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 Grammitik zu Grunde

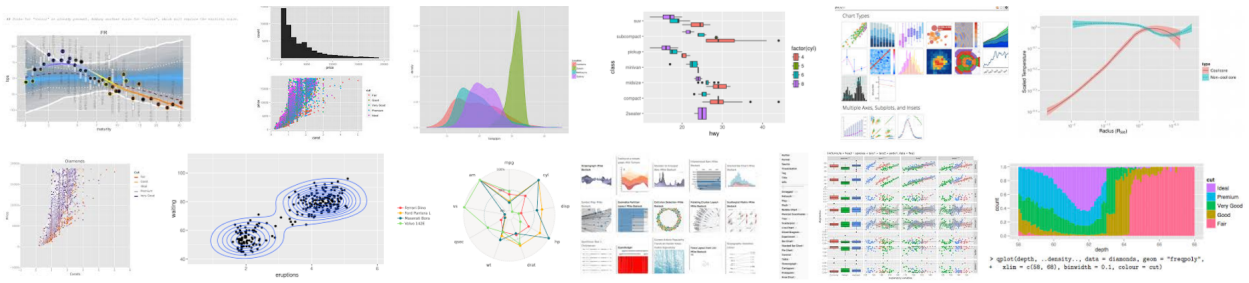


Figure 1:

Basiseinführung `ggplot2`

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

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

```
library(ggplot2)
```

Der `diamonds` Datensatz

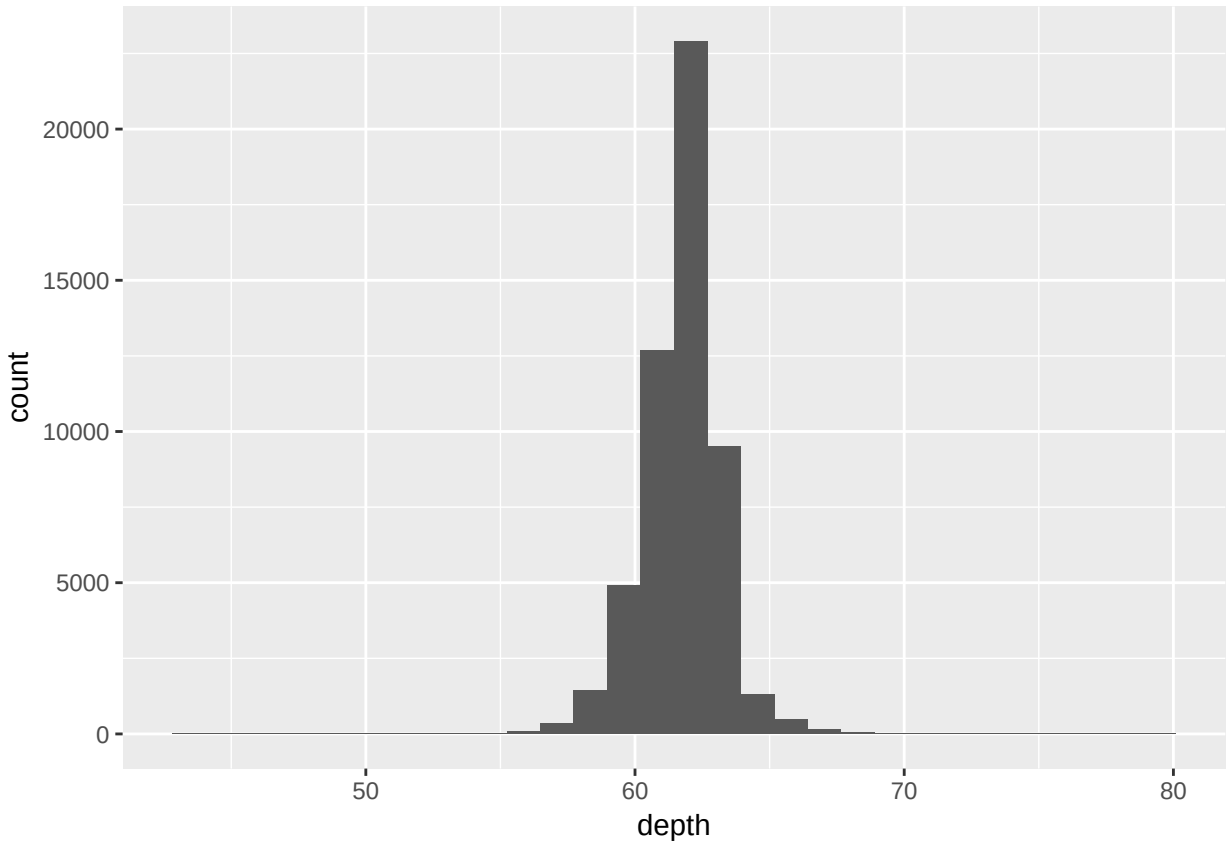
```
head(diamonds)
```

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`

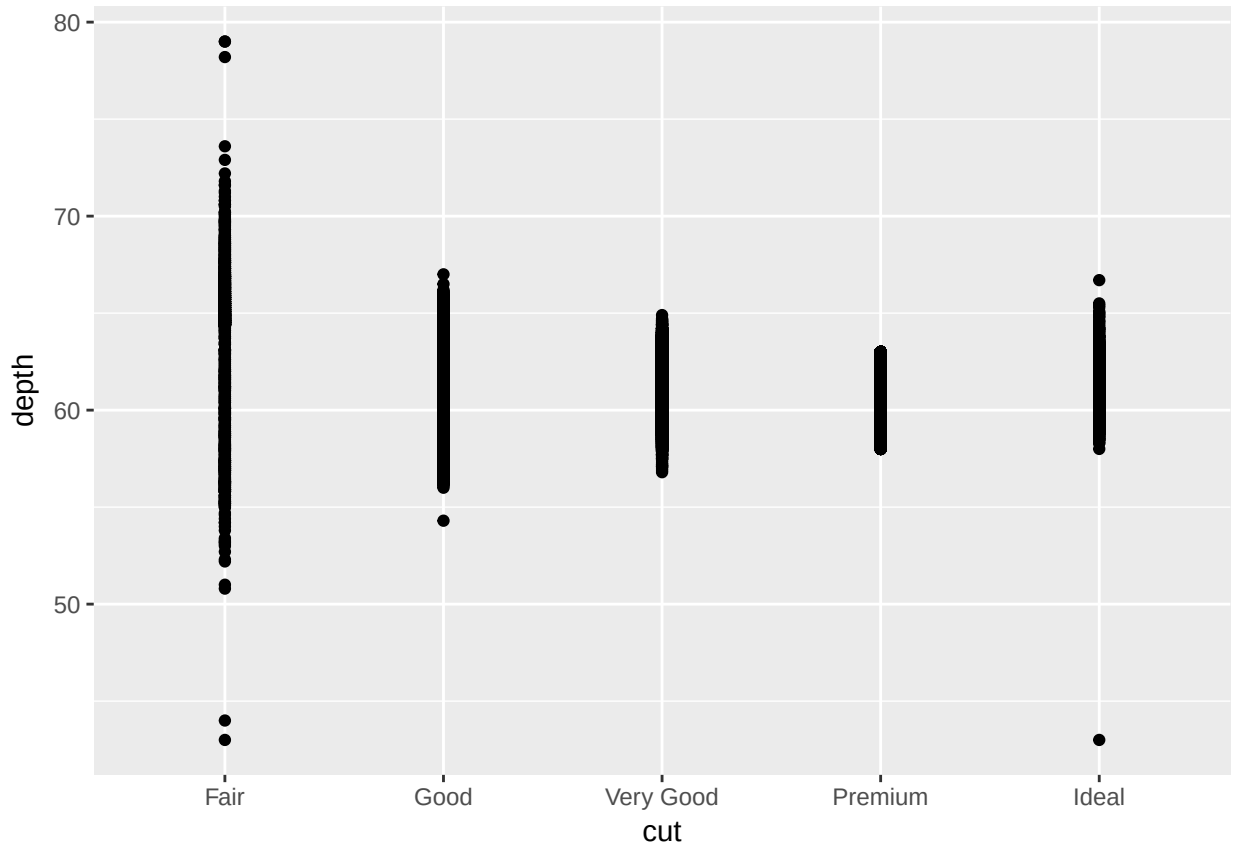
- Die Funktion `qplot` wird für schnelle Graphiken verwendet (quick plots)
- bei der Funktion `ggplot` kann man alles bis ins Detail kontrollieren

