Intro Datenanalyse - zweiter Teil

Jan-Philipp Kolb

07 Mai, 2019

Einfache Graphiken erstellen

Ein Plot sagt mehr als 1000 Worte

- Grafisch gestützte Datenanalyse ist toll
- Gute Plots können zu einem besseren Verständnis beitragen
- Einen Plot zu generieren geht schnell
- Einen guten Plot zu machen kann sehr lange dauern
- Mit R Plots zu generieren macht Spaß
- Mit R erstellte Plots haben hohe Qualität
- Fast jeder Plottyp wird von R unterstützt
- R kennt eine große Menge an Exportformaten für Grafiken

Plot ist nicht gleich Plot

- Bereits das base Package bringt eine große Menge von Plot Funktionen mit
- Das lattice Packet erweitert dessen Funktionalität
- Eine weit über diese Einführung hinausgehende Übersicht findet sich in Murrell, P (2006): R Graphics.

CRAN Task Views

- Zu einigen Themen sind alle Möglichkeiten in R zusammengestellt. (Übersicht der Task Views)
- Zur Zeit gibt es 35 Task Views
- Alle Pakete eines Task Views können mit folgendem Befehl installiert werden:

```
install.packages("ctv")
library("ctv")
install.views("Bayesian")
```

CRAN Task Views

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
ExtremeValue	Extreme Value Analysis
Finance	Empirical Finance

Task View zu Thema Graphiken

CRAN Task View: Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

Maintainer:	Nicholas Lewin-Koh
Contact:	nikko at hailmail.net
Version:	2015-01-07
URL:	https://CRAN.R-project.org/view=Graphics

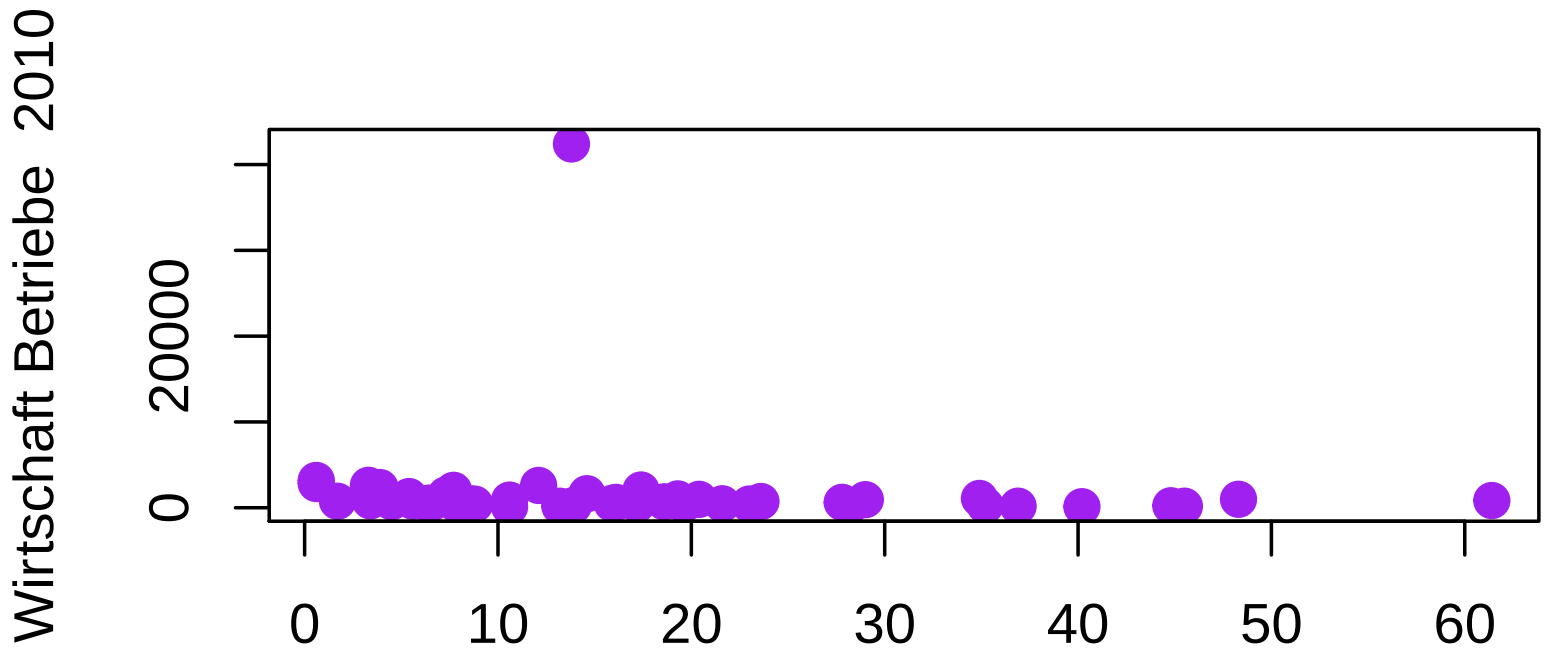
R is rich with facilities for creating and developing interesting graphics. Base R contains functionality for many plot types including coplots, mosaic plots, biplots, and the list goes on. There are devices such as postscript, png, jpeg and pdf for outputting graphics as well as device drivers for all platforms running R. lattice and grid are supplied with R's recommended packages and are included in every binary distribution. lattice is an R implementation of William Cleveland's trellis graphics, while grid defines a much more flexible graphics environment than the base R graphics.

Grafiken für bedingte, bi- und multivariate Verteilungen - Scatterplots

- Funktion `plot()` ist eine generische Funktion - bspw. kann ein einfacher Scatterplot erstellt werden
- Für einen solchen muss `plot()` ein x und ein y Beobachtungsvektor übergeben werden
- Um die Farbe der Plot-Symbole anzupassen gibt es die Option `col` (Farbe als character oder numerisch)
- Die Plot-Symbole selbst können mit `pch` (plotting character) angepasst werden (character oder numerisch)
- Die Achenbeschriftungen (labels) werden mit `xlab` und `ylab` definiert

Beispiel - Beschäftigten Datensatz

```
plot(dat[,7],dat[,4],ylab=colnames(dat)[4],
     xlab=colnames(dat)[7],pch=20,cex=2,col="purple")
```



/irtschaft Beschäftigte in Betrieben im Produzierenden Gewerbe in %

Datensatz

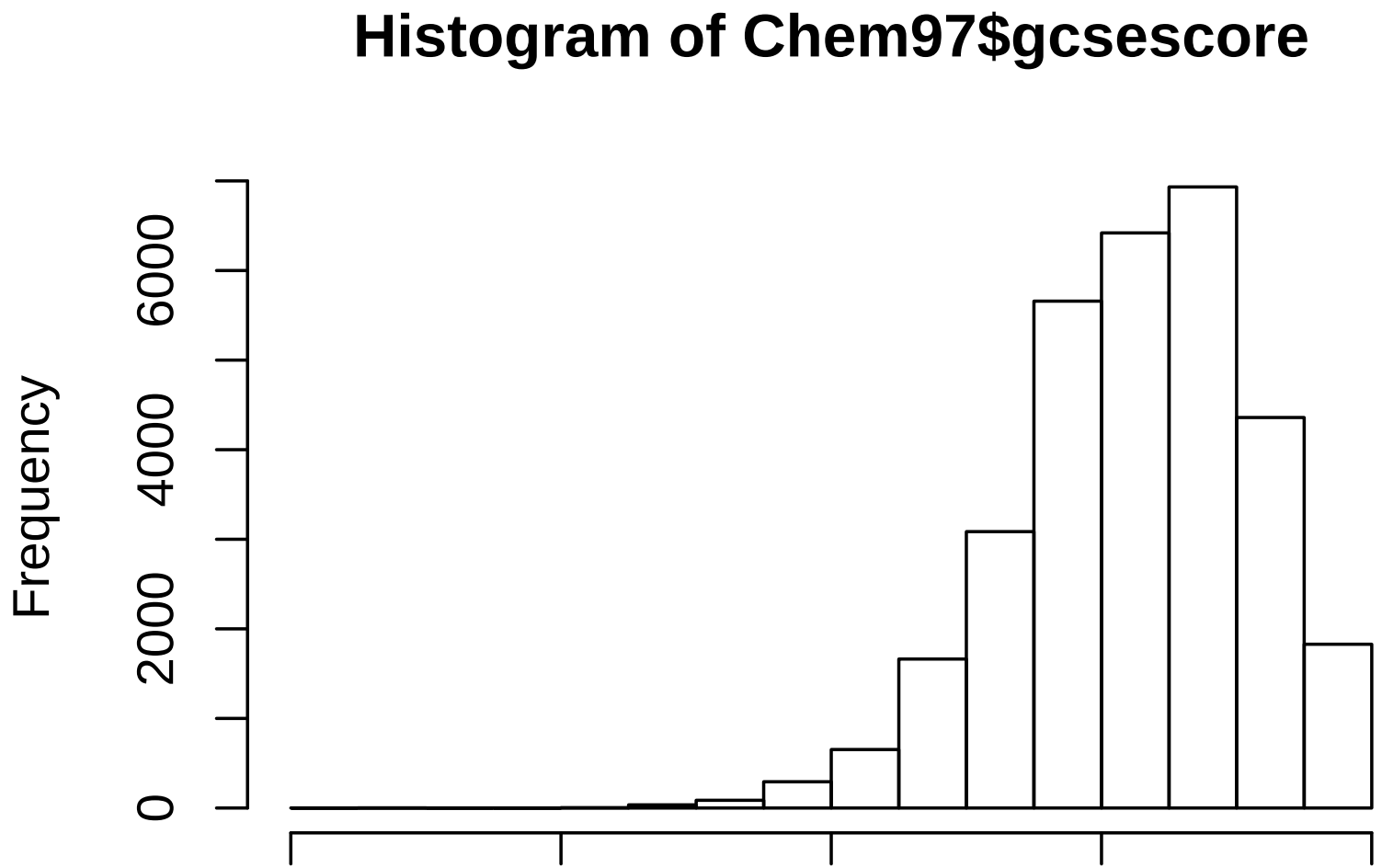
```
library(mlmRev)
data(Chem97)
```

- [lea] Local Education Authority - a factor
- [school] School identifier - a factor
- [student] Student identifier - a factor
- [score] Point score on A-level Chemistry in 1997
- [gender] Student's gender
- [age] Age in month, centred at 222 months or 18.5 years
- [gcsescore] Average GCSE score of individual.
- [gcsecnt] Average GCSE score of individual, centered at mean.

Histogramm - Die Funktion hist()

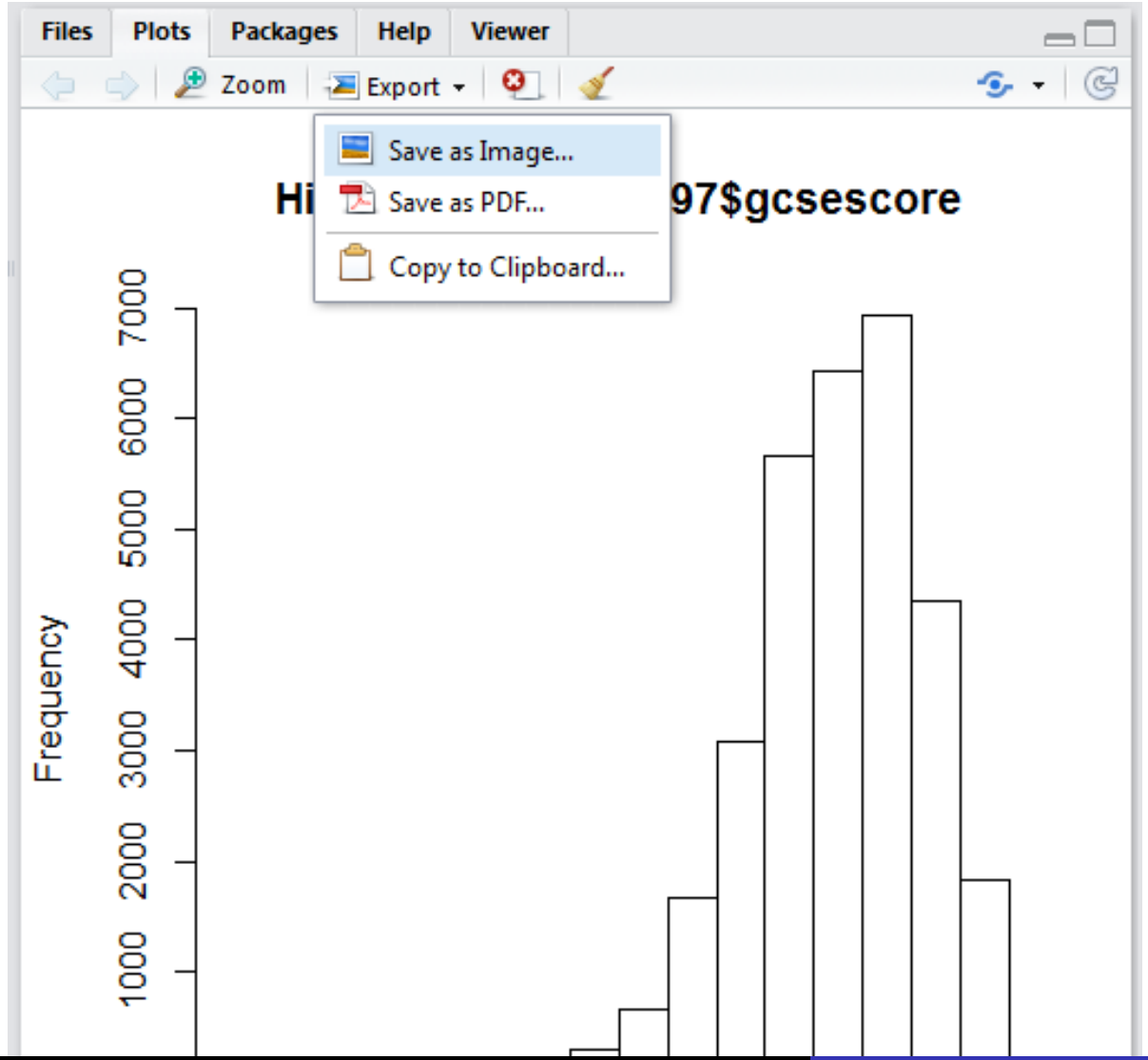
Wir erstellen ein Histogramm der Variable gcsescore:

```
hist(Chem97$gcsescore)
```



Graphik speichern

- Mit dem button Export in Rstudio kann man die Grafik speichern.



Befehl um Graphik zu speichern

- Alternativ auch bspw. mit den Befehlen `png`, `pdf` oder `jpeg`

```
png("Histogramm.png")
hist(Chem97$gcsescore)
dev.off()
```

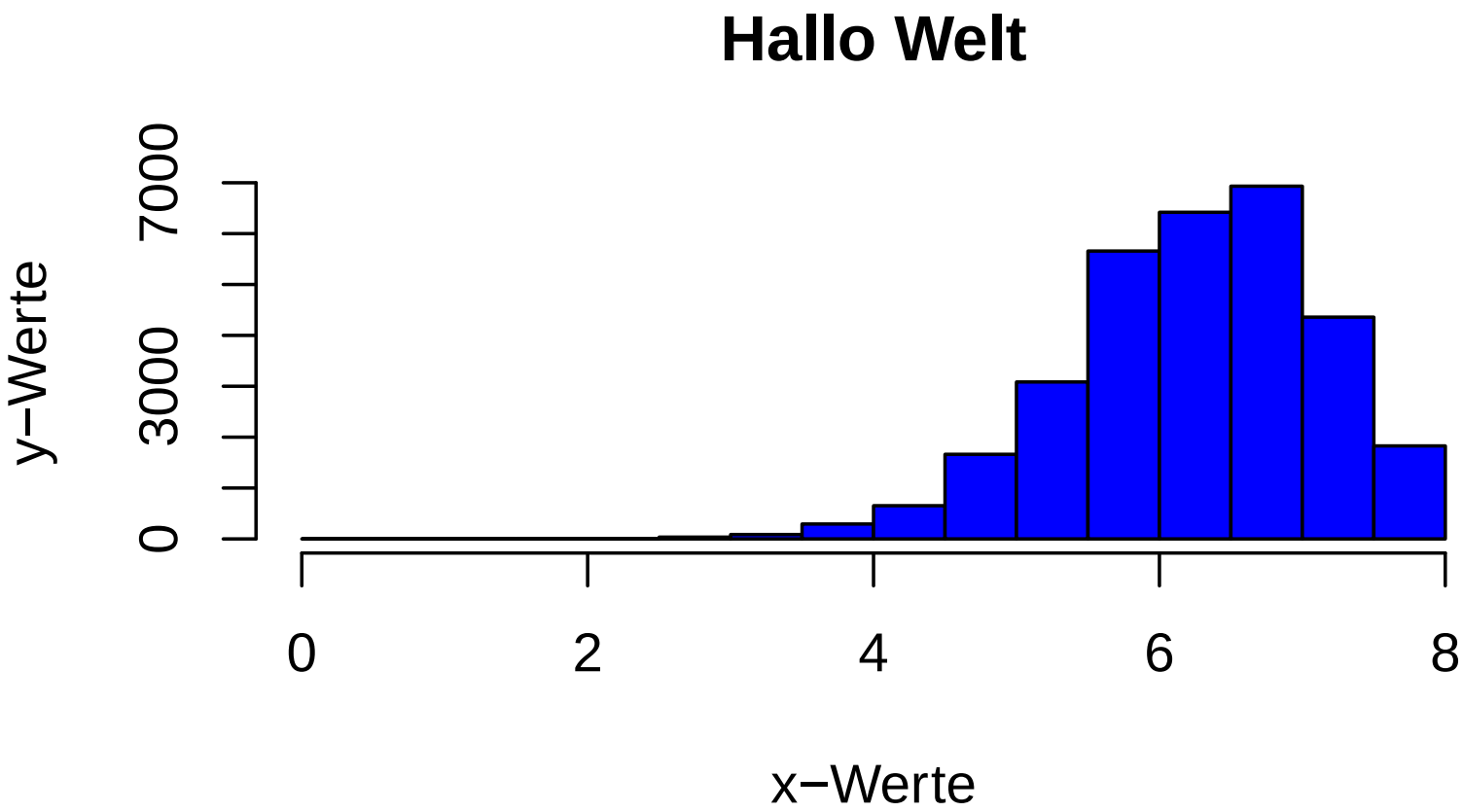
Histogramme

- Die Funktion `hist()` plottet ein Histogramm der Daten
- Der Funktion muss mindestens ein Beobachtungsvektor übergeben werden
- `hist()` hat noch sehr viel mehr Argumente, die alle (sinnvolle) default values haben

Argument	Bedeutung	Beispiel
main	Überschrift	main=‘Hallo Welt’
xlab	x-Achsenbeschriftung	xlab=‘x-Werte’
ylab	y-Achsenbeschriftung	ylab=‘y-Werte’
col	Farbe	col=‘blue’

Histogramm

```
hist(Chem97$gcsescore,col="blue",
     main="Hallo Welt",ylab="y-Werte", xlab="x-Werte")
```



Weitere Argumente:

