

# EINFÜHRUNG IN R - EINFACH GRAPHIKEN

Jan-Philipp Kolb

11 Juni, 2019

# EINE GRAPHIK SAGT MEHR ALS 1000 WORTE.

```
sysinfo <- Sys.info()
if (sysinfo["nodename"]=="MAC14077"){
  GESIS <- T
}else{
  GESIS <- F
}
athome <- !GESIS
```

## AUSSAGEN ZU GRAPHEN IN R

- Die grafische Datenanalyse ist großartig.
- Gute Graphiken können zu einem besseren Verständnis beitragen.
- Die Erzeugung eines Plot ist einfach.
- Einen guten Plot zu erstellen, kann sehr lange dauern.
- Das Erstellen von Plots mit R macht Spaß.
- Mit R erstellte Diagramme haben eine hohe Qualität.
- Fast jedes Graphikformat wird von R unterstützt.
- Eine große Anzahl von Exportformaten ist in R verfügbar.

# NICHT ALLE DIAGRAMME SIND GLEICH.

- Das Basispaket enthält bereits eine Vielzahl von Plotfunktionen.
- Andere Pakete wie `lattice`, `ggplot2`, etc. erweitern diese Funktionalität.

## HANDBÜCHER, DIE WEIT ÜBER DIESE EINFÜHRUNG HINAUSGEHEN:

- Murrell, P (2006): R Graphics.
- R Development Core Group **Graphiken mit R**
- Wiki zu **R Programmierung/Graphiken**
- Martin Meermeyer **Creating Reproducible Publication Quality Graphics with R: A Tutorial**
- Institute for Quantitative Social Science at Harvard - **R Graphik Tutorial**

# TASK VIEW FÜR GRAPHIKEN

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

**Maintainer:** Nicholas Lewin-Koh

**Contact:** nikko at hainmail.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.

R's base graphics are implemented in the same way as in the S3 system developed by Becker, Chambers, and Wilks. There is a static device, which is treated as a static canvas and objects are drawn on the device through R plotting commands. The device has a set of global parameters such as margins and layouts which can be manipulated by the user using `par()` commands. The R graphics engine does not maintain a user visible graphics list, and there is no system of double buffering, so objects cannot be easily edited without redrawing a whole plot. This situation may change in R 2.7.x, where developers are working on double buffering for R devices. Even so, the base R graphics can produce many plots with extremely fine graphics in many specialized instances.

One can quickly run into trouble with R's base graphic system if one wants to design complex layouts where scaling is maintained properly on resizing. nested graphs are desired or more interactivity is needed. grid was designed by Paul Murrell to overcome some of these limitations and as a result packages like [lattice](#), [gridplot2](#), [grid](#) or [hexbin](#) use grid for the underlying primitives. When using plots designed with grid one needs to keep in mind that grid is based on a system of viewports and graphic objects. To add objects one needs to use grid commands, e.g., `grid.polygon()` rather than `polygon()`. Also grid maintains a stack of viewports from the device and one needs to make sure the desired viewport is at the top of the stack. There is a great deal of explanatory documentation included with grid as vignettes.

The graphics packages in R can be organized roughly into the following topics, which range from the more user oriented at the top to the more developer oriented at the bottom. The categories are not mutually exclusive but are for the convenience of presentation:

<https://cran.r-project.org/web/views/Graphics.html>

# DATEN IMPORTIEREN

```
load("../data/bauenwohnen_teil.RData")
```

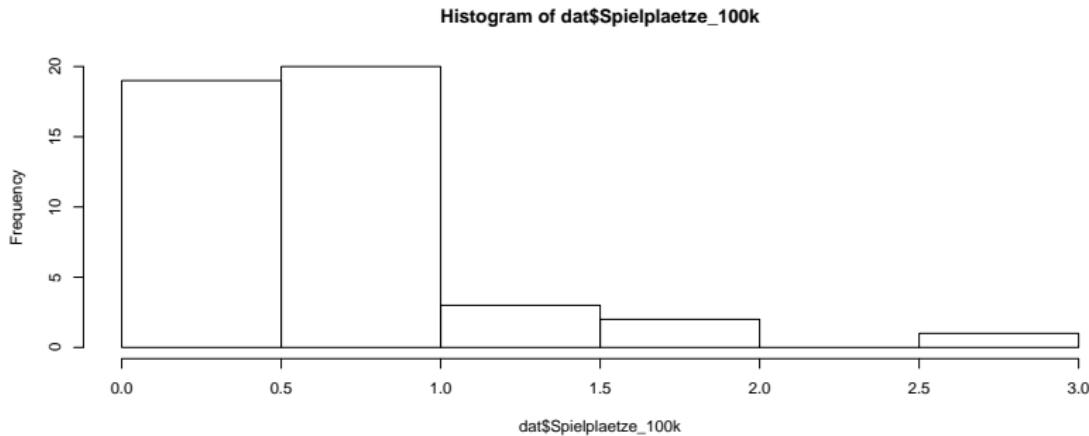
Stadtteil	bautaet	Spielplaetze_100k	Spiel100K	baugenehm12	wohnungsbestand	wohnraumversorgung_k
Altstadt	NA	2.7	viele	NA	282	1.631
Innenstadt	NA	2.0	viele	NA	352	1.490
Bahnhofsviertel	5	0.7	wenig	NA	152	1.310
Westend-Sued	45	0.2	wenig	NA	1291	1.540
Westend-Nord	40	0.9	wenig	NA	688	1.760
Nordend-West	7	0.3	wenig	NA	2361	1.650
Nordend-Ost	9	0.6	wenig	NA	2024	1.560
Ostend	14	0.4	wenig	NA	1595	1.610
Bornheim	93	0.3	wenig	NA	2157	1.650
Gutleutviertel	17	1.8	mittel	NA	291	1.650
Gallus	340	0.7	wenig	NA	1948	1.730
Bockenheim	150	0.8	wenig	NA	2641	1.600
Sachsenhausen-Nord	87	0.4	wenig	NA	2391	1.680
Sachsenhausen-Sued	70	0.6	wenig	NA	3146	1.600

# HISTOGRAMM - DIE FUNKTION HIST()