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



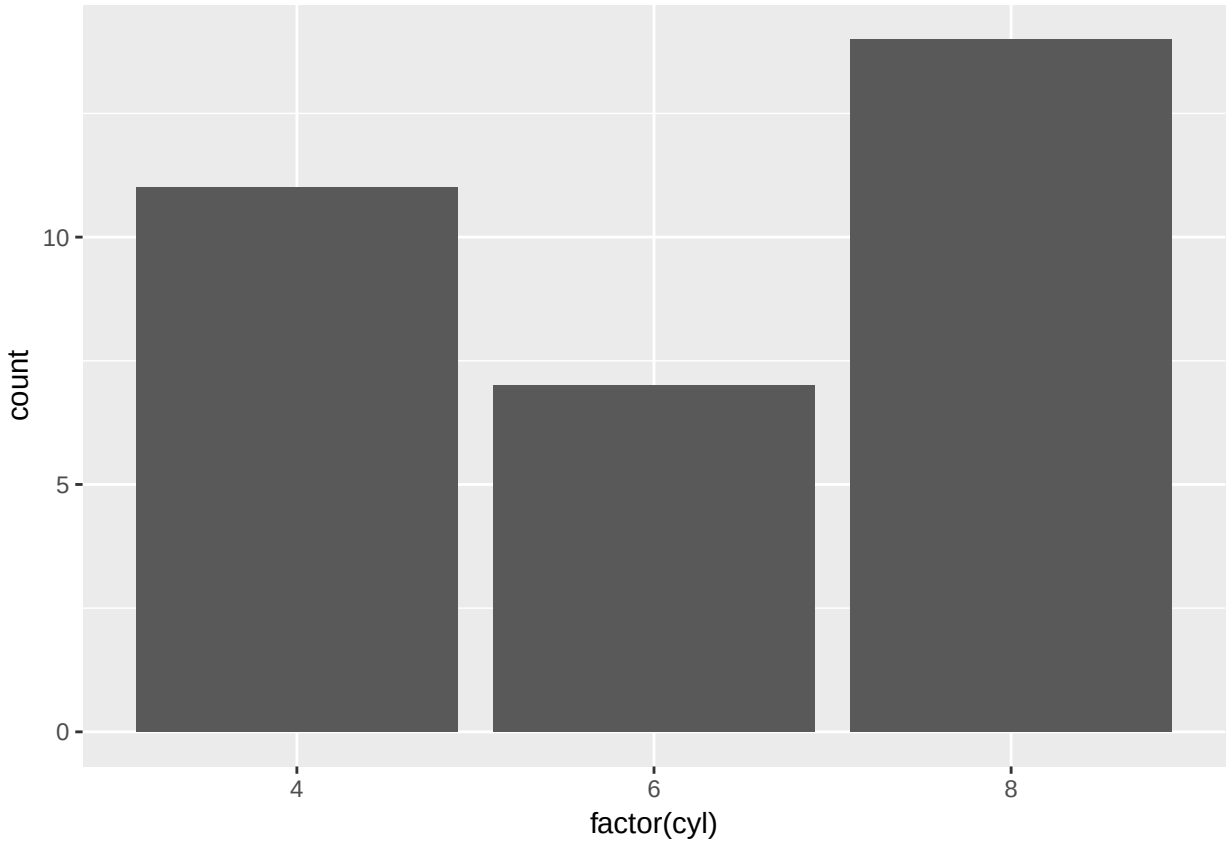
Ein Balkendiagramm

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



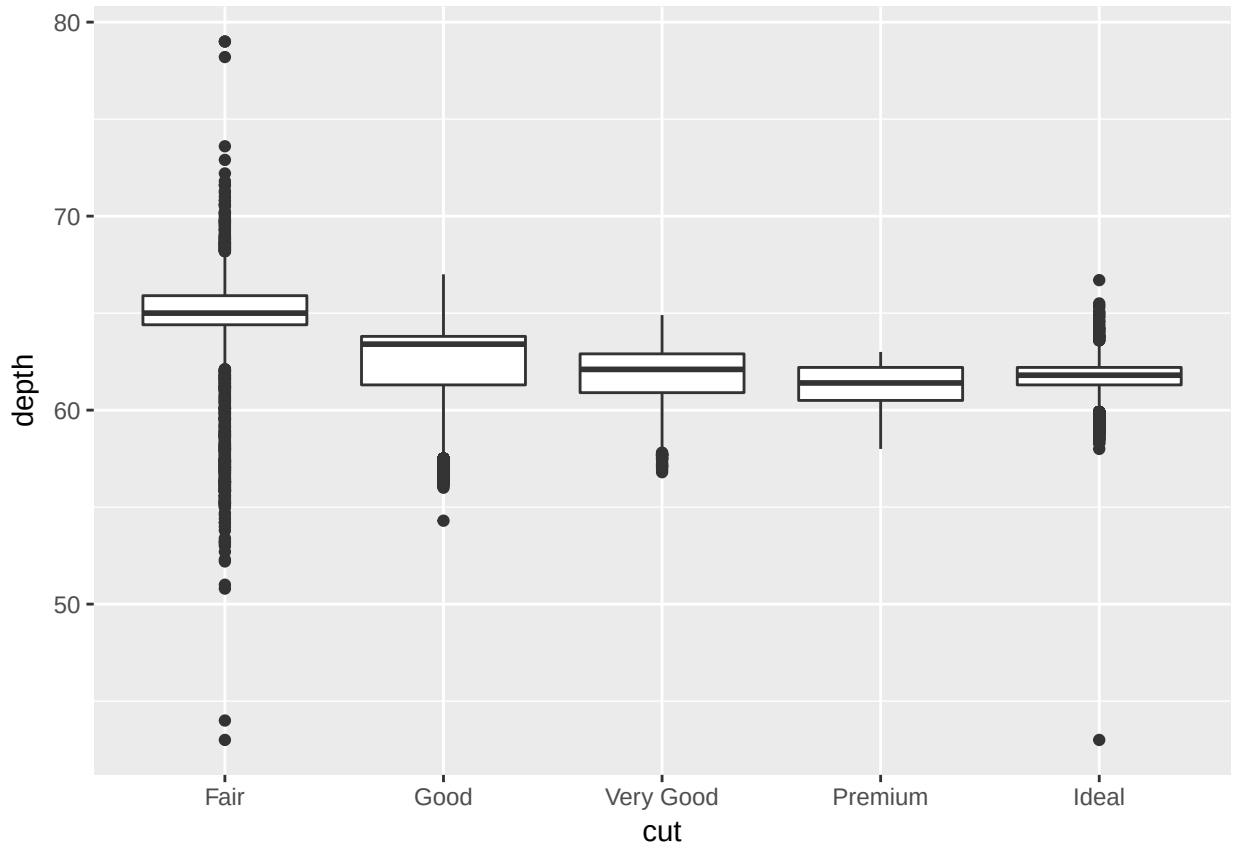
Ein weiteres Balkendiagramm

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



