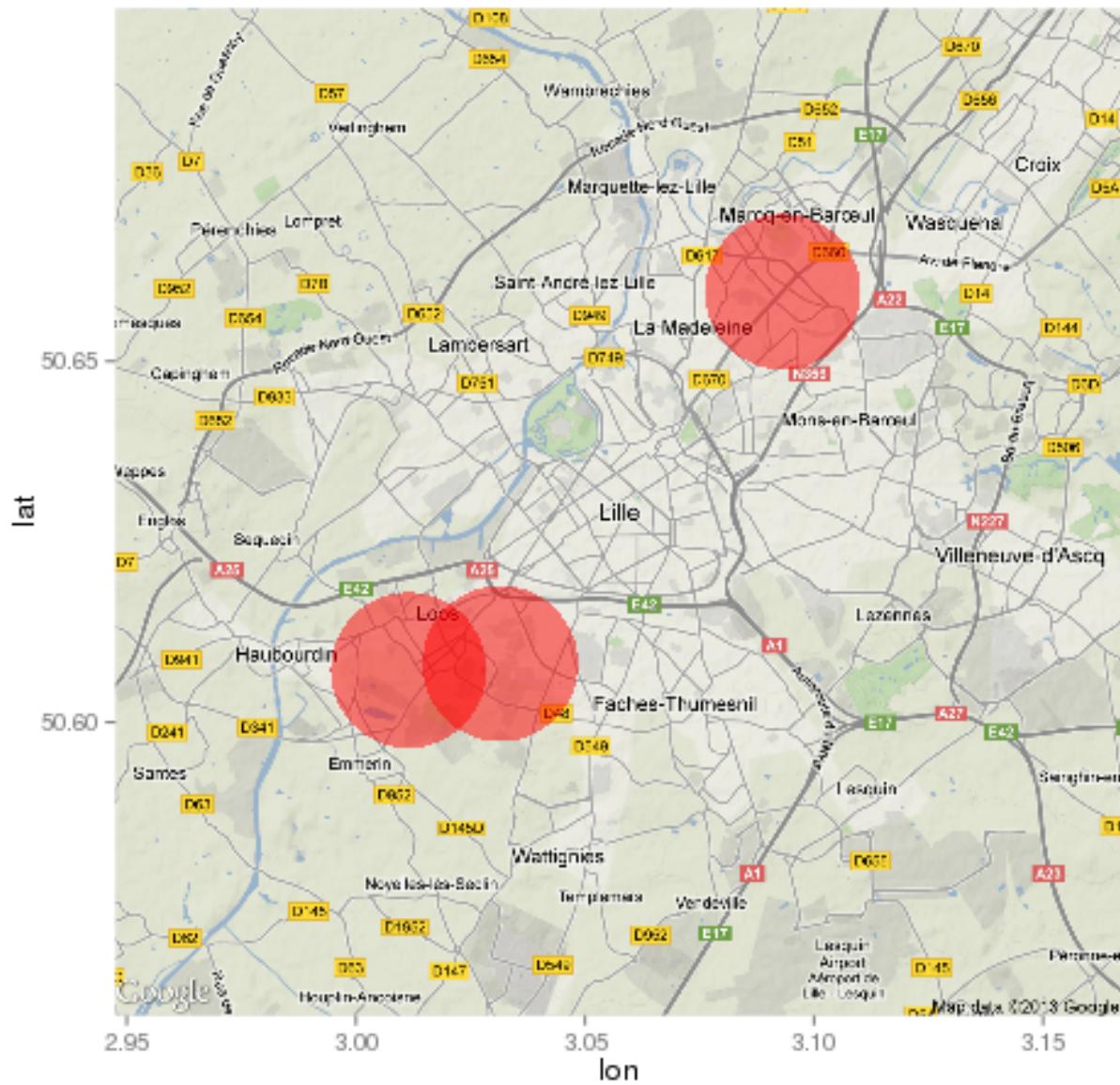


Figure 7:

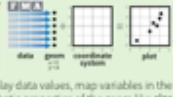
Data Visualization with ggplot2

Cheat Sheet



Basics

ggplot2 is based on the [grammar of graphics](#), the idea you can build every graph from the same few components: a **data** set, a set of **geoms**—visual objects that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like `size`, `color`, and `x` and `y` locations.



