

R für die Sozialwissenschaften - Teil 2

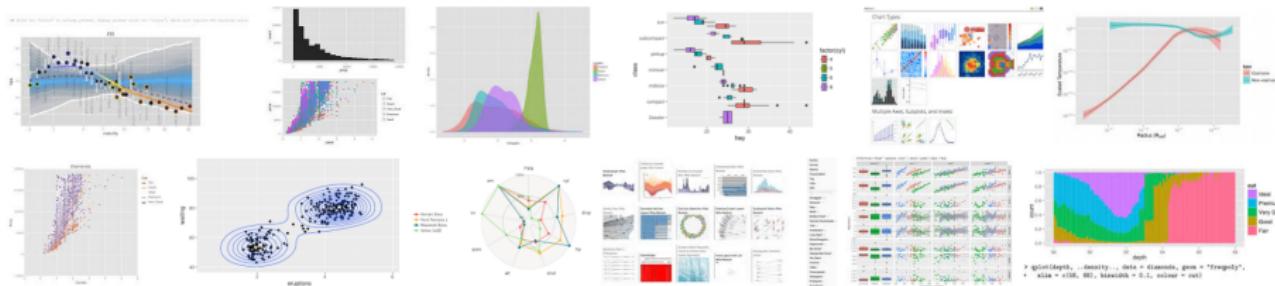
Jan-Philipp Kolb

23 Juni 2017

ggplot und ggmap

Das Paket ggplot2

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



Das Paket `ggplot2` installieren und laden

- Basiseinführung `ggplot2`

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

```
library(ggplot2)
```

Der diamonds Datensatz

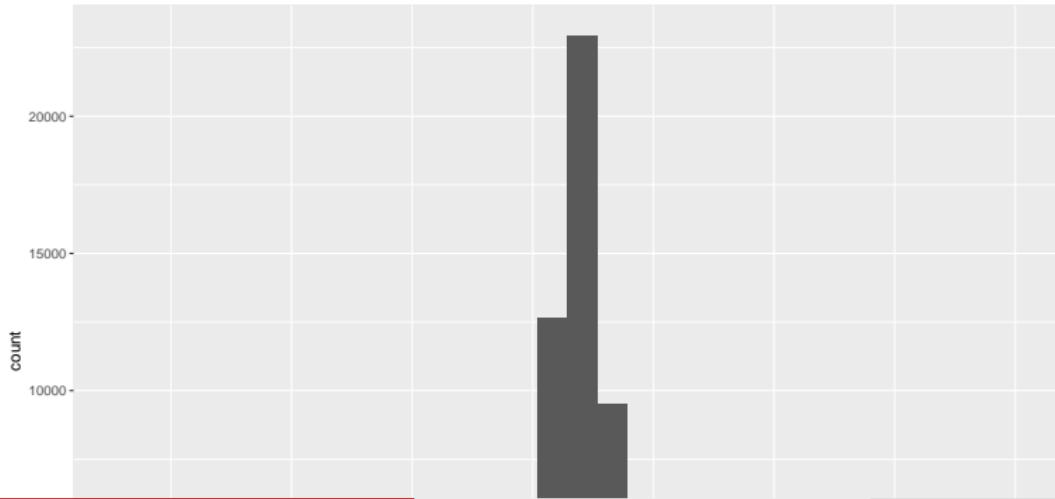
`head(diamonds)`

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

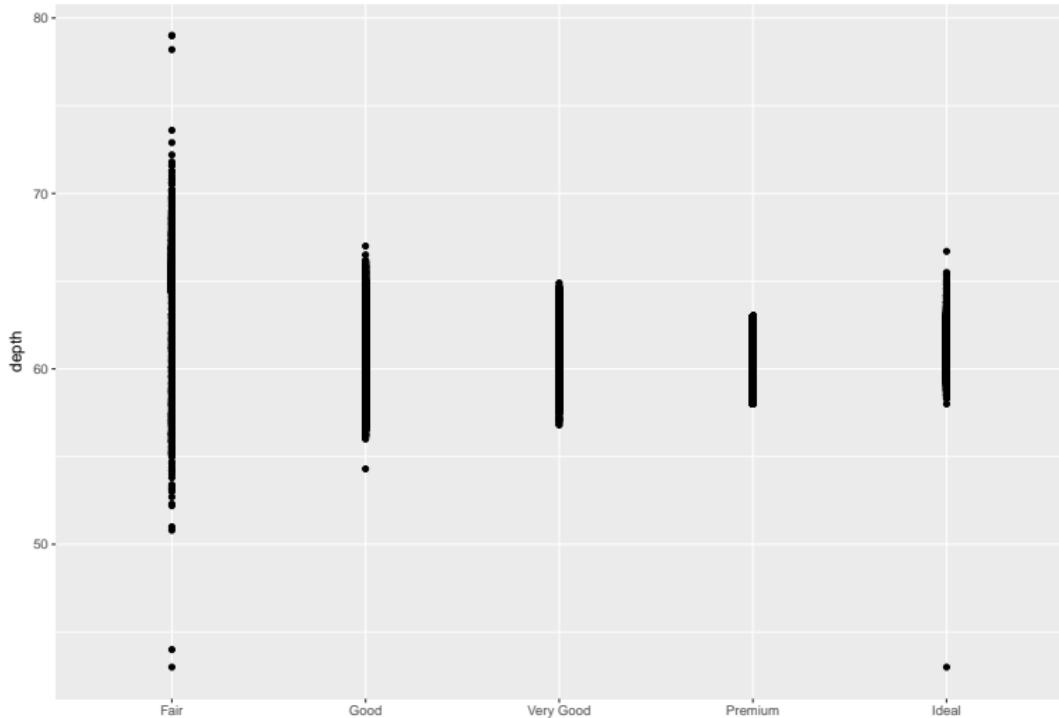
- Die Funktion `qplot` wird für schnelle Graphiken verwendet (quick plots)
- bei der Funktion `ggplot` kann man alles bis ins Detail kontrollieren