Barplot

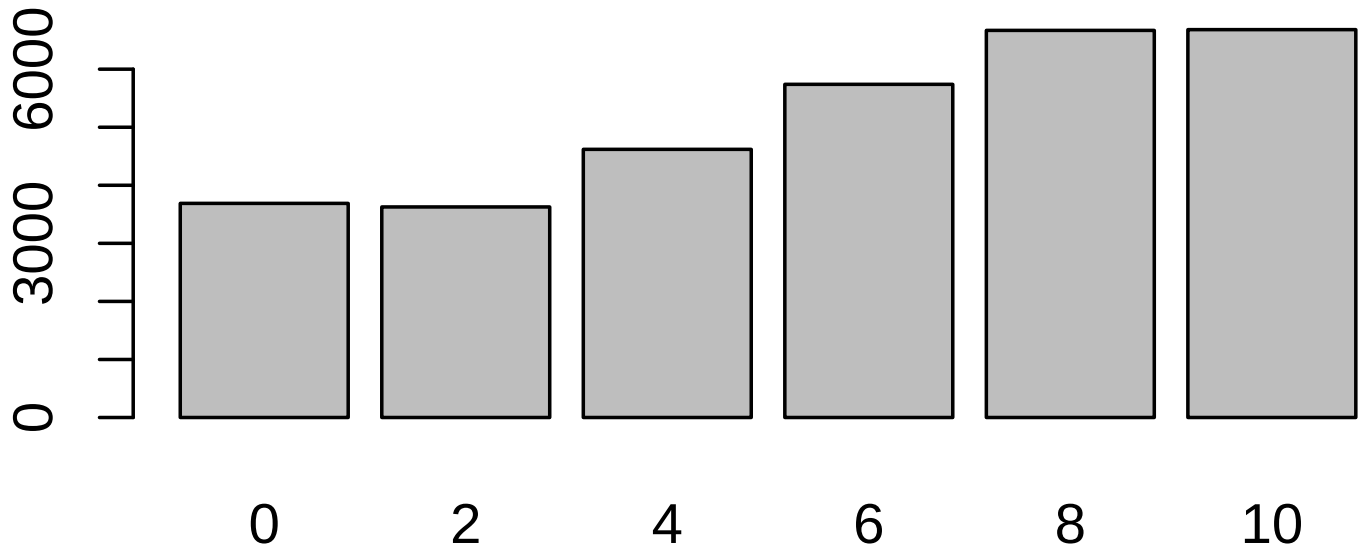
- Die Funktion `barplot()` erzeugt aus einer Häufigkeitstabelle einen Barplot
- Ist das übergebene Tabellen-Objekt zweidimensional wird ein bedingter Barplot erstellt

```
tabScore <- table(Chem97$score)
```

```
barplot(tabScore)
```

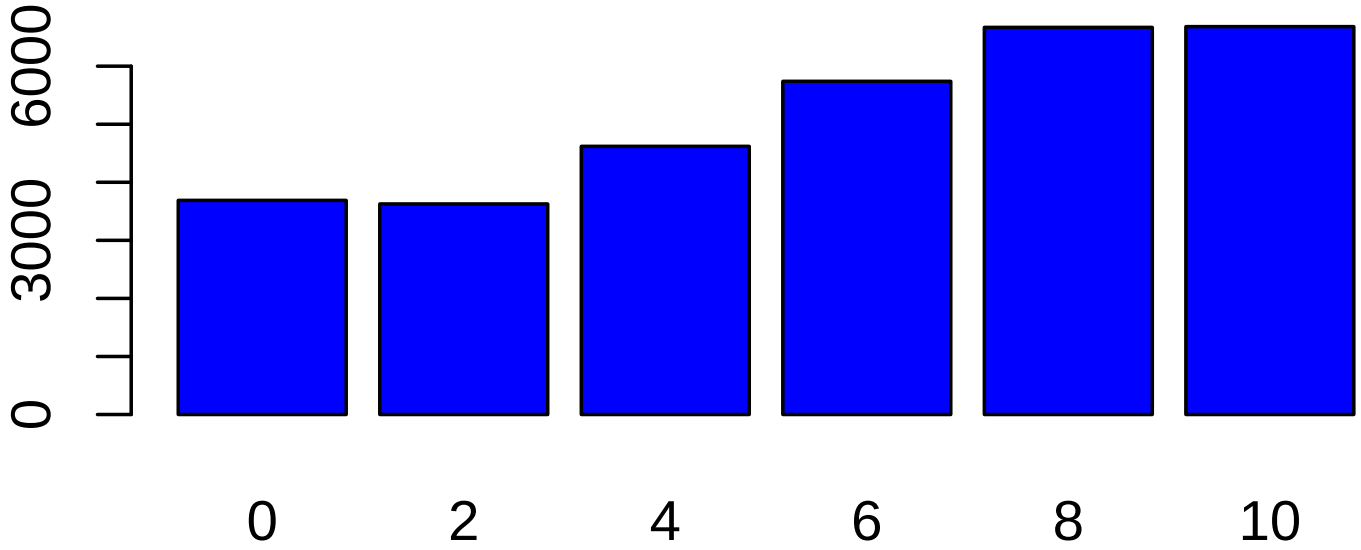
Barplots und barcharts

```
barplot(tabScore)
```



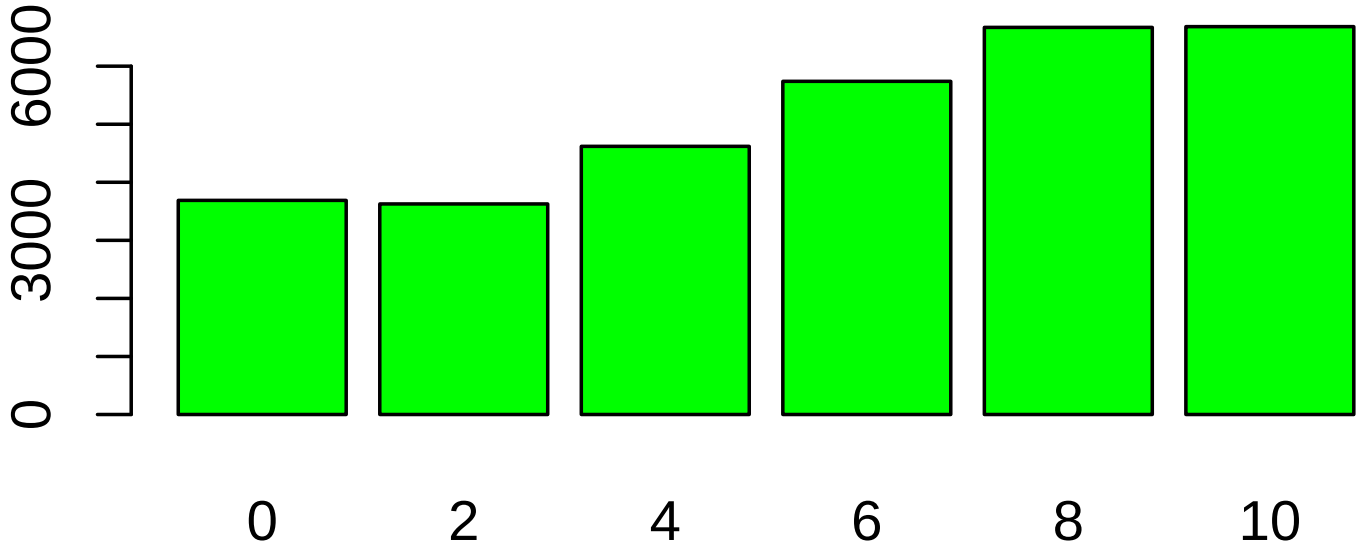
Mehr Farben:

```
barplot(tabScore,col=rgb(0,0,1))
```



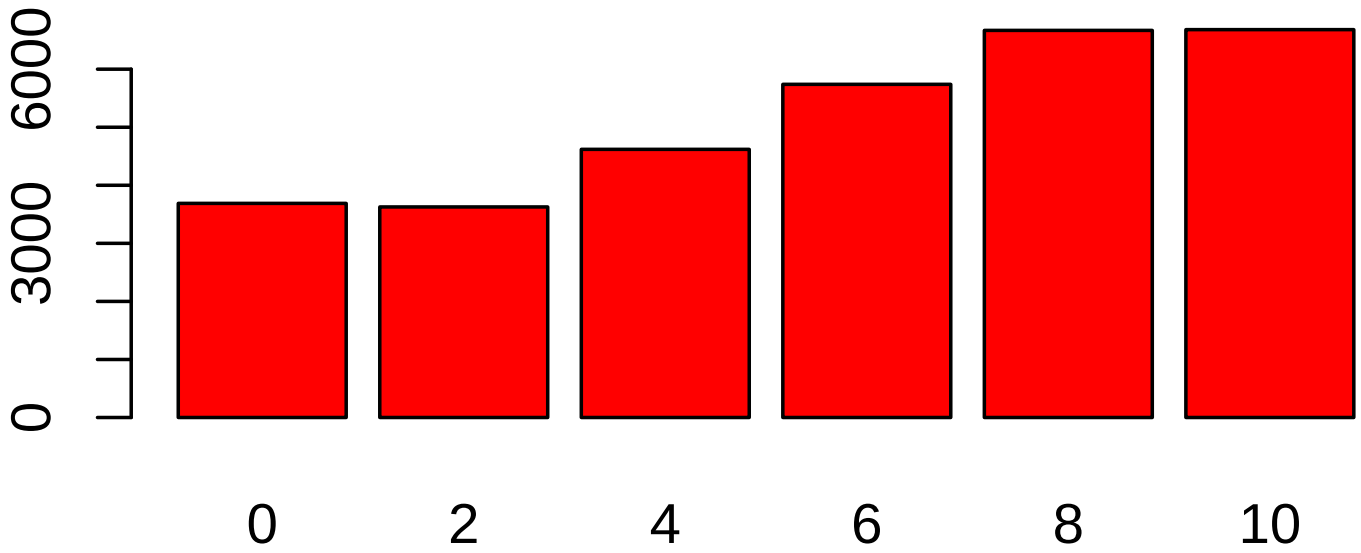
Grüne Farbe

```
barplot(tabScore,col=rgb(0,1,0))
```



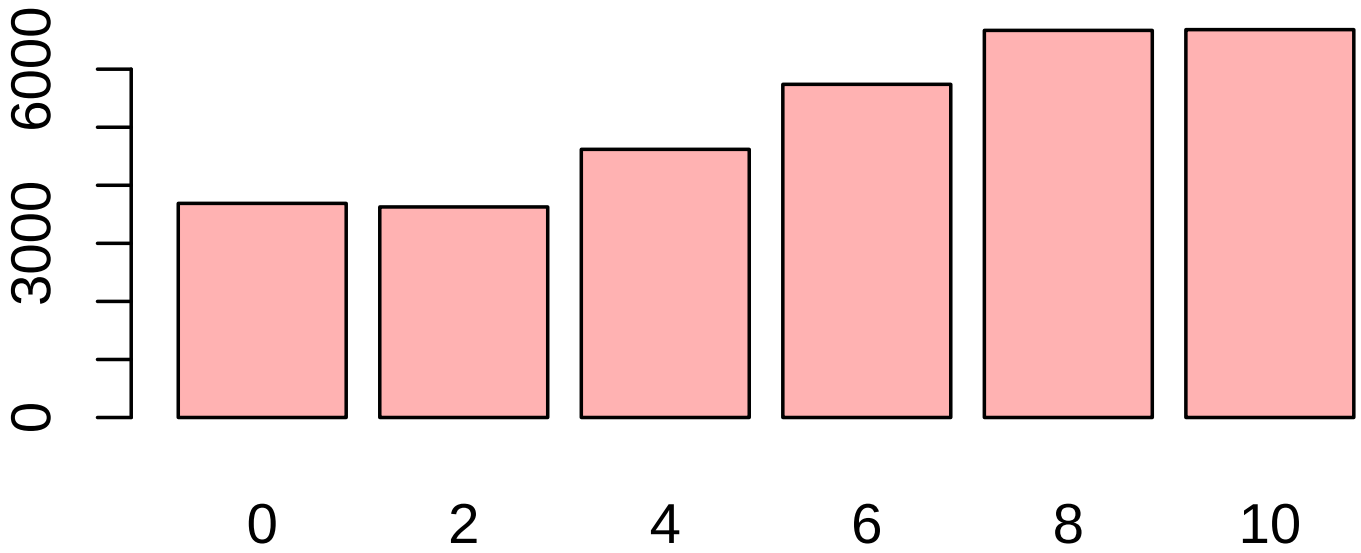
Rote Farbe

```
barplot(tabScore,col=rgb(1,0,0))
```



Transparent

```
barplot(tabScore,col=rgb(1,0,0,.3))
```



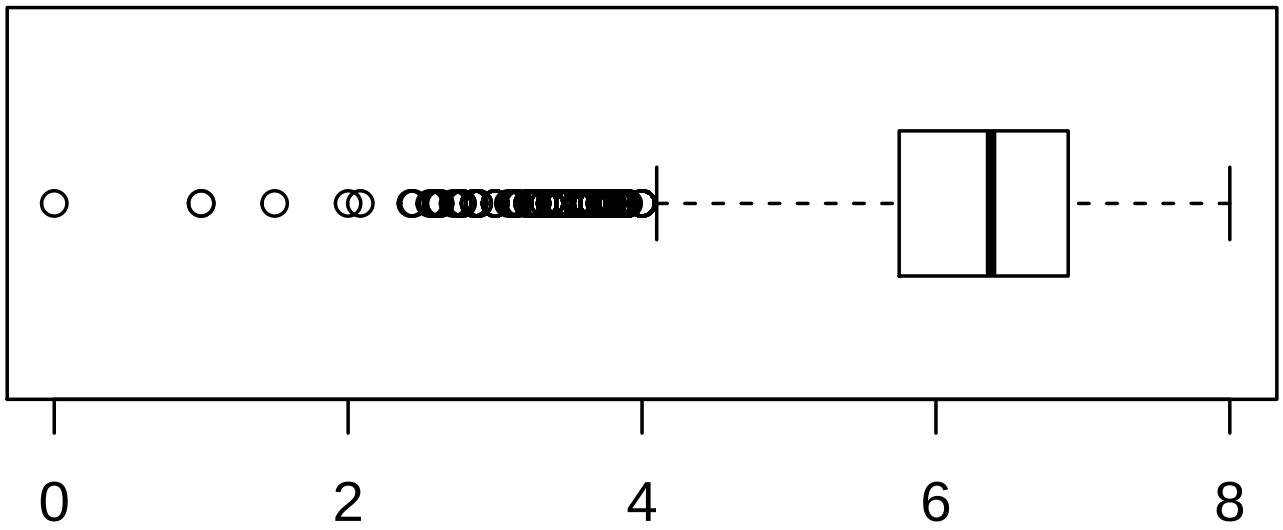
Boxplot

- Einen einfachen Boxplot erstellt man mit `boxplot()`
- Auch `boxplot()` muss mindestens ein Beobachtungsvektor übergeben werden

```
?boxplot
```

Horizontaler Boxplot

```
boxplot(Chem97$gcsescore,
horizontal=TRUE)
```



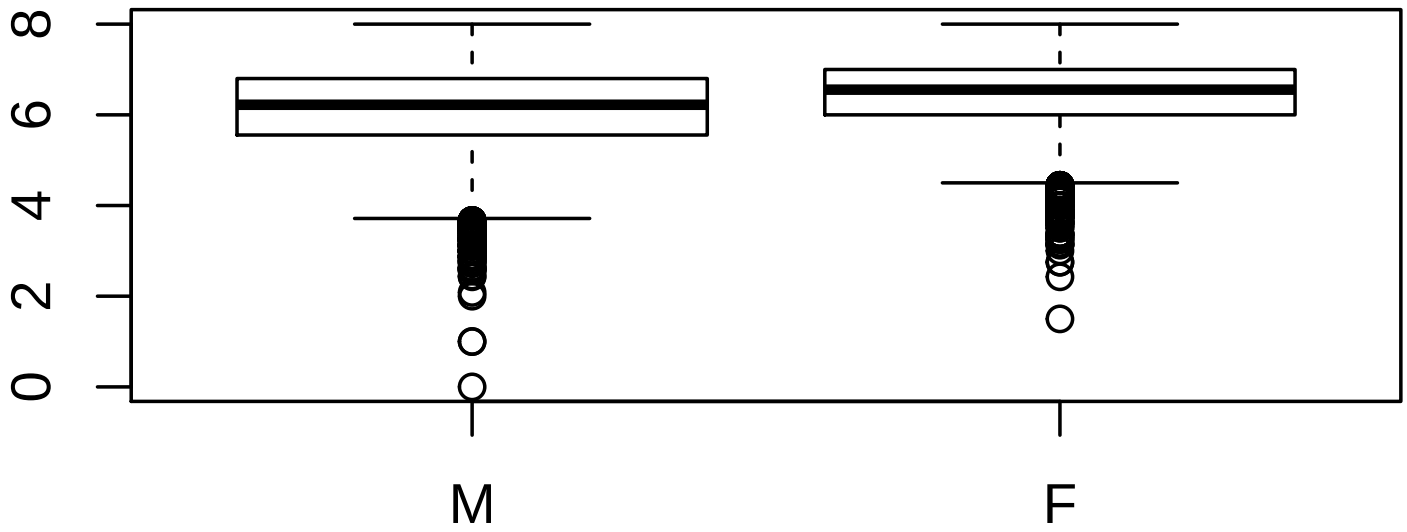
- Erklärung zu Boxplots

Gruppierte Boxplots

- Ein sehr einfacher Weg, einen ersten Eindruck über bedingte Verteilungen zu bekommen ist über sog. Gruppierte notched Boxplots
- Dazu muss der Funktion `boxplot()` ein sog. Formel-Objekt übergeben werden
- Die bedingende Variable steht dabei auf der rechten Seite einer Tilde

Beispiel grupierter Boxplot

```
boxplot(Chem97$gcsescore~Chem97$gender)
```



Alternativen zu Boxplot

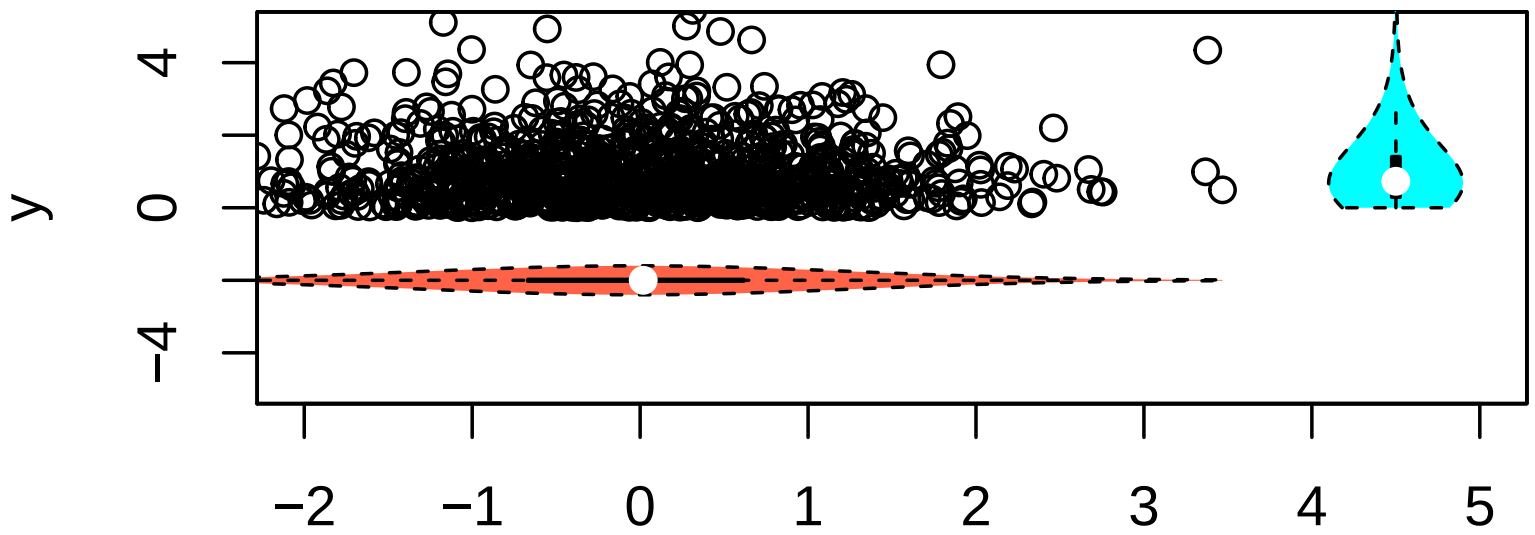
`vioplot`

- Baut auf Boxplot auf
- Zusätzlich Informationen über Dichte der Daten
- Dichte wird über Kernel Methode berechnet.
- weißer Punkt - Median
- Je weiter die Ausdehnung, desto größer ist die Dichte an dieser Stelle.

```
# Beispieldaten erzeugen
x <- rnorm(1000)
y <- rexp(1000,1)
```