Build a graph with `qplot()` or `ggplot()`

```
qplot(x + cty, y + hwy, color = cyl, data = mpg, geom = "point")
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.
```

`ggplot(data = mpg, aes(x = cty, y = hwy))`

Begins a plot that you finish by adding layers to. No defaults, but provides more control than `qplot()`.

```
ggplot(mpg, aes(hwy, cty)) +
  geom_point(aes(color = cyl)) + # layer 1: geom and stat
  geom_smooth(method = "lm") + # layer 2: geom and stat
  scale_color_gradient() + # layer 3: geom and mappings
  theme_bw()
```

Add a new layer to a plot with a `geom_*` or `stat_*` function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

`last_plot()`

Returns the last plot

`ggsave("plot.png", width = 5, height = 5)`

Saves last plot as 5" x 5" file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

```
a <- ggplot(mpg, aes(cty, hwy))
#+ geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size
b + geom_area(stat = "density")
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, linetype, size, weight
b + geom_density2d()
x, y, alpha, color, fill
```

Discrete

```
b <- ggplot(mpg, aes(cty))
b + geom_bar()
x, y, alpha, color, fill, linetype, size, weight
```

Graphical Primitives

```
c <- ggplot(mpg, aes(long, lat))
c + geom_polygon(aes(group = group))
x, y, alpha, color, fill, linetype, size
```

```
d <- ggplot(economics, aes(date, unemploy))
d + geom_path(linewidth = "butt",
  linejoin = "round", lineend = "square")
x, y, alpha, color, linetype, size
```

```
d + geom_rect(ymin = round(unemploy - 900),
  ymax = unemploy + 900)
x, ymin, ymax, alpha, color, fill, linetype, size
```

```
e <- ggplot(seals, aes(x = long, y = lat))
e + geom_segment(aes(
  xend = long + delta_long,
  yend = lat + delta_lat))
x, rend, y, rend, alpha, color, linetype, size
```

```
e + geom_rect(aes(xmin = long, ymin = lat,
  xmax = long + delta_long,
  ymax = lat + delta_lat))
xmin, ymin, ymax, alpha, color, fill,
linetype, size
```

Two Variables

Continuous X, Continuous Y

```
f <- ggplot(mpg, aes(cty, hwy))
f + geom_blank()
#+ geom_jitter()
x, y, alpha, color, fill, shape, size
f + geom_point()
x, y, alpha, color, fill, shape, size
f + geom_quantile()
x, y, alpha, color, linetype, size, weight
f + geom_rug(sides = "lf")
alpha, color, linetype, size
f + geom_smooth(method = lm)
x, y, alpha, color, fill, linetype, size, weight
f + geom_step(direction = "hv")
x, y, alpha, color, linetype, size
```

A **B** **C**

Continuous Bivariate Distribution

```
i <- ggplot(movies, aes(year, rating))
i + geom_bin2d(binwidth = c(5, 0.5))
xmax, xmin, ymax, ymin, alpha, color, fill,
linetype, size, weight
i + geom_density2d()
x, y, alpha, colour, linetype, size
i + geom_hex()
x, y, alpha, colour, fill, size
```

Continuous Function

```
j <- ggplot(economics, aes(date, unemploy))
j + geom_area()
x, y, alpha, color, fill, linetype, size
j + geom_line()
x, y, alpha, color, linetype, size
j + geom_step(direction = "hv")
x, y, alpha, color, linetype, size
```

Visualizing error

```
df <- data.frame(gpp = c("A", "B"), fit = 4.5, se = 1.2)
k <- ggplot(df, aes(gpp, fit, ymin = fit - se, ymax = fit + se))
k + geom_crossbar(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, linetype, size
k + geom_errorbar()
x, ymin, ymax, alpha, color, linetype, size, width (also geom_errorbarh())
k + geom_linerange()
x, ymin, ymax, alpha, color, linetype, size
k + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, linetype, size
```

Maps

```
data <- data.frame(murder = USArrests$Murder,
  state = USArrests$State, arrests = USArrests$Arrests)
murder, max, data, state
l <- ggplot(data, aes(state))
l + geom_map(map = map)
l + expand_limits(x = map$long, y = map$lat)
map_id, alpha, color, fill, linetype, size
```

Three Variables

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(lat, long))
#+ geom_raster(aes(z = z), hjust = 0.5,
  vjust = 0.5, interpolate = FALSE)
x, y, alpha, fill
```

```
m + geom_contour(aes(z = z))
x, y, alpha, colour, linetype, size, weight
```

D

Learn more at docs.ggplot2.org • ggplot2 0.9.3.1 • Updated: 3/15

Figure 8: Cheatsheet

- Wie man eine schnelle Karte erzeugt
- Wie man geokodiert
- Wie man eine Distanz berechnet