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



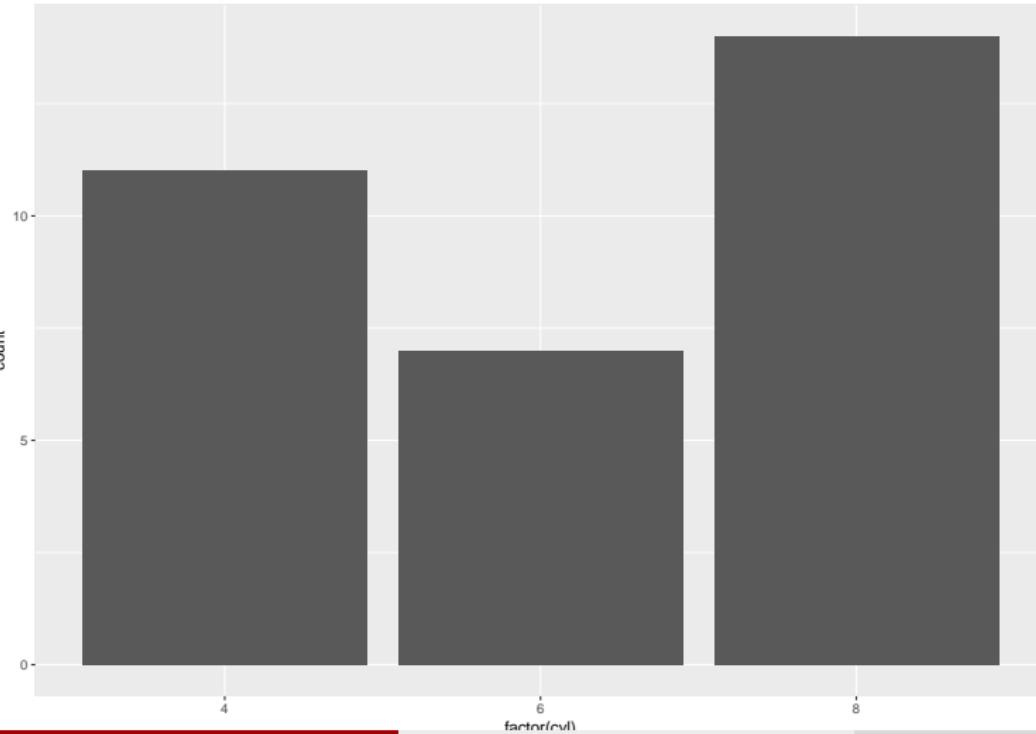
Ein Balkendiagramm

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



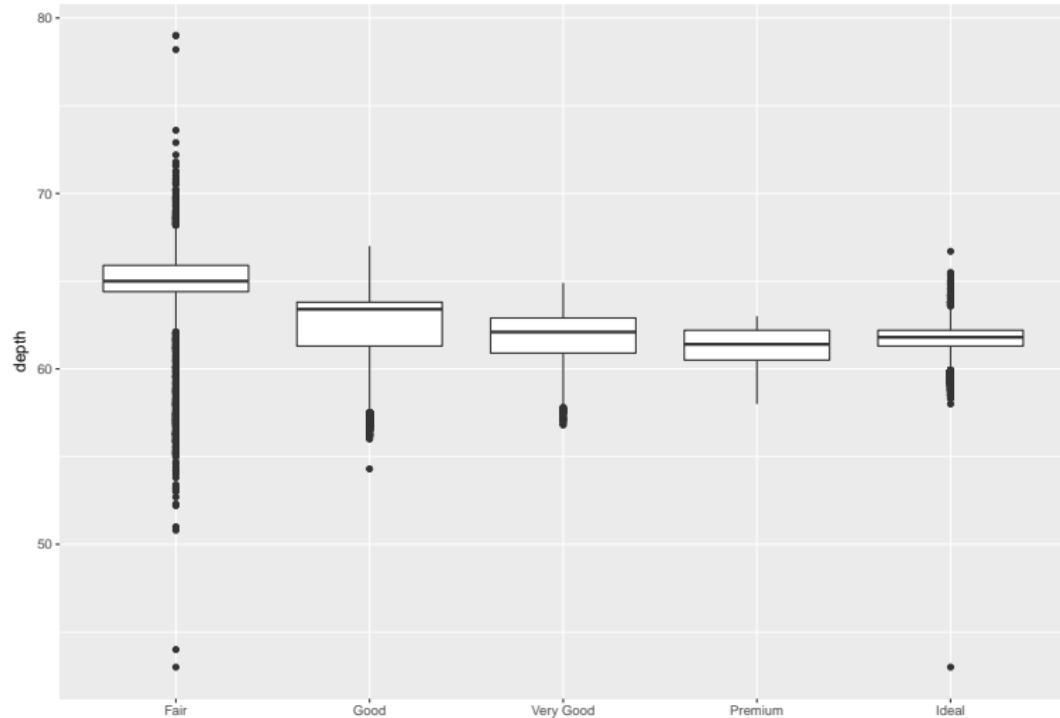
Ein weiteres Balkendiagramm

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



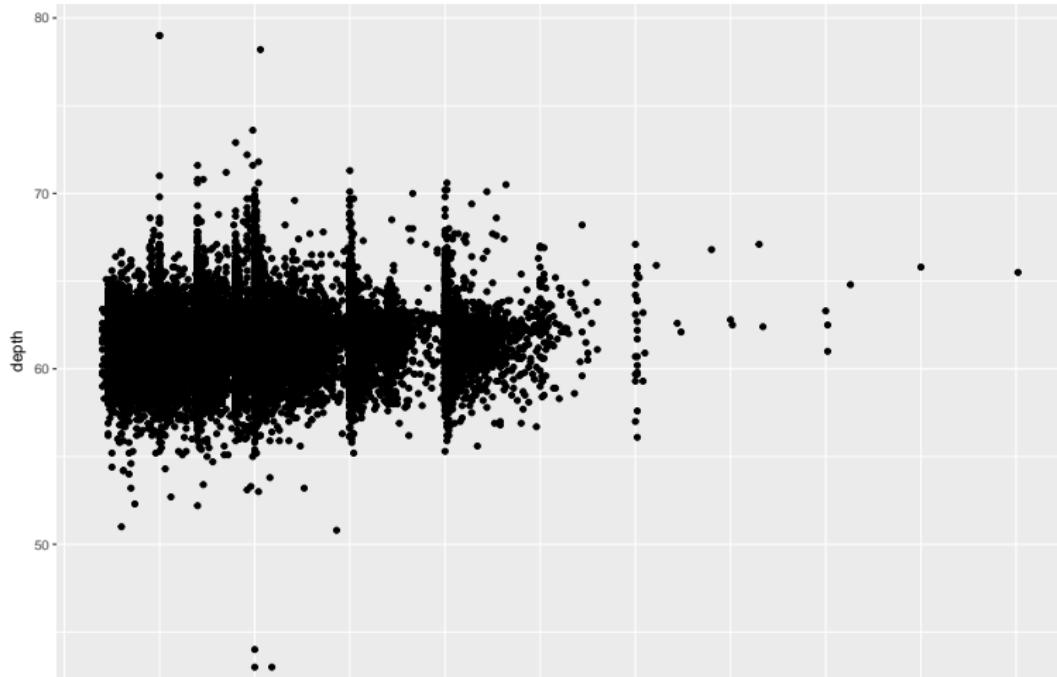
Boxplot

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



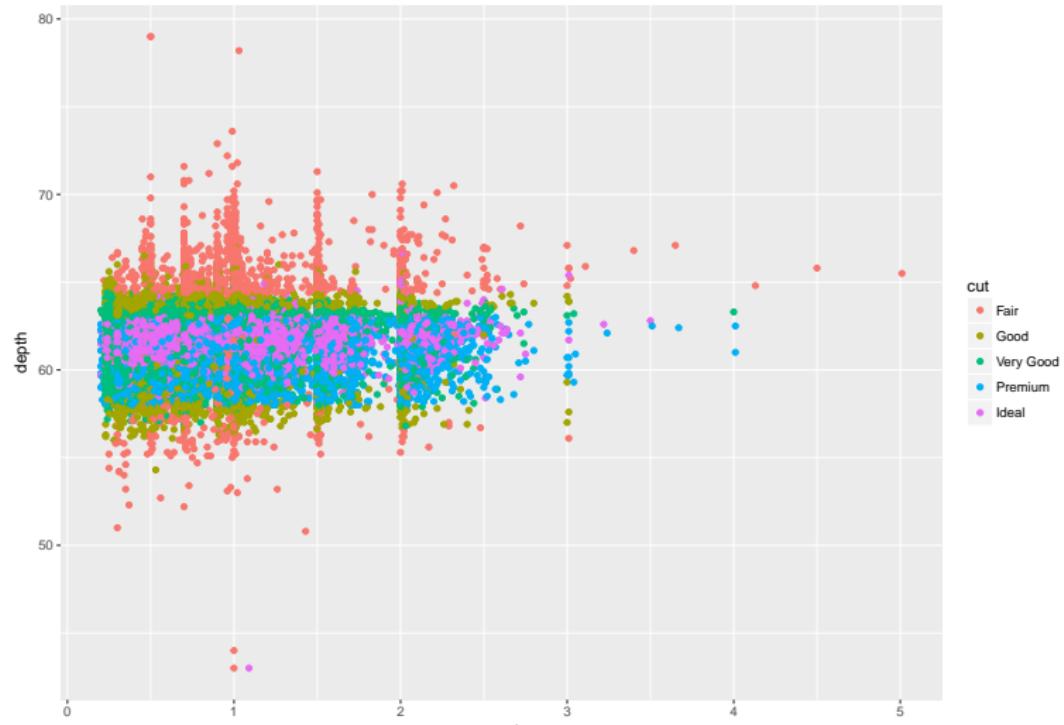
Scatterplot

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



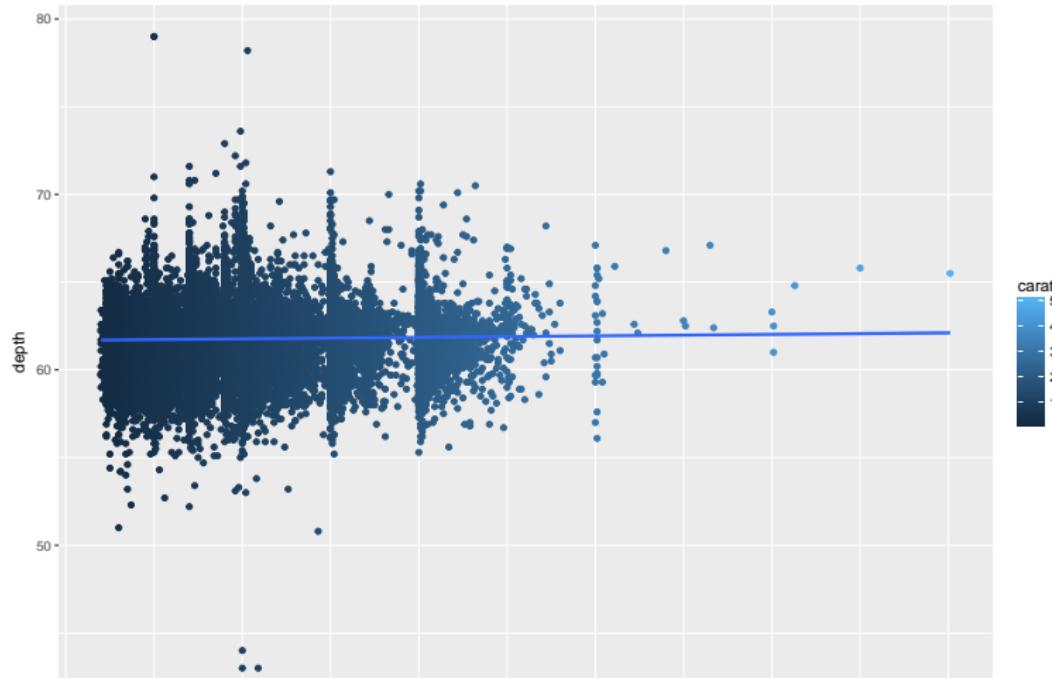
Farbe hinzufügen:

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



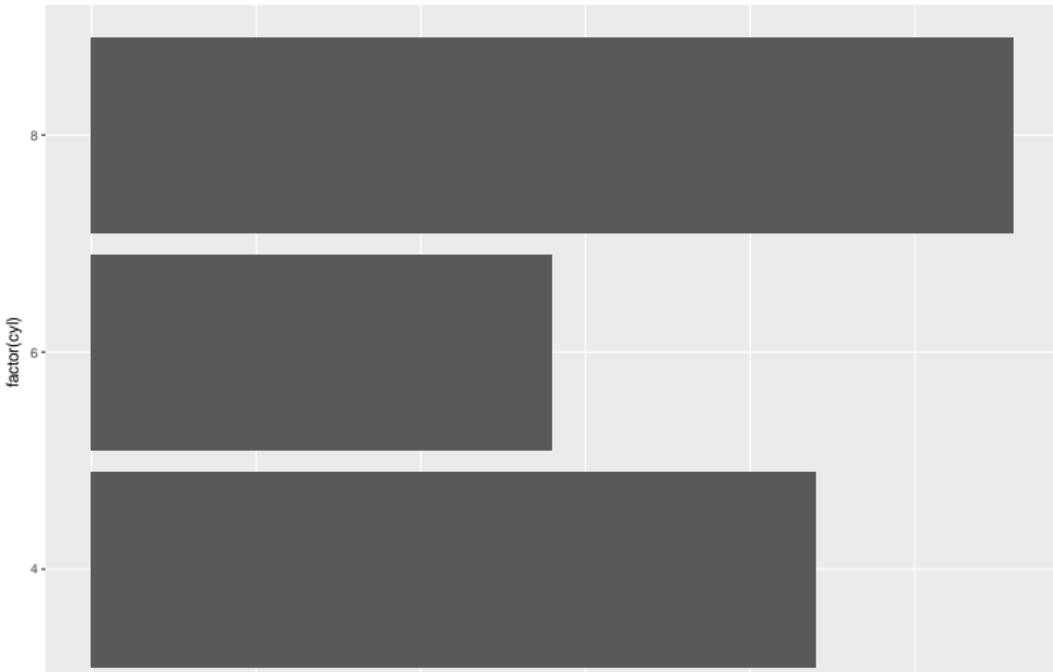
Trendlinie hinzufügen

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



Graphik drehen

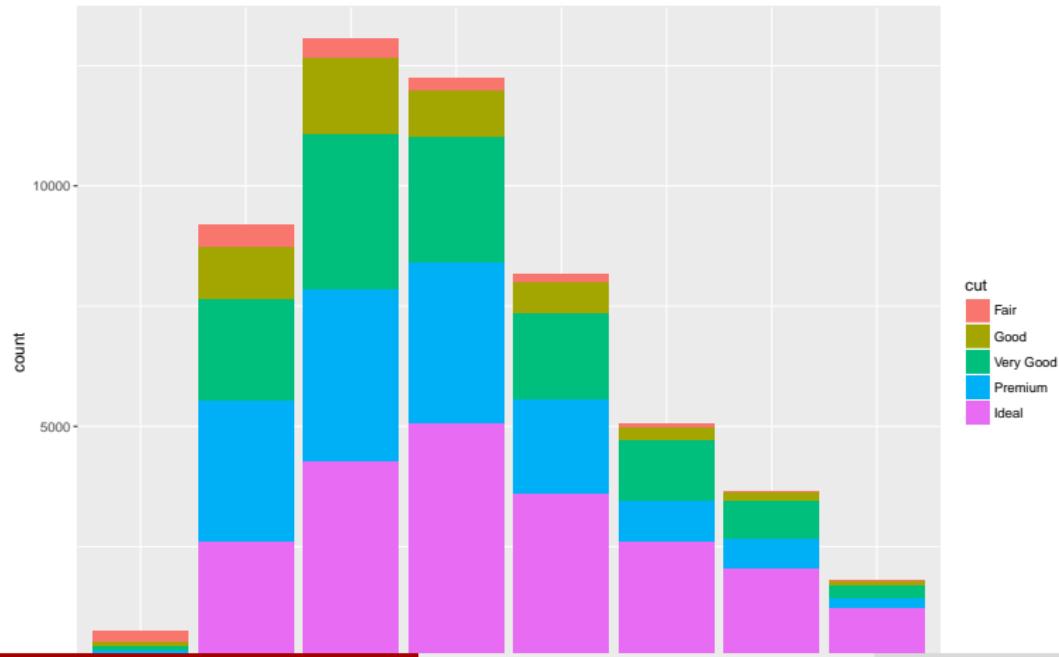
```
qplot(factor(cyl), data=mtcars, geom="bar") +  
coord_flip()
```



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

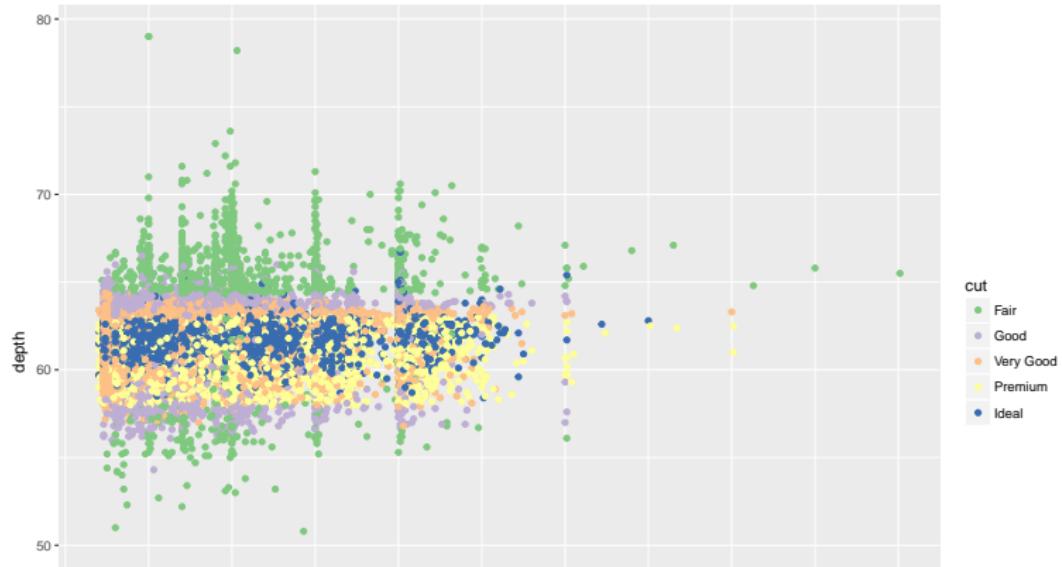
```
install.packages("RColorBrewer")

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

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

- Einführung in ggplot2

Verschiedene Kartentypen

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:

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")
```

Das Beispiel kann man direkt in die Konsole kopieren:

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

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>)



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

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")
```

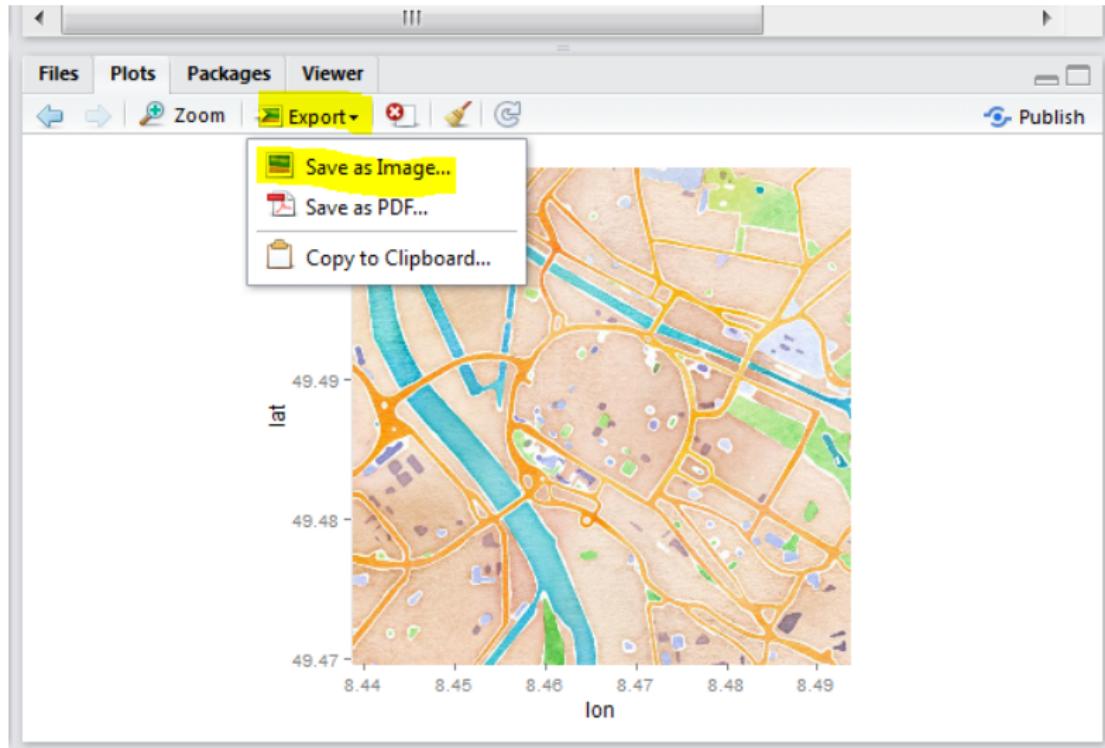
ggmap - maptype toner-hybrid

```
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

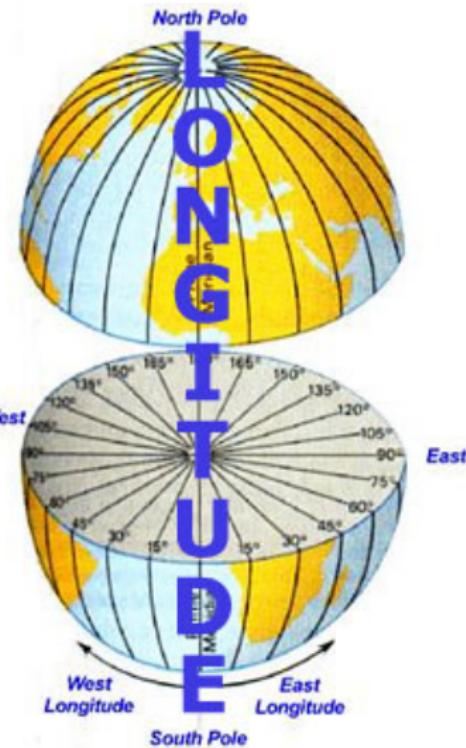
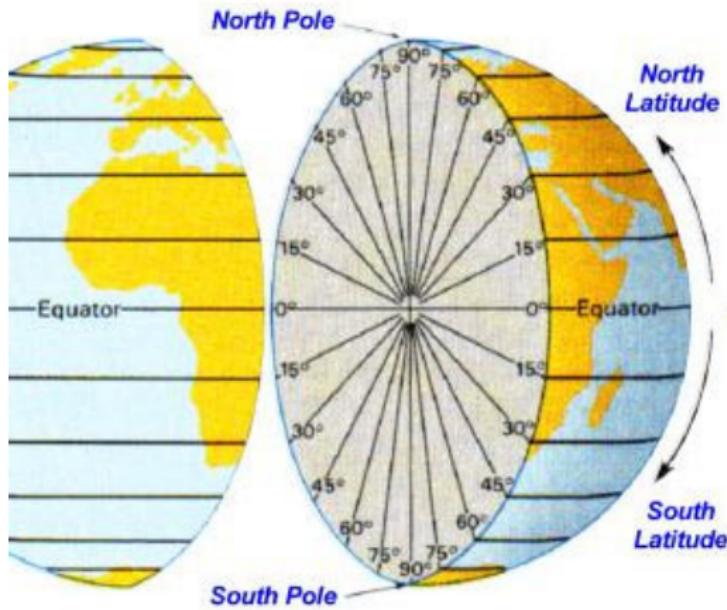
Geocoding (. . .) uses a description of a location, most typically a postal address or place name, to find geographic coordinates from spatial reference data . . .