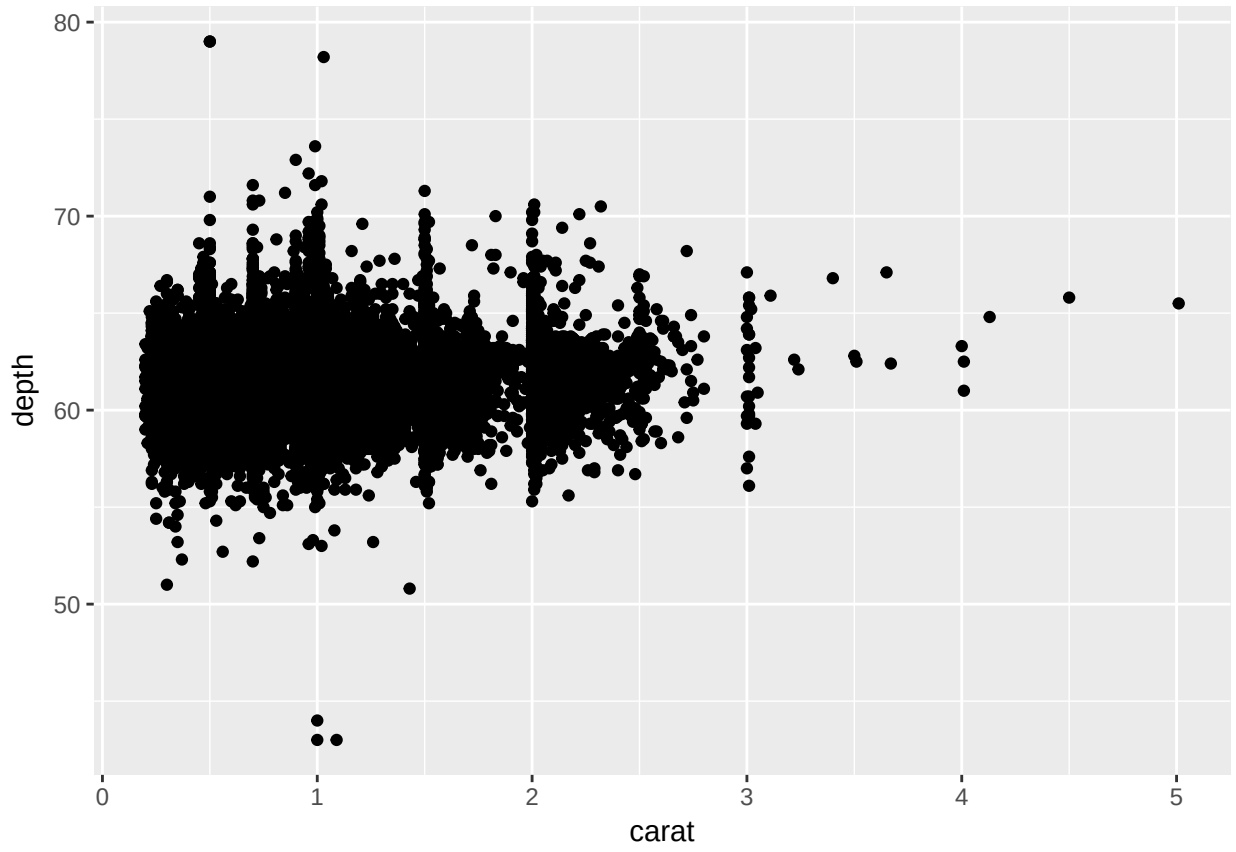
Boxplot

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



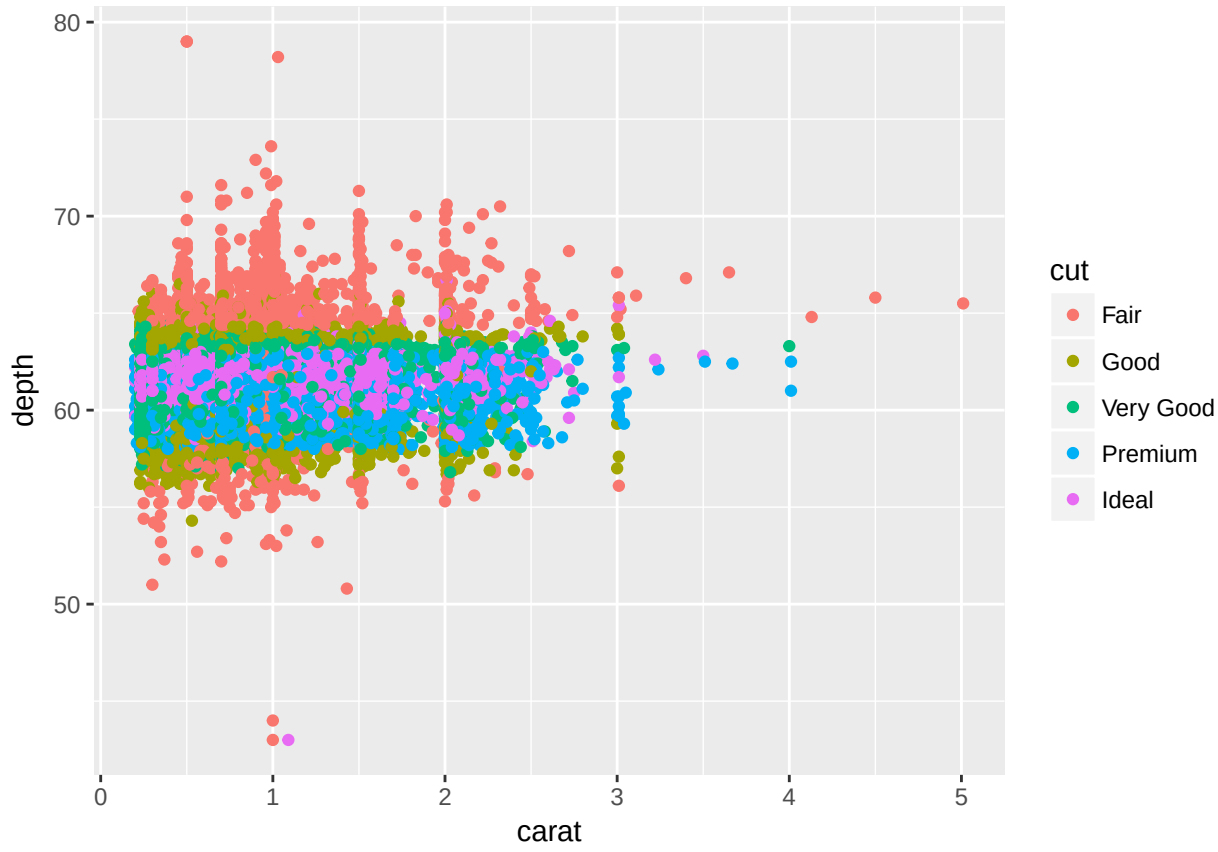
Scatterplot

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



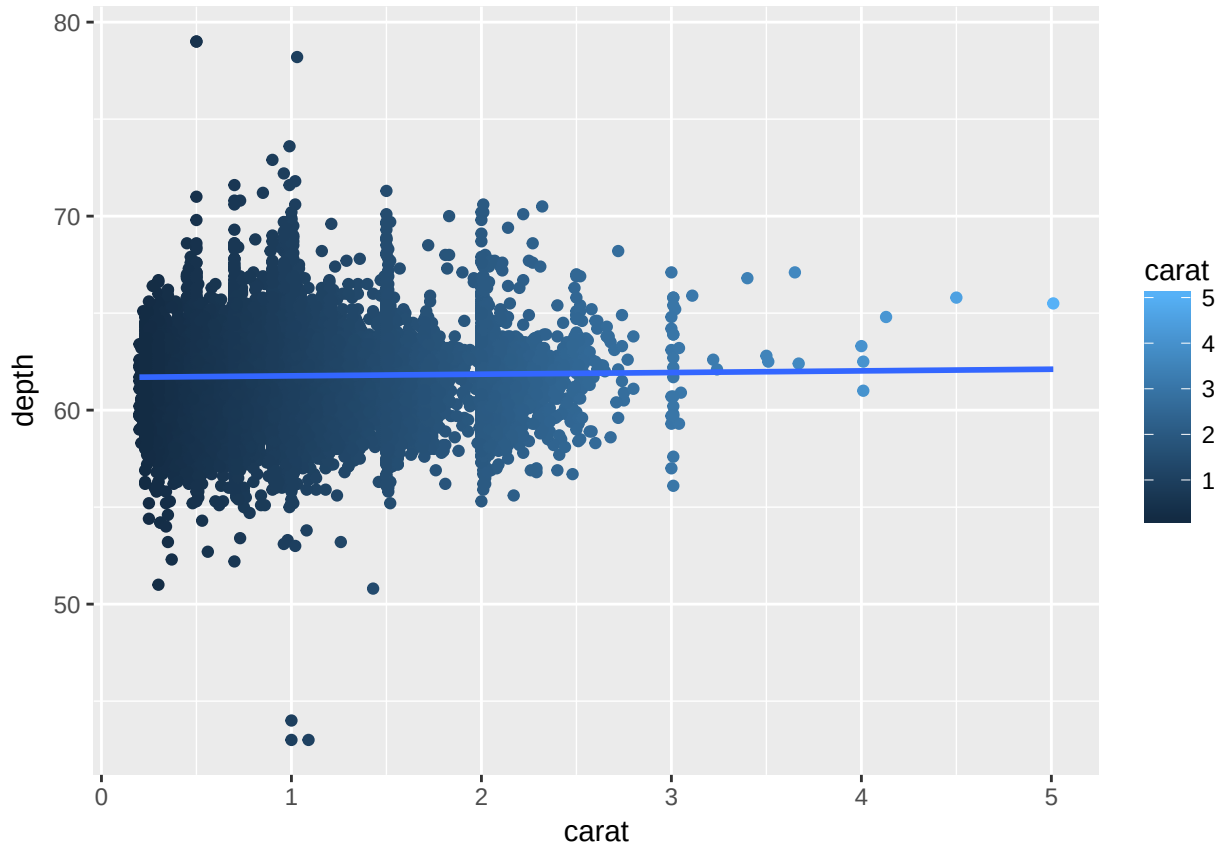
Farbe hinzu:

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



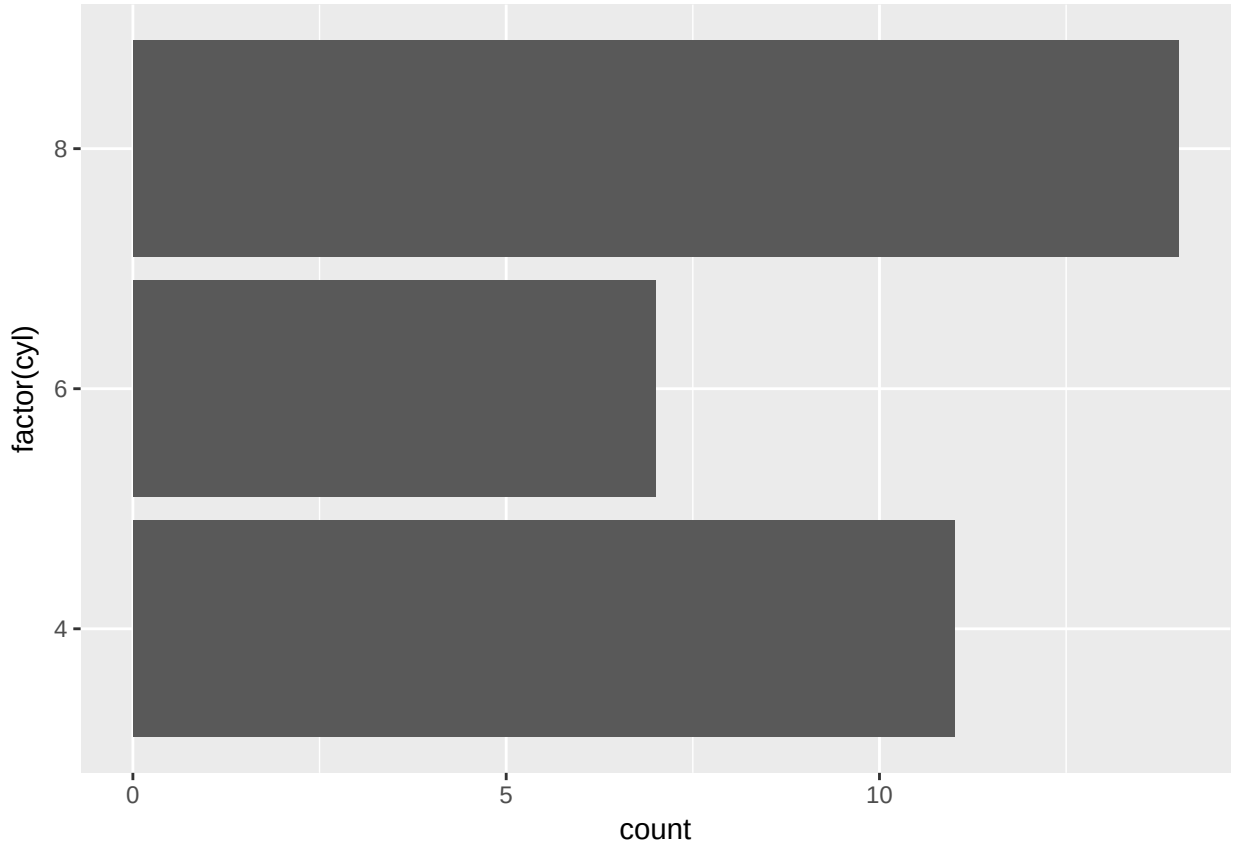
Trendlinie hinzufügen

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



Graphik drehen

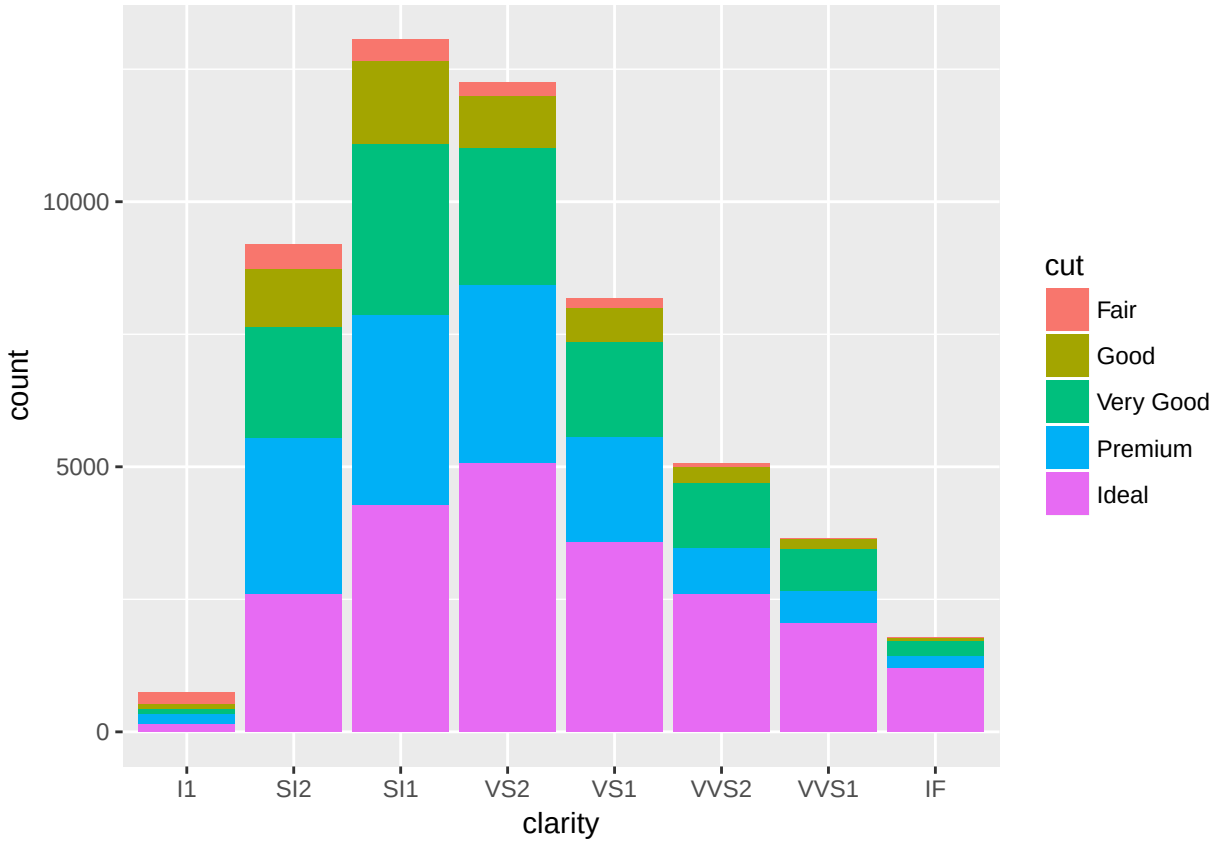
```
qplot(factor(cyl), data=mtcars, geom="bar") +
coord_flip()
```



Wie nutzt man ggplot

- die aestetics:

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



Farben selber wählen

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

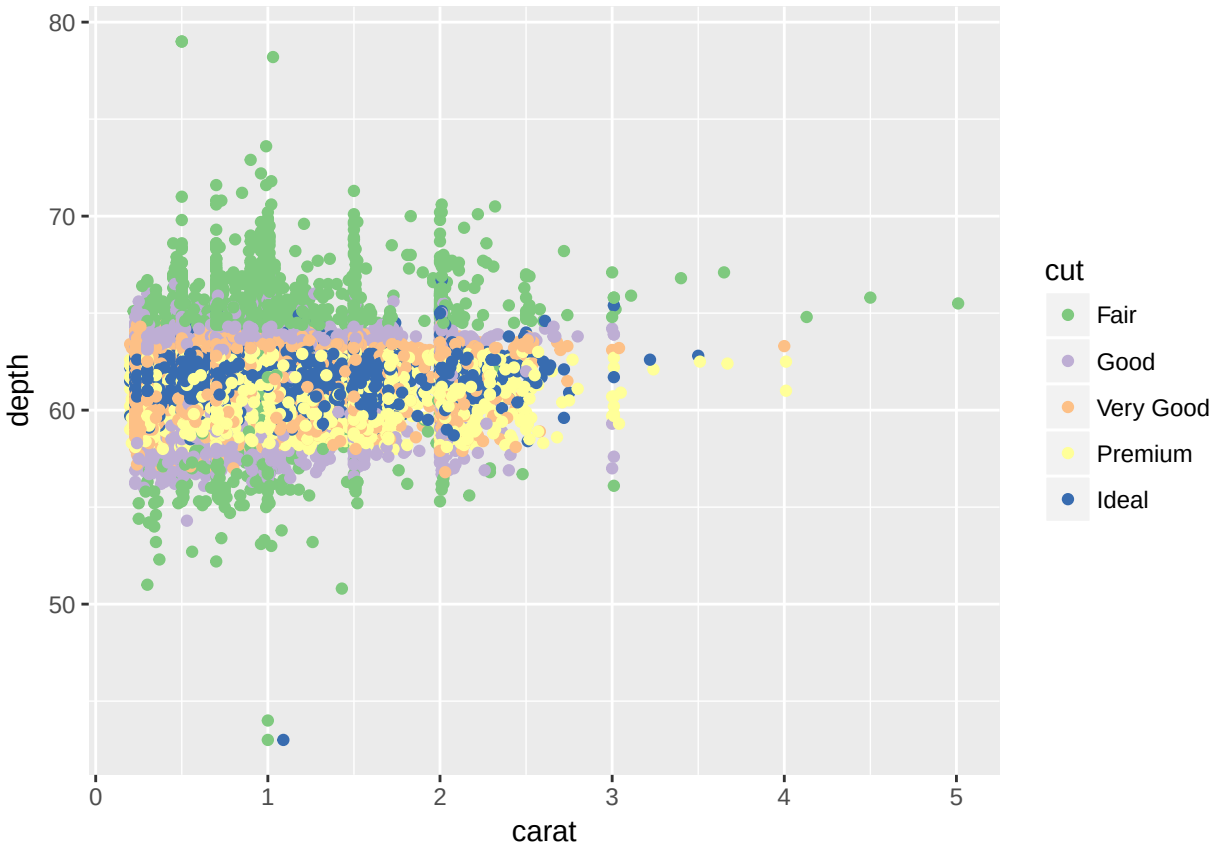
```
install.packages("RColorBrewer")
```

```
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

- ggplot2 Basics

- Noam Ross - Quick Introduction to ggplot2
- Plugin ggplot2

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")
```