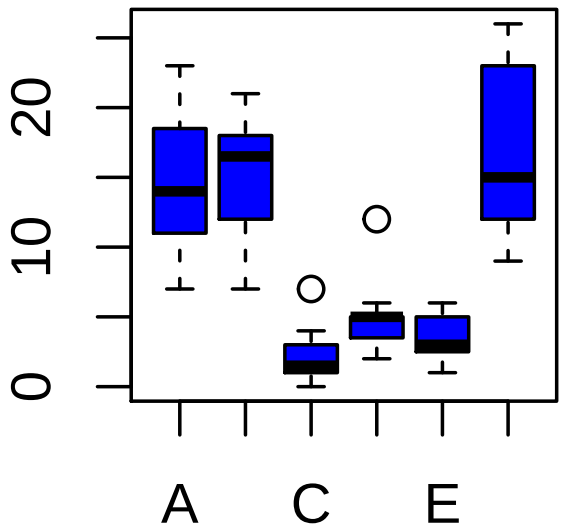
Beispiel vioplot

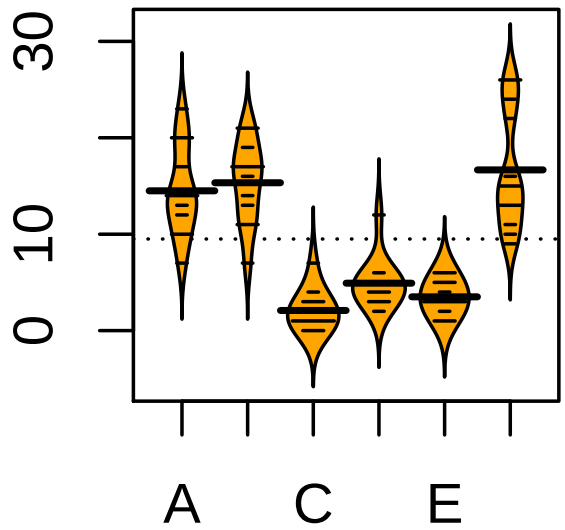
```
library(vioplot)
plot(x, y, xlim=c(-2,5), ylim=c(-5,5))
vioplot(x, col="tomato", horizontal=TRUE, at=-2,
        add=TRUE,lty=2, rectCol="gray")
vioplot(y, col="cyan", horizontal=FALSE, at=4.5,
        add=TRUE,lty=2)
```



Alternativen zum Boxplot

```
library(beanplot)
par(mfrow = c(1,2))
boxplot(count~spray,data=InsectSprays,col="blue")
beanplot(count~spray,data=InsectSprays,col="orange")
```

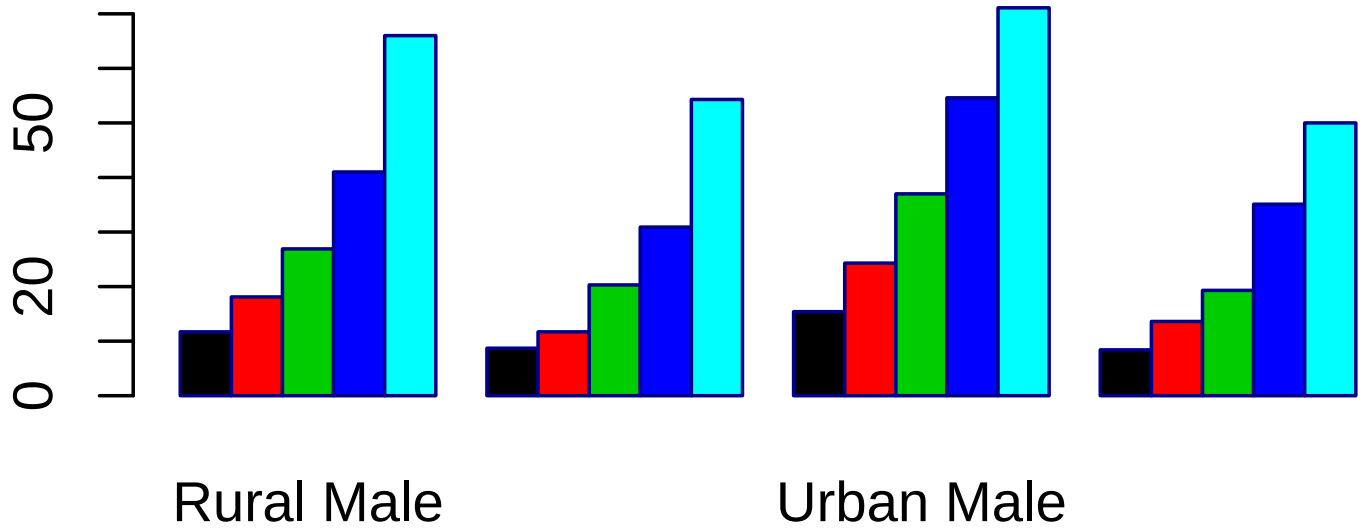




Aufgabe Balkendiagramm

Aufgabe - einfache Grafiken

- Laden Sie den Datensatz `VADeaths` und erzeugen Sie den folgenden plot:



Die lineare Regression

Die lineare Regression

John H. Maindonald and W. John Braun - **Data Analysis and Graphics Data and Functions**

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

Lineare Regression in R - Beispieldatensatz

```
data(mtcars)
```

Hilfe für den `mtcars` Datensatz:

```
?mtcars
```

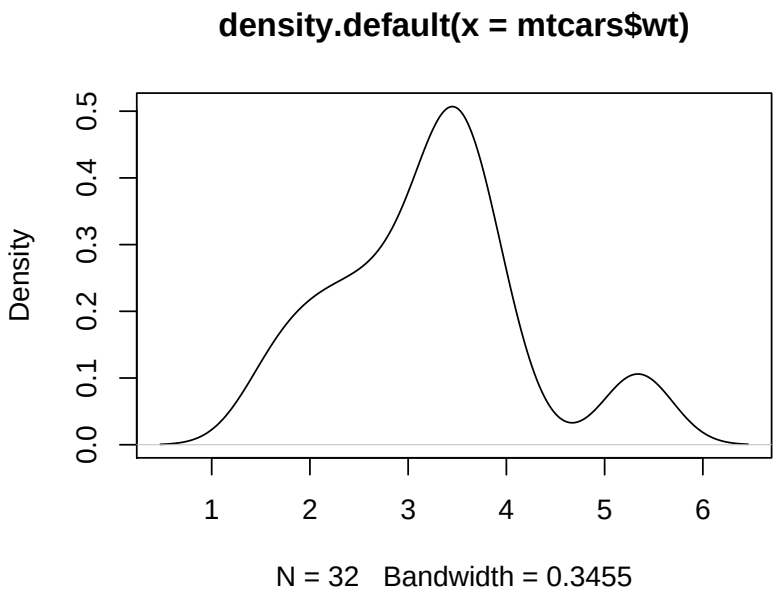
	mpg	cyl	disp	hp	drat	wt	model
21.0	6	160	110	3.90	2.620	Mazda RX4	
21.0	6	160	110	3.90	2.875	Mazda RX4 Wag	
22.8	4	108	93	3.85	2.320	Datsun 710	
21.4	6	258	110	3.08	3.215	Hornet 4 Drive	
18.7	8	360	175	3.15	3.440	Hornet Sportabout	
18.1	6	225	105	2.76	3.460	Valiant	

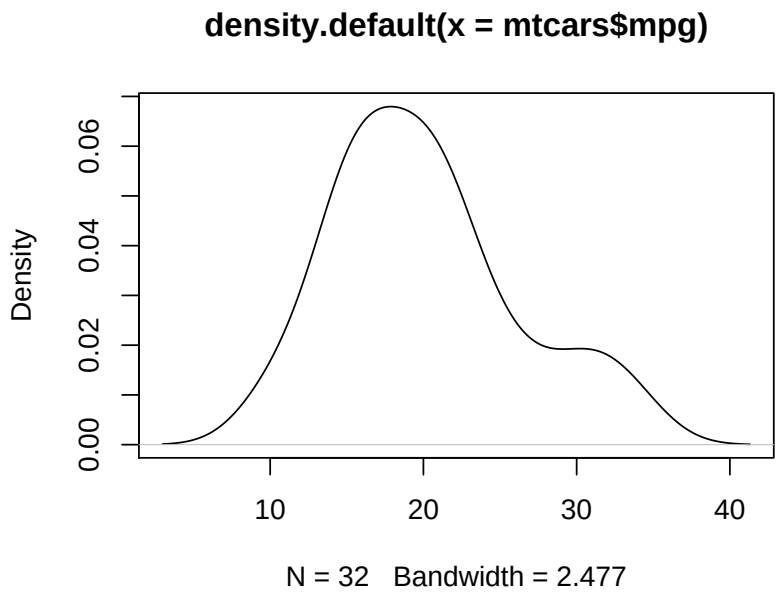
Variablen des `mtcars` Datensatzes

- mpg - Miles/(US) gallon
- cyl - Number of cylinders
- disp - Displacement (cu.in.)
- hp - Gross horsepower
- drat - Rear axle ratio
- wt - Weight (1000 lbs)
- qsec - 1/4 mile time
- vs - Engine (0 = V-shaped, 1 = straight)
- am - Transmission (0 = automatic, 1 = manual)
- gear - Number of forward gears
- carb - Number of carburetors

Verteilungen für zwei Variablen von `mtcars`

```
par(mfrow=c(1,2))
plot(density(mtcars$wt)); plot(density(mtcars$mpg))
```





Ein einfaches Regressionsmodell

Abhängige Variable - Meilen pro Gallone (mpg)

Unabhängige Variable - Gewicht (wt)

```
m1 <- lm(mpg ~ wt,data=mtcars)
m1
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Coefficients:
## (Intercept)           wt
##      37.285       -5.344
```

Die Modellformel

Modell ohne Achsenabschnitt

```
m2 <- lm(mpg ~ - 1 + wt,data=mtcars)
summary(m2)$coefficients
```

```
##    Estimate Std. Error  t value    Pr(>|t|)
## wt 5.291624  0.5931801 8.920771 4.55314e-10
```

Weitere Variablen hinzufügen

```
m3 <- lm(mpg ~ wt + cyl,data=mtcars)
summary(m3)$coefficients
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 39.686261  1.7149840 23.140893 3.043182e-20
## wt          -3.190972  0.7569065 -4.215808 2.220200e-04
## cyl         -1.507795  0.4146883 -3.635972 1.064282e-03
```

Summary des Modells

```
summary(m3)
```

```
##
## Call:
## lm(formula = mpg ~ wt + cyl, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2893 -1.5512 -0.4684  1.5743  6.1004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  39.6863     1.7150  23.141  < 2e-16 ***
## wt           -3.1910     0.7569  -4.216 0.000222 ***
## cyl          -1.5078     0.4147  -3.636 0.001064 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
##
```

R arbeitet mit Objekten

- `m3` ist nun ein spezielles Regressions-Objekt
- Auf dieses Objekt können nun verschiedene Funktionen angewendet werden

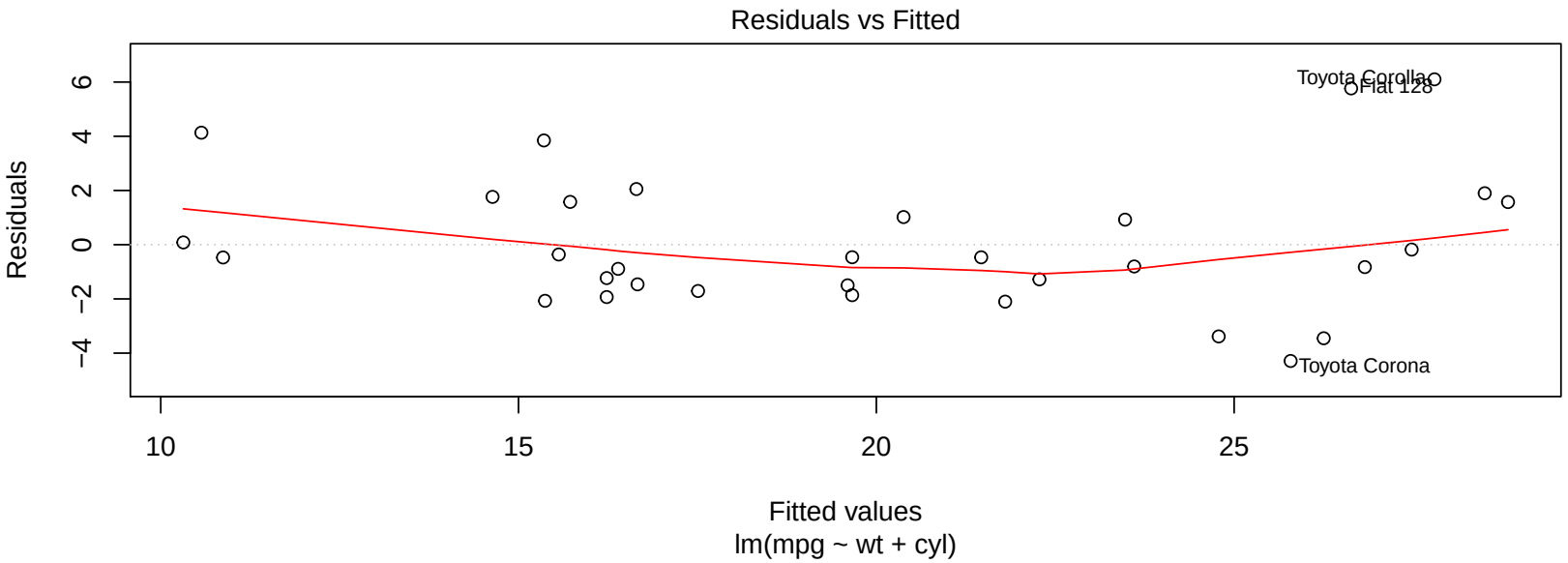
```
predict(m3) # Vorhersage
```

```
##           Mazda RX4       Mazda RX4 Wag           Datsun
##            22.27914            21.46545              26.25
##      Hornet 4 Drive   Hornet Sportabout              Vali
##            20.38052            16.64696              19.59
##         Duster 360          Merc 240D               Merc
##            16.23213            23.47588              23.60
##          Merc 280          Merc 280C             Merc 45
##            19.66255            19.66255              14.63
##         Merc 450SL         Merc 450SLC  Cadillac Fleetw
##            15.72158            15.56203              10.87
## Lincoln Continental   Chrysler Imperial               Fiat
##            10.31607            10.56816              26.63
```

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)

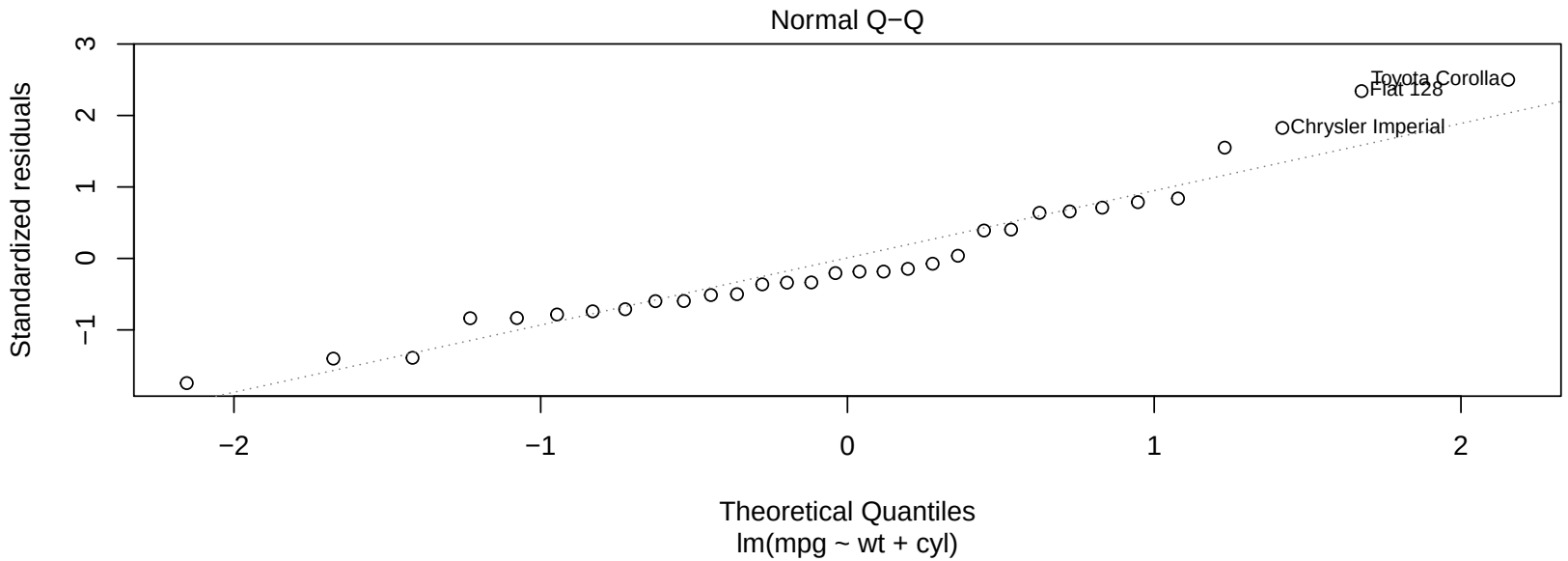
```
plot(m3,1)
```



Residuenplot

- Wenn Residuen normalverteilt sind sollten sie auf Linie sein.

```
plot(m3,2)
```



Weitere Möglichkeiten die Formel zu spezifizieren

Interaktionseffekt

```
# effect of cyl and interaction effect:
m3a<-lm(mpg~wt*cyl,data=mtcars)

# only interaction effect:
m3b<-lm(mpg~wt:cyl,data=mtcars)
```

Den Logarithmus nehmen

```
m3d<-lm(mpg~log(wt),data=mtcars)
```

Ein Modell mit Interaktionseffekt

disp - Hubraum

```
m3d<-lm(mpg~wt*disp,data=mtcars)
m3dsum <- summary(m3d)
m3dsum$coefficients
```