Wikipedia - Geocoding

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

Latitude und Longitude

LATITUDE



Koordinaten verschiedener Orte in Deutschland

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))
```

Die Distanz zwischen zwei Punkten

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

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

Eine andere Distanz bekommen

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

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
```

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/>

Data Visualization with ggplot2

Cheat Sheet

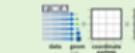


Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a set of **geom**-visual marks 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** & **y** locations.



Build a graph with **ggplot()** or **ggplot2()**

problem mapping data
solution mapping data
geom
ggplot(mapping = c(x,y), data = my_data, aes(x=x, y=y))
Creates a complete plot with given data, geom, and mapping. Supports many visual details.

ggplot2(data = mpg, aes(~mpg, ~hwy))
Begin a plot that you finish by adding layers to. No defaults, but provides more control than ggplot().

ggplot(mpg, aes(hwy, cyl)) +
 geom_point() +
 geom_smooth(method = "lm") +
 geom_text(label = "Slope = 0.001",
 stat = "text", size = 10)
 # additional styling

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

last_plot()
Returns the last plot
ggplot("plot.png", width = 5, height = 5)

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(hwy))

a + geom_area(stat = "bin")

x, y, alpha, color, fill, linetype, size

a + geom_density()

x, y, alpha, color, fill

a + geom_freqpoly()

x, y, alpha, color, linetype, size

b <- ggplot(mpg, aes(hwy))

b + geom_histogram(binwidth = 5)

x, y, alpha, color, fill, linetype, size, weight

b + geom_hex()

x, y, alpha, color, fill

b + geom_lattice()

x, y, alpha, color, fill, shape, size

b + geom_dotplot()

x, y, alpha, color, fill, count, density

b + geom_hexbin()

x, y, alpha, color, fill, shape, size, weight

b + geom_hex()

x, y, alpha, color, fill, size

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

b + geom_hex()

x, y, alpha, color, fill, size, weight

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:

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

Die lineare Regression

Die lineare Regression

Maindonald - Data Analysis

- Einführung in R
- Datenanalyse
- Statistische Modelle
- Inferenzkonzepte
- Regression mit einem Prädiktor
- Multiple lineare Regression
- Ausweitung des linearen Modells
- ...

Lineare Regression in R - Beispieldatensatz

John H. Maindonald and W. John Braun

DAAG - Data Analysis and Graphics Data and Functions

```
install.packages("DAAG")
```

```
library("DAAG")
data(roller)
```

help on roller data:

```
?roller
```

Das lineare Regressionsmodell in R

Schätzen eines Regressionsmodells:

```
roller.lm <- lm(depression ~ weight, data = roller)
```

So bekommt man die Schätzwerte:

```
summary(roller.lm)
```

```
##  
## Call:  
## lm(formula = depression ~ weight, data = roller)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -8.180 -5.580 -1.346  5.920  8.020  
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)  
##
```

Summary des Modells

```
summary(roller.lm)

##
## Call:
## lm(formula = depression ~ weight, data = roller)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.180  -5.580  -1.346   5.920   8.020
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.0871     4.7543  -0.439  0.67227
## weight       2.6667     0.7002   3.808  0.00518 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
```

R arbeitet mit Objekten

- `roller.lm` ist nun ein spezielles Regressions-Objekt
- Auf dieses Objekt können nun verschiedene Funktionen angewendet werden

```
predict(roller.lm) # Vorhersage
```

```
##          1          2          3          4          5          6
## 2.979669 6.179765 6.713114 10.713233 12.046606 14.180002
##          8          9         10
## 18.180121 24.046962 30.980502
```

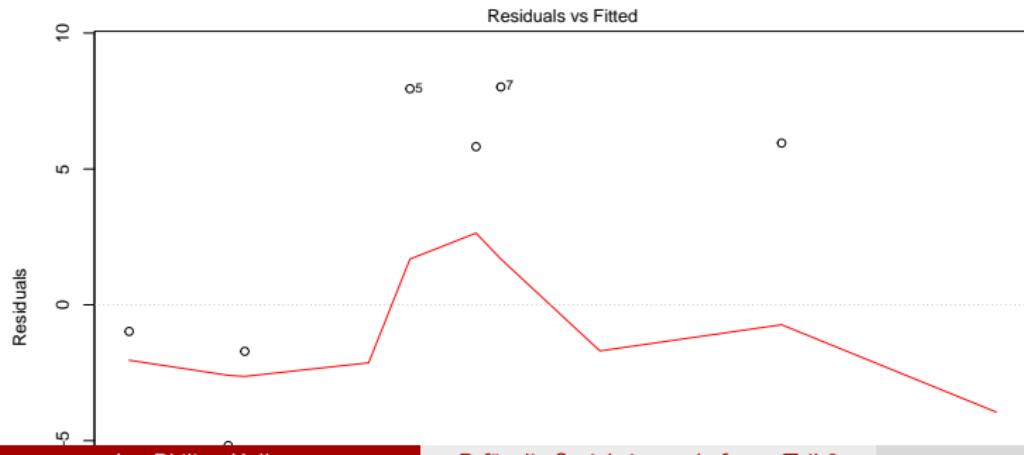
```
resid(roller.lm) # Residuen
```

```
##          1          2          3          4          5          6
## -0.9796695 -5.1797646 -1.7131138 -5.7132327 7.9533944 5.8
##          7          8          9         10
## 8.0199738 -8.1801213 5.9530377 -5.9805017
```

Residuenplot

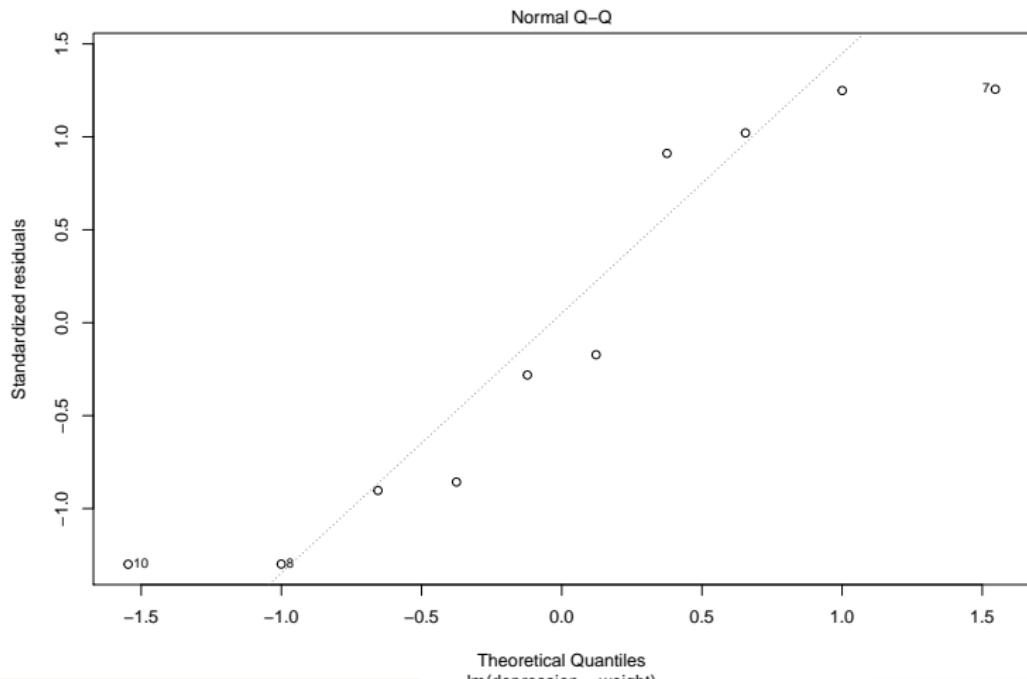
- Sind Annahmen des linearen Regressionsmodells verletzt?
- Dies ist der Fall, wenn ein Muster abweichend von einer Linie zu erkennen ist.
- Hier ist der Datensatz sehr klein

```
plot(roller.lm, 1)
```



Residuenplot

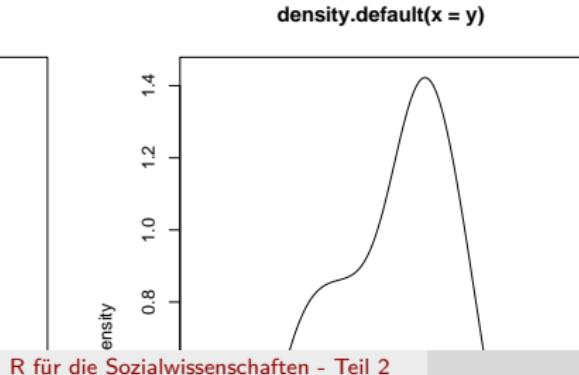
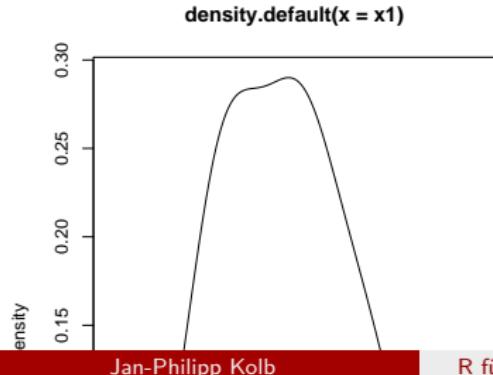
```
plot(roller.lm, 2)
```