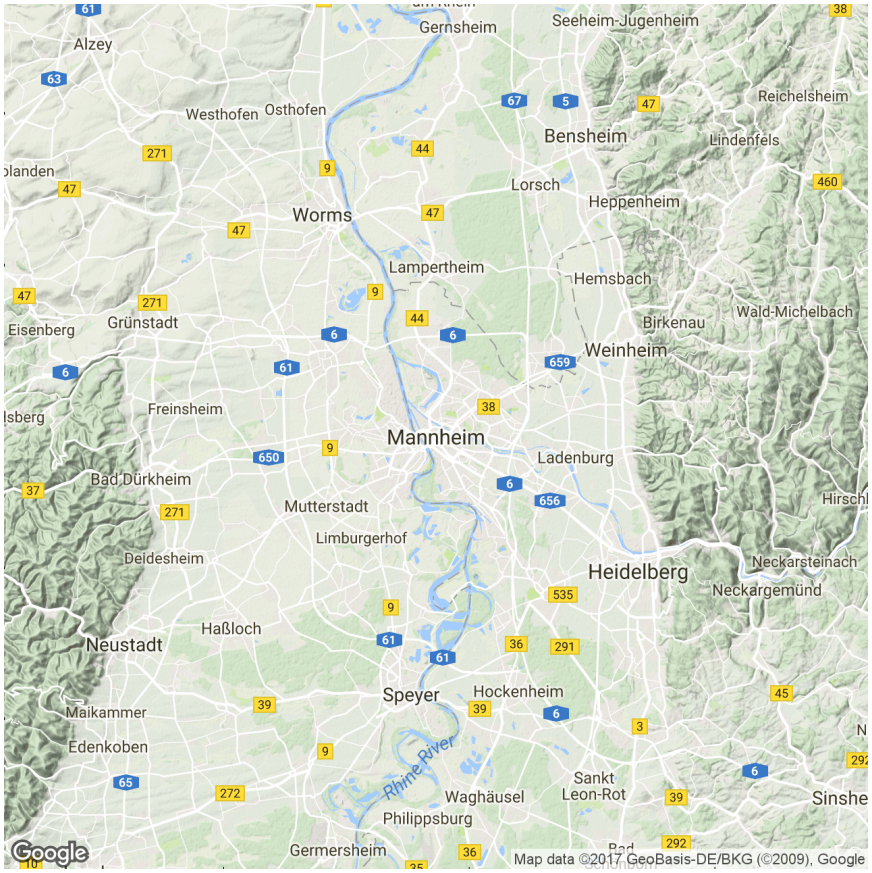
Paket ggmap - Hallo Welt

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

```
library(ggmap)
```

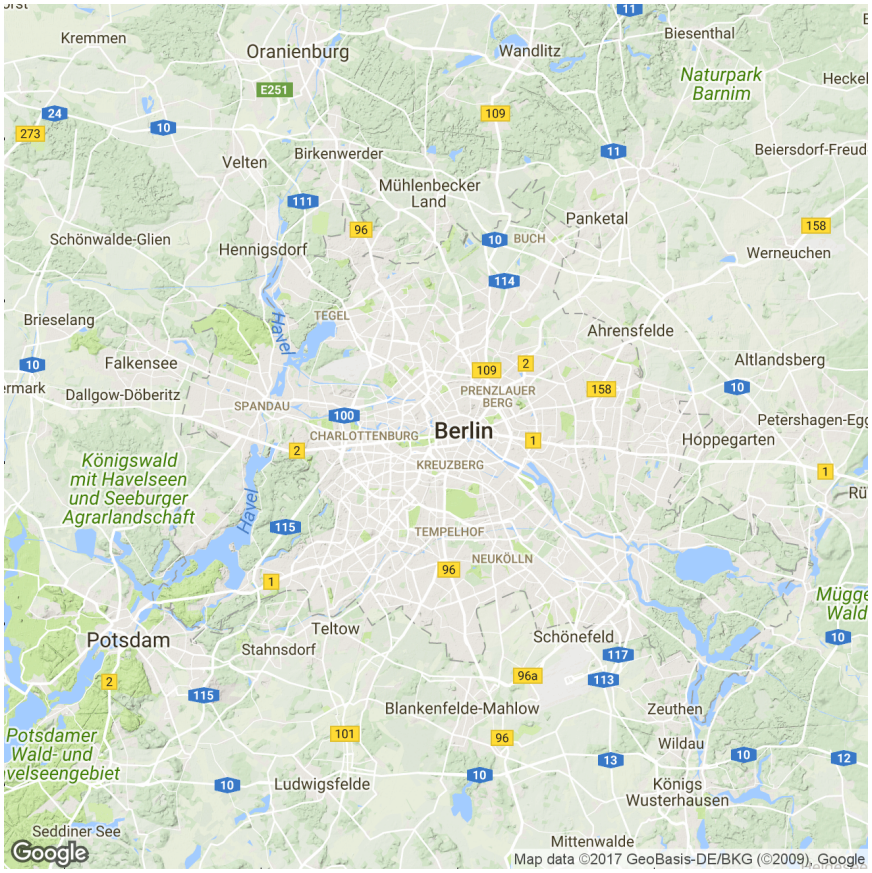
Und schon kann die erste Karte erstellt werden:

```
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*

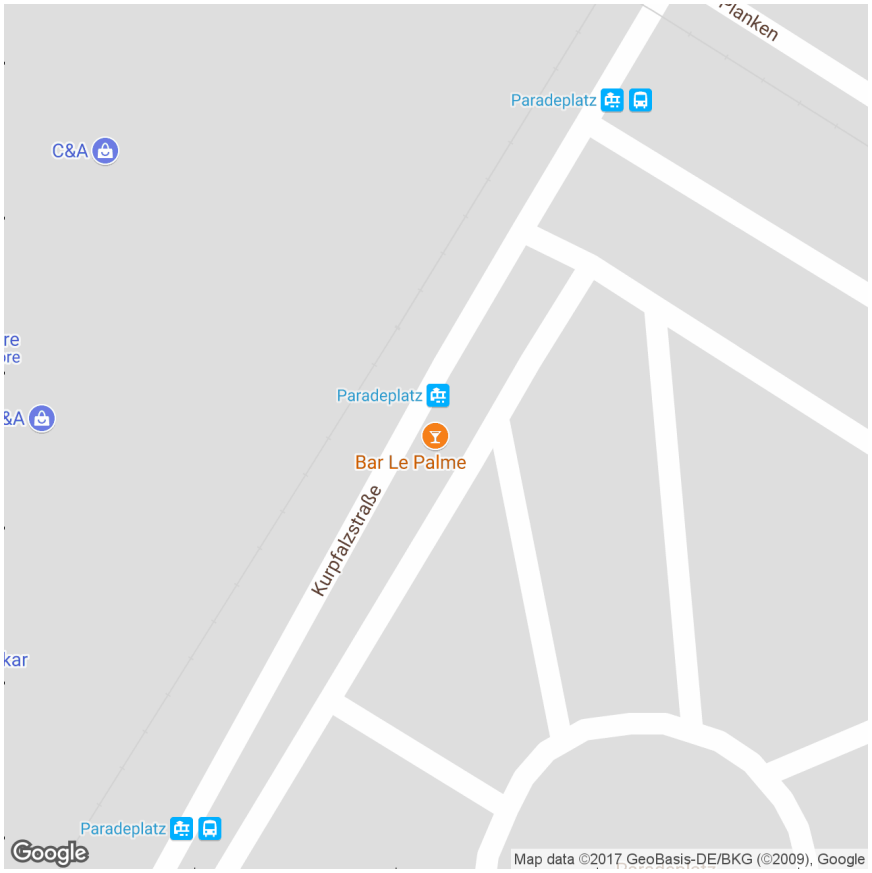
```
qmap("Mannheim", zoom = 12)
```

Näher rankommen

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

Ganz nah dran

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



ggmap - maptype satellite

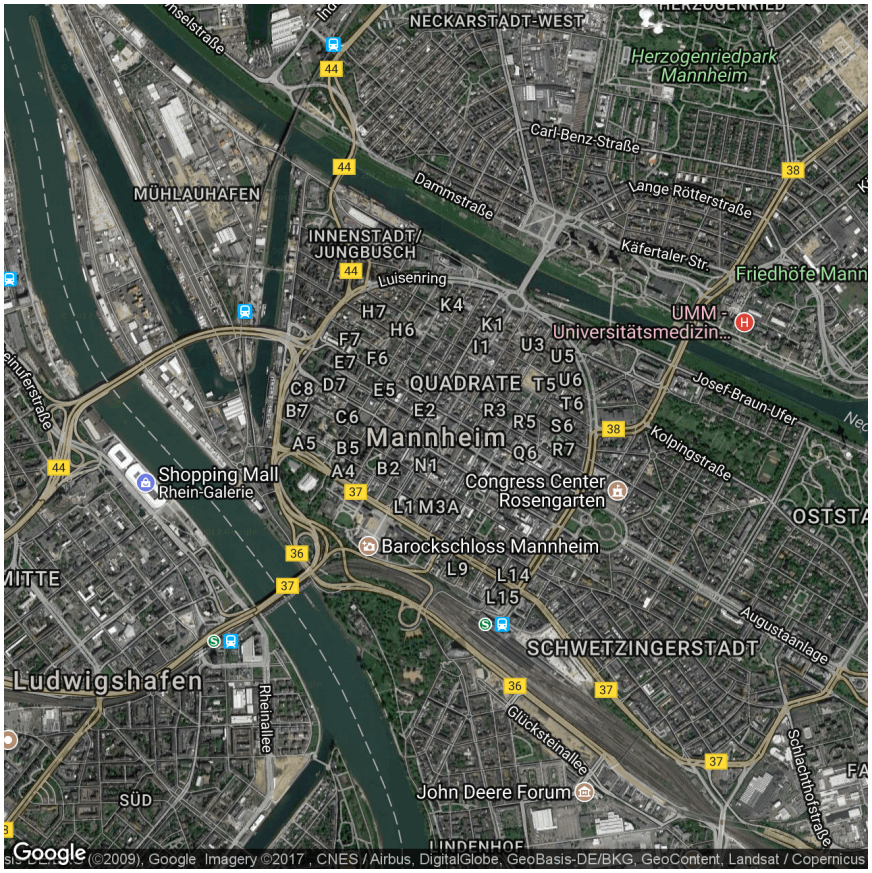
```
qmap('Mannheim', zoom = 14, maptype="satellite")
```

ggmap - maptype satellite zoom 20

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

ggmap - maptype hybrid

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

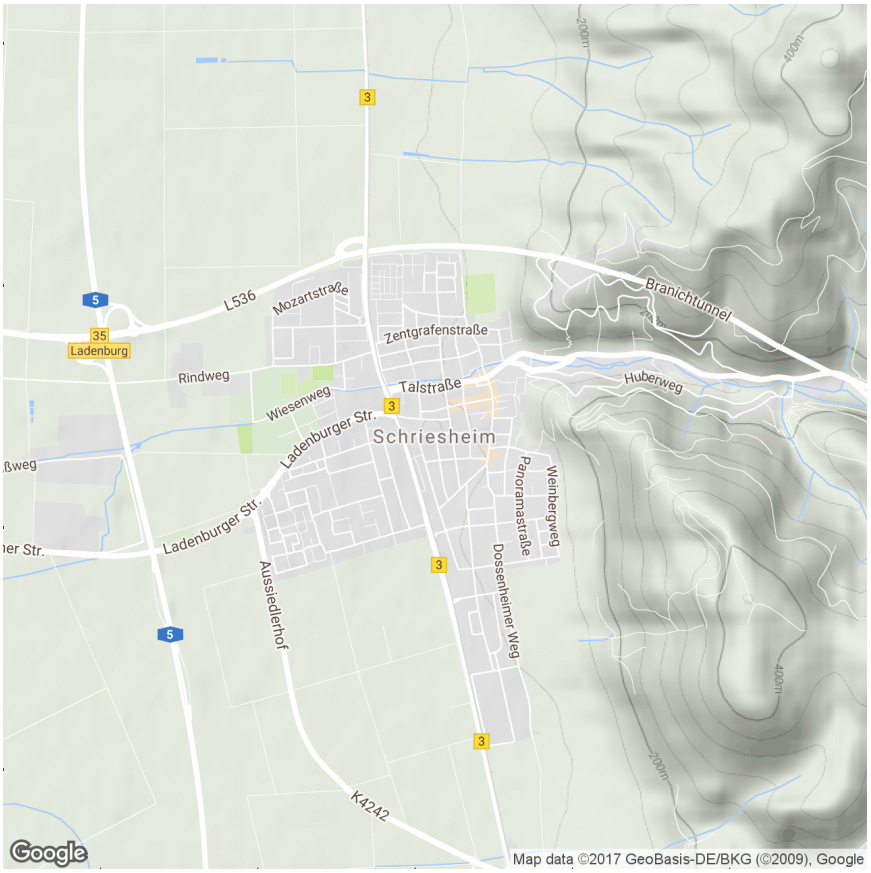


Terrain/physical maps

- 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

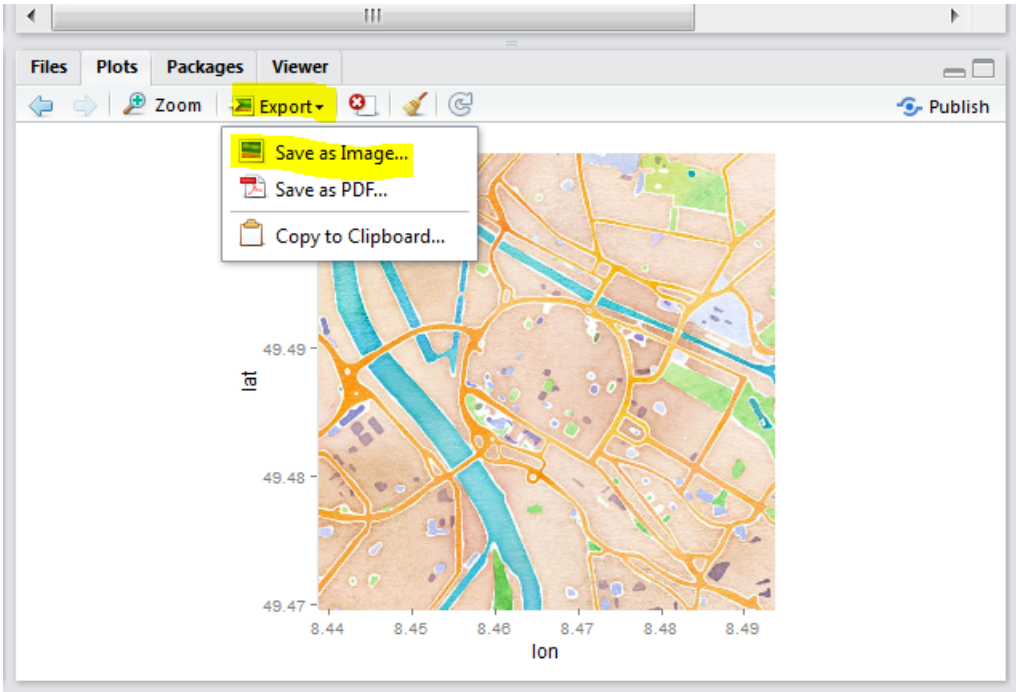


Figure 5: RstudioExport

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     miles seconds  minutes
## 1 Q1, 4 Mannheim B2, 1 Mannheim 749 0.749 0.4654286     215 3.583333
##       hours
## 1 0.05972222
```