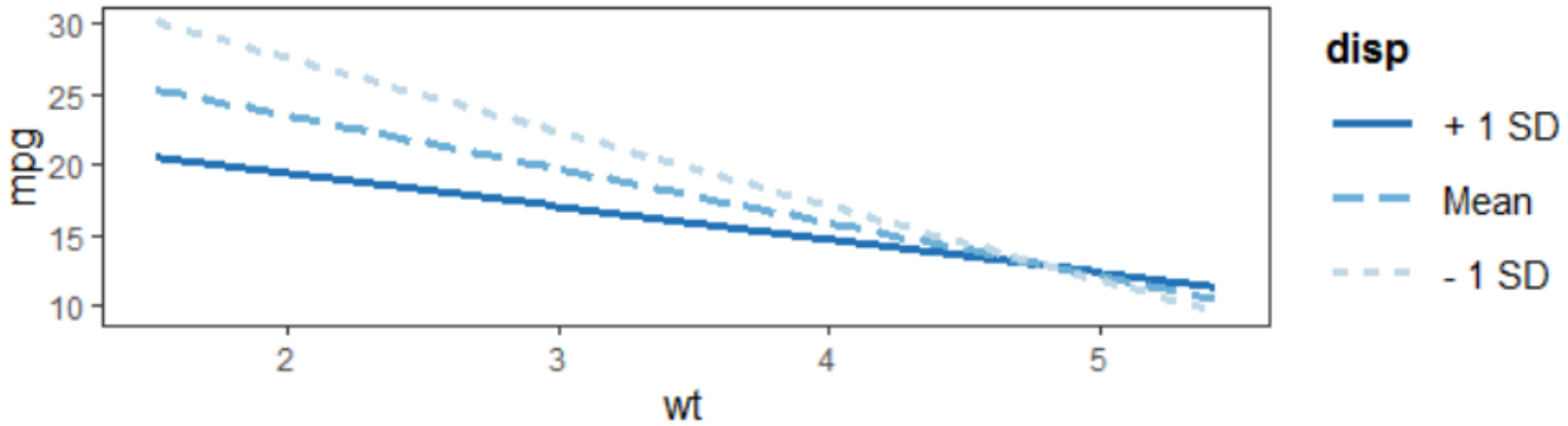
```
##                Estimate  Std. Error   t value     Pr(>|t
## (Intercept) 44.08199770 3.123062627 14.114990 2.955567e-
## wt          -6.49567966 1.313382622 -4.945763 3.216705e-
## disp        -0.05635816 0.013238696 -4.257078 2.101721e-
## wt:disp      0.01170542 0.003255102  3.596022 1.226988e-
```

Interaktionen untersuchen

```
install.packages("jtools")
```

```
library(jtools)
interact_plot(m3d, pred = "wt", modx = "disp")
```

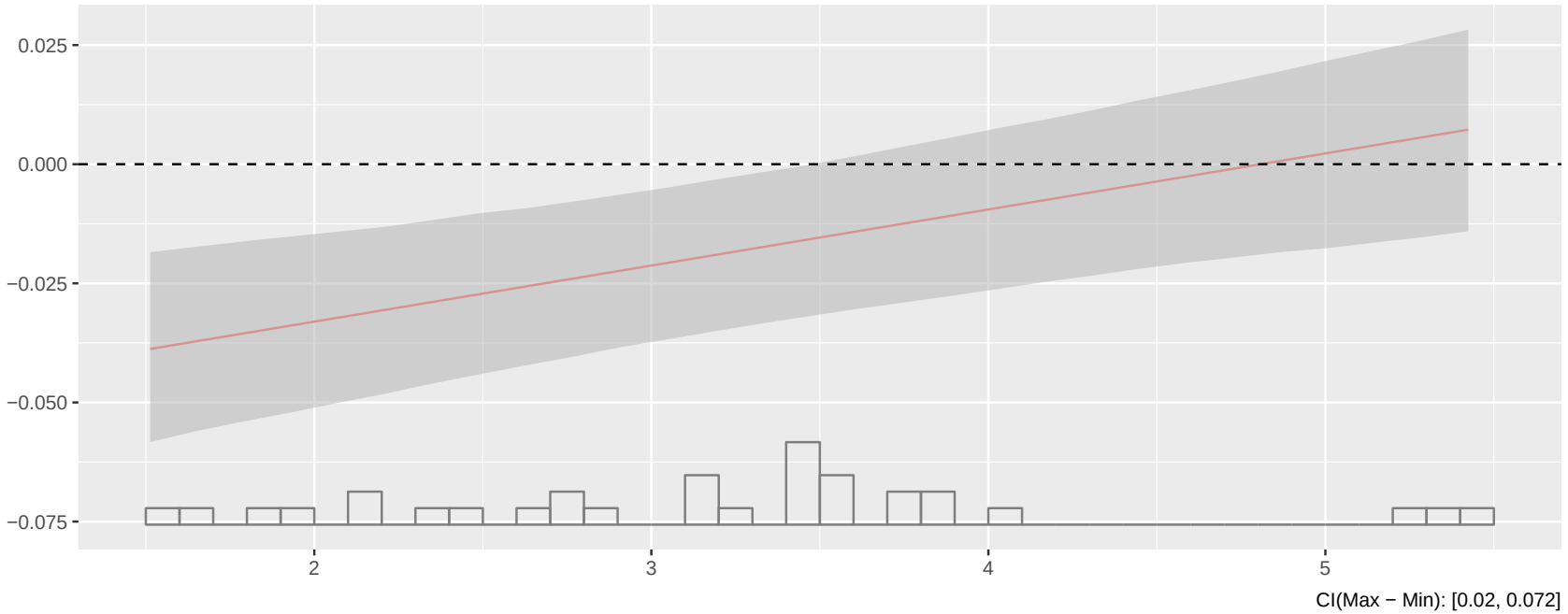
- Mit einem kontinuierlichen Moderator (in unserem Fall `Disp`) erhält man drei Zeilen - 1 Standardabweichung über und unter dem Mittelwert und der Mittelwert selbst.



Das Paket interplot

```
library(interplot)
```

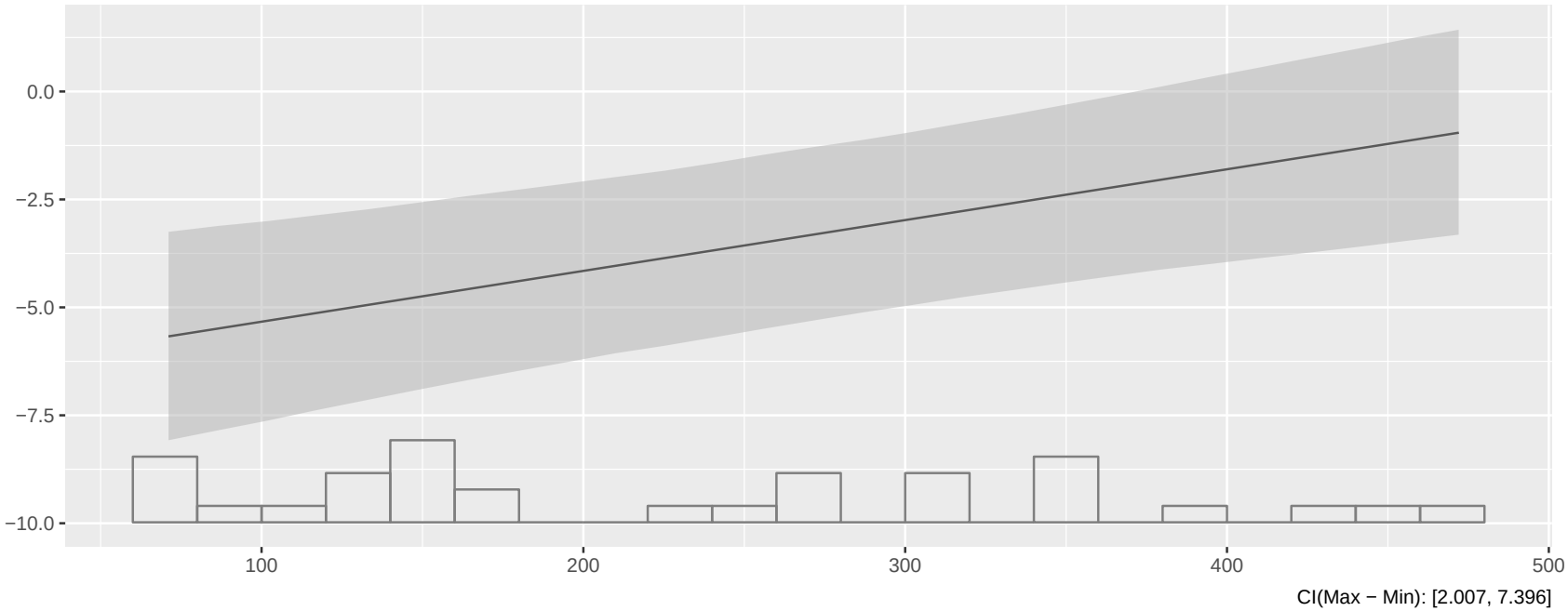
```
interplot(m = m3d, var1 = "disp", var2 = "wt", hist = TRUE)
  aes(color = "pink") + theme(legend.position="none") +
  geom_hline(yintercept = 0, linetype = "dashed")
```



Noch ein interplot

- Effekt wird auf die y-Achse geplottet - **wt** auf der x-Achse

```
interplot(m = m3d, var1 = "wt", var2 = "disp", hist = TRUE)
```



- Eine detailliertere Beschreibung ist in der **interplot Vignette** zu bekommen.

Beispiel: Objektorientierung

- `m3` ist nun ein spezielles Regressionsobjekt
- Verschiedene Funktionen können auf dieses Objekt angewendet werden

```
predict(m3) # Prediction
resid(m3) # Residuals
```

```
##          Mazda RX4      Mazda RX4 Wag         Datsun 710
##           22.27914           21.46545           26.25203
## Hornet Sportabout            Valiant
##           16.64696           19.59873

##          Mazda RX4      Mazda RX4 Wag         Datsun 710
##         -1.2791447         -0.4654468         -3.4520262
## Hornet Sportabout            Valiant
##          2.0530424         -1.4987281
```

Eine Modellvorhersage machen

```
pre <- predict(m1)
head(mtcars$mpg)
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1
```

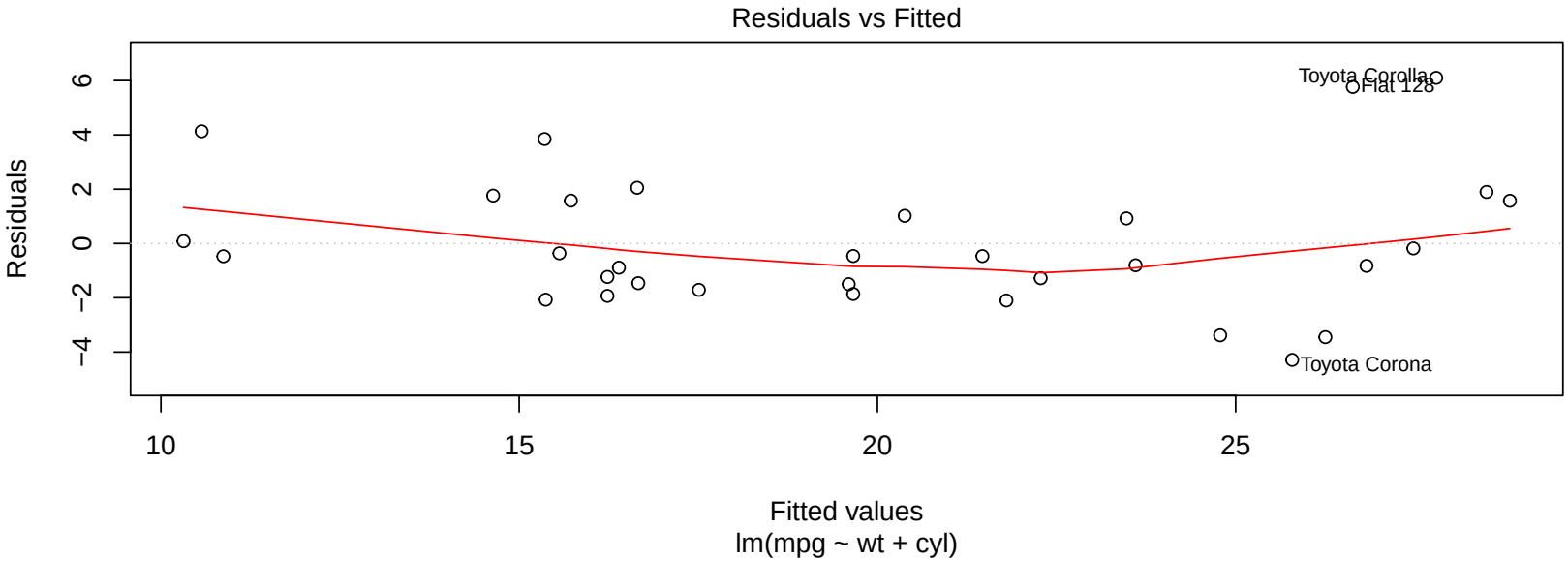
```
head(pre)
```

```
##          Mazda RX4      Mazda RX4 Wag         Datsun 710
##           23.28261           21.91977           24.88595
## Hornet Sportabout            Valiant
##           18.90014           18.79325
```

Residuenplot - Modellannahmen verletzt?

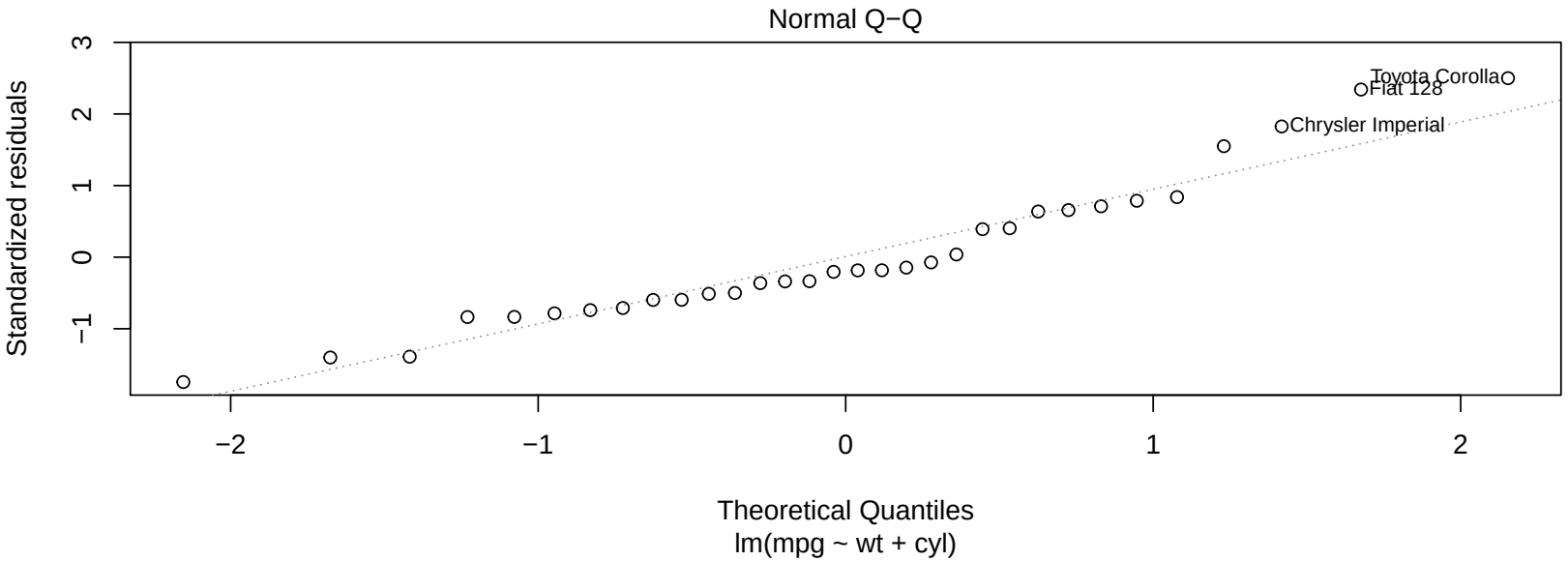
- Gibt es ein Muster in der Abweichung von der Linie

```
plot(m3,1)
```



Residuenplot

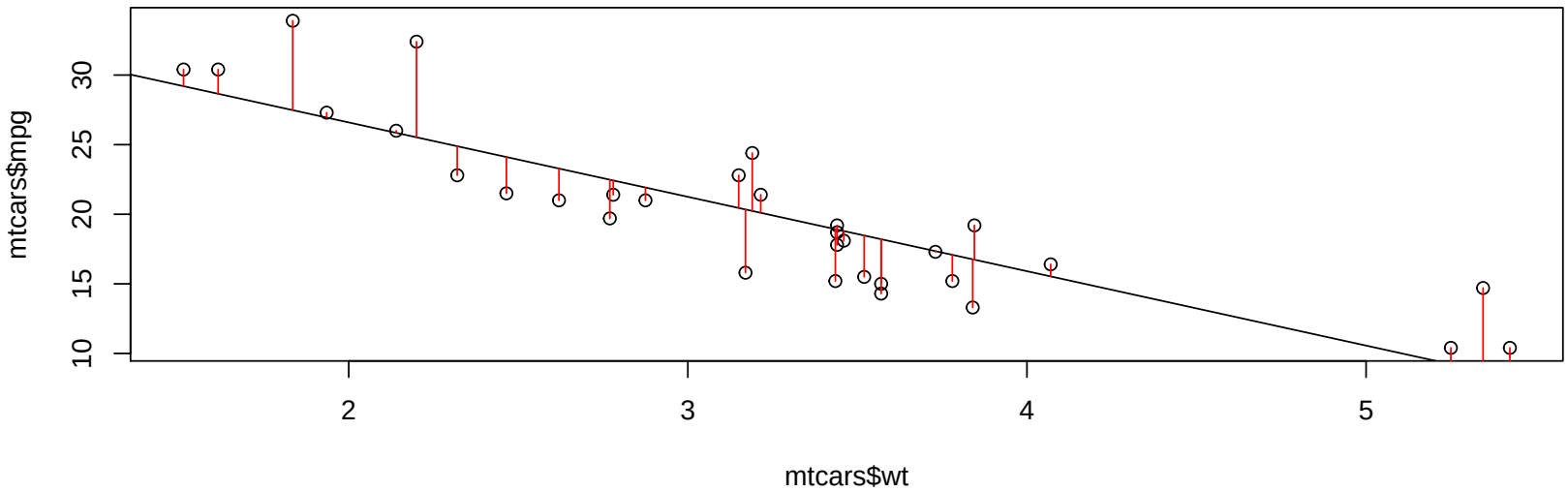
```
plot(m3,2)
```



- Bei Normalverteilung liegen Residuen auf gleicher Linie

Regressionsdiagnostik mit Basis-R

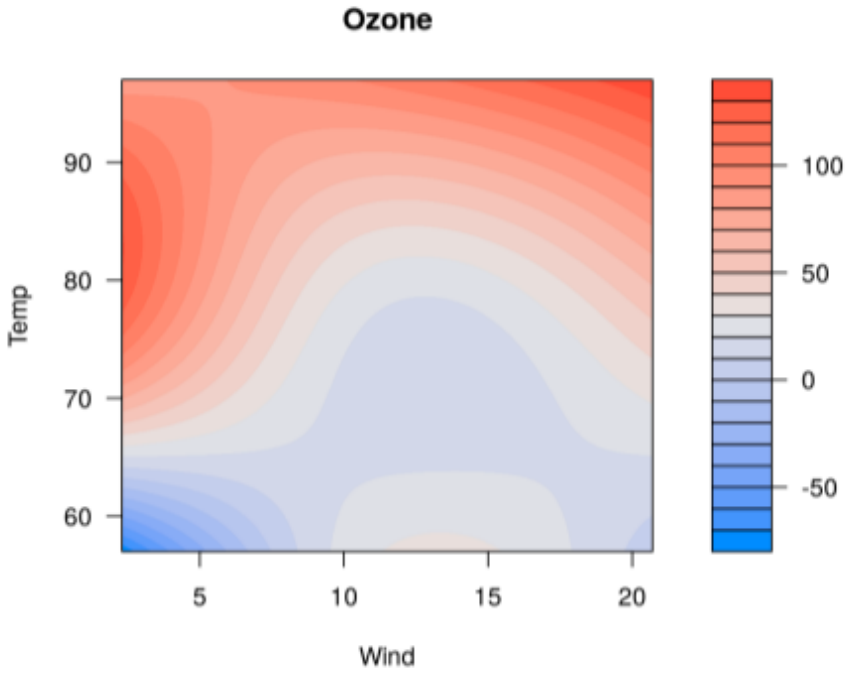
```
plot(mtcars$wt,mtcars$mpg)
abline(m1)
segments(mtcars$wt, mtcars$mpg, mtcars$wt, pre, col="red")
```