Regressionsdiagnostik mit Basis-R

Ein einfaches Modell

```
N <- 5  
x1 <- rnorm(N)  
y <- runif(N)  
  
par(mfrow=c(1,2))  
plot(density(x1))  
plot(density(y))
```



Modellvorhersage machen

```
mod1 <- lm(y~x1)
pre <- predict(mod1)
y

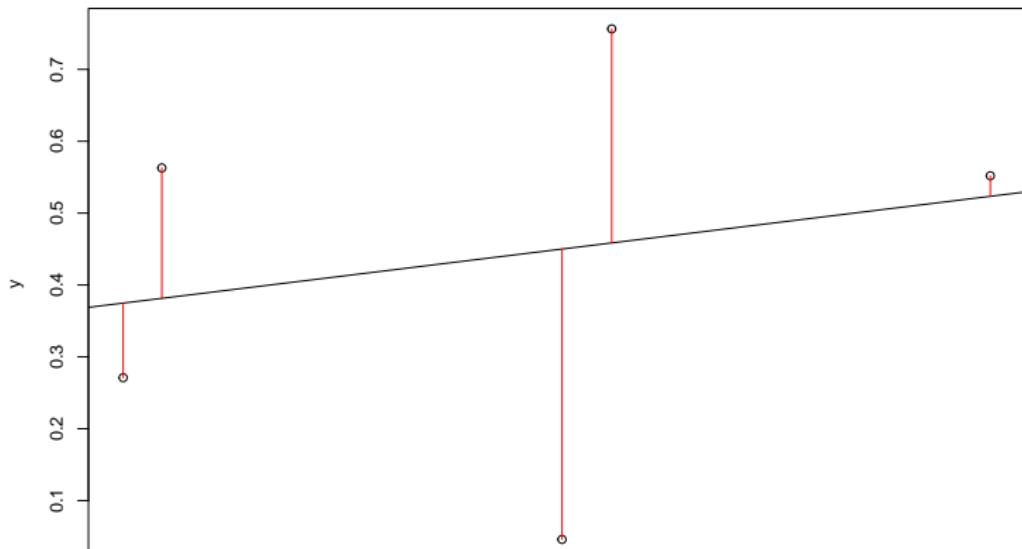
## [1] 0.2710050 0.7564149 0.0460009 0.5628672 0.5519674

pre

##          1           2           3           4           5
## 0.3748280 0.4585398 0.4500227 0.3814526 0.5234123
```

Regressionsdiagnostik mit Basis-R

```
plot(x1,y)
abline(mod1)
segments(x1, y, x1, pre, col="red")
```



Beispieldaten Luftqualität

```
library(datasets)
?airquality
```

```
airquality {datasets}
```

New York Air Quality Measurements

Description

Daily air quality measurements in New York, May to September 1973.

Usage

```
airquality
```

Format

A data frame with 154 observations on 6 variables.

```
[,1] Ozone    numeric Ozone (ppb)
[,2] Solar.R  numeric Solar R (lang)
[,3] Wind     numeric Wind (mph)
[,4] Temp     numeric Temperature (degrees F)
[,5:12] Month   numeric Month (1-12)
```

Das visreg-Paket

Ein Modell wird auf dem airquality Datensatz geschätzt

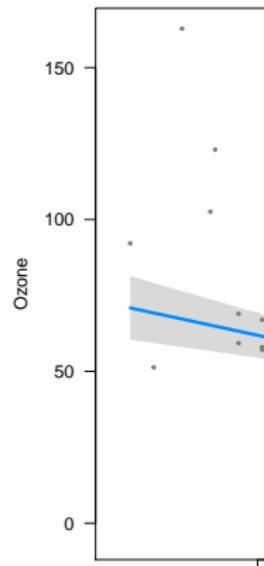
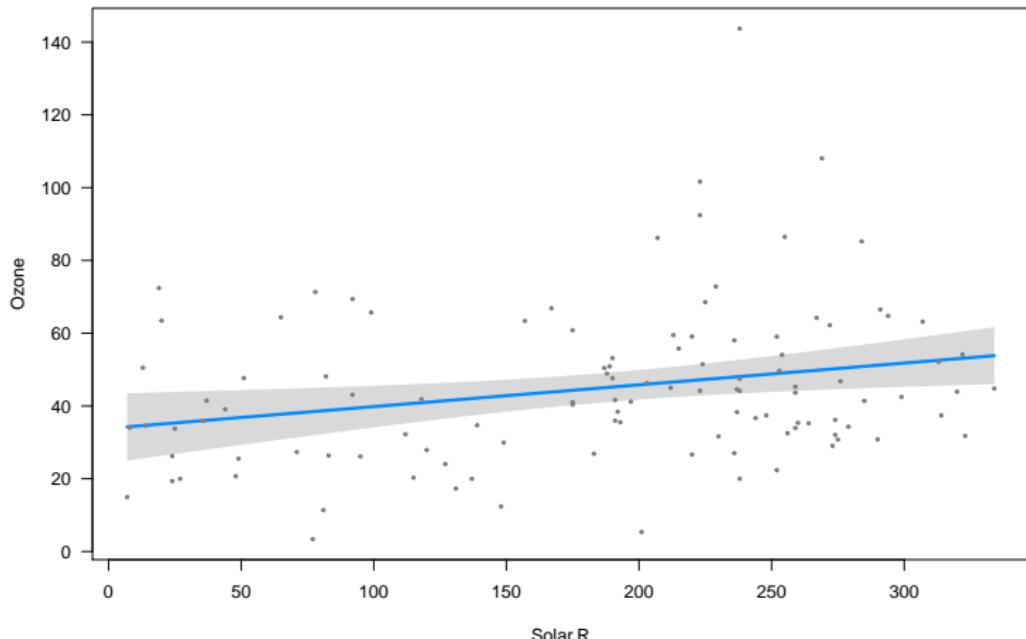
```
install.packages("visreg")

library(visreg)
fit <- lm(Ozone ~ Solar.R + Wind + Temp, data = airquality)
summary(fit)

##
## Call:
## lm(formula = Ozone ~ Solar.R + Wind + Temp, data = airquality)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -40.485 -14.219 - 3.551  10.097  95.619 
##
## Coefficients:
```

Visualisierung

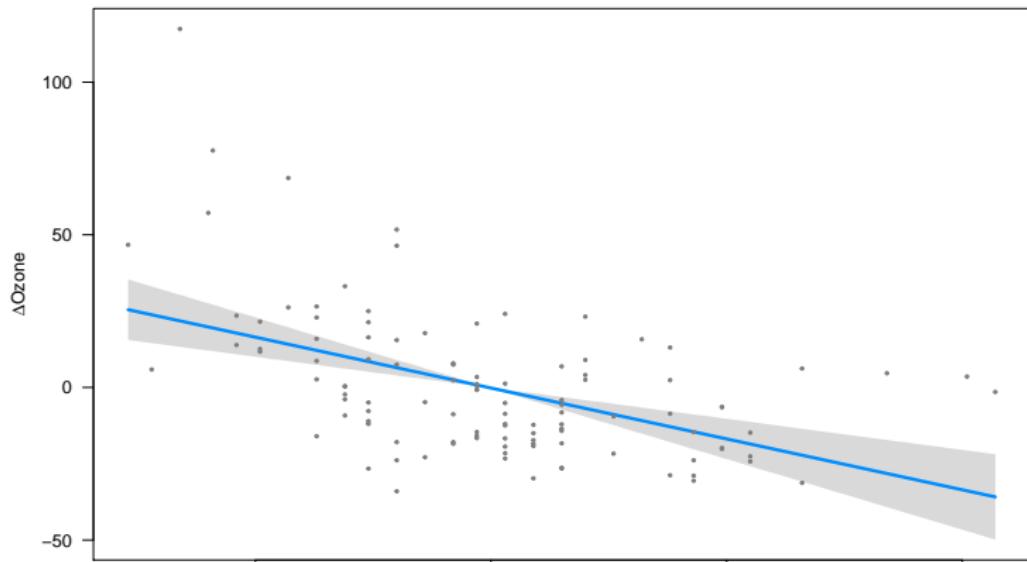
`visreg(fit)`



Und dann mit visreg visualisiert.

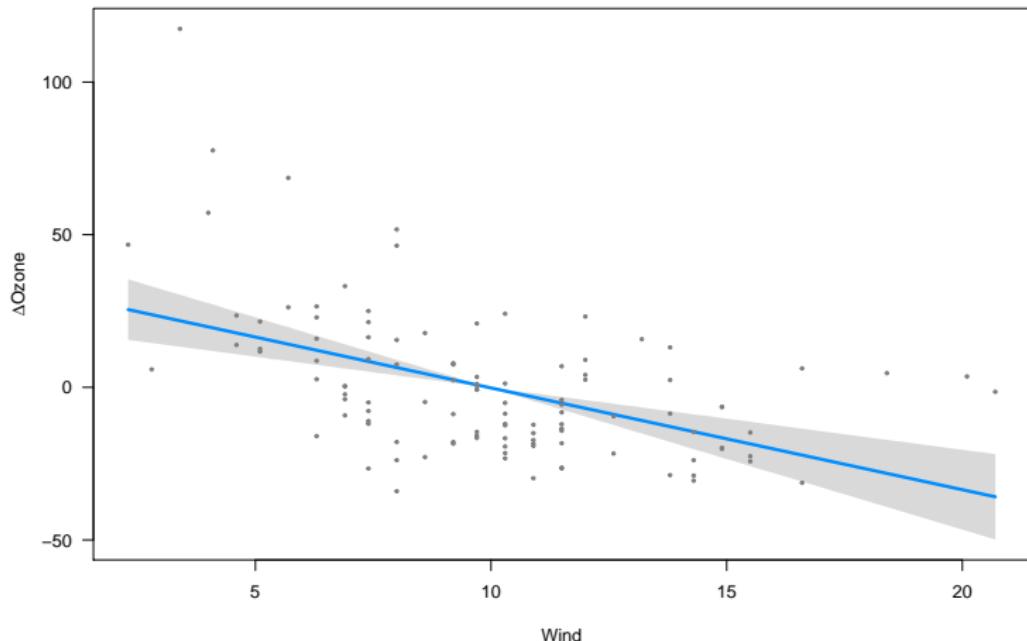
- Zweites Argument - Spezifikation erklärende Variable für Visualisierung

```
visreg(fit, "Wind", type = "contrast")
```