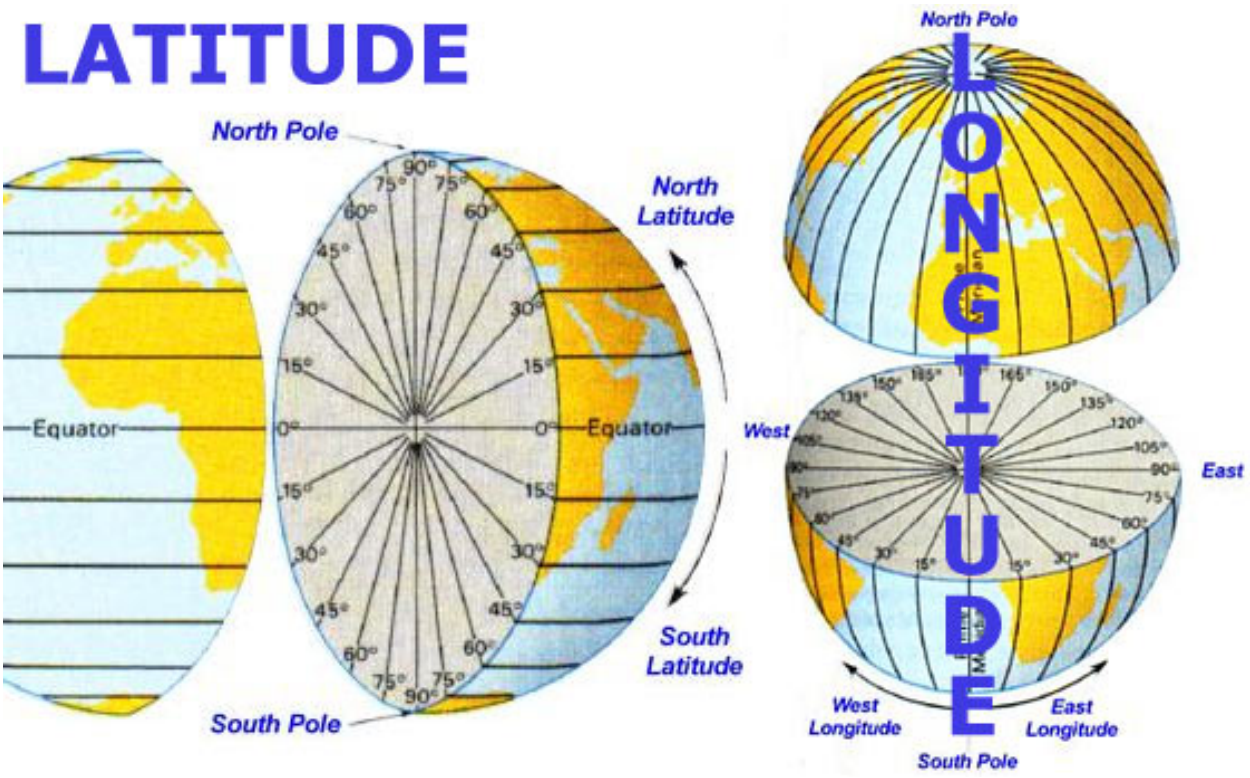


Figure 6: LatLon

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

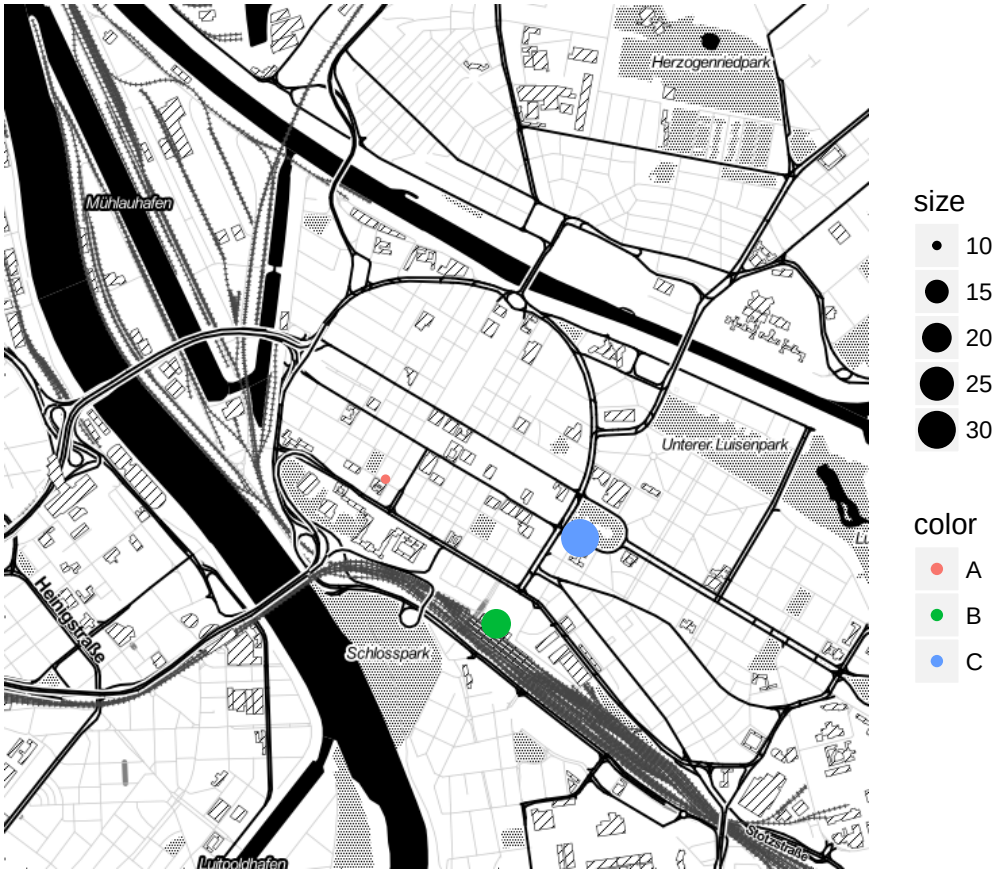
```
MA_map +
geom_point(aes(x = lon, y = lat),col="red",
data = ListPOI)
```

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

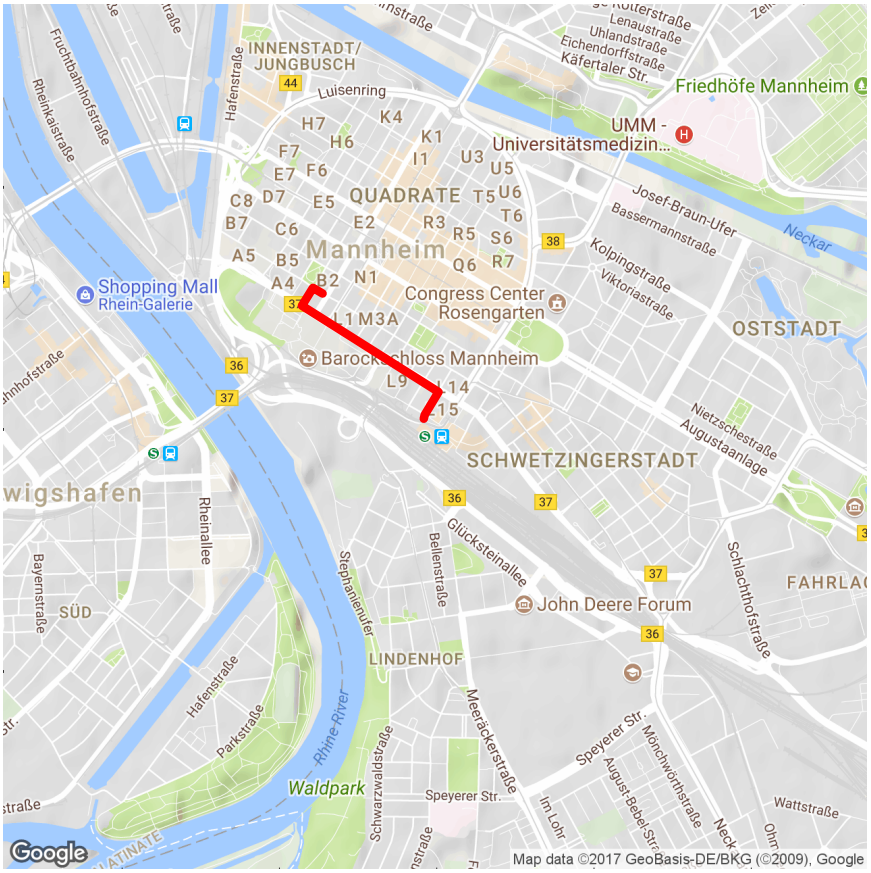
```
from <- "Mannheim Hbf"
to <- "Mannheim B2 , 1"
route_df <- route(from, to, structure = "route")
```