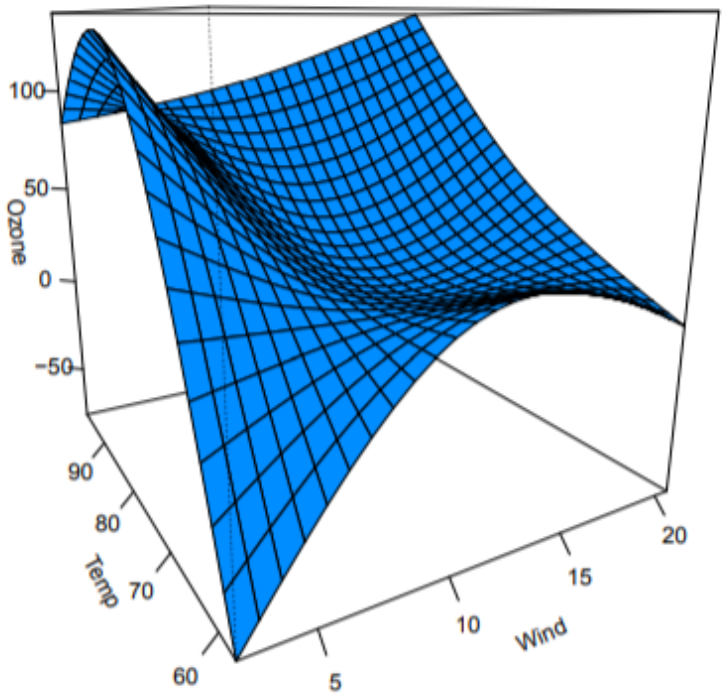


Das visreg-Paket

```
install.packages("visreg")
```

```
library(visreg)
```

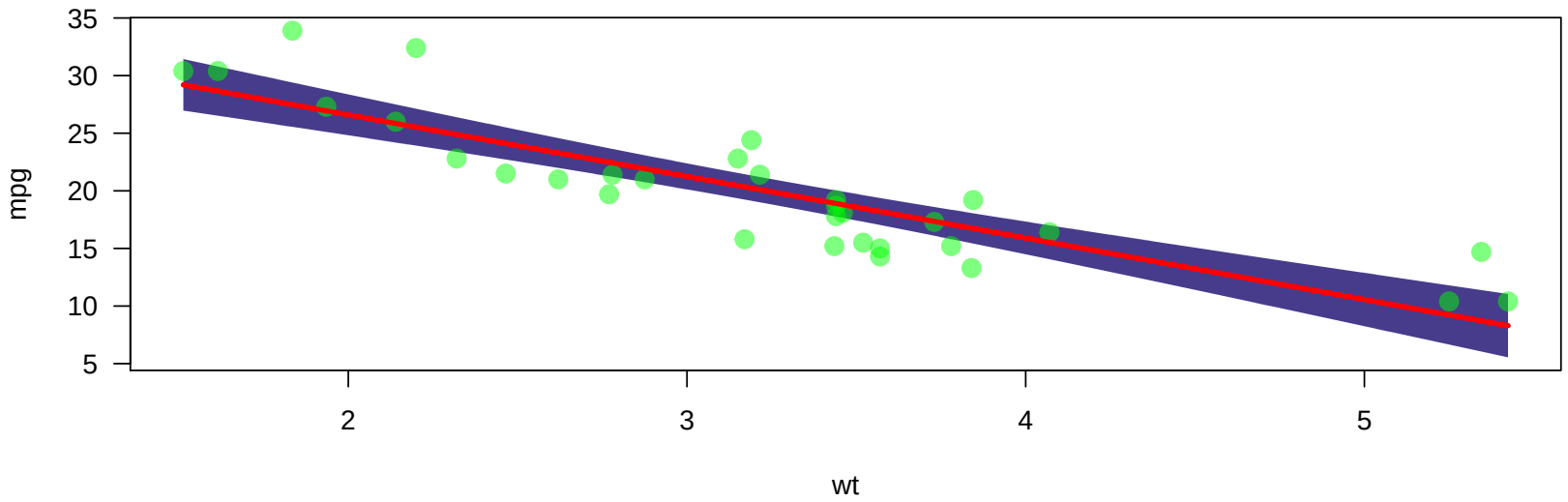




Das visreg-Paket

- Das Default-Argument für `type` ist `conditional`.
- Scatterplot von `mpg` und `wt` mit Regressionslinie und Konfidenzbändern

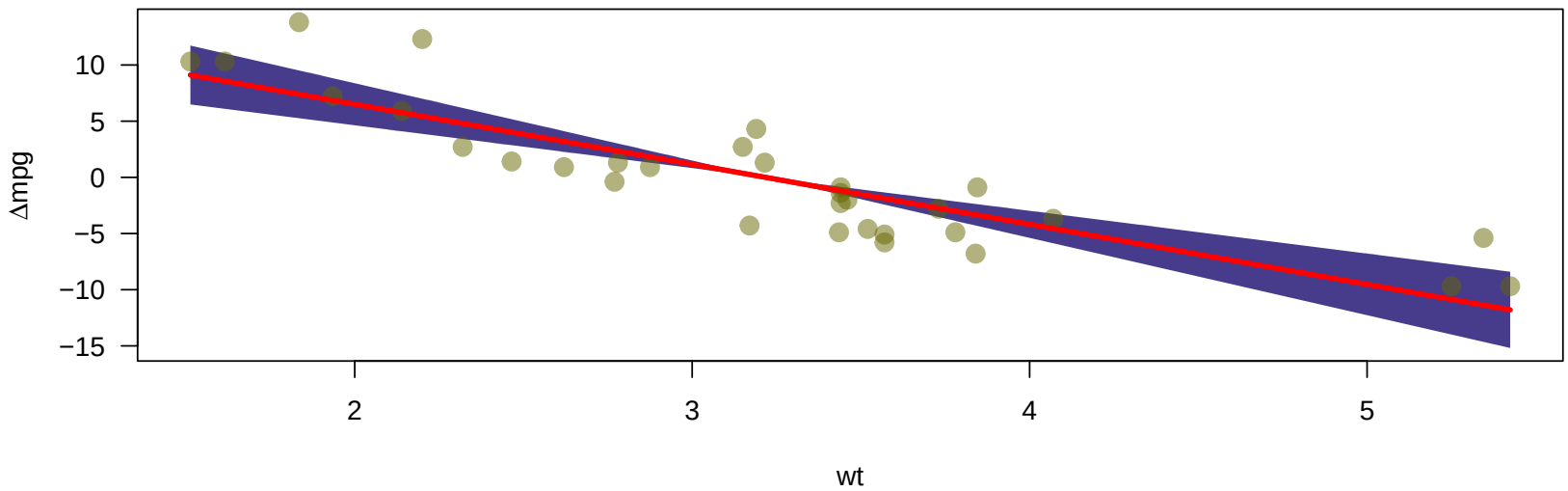
```
visreg(m1, "wt", type = "conditional")
```



Visualisierung mit visreg

- Zweites Argument - Spezifikation der Kovariaten in der Graphik
- Das Diagramm zeigt die Auswirkung auf den erwarteten Wert des Regressors, wenn die Variable x von einem Referenzpunkt auf der x-Achse wegbewegt wird (bei numerischen Variablen der Mittelwert).

```
visreg(m1, "wt", type = "contrast")
```



Regression mit Faktoren

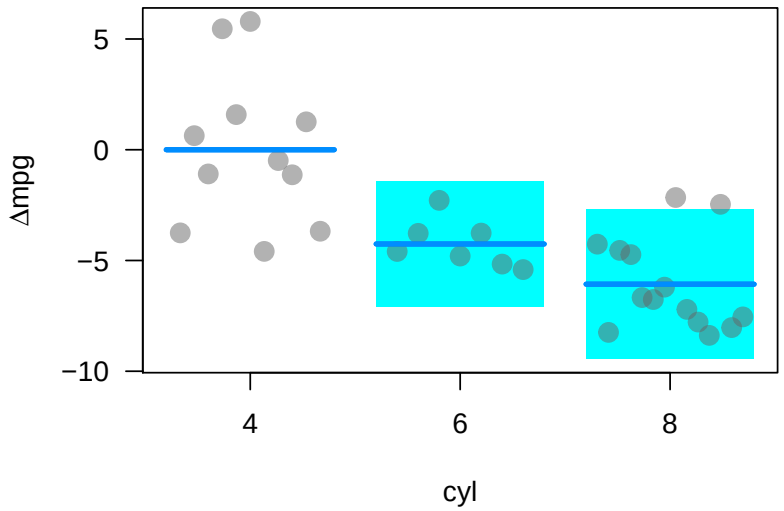
- Die Effekte von Faktoren können auch mit `visreg` visualisiert werden:

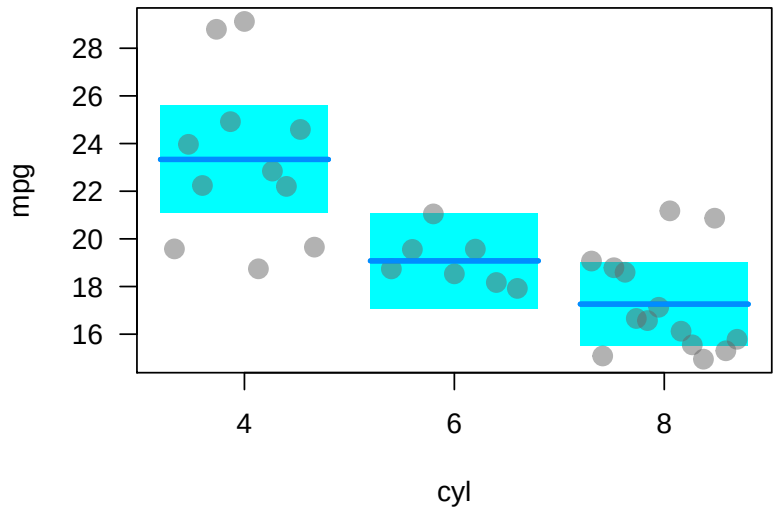
```
mtcars$cyl <- as.factor(mtcars$cyl)
m4 <- lm(mpg ~ cyl + wt, data = mtcars)
# summary(m4)
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 33.990794  1.8877934 18.005569 6.257246e-17
## cyl6        -4.255582  1.3860728 -3.070244 4.717834e-03
## cyl8        -6.070860  1.6522878 -3.674214 9.991893e-04
## wt          -3.205613  0.7538957 -4.252065 2.130435e-04
```

Effekte von Faktoren

```
par(mfrow=c(1,2))
visreg(m4, "cyl", type = "contrast")
visreg(m4, "cyl", type = "conditional")
```





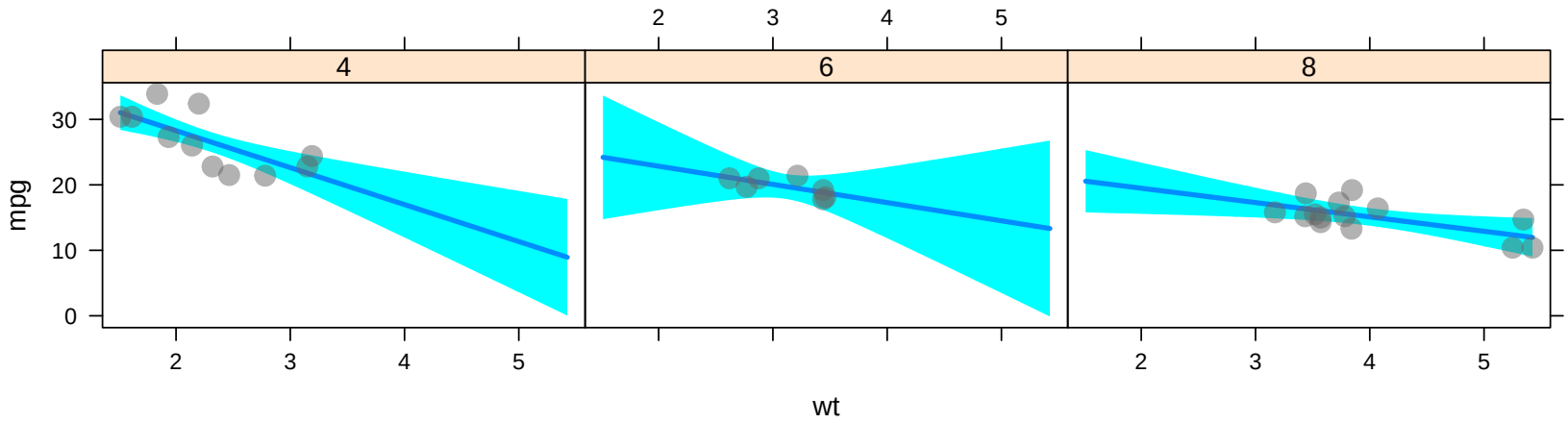
Das Paket visreg - Interaktionen

```
m5 <- lm(mpg ~ cyl*wt, data = mtcars)
# summary(m5)
```

```
##                Estimate Std. Error    t value     Pr(>|t
## (Intercept)  39.571196   3.193940 12.3894599 2.058359e-1
## cyl6        -11.162351   9.355346 -1.1931522 2.435843e-0
## cyl8        -15.703167   4.839464 -3.2448150 3.223216e-0
## wt           -5.647025   1.359498 -4.1537586 3.127578e-0
## cyl6:wt       2.866919   3.117330  0.9196716 3.661987e-0
## cyl8:wt       3.454587   1.627261  2.1229458 4.344037e-0
```

Den Graphikoutput mit `layout` kontrollieren

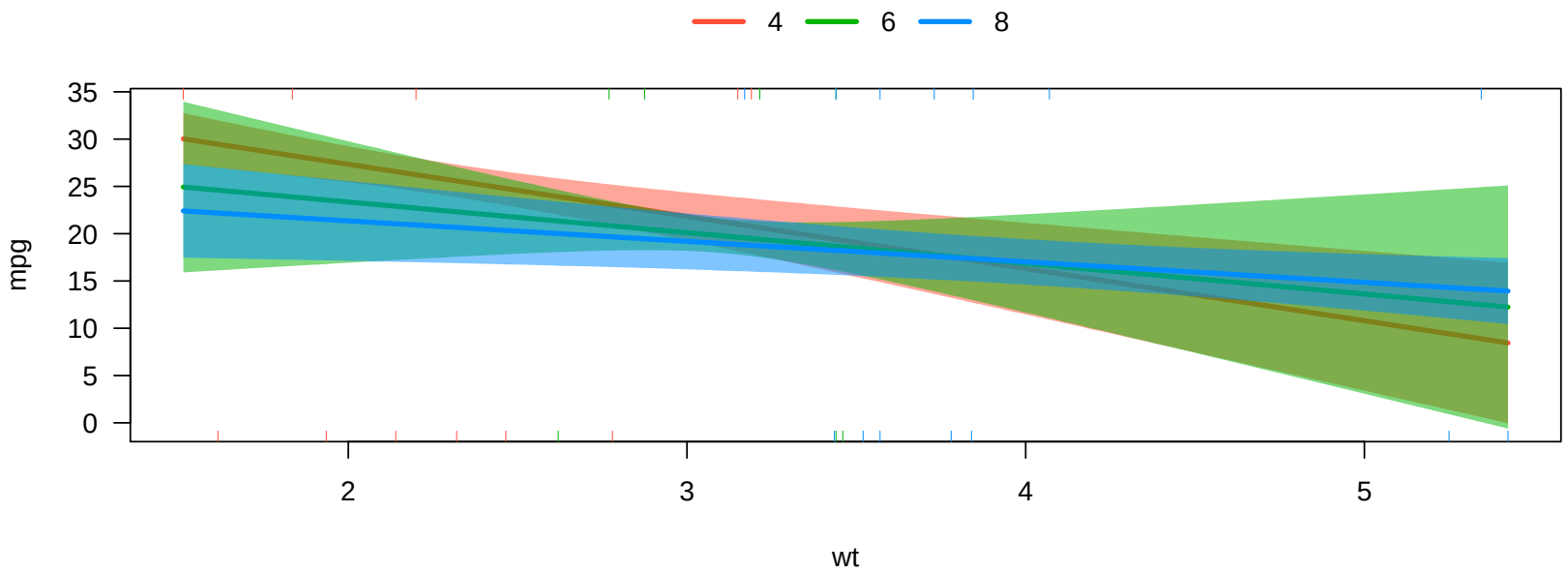
```
visreg(m5, "wt", by = "cyl",layout=c(3,1))
```



Das Paket `visreg` - Interaktionseffekte übereinander legen

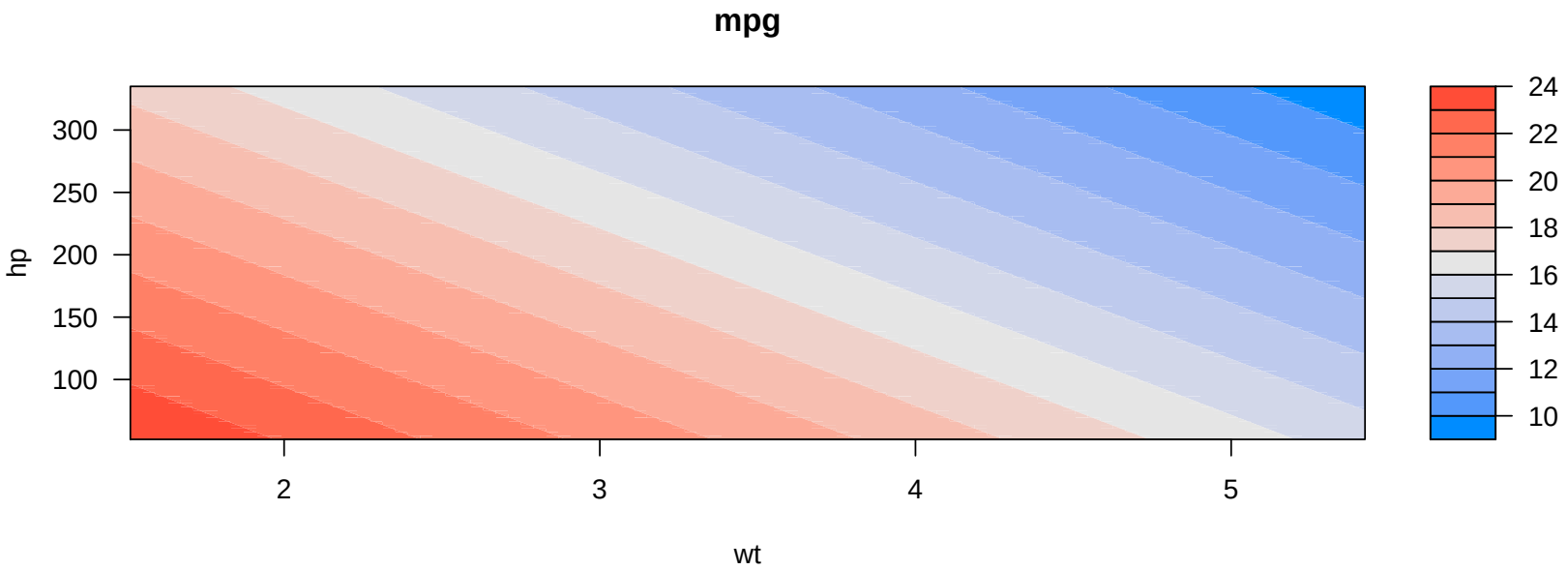
```
m6 <- lm(mpg ~ hp + wt * cyl, data = mtcars)
```

```
visreg(m6, "wt", by="cyl", overlay=TRUE, partial=FALSE)
```



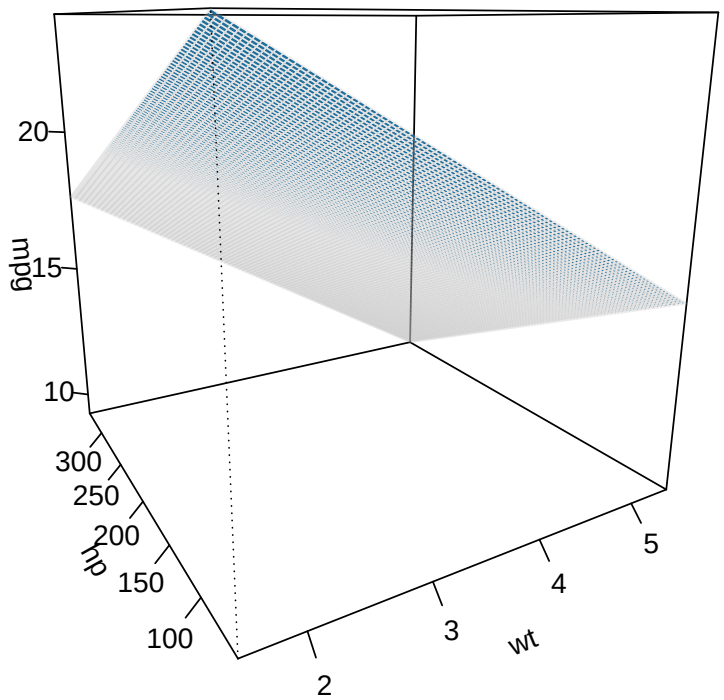
Das Paket visreg - visreg2d

```
visreg2d(m6, "wt", "hp", plot.type = "image")
```



Das Paket visreg - surface

```
visreg2d(m6, "wt", "hp", plot.type = "persp")
```



B3A Aufgabe lineare Regression

Der Datensatz `toycars` beschreibt die Route von drei Spielzeugautos, die Rampen in verschiedenen Winkeln absteigen.