Visualisierung mit dem Paket visreg

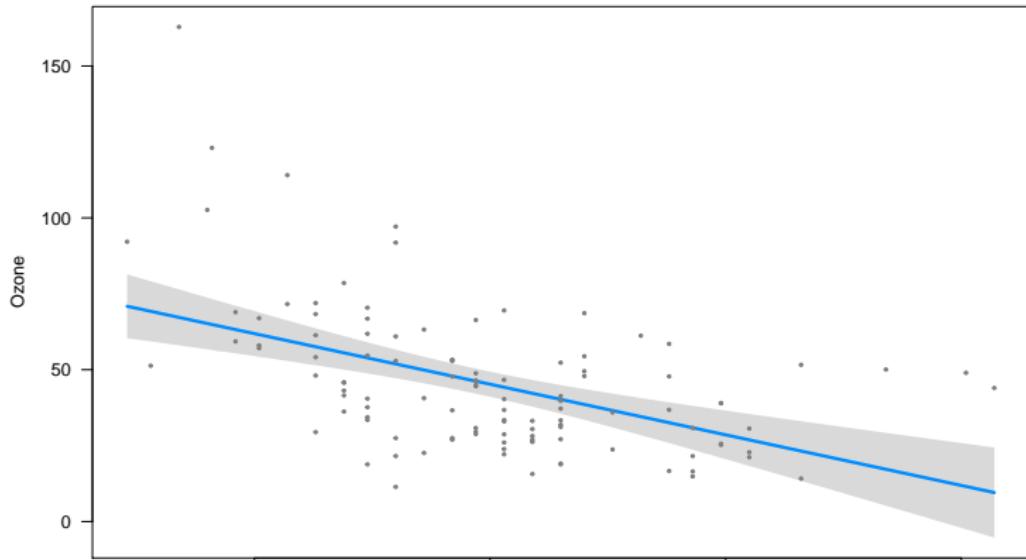
```
visreg(fit, "Wind", type = "contrast")
```



Das visreg-Paket

- Das Default-Argument für type ist conditional.

```
visreg(fit, "Wind", type = "conditional")
```



Regression mit Faktoren

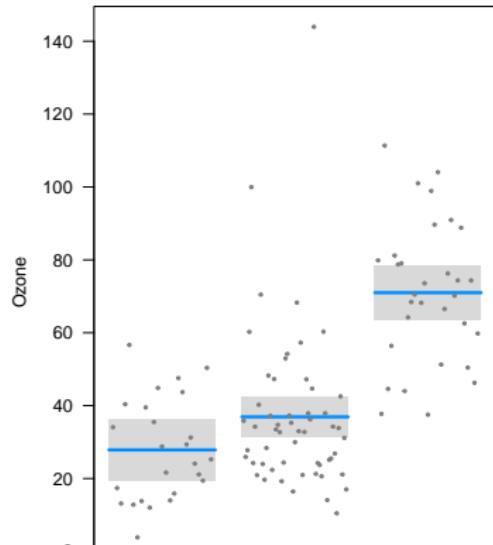
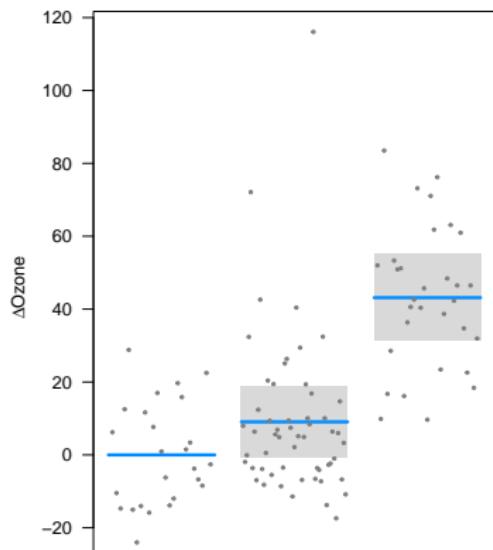
Mit visreg können die Effekte bei Faktoren visualisiert werden.

```
airquality$Heat <- cut(airquality$Temp, 3,
  labels=c("Cool", "Mild", "Hot"))
fit.heat <- lm(Ozone ~ Solar.R + Wind + Heat,
  data = airquality)
summary(fit.heat)

##
## Call:
## lm(formula = Ozone ~ Solar.R + Wind + Heat, data = airquality)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -33.473 -12.794  -2.686   8.461 107.035
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.6791   11.5340  3.000 ** 0.0032 **
## Solar.R     0.1239    0.0794  1.550   0.1254
## Wind        0.0083    0.0083  1.000   0.3157
## Heat        0.0000    0.0000  0.000   1.0000
```

Effekte von Faktoren

```
par(mfrow=c(1,2))
visreg(fit.heat, "Heat", type = "contrast")
visreg(fit.heat, "Heat", type = "conditional")
```



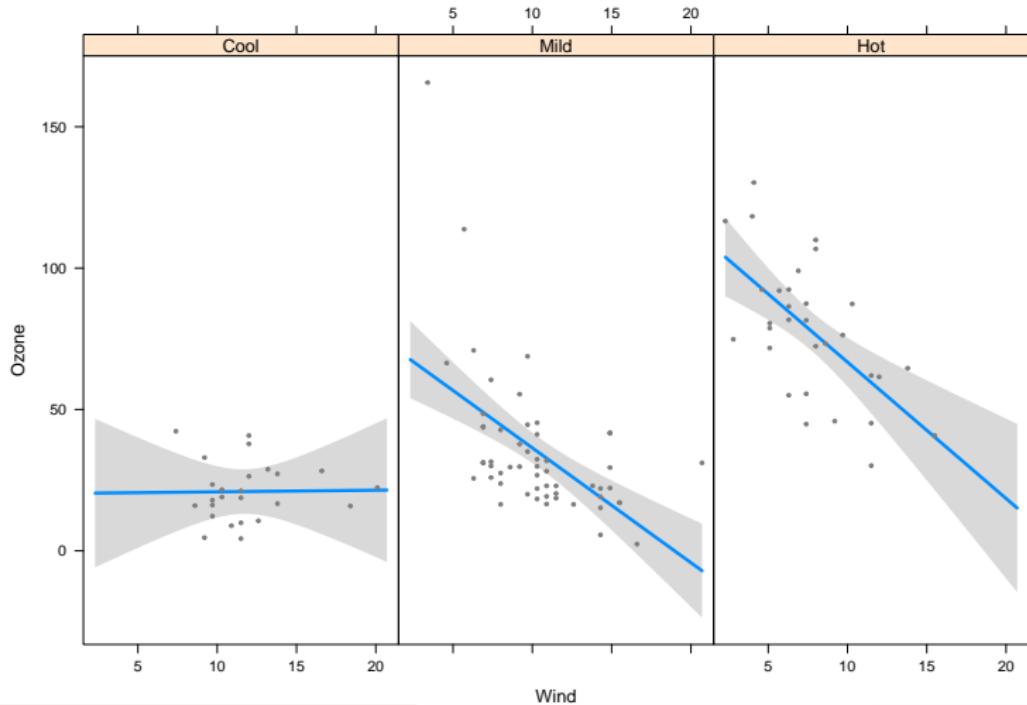
Das Paket visreg - Interaktionen

```
airquality$Heat <- cut(airquality$Temp, 3,
labels=c("Cool", "Mild", "Hot"))
fit <- lm(Ozone ~ Solar.R + Wind * Heat, data = airquality)
summary(fit)

##
## Call:
## lm(formula = Ozone ~ Solar.R + Wind * Heat, data = airquality)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -34.472 -11.640  -1.919   7.403 102.428
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.48042   17.38219   0.258 0.797102
```

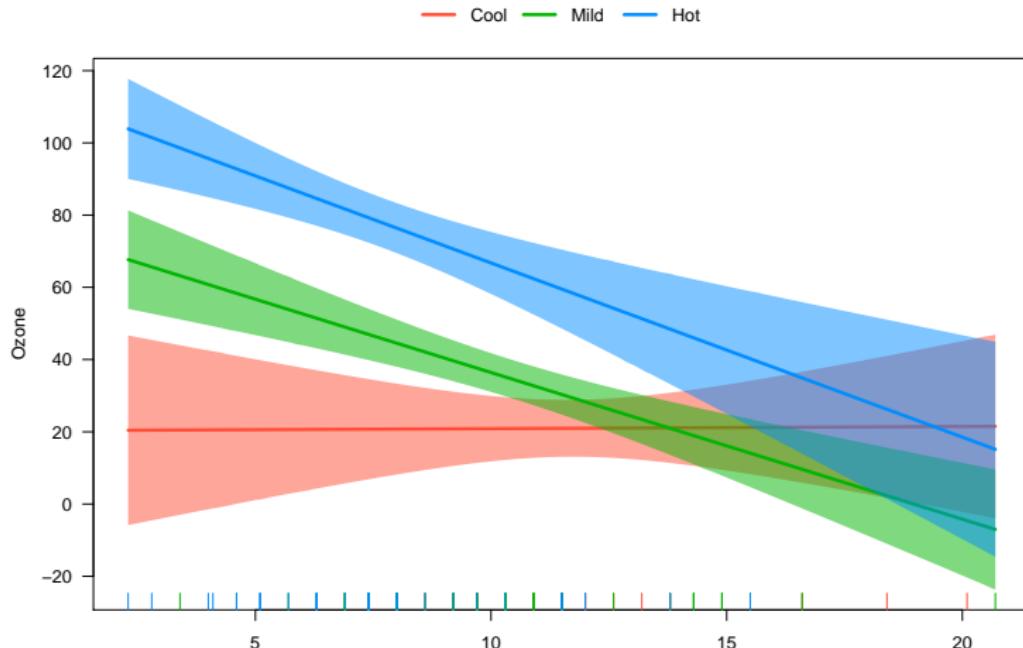
Steuern der Graphikausgabe mittels layout

```
visreg(fit, "Wind", by = "Heat", layout=c(3,1))
```