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:

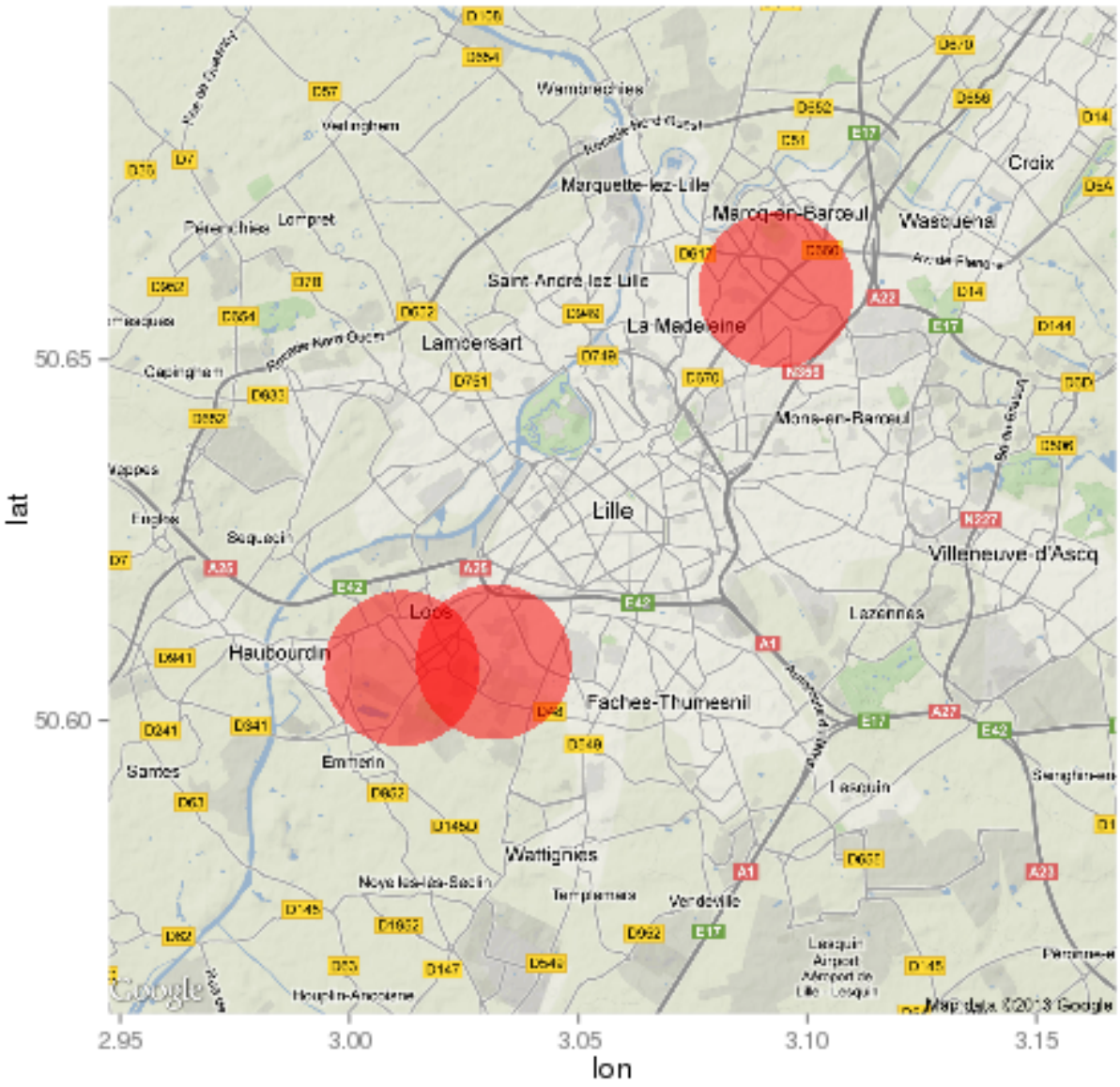

Marquette-lez-Lille
Marcq-en-Barœul
Wasquehal
Croix
Saint-André-lez-Lille
La Madeleine
Lambersart
Mons-en-Barœul
Lille
Villeneuve-d'Ascq
Lezennes
Loos
Haubourdin
Fâches-Thumesnil
Emmerin
Sames
Wattignies
Templemars
Verdrel
Lesquin
Noyelles-lès-Seclin
Houplin-Ancoisne
Google
Map data ©2013 Google
lat
50.65
50.60
2.95
3.00
3.05
3.10
3.15
lon

Figure 7:

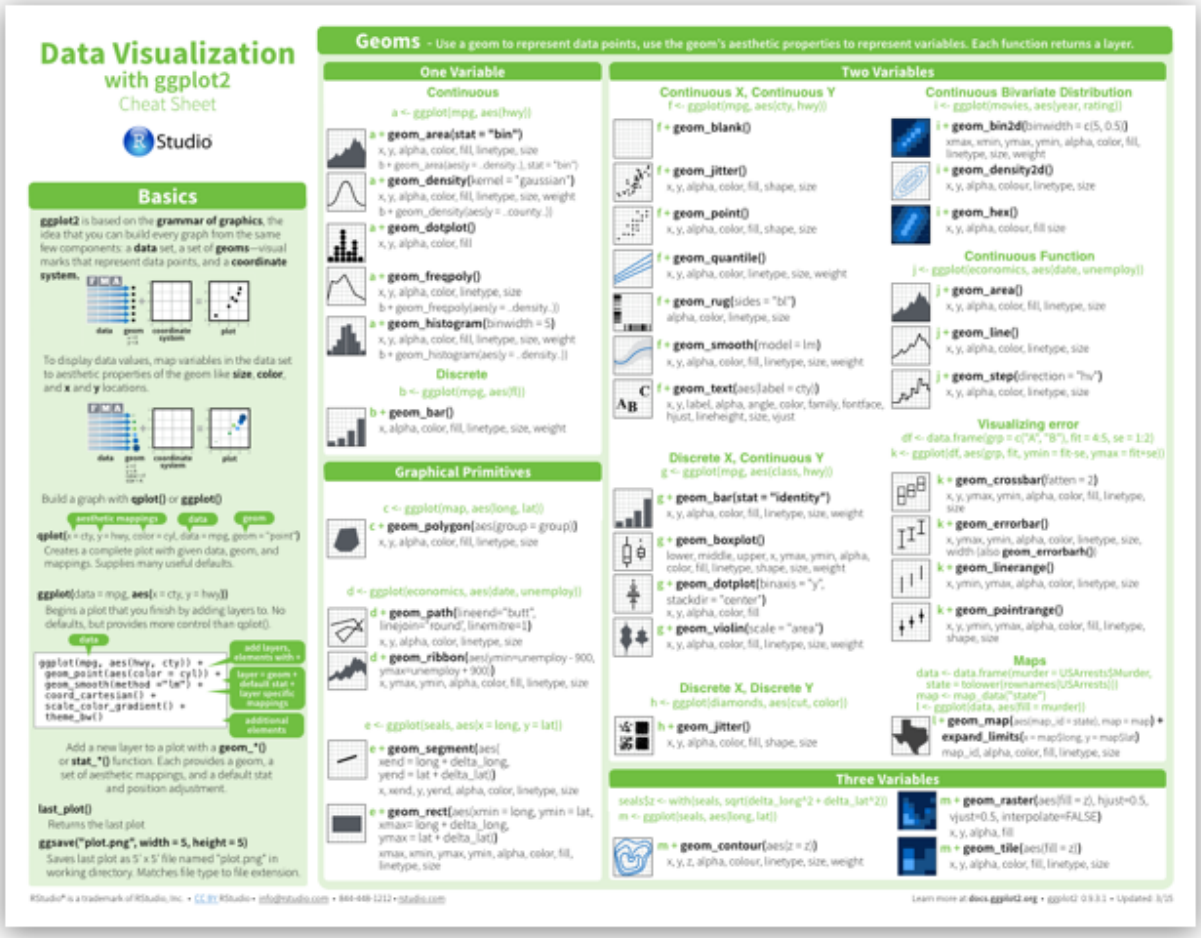


Figure 8: Cheatsheet

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