- angle: Rampenwinkel
- distance: Entfernung die von dem Spielzeugauto zurück gelegt wird.
- car: Autotyp (1, 2 or 3)

a) Lese den Datensatz `toycars` ein und konvertiere die Variable `car` des Datensatzes in einen Faktor (`as.factor`).

(b) Erstelle drei Box-Plots, in denen die von den Autotypen zurückgelegte Strecke visualisiert wird.

B3A Aufgabe lineare Regression II

(c) Schätze für jeden Autotyp getrennt die Parameter des folgenden linearen Modell; nutze dafür die Funktion `lm()`

$$distance_i = \beta_0 + \beta_1 \cdot angle_i + \epsilon_i$$

(d) Überprüfe die Anpassung des Modells indem Du die drei Regressionslinien in den Scatterplot einzeichnest (`distance` gegen `angle`). Spricht das

$$R^2$$

für eine gute Modellanpassung?

Einen schönen Output mit dem Paket **stargazer**

erzeugen

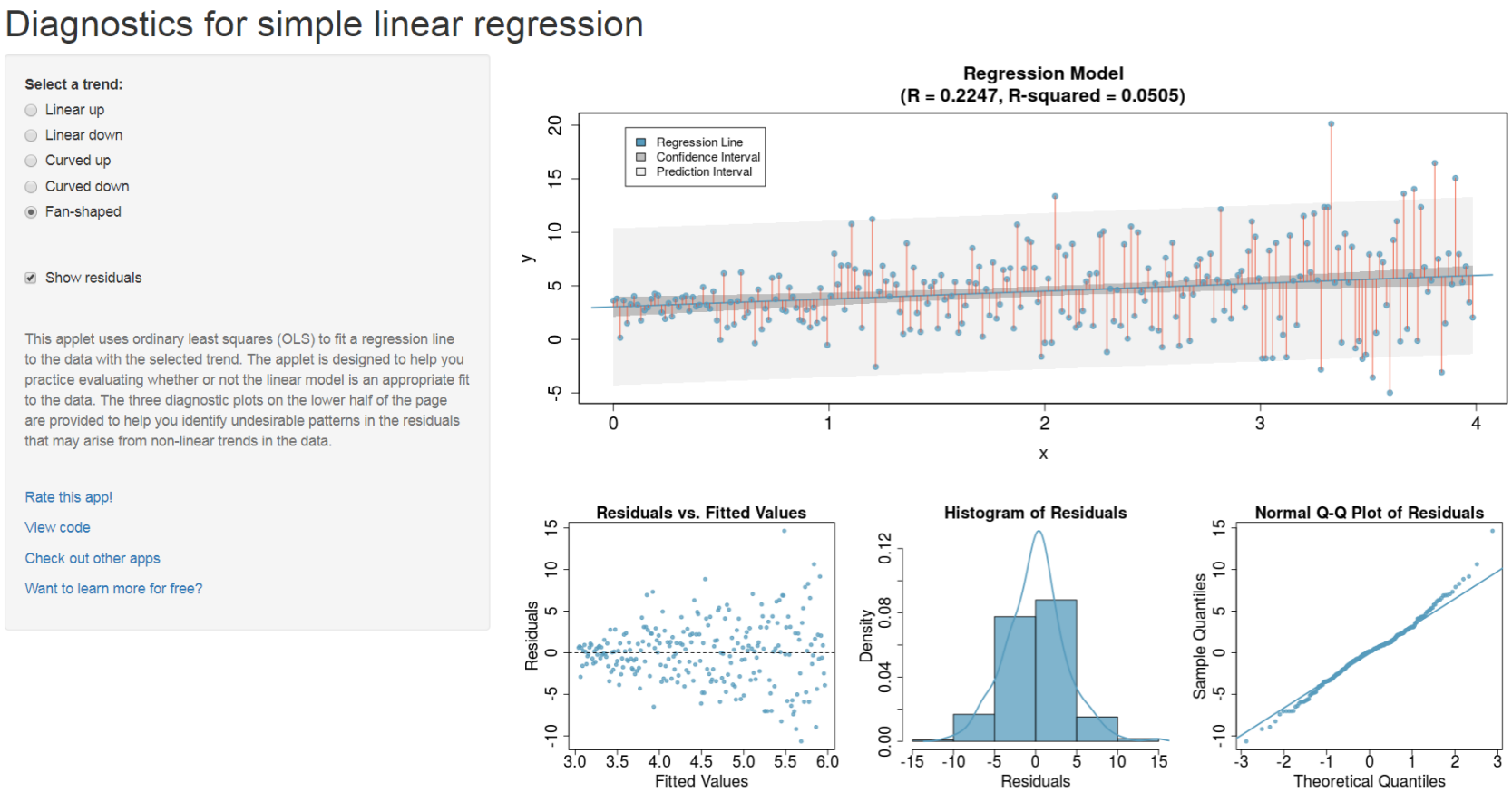
```
library(stargazer)
stargazer(m3, type="html")
```

Beispiel HTML Outputs:

	Dependent variable:
	mpg
wt	-3.125***
	(0.911)
cyl	-1.510***
	(0.422)
am	0.176
	(1.304)
Constant	39.418***
	(2.641)

Shiny App - Diagnostiken für die einfache lineare Regression

https://gallery.shinyapps.io/slr_diag/



- Shiny App - **Eine einfache lineare Regression**
- Shiny App - **Multikollinearität in multipler**

Links - lineare Regression

- Regression - **r-bloggers**
- Das komplette Buch von **Faraway**- sehr intuitiv geschriebenes Buch
- Gute Einführung auf **Quick-R**
- **Multiple Regression**
- **15 Arten von Regressionen die man kennen sollte**
- **`ggeffects` - Erzeuge saubere Datensätze mit marginellen Effekten für 'ggplot' aus Modell Outputs**

Die logistische Regression

Agresti - Categorical Data Analysis (2002)

- Sehr intuitiv geschriebenes Buch
- Sehr detailliertes Skript von **Laura A. Thompson**
- Das Buch behandelt die kategoriale Datenanalyse ganz grundsätzlich.

Faraway Bücher zum Thema Regression

- Logistische Regression eingängig erklärt
- Beispiel mit R-Code
  - Faraway - **Extending the linear model with R**
  - Faraway - **Practical Regression and Anova using R**

Importieren des GESIS Panels Datensatzes

```
library(readstata13)
path <- "D:/Daten/GitLab/IntroDataAnalysis/data/"
datf <- read.dta13(paste0(path,"ZA5666_v1-0-0_Stata14.dta"),
                   convert.factors = F)
```

Das Argument `convert.factors`:

- `logical`. Wenn TRUE, werden Faktoren aus dem Stata Werte Labeln erzeugt.

Eine Funktion um fehlende Werte zu rekodieren

```
code_miss <- function(var){
  misvals <- c(-11,-22,-33,-44,-55,-66,-77,-88,-99,-111)
  var[var %in% misvals] <- NA
  return(var)
}
```

Variablen für das glm

- a11d056z: Altersgruppe

```
table(datf$a11d056z)
```

```
##
## -99   1   2   3   4   5   6   7   8   9  10  11  12  13
##   5  31  87 101  91  83 100 163 159 133  64  56 105  44
```

```
age <- code_miss(datf$a11d056z)
```

```
table(age)
```

```
## age
##   1   2   3   4   5   6   7   8   9  10  11  12  13
##  31  87 101  91  83 100 163 159 133  64  56 105  44
```

GP Variable a11d094a: Kinder unter 16 Jahre

Leben in Ihrem Haushalt Kinder unter 16 Jahren?

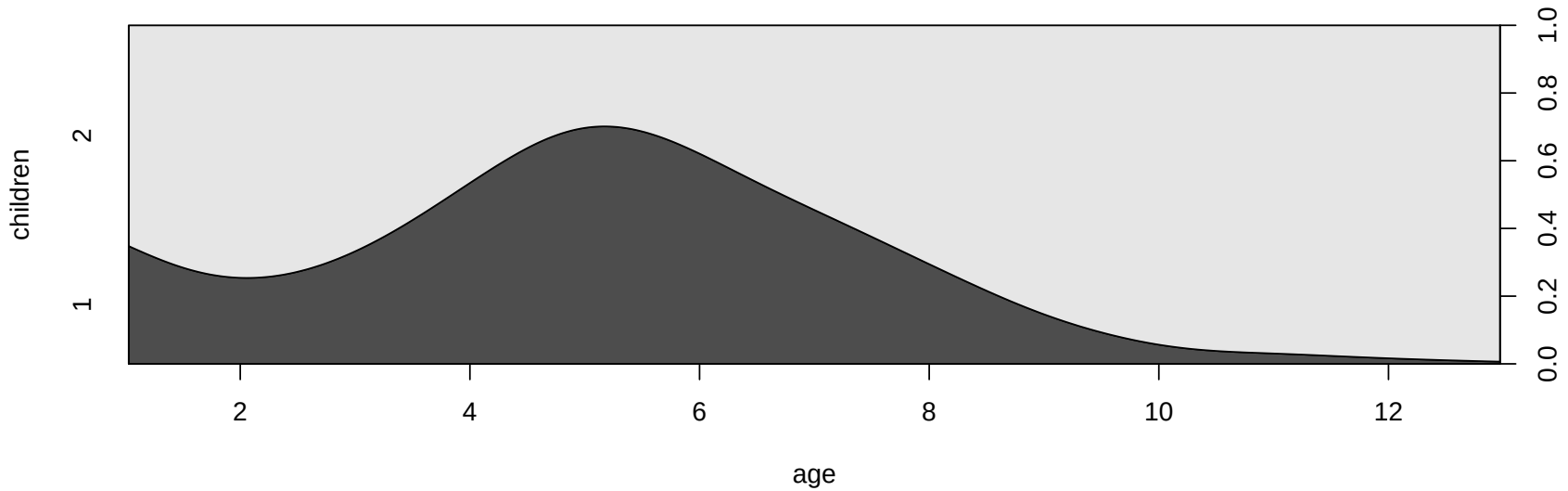
- 1 Ja
- 2 Nein

```
children <- as.factor(code_miss(datf$a11d094a))
table(children)
```

```
## children
##   1   2
## 325 681
```

Conditional Density Plot (GESIS Panel)

```
cdplot(children ~ age)
```



Binäre abhängige Variablen im `glm`

- Die **logistische Regression** ghört zur Klasse der generalisierten linearen Modellen (GLM)
- Die Funktion zur Schätzung eines Modells dieser Klasse heißt `glm()`

Ein `glm` spezifizieren

- Formel-Objekt
- die Klasse (binomial, gaussian, gamma)
- mit einer Link Funktion (logit, probit, cauchit, log, cloglog)

muss spezifiziert

Logistische Regression mit R

```
glm_1 <- glm(children ~ age,
                    family = binomial())
```

```
sum_glm1 <- summary(glm_1)
sum_glm1$coefficients
```

```
##                Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -0.7194058 0.16384386 -4.390801 1.129338e-05
## age          0.2225862 0.02376266  9.367056 7.458415e-21
```

Die Koeffizienten interpretieren

Wir betrachten das logistische Modell der Kinder im Haushalt als eine Funktion des Alters.

```
sum_glm1$coefficients
```

```
##                Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -0.7194058 0.16384386 -4.390801 1.129338e-05
## age          0.2225862 0.02376266  9.367056 7.458415e-21
```

- Die Schätzungen und Standardfehler werden mit Log Odds angegeben, nicht mit der Wahrscheinlichkeit.
- Die p-Werte bedeuten das Gleiche, wie bei der linearen Regression.

Der inverse Logit

```
sum_glm1$coefficients
```

```
##                Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -0.7194058 0.16384386 -4.390801 1.129338e-05
## age          0.2225862 0.02376266  9.367056 7.458415e-21
```

- Die Koeffizienten können nicht so einfach interpretiert werden
- Wir müssen den inversen Logit verwenden, um etwas auszusagen.

Werte für die Log-odds von 0.2225862 sind das Gleiche, wie die Wahrscheinlichkeit: 0.5554179.

```
faraway::ilogit(sum_glm1$coefficients[1,1])
```

```
## [1] 0.3275238
```

Zum Achsenabschnitt in einem logistischen Modell

- Es ist möglich, dass der Schätzwert für den Achsenabschnitt kleiner als null ist.
- Das bedeuted, dass die log-odds negativ sind und NICHT die Wahrscheinlichkeit.
- Ein Log-Odd Wert von 0 bedeuted eine Wahrscheinlichkeit von 0.5.

Log-odds und die Wahrscheinlichkeit

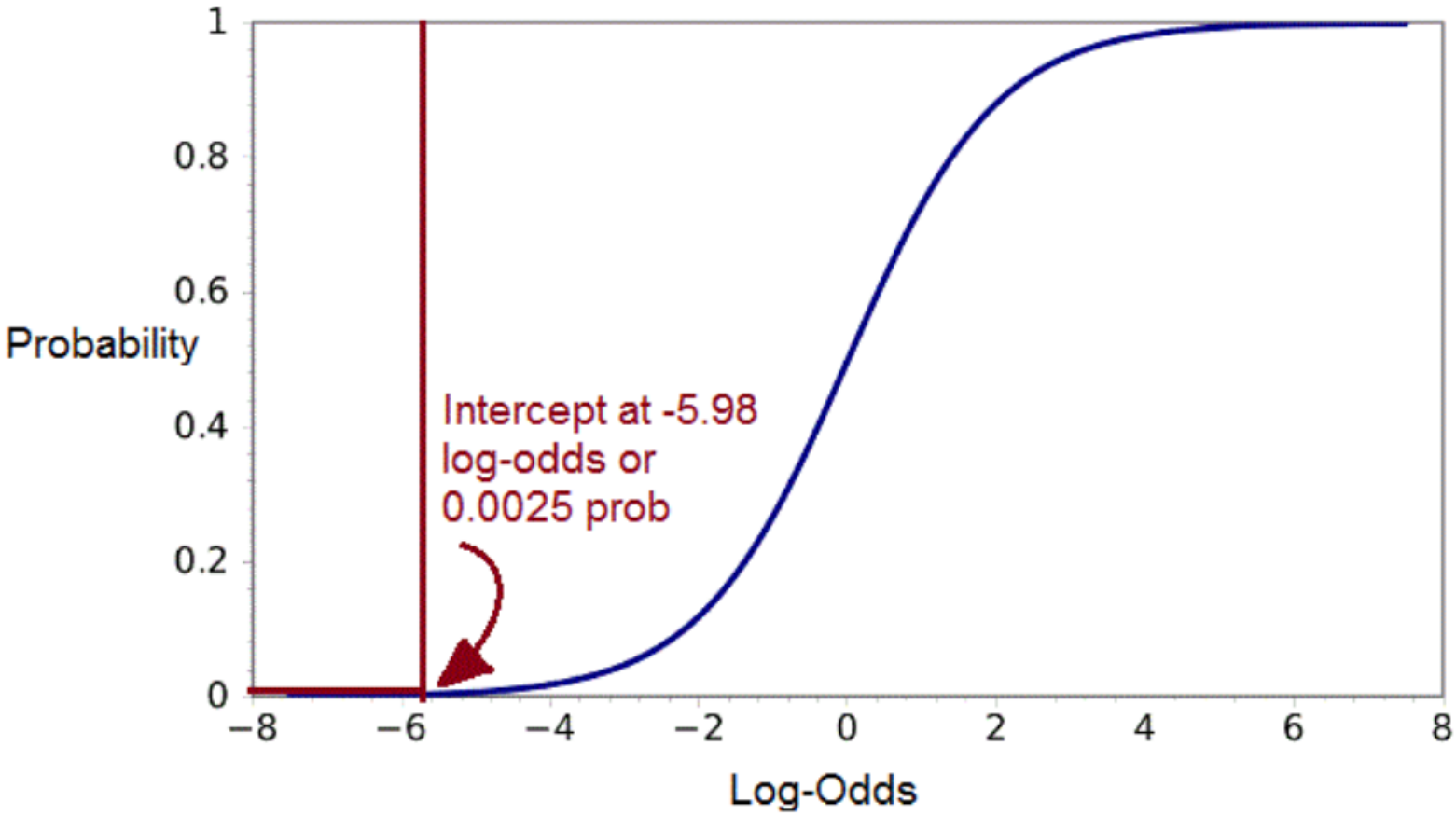
- Die Log-odds steigen an, wenn die Wahrscheinlichkeit auch ansteigt.

Daraus folgt...

- Ein postivier Steigungskoeffizeint bedeutet, dass der Response-Wert mit zunehmenden Wert für die erklärende Variable auch zunimmt.
- In unserem Fall heißt das: Die Wahrscheinlichkeit, dass sich im Haushalt Kinder befinden steigt mit dem Alter des Befragten.

Das Ergebnis graphisch darstellen

Es resultiert eine Sigmoid-Kurve, anstatt einer Gerade mit konstanter Steigungsrate wie bei der linearen Regression.