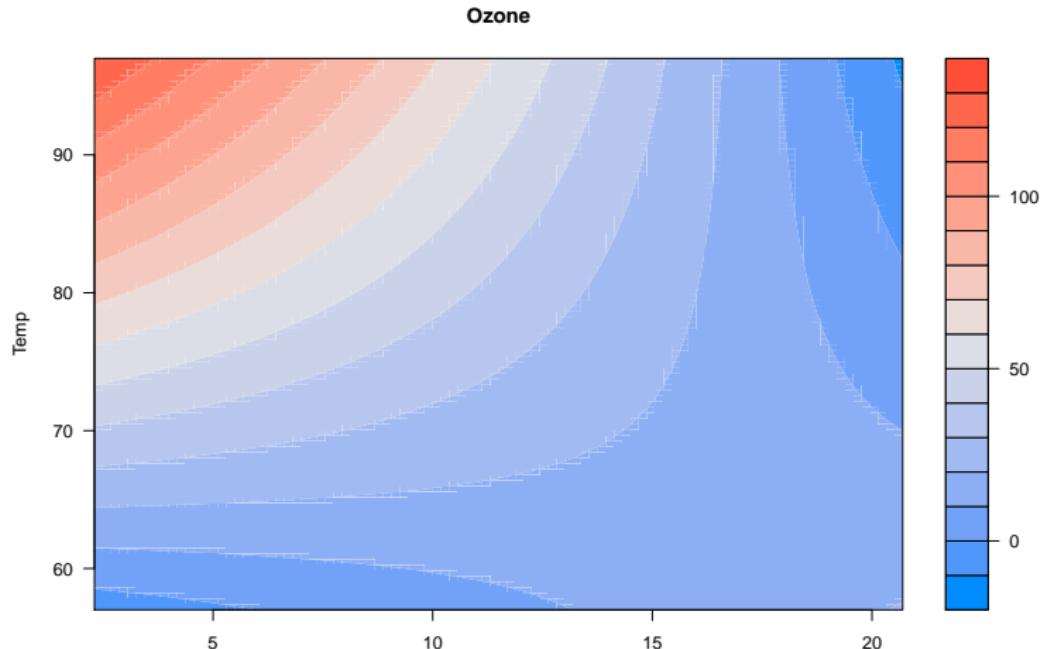
Das Paket visreg - Interaktionen overlay

```
fit <- lm(Ozone ~ Solar.R + Wind * Heat, data = airquality)
visreg(fit, "Wind", by="Heat", overlay=TRUE, partial=FALSE)
```



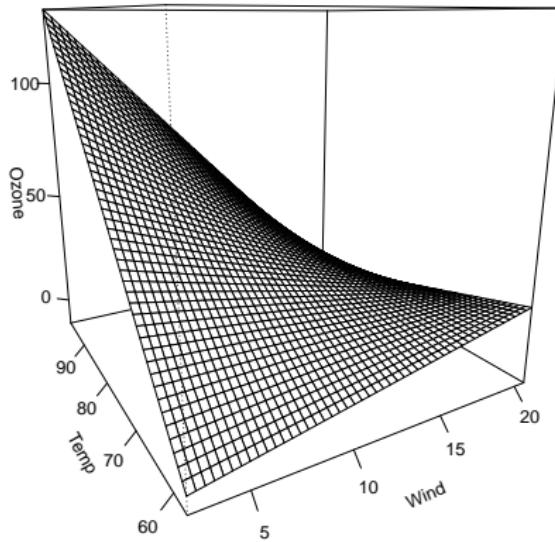
Das Paket visreg - visreg2d

```
fit2 <- lm(Ozone ~ Solar.R + Wind * Temp, data = airquality)
visreg2d(fit2, "Wind", "Temp", plot.type = "image")
```



Das Paket visreg - surface

```
visreg2d(fit2, "Wind", "Temp", plot.type = "persp")
```



Linkliste - lineare Regression

- Regression - r-bloggers
- Das Komplette Buch von Faraway- sehr intuitiv geschrieben.
- Gute Einführung auf Quick-R
- Multiple Regression
- Basis Regression - How to go about interpreting regression coefficients

Die logistische Regression



- Sehr intuitiv geschriebenes Buch
- Sehr ausführliches begleitendes Skript von Thompson
- Das Skript eignet sich um die kategoriale Datenanalyse nachzuvollziehen

Texts in Statistical Science

Linear Models with R

- Logistische Regressionen gut erklärt
- Beispiele mit R-code
 - Faraway - Extending the linear model with r
 - Faraway - Practical Regression and Anova using R

Binäre AVs mit `glm`

- Die logistische Regression gehört zur Klasse der generalisierten linearen Modelle (GLM)
- Die Funktion zur Schätzung eines Modells dieser Klasse in heißt `glm()`
- `glm()` muss 1. ein Formel-Objekt mitgegeben werden und 2. die Klasse (binomial, gaussian, Gamma) samt link-Funktion (logit, probit, cauchit, log, cloglog)

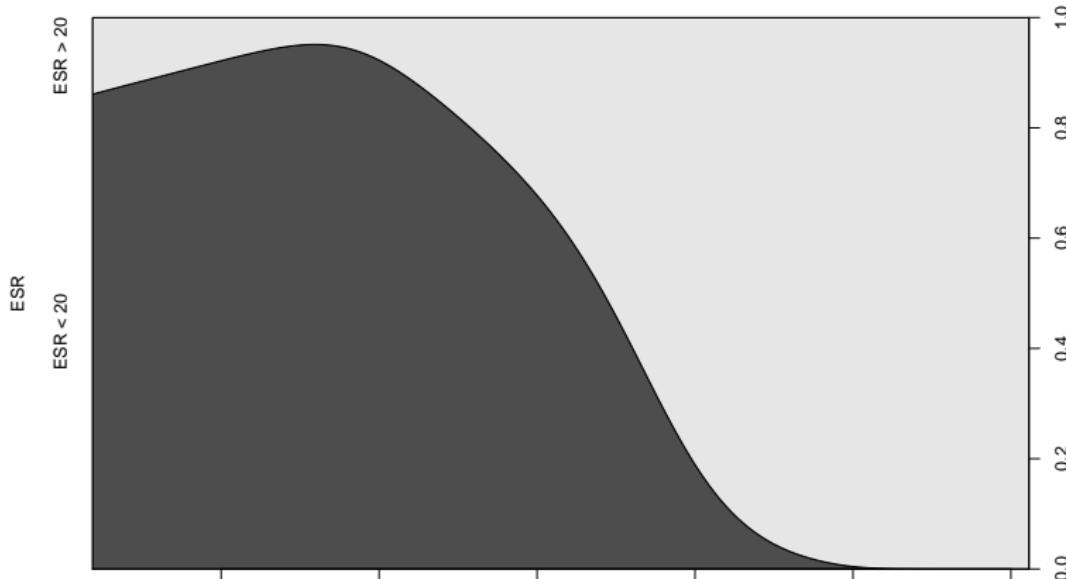
Beispieldaten für die logistische Regression

```
install.packages("HSAUR")  
  
library("HSAUR")  
data("plasma", package = "HSAUR")
```

Logistische Regression mit R

- Kategoriale Daten:

```
cdplot(ESR ~ fibrinogen, data = plasma)
```



Logistische Regression mit R

```
plasma_glm_1 <- glm(ESR ~ fibrinogen, data = plasma,  
                      family = binomial())
```

Weitere Beispieldaten

```
install.packages("faraway")
```

```
library("faraway")
```

```
data(orings)
```

temp	damage
53	5
57	1
58	1

Generalisierte Regression mit R - weitere Funktionen

- Logistisches Modell mit Probit-Link:

```
probitmod <- glm(cbind(damage, 6-damage) ~ temp,  
                  family=binomial(link=probit), orings)
```

- Regression mit Zähldaten:

```
modp <- glm(Species ~ ., family=poisson, gala)
```

- Proportional odds logistic regression im Paket MASS:

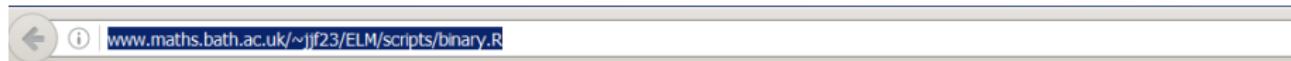
```
library("MASS")  
house.plr<-polr(Sat~Infl, weights=Freq, data=housing)
```

Linkliste - logistische Regression

- Einführung in logistische Regression



- Code zum Buch von Faraway



```
library(faraway)
data(orings)
plot(damage/6 ~ temp, orings, xlim=c(25,85), ylim = c(0,1), xlab="Temperature", ylab="Prob of damage")
lmod <- lm(damage/6 ~ temp, orings)
abline(lmod)
logitmod <- glm(cbind(damage,6-damage) ~ temp, family=binomial, orings)
summary(logitmod)
plot(damage/6 ~ temp, orings, xlim=c(25,85), ylim = c(0,1), xlab="Temperature", ylab="Prob of damage")
```