Logistische Regressionsformel

Das Modell als Formel:

Log-Odds(Children) = -0.7194058 + 0.2225862(Age) + Fehler

- Wir können Werte in die Formel einsetzen um die vorhergesagten Log-Odds für unterschiedliche Altersklassen zu bekommen.

Beispiel: Log-Odds für die Altersgruppe 5

-0.7194058 + 0.2225862*(5) = 0.3935251

Wahrscheinlichkeit für Kinder in der Altersgruppe 5

```
ilogit(0.3935251)
```

```
## [1] 0.597131
```

Die Ergebnisse interpretieren

```
anova(glm_1, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: children
##
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                  1000       1259
## age   1   98.956        999       1160 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
```

Die Devianz

Abweichung vom Idealwert

- Zweimal die Differenz zwischen der maximalen Log-Likelihood $\ell^{(M)}$ und dem Wert für das angepasste Modell
- Je niedriger die Devianz, desto besser.

```
sum_glm1 <- summary(glm_1)
sum_glm1$deviance
```

```
## [1] 1160.044
```

Mc Fadden's R^2

```
library(pscl)
pR2(glm_1)
```

```
##              llh       llhNull            G2      McFadden
## -580.02210772 -632.93066002  105.81710461     0.08359297
##              r2CU
##     0.13978426
```

`llh`	The log-likelihood from the fitted model
`llhNull`	The log-likelihood from the intercept-only restricted model
`G2`	Minus two times the difference in the log-likelihoods
`McFadden`	McFadden's pseudo r-squared
`r2ML`	Maximum likelihood pseudo r-squared
`r2CU`	Cragg and Uhler's pseudo r-squared

Großstadtnähe Wohngegend

Wie weit ist es von Ihrer Wohnung bis ins Zentrum der nächsten Großstadt?

- 1 - Im Großstadtzentrum
- 6 - 60 km und mehr

```
region <- code_miss(datf$bczd001a)
table(region)
```

```
## region
##   1   2   3   4   5   6
##  87 191 279 157 126 165
```

Zufriedenheit Leben in Wohnort

Alles in allem, wie zufrieden sind Sie mit dem Leben in [Wohnort]?

- 1 - Sehr zufrieden
- 5 - Sehr unzufrieden

```
satisfactionplace <- datf$a11c019a
table(satisfactionplace)
```

```
## satisfactionplace
##   1   2   3   4   5
## 553 534  99  30   6
```

Ein anderes Modell

```
glm_2 <- glm(children ~ age + satisfactionplace*region,
                        family = binomial())
```

```
pseudor2 <- pR2(glm_2)
pseudor2["McFadden"]
```

```
## McFadden
## 0.258121
```

Eine weitere Variable aus dem GESIS Panel Datensatz

- Anzahl Tattoos:

```
Tatoos <- code_miss(datf$bdao067a)
Tatoos[Tatoos==97]<-0
```

```
table(Tatoos)
```

```
## Tatoos
##   0   1   2   3   4   5   6
## 871  56  28  13   7   4   8
```

Generalisierte Regression mit R - Mehr Funktionen

- Logistisches Modell mit einem Probit Link:

```
probitmod <- glm(children ~ age,
    family=binomial(link=probit))
```

- Regression mit Zähldaten:

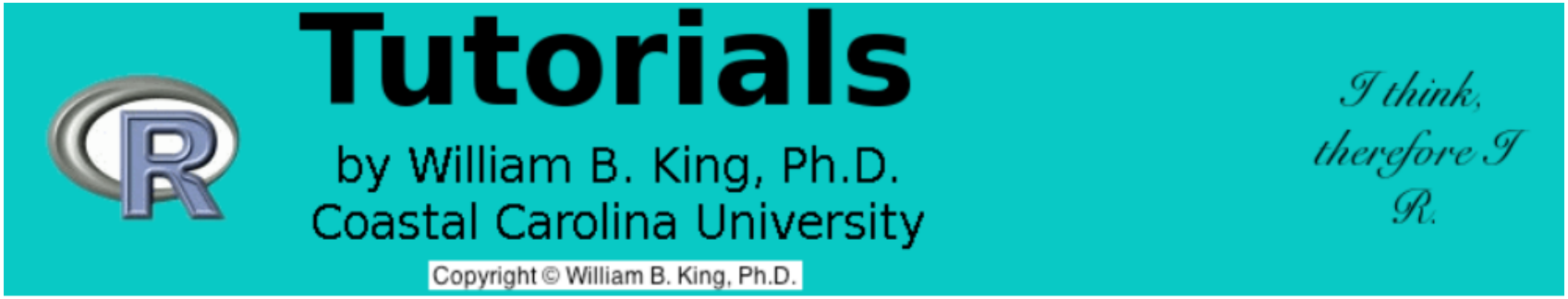
```
modp <- glm(Tatoos ~ age,family=poisson)
```

- Proportional Odds logistic Regression aus dem Paket `MASS`:

```
library("MASS")
mod_plr<-polr(a11c020a ~ a11d096b ,data=dat)
```

Linkliste - logistische Regression

- Einführung in die **logistische Regression**



- **Code zum Buch von Faraway**

www.maths.bath.ac.uk/~jjf23/ELM/scripts/binary.R

```
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")
```

- **Kategoriale Daten: - Durchführung logistische Regression in R**

Aufgabe Datenanalyse

Aufgabe - Datenanalyse

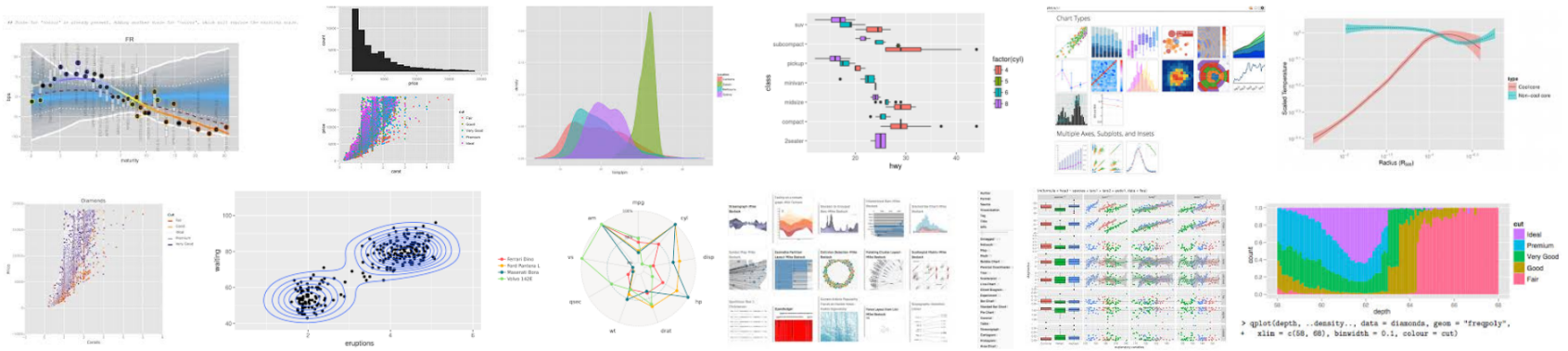
- Laden Sie einen Datensatz Ihrer Wahl - entweder einen eigenen oder einen der vorgestellten Datensätze
- Berechnen Sie einfache Statistiken auf den wichtigsten Variablen (Mittelwert, Median, Standardabweichung)
- Erzeugen Sie eine zweidimensionale Häufigkeitstabelle
- Führen Sie eine Regression mit sinnvoll gewählten abhängiger und unabhängiger Variablen auf den Daten durch
- Erzeugen Sie einen Lattice-plot

Zurück zur Gliederung.

Grafiken mit ggplot

Das Paket ggplot2

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



Einführung in ggplot2

http://www.r-bloggers.com/basic-introduction-to-ggplot2/

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

```
library(ggplot2)
```

```
?ggplot2
```

ggplot2-package {ggplot2} R Documentation

ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics

Description

A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

Author(s)

Maintainer: Hadley Wickham hadley@rstudio.com

Authors:

- Winston Chang winston@rstudio.com

Other contributors:

- RStudio [copyright holder]

Der `diamonds` Datensatz

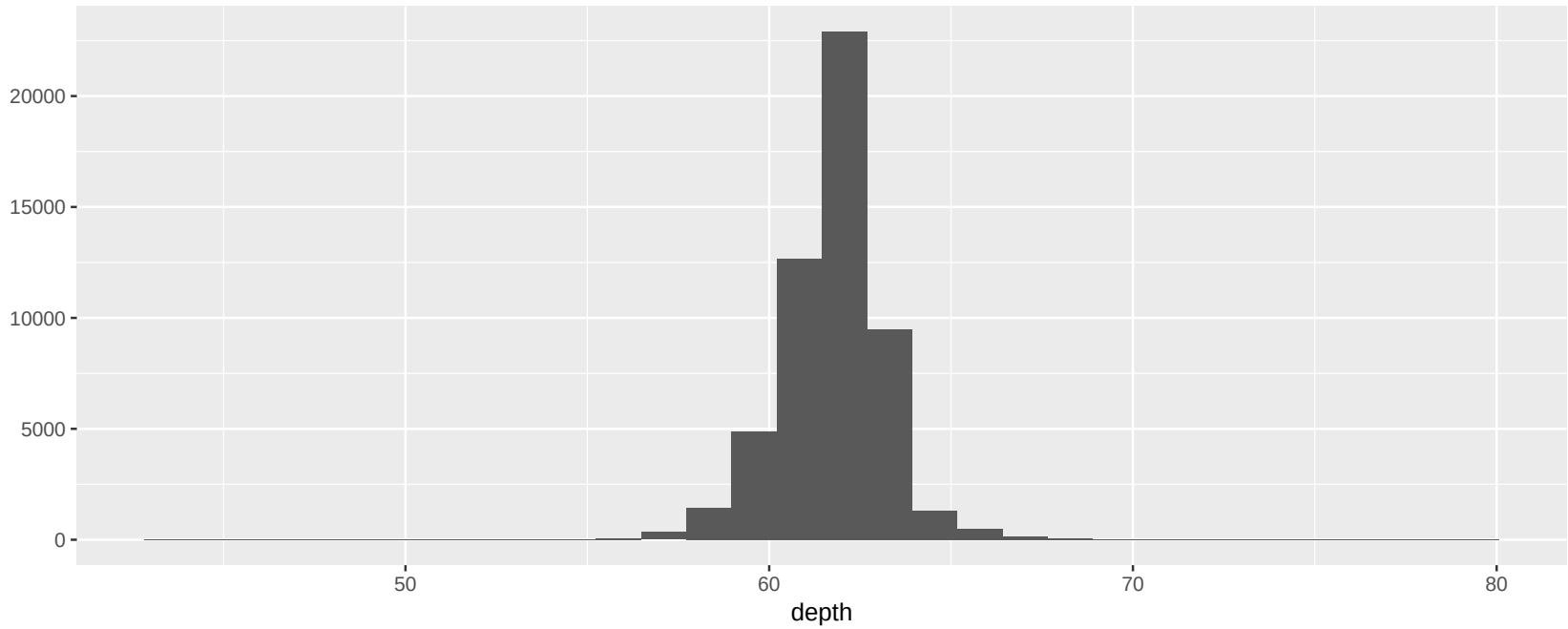
```
head(diamonds)
```

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

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

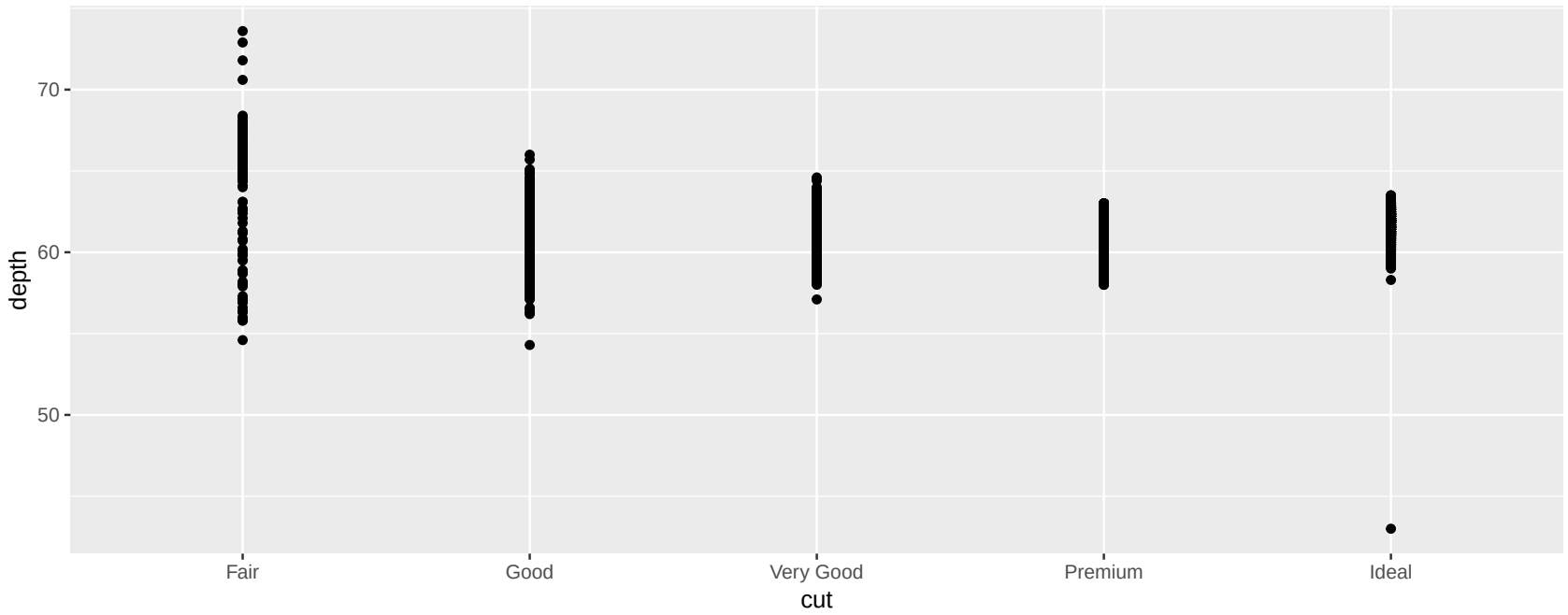
```
# histogram
qplot(depth, data=diamonds)
```




```
diamonds <- diamonds[sample(1:nrow(diamonds),5000),]
```

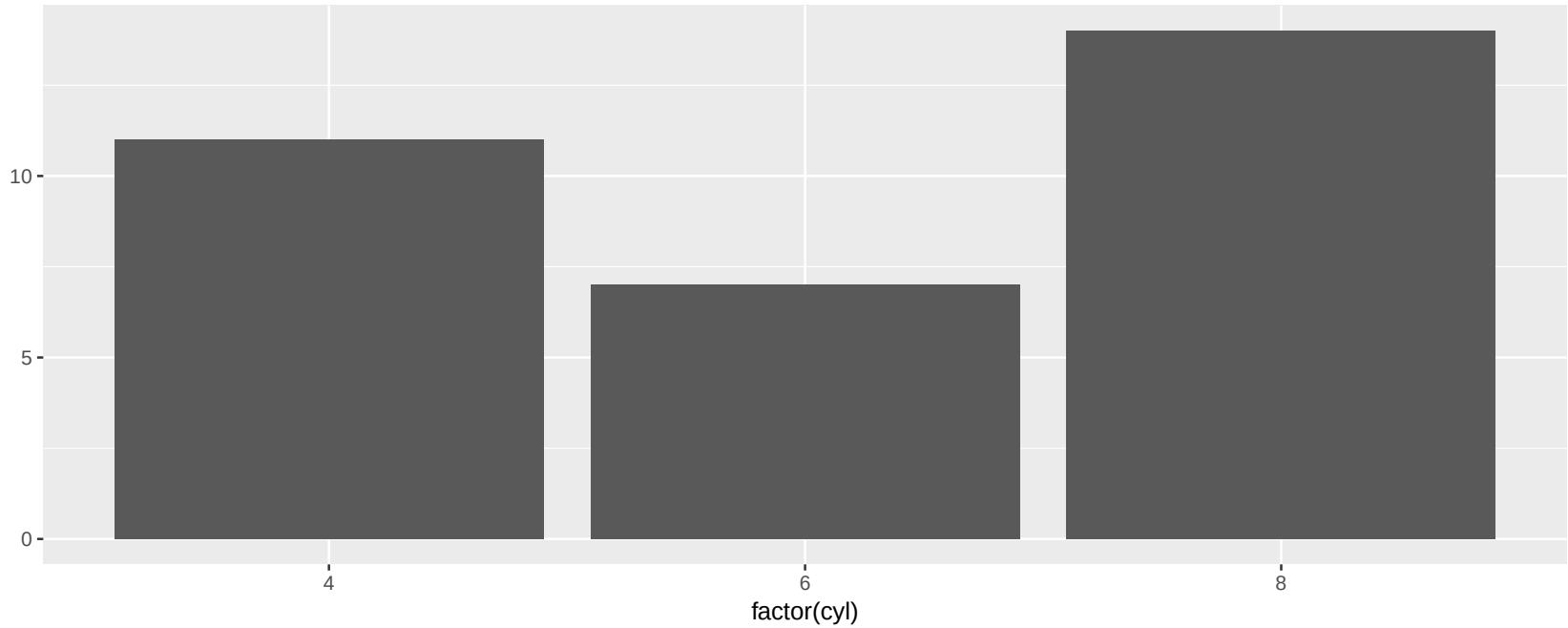
Ein Balkendiagramm

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



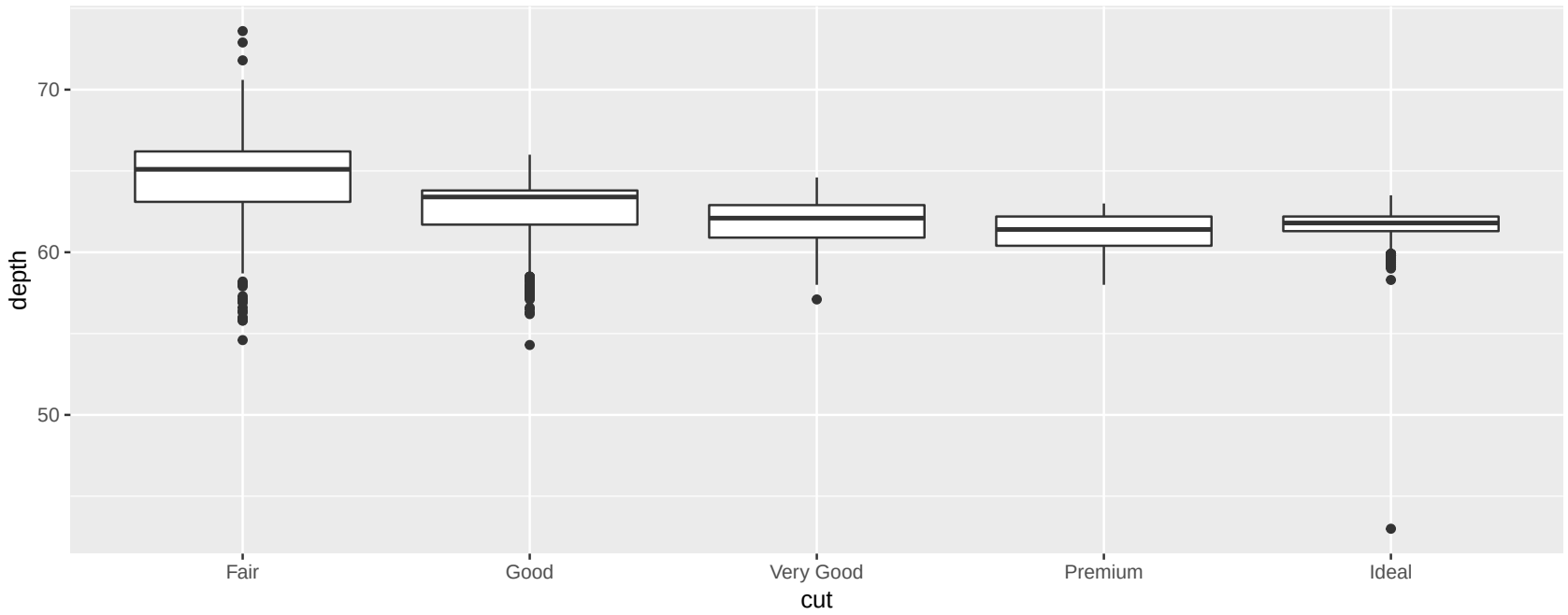
Ein weiteres Balkendiagramm

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



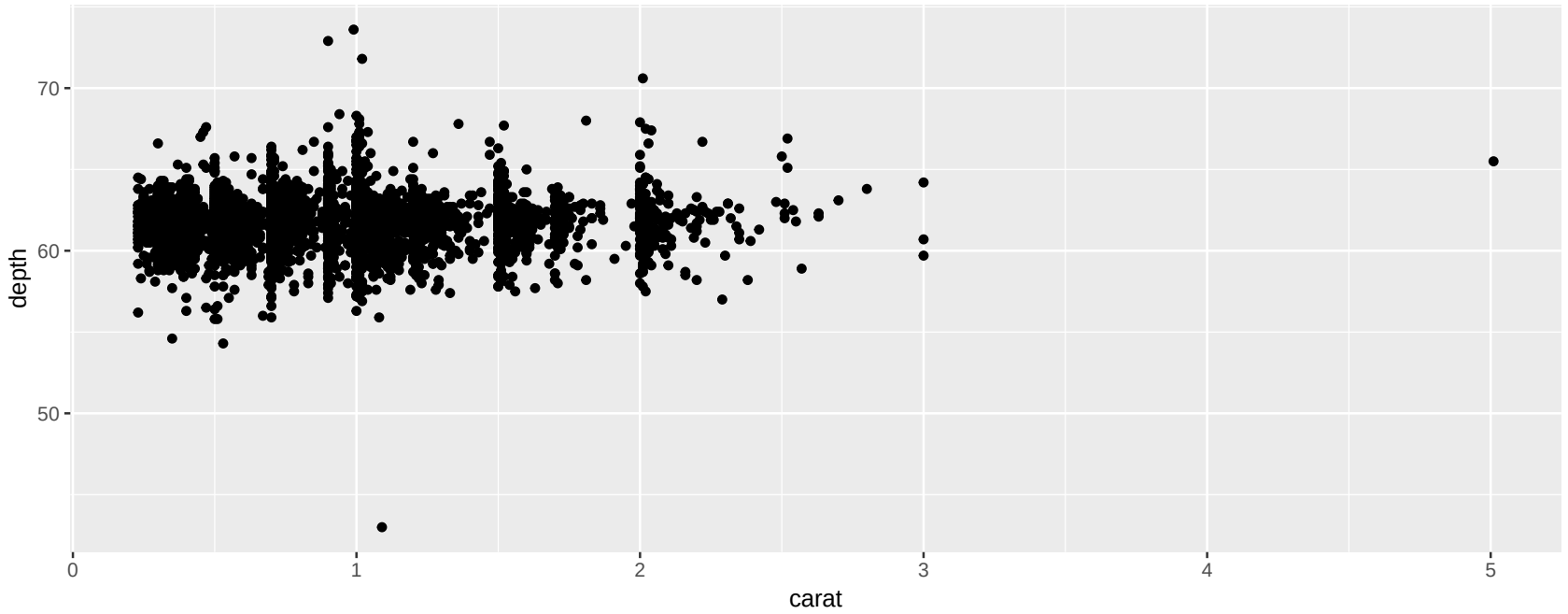
Boxplot

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



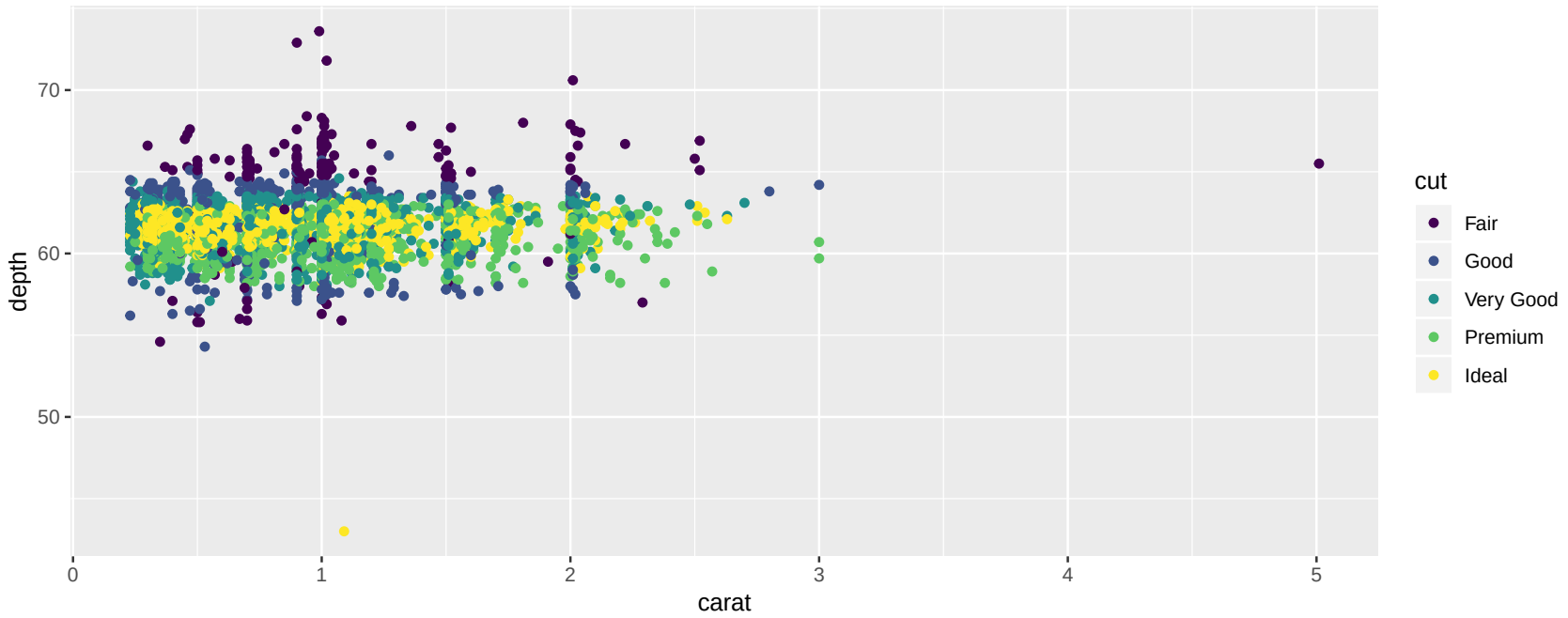
Scatterplot

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



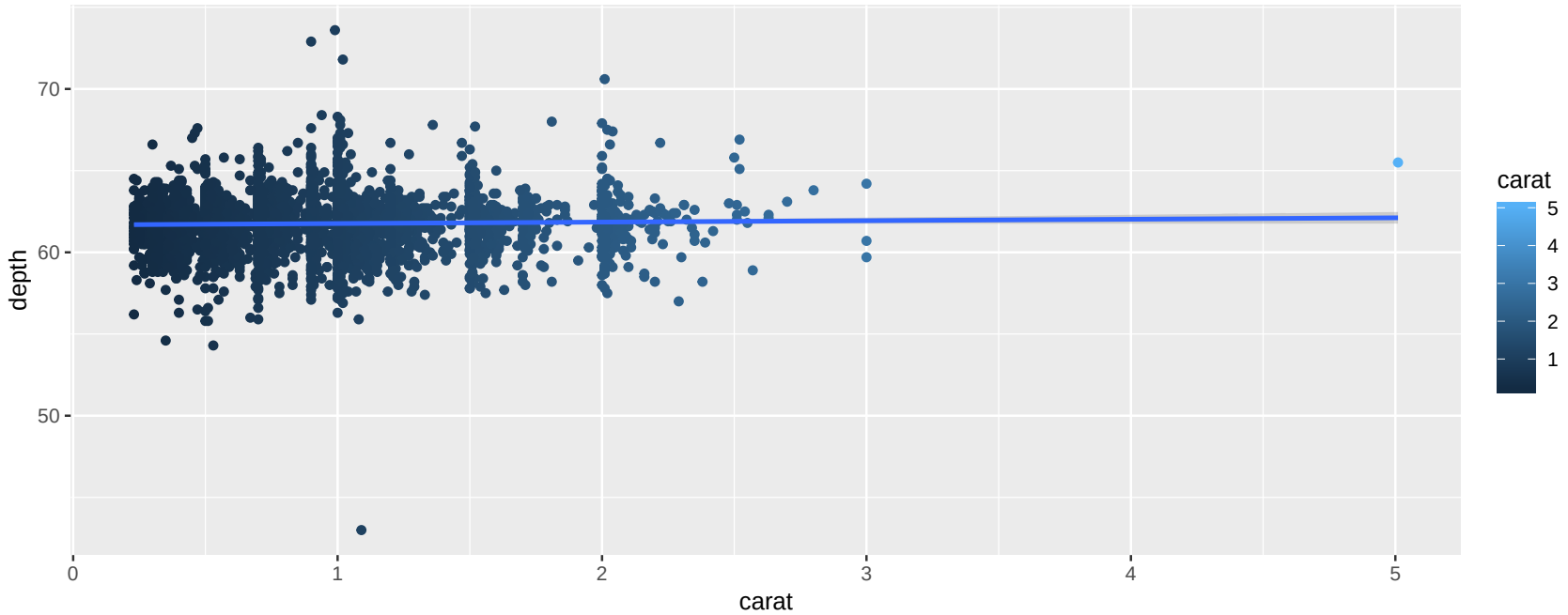
Farbe hinzu:

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



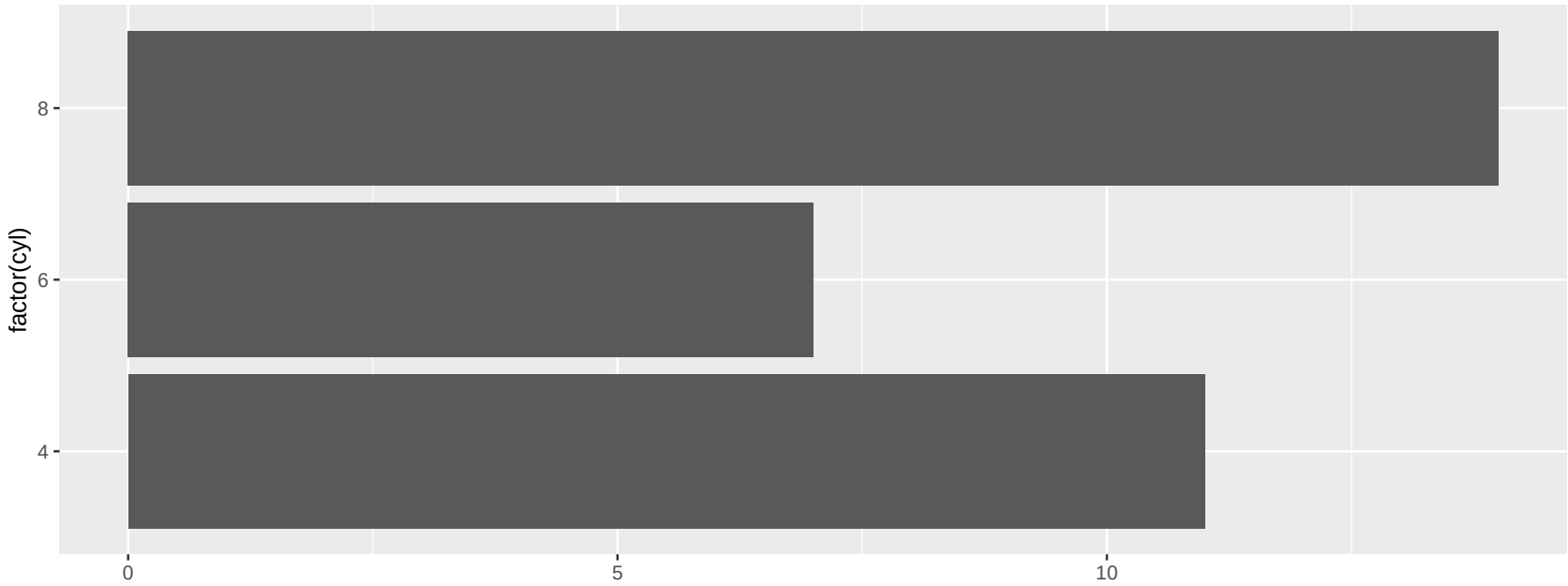
Trendlinie hinzufügen

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



Graphik drehen

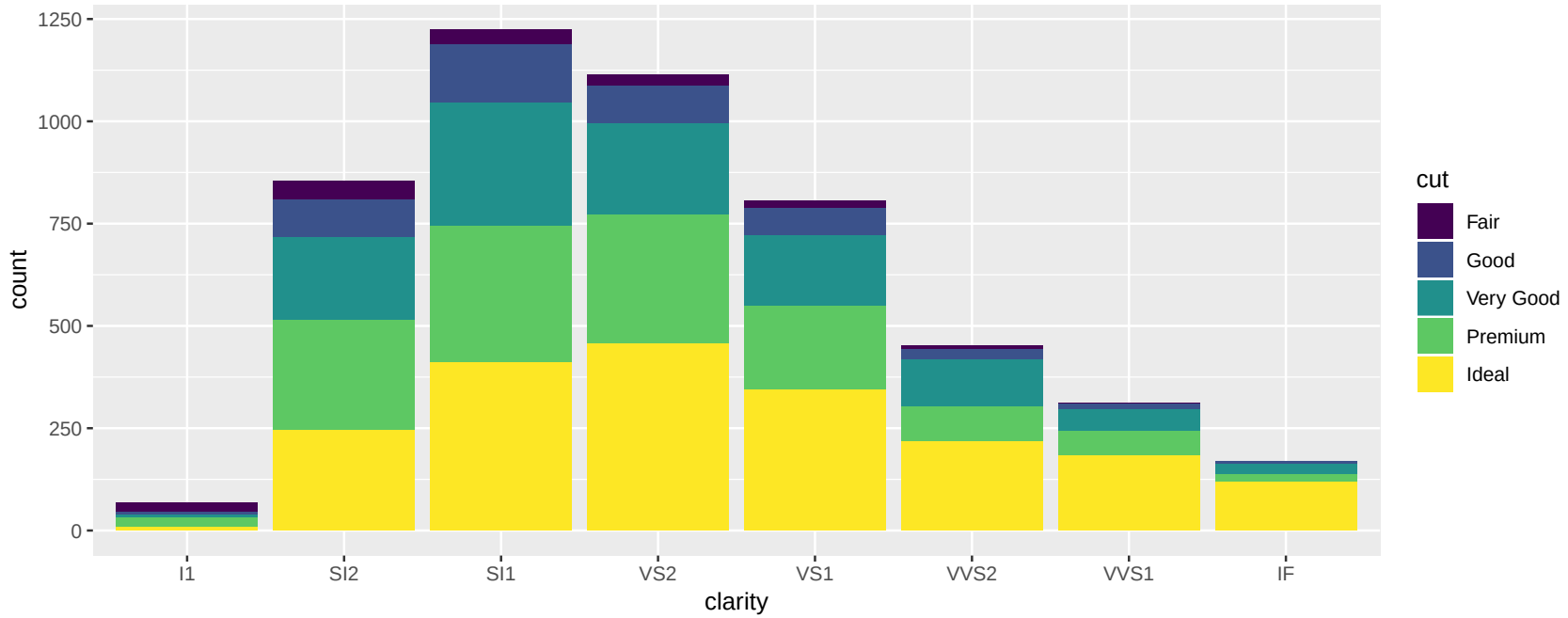
```
qplot(factor(cyl), data=mtcars, geom="bar") +
coord_flip()
```



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

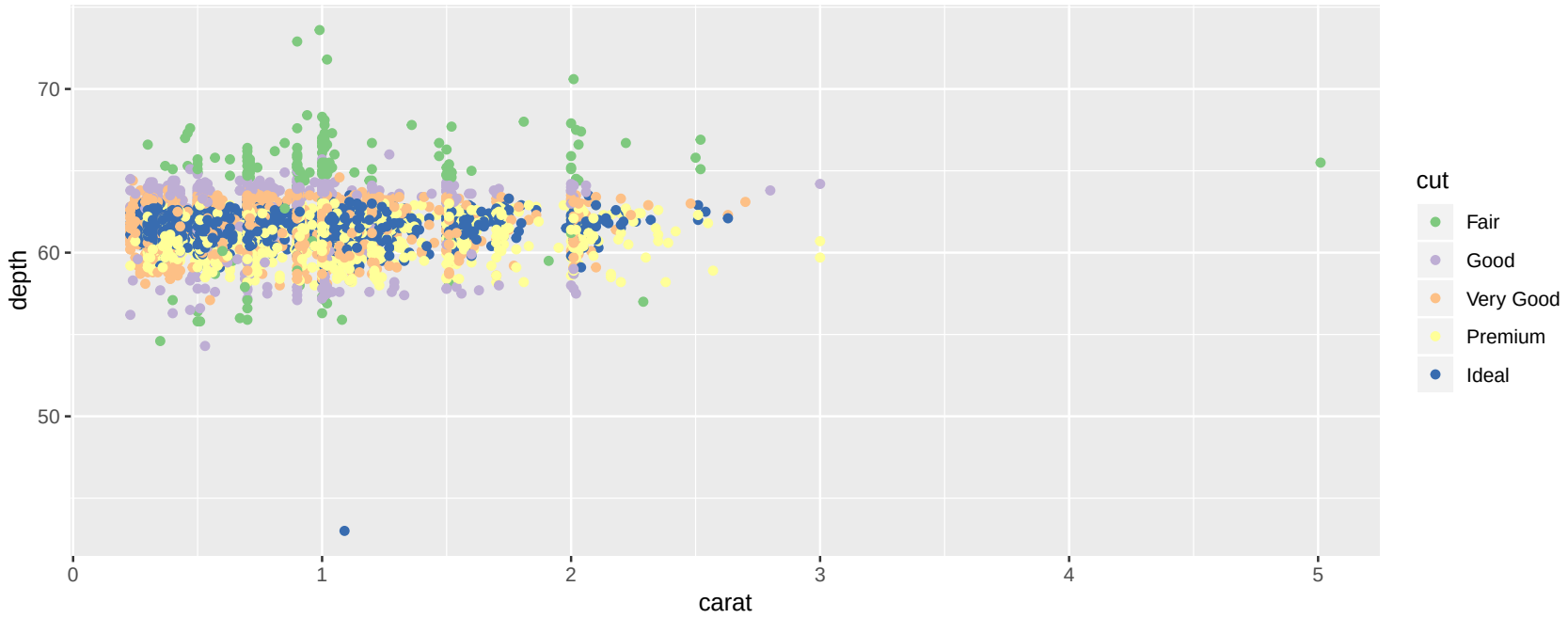
```
install.packages("RColorBrewer")
```

```
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")
```

- 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

Introduction to ggplot2