Mehrebenenmodelle

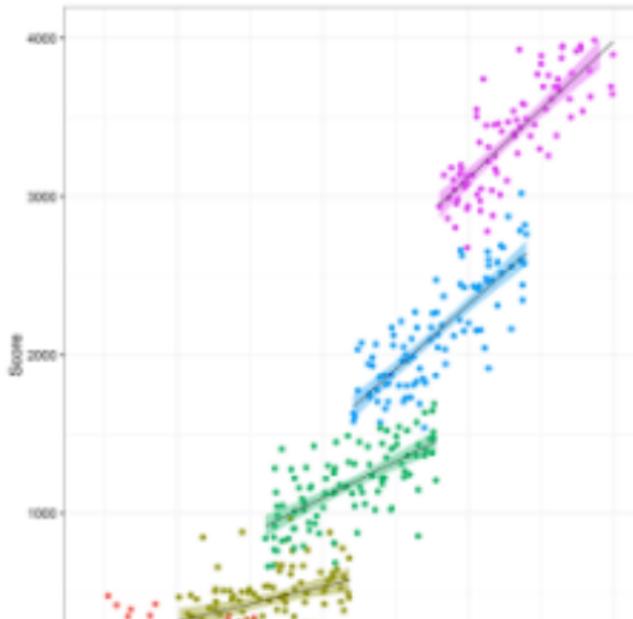
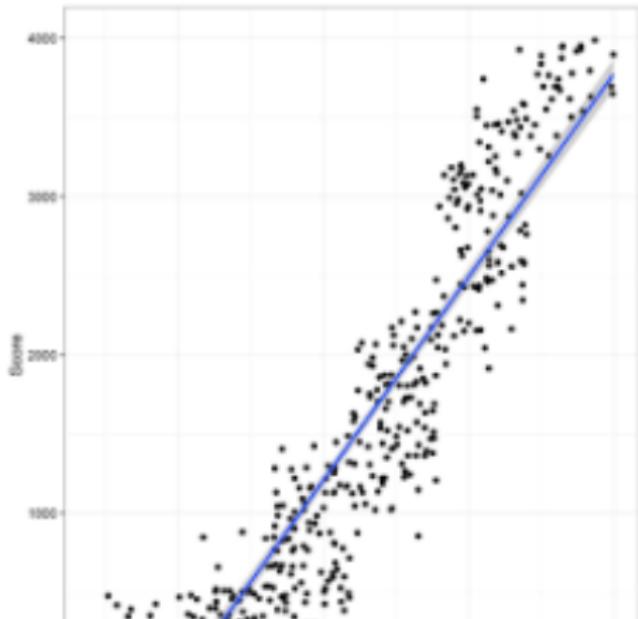
Wie sehen die Daten aus?

- Beispiel Mehrebenenstruktur der Daten



Andres Gutierrez - Multilevel Modeling of Educational Data using R

- Lineare Modelle erkennen den Cluster-Effekt aufgrund der Intraklassen Korrelation nicht



Beispiel Mehrebenenmodelle

Untersuchungsgegenstand

- Es sollen die Kenntnisse (Fähigkeiten) von Grundschülern in Mathematik gemessen werden.
- Dazu werden in einem Schulbezirk zunächst Schulen ausgewählt und anschließend Klassen.
- Innerhalb der Klassen soll schließlich jeweils eine Stichprobe gezogen und diese getestet werden.
- Geht man zunächst von einer zufälligen Auswahl von Klassen aus, dann ist die Level-1-Variation durch die Schüler und die Level-2-Variation durch die Klassen bestimmt.

Fragen hierzu

- Wie wäre die Auswahl der Schulen zu berücksichtigen?
- Wie kann zusätzlich eine Unterscheidung nach privaten und staatlichen Schulen in die Modellierung eingebracht werden?

Beispiel in Goldstein (2010), Kapitel 1.2

Evaluierung der Effektivität von Schulen

Mehrebenen-Modelle:

- Schüler
- Klassenverbände
- Schulamtsbezirke oder Bundesländer

Unterscheidung

- Modelle mit vielen Parametern, die wiederum modelliert werden können
- Regressionen mit Koeffizienten, die zwischen Gruppen variieren können

Bibliotheken

```
install.packages("lme4")
install.packages("sjPlot")

library(ggplot2)
library(gridExtra)
library(lme4)
library(sjPlot)
library(dplyr)
```

Beispieldaten

```
mlexdat <- read.csv("https://github.com/Japhilko/RSocialScienc
```

X	SES	Score	ID
1	18.62733	-55.120574	A
2	33.64915	-92.375273	A
3	22.26931	-48.783447	A
4	36.49052	38.099329	A
5	38.21402	339.701754	A
6	11.36669	2.286978	A

Formalistisch

- Bei der Analyse von Daten mit diesen hierarchischen Strukturen, sollte man immer zunächst ein Null-Modell anpassen
- Somit kann man die Variation erfassen, die auf die Schulen zurückzuführen ist:
- Das passende Modell sieht folgendermaßen aus:

Die Gesamtvariation wird in zwei Teile zerlegt:

- Variation zwischen Schülern (innerhalb der Schulen) und
- zwischen den Schulen (zwischen den Schulen).

Der R-code für dieses Nullmodell

- das einfachste Multilevel Modell
- nach dem vertikalen Strich wird die Gruppen Variable spezifiziert
- die Default Schätzmethode ist restricted maximum likelihood (REML)
- Man kann aber auch maximum likelihood Schätzung spezifizieren (ML)

```
HLM0 <- lmer(Score ~ (1 | ID), data = mlexdat)
```

Nullmodell Ergebnis

```
coef(HLM0)
```

```
## $ID  
## (Intercept)  
## A      45.7893  
## B      430.7218  
## C     1182.1760  
## D     2145.2329  
## E     3489.1408  
##  
## attr(,"class")  
## [1] "coef.mer"
```

```
summary(HLM0)
```

```
## Linear mixed model fit by REML ['lmerMod']  
## Formula: Score ~ (1 | ID)
```

Interpretation des Nullmodells

- 96 Prozent Variation zwischen den Schulen
- 4 Prozent Variation innerhalb der Schulen

$100 * 87346 / (87346 + 1931757)$

[1] 4.32598

- Die Schätzung der zufälligen Effekte zeigt, dass die Variation zwischen den Schulen (Intraklassen Korrelation) fast 96 Prozent beträgt
- Während der Anteil der Variation zwischen den Studierenden nur etwas mehr als 4 Prozent ausmacht.
- Das Null-Modell behauptet also , dass Leistungsträger zu bestimmten Schulen gehen und Studierende mit geringerem Leistungsniveau nicht diese Schulen besuchen.
- Mit anderen Worten, die Schule bestimmt das Testergebnis.

Ein weiteres Modell

- Das Null Modell schließt keine erklärenden Variablen ein.
- Allerdings könnte der sozioökonomischen Status (SES) der Schüler auch eine Rolle spielen.
- Die folgenden Ausdrücke geben ein verfeinertes Modell mit zufälligen Achsenabschnitten und Steigung für jede der Schulen:

Rcode für dieses Modell

```
HLM1 <- lmer(Score ~ SES + (SES | ID), data = mlexdat)
coef(HLM1)

## $ID
##   (Intercept)      SES
## A    36.46401  0.3798185
## B    37.21549  9.7596237
## C    38.10719 20.8897245
## D    38.85566 30.2320132
## E    39.70159 40.7907386
##
## attr(,"class")
## [1] "coef.mer"

summary(HLM1)

## Linear mixed model fit by REML ['lmerMod']
```

```
# 1% - BS variance  
# 99% - WS variance  
100 * 40400.24 / (40400.24 + 257.09 + 1.65)  
  
## [1] 99.36363  
  
# Percentage of variation explained by SES between schools  
1 - ((257.09 + 1.65) / 1931757)  
  
## [1] 0.9998661  
  
# Percentage of variation explained by SES within schools  
1 - (40400.24 / 87346)  
  
## [1] 0.5374689
```

Auf der einen Seite stellen wir fest, dass die SES 99 Prozent der Unterschiede zwischen den Schulen erklärt; Auf der anderen Seite erklärt die SES 53 Prozent der Abweichungen innerhalb der Schulen.

Was heißt das? Schulsegregation

Multilevel Model Specification

```
library(lme4)
library(mlmRev)
names(Exam)

## [1] "school"    "normexam"   "schgend"    "schavg"     "vr"
## [7] "standLRT"  "sex"        "type"       "student"
```

random intercept, fixed predictor in individual level

- Für das nächste Modell fügen wir dem einzelnen Level einen Prädiktor hinzu.
- Wir tun dies, indem wir die '1' im Nullmodell durch den Prädiktor (hier: standLRT) ersetzen.
- Es wird immer ein Intercept angenommen, also wird es noch hier geschätzt.
- Es muss nur angegeben werden, wenn keine anderen Prädiktoren angegeben sind.
- Da wir nicht wollen, dass der Effekt des Prädiktors zwischen den Gruppen variiert, bleibt die Spezifikation des zufälligen Teils des Modells mit dem vorherigen Modell identisch.

```
lmer(normexam ~ standLRT + (1 | school), data=Exam)
```

random intercept, random slope

Das nächste Modell, das angegeben wird, ist ein Modell mit einem zufälligen Intercept auf individueller Ebene und ein Prädiktor, der zwischen Gruppen variieren darf. Mit anderen Worten, die Wirkung der Hausaufgaben auf die Partitur auf einem Mathe-Test variiert zwischen den Schulen. Um dieses Modell zu schätzen, wird das '1', das den Intercept im zufälligen Teil der Modellspezifikation angibt, durch die Variable ersetzt, von der wir den Effekt zwischen den Gruppen variieren wollen.

Varying intercept model

```
MLexamp.6 <- lmer(extro ~ open + agree + social + (1 | school))
```

Varying slope model

```
MLexamp.9 <- lmer(extro ~ open + agree + social + (1 + open |
```

Paket lmer

```
lmer(y ~ 1 + (1 | subjects), data=data)
# nlme
lme(y ~ 1, random = ~ 1 | subjects, data=data)
```

Links

- Uncertainty in parameter estimates using multilevel models
- Multilevel models with R
- Ein Beispieldatensatz
- Multilevel Modeling of Educational Data using R (Part 1)
- Vignette für lme4
- Mixed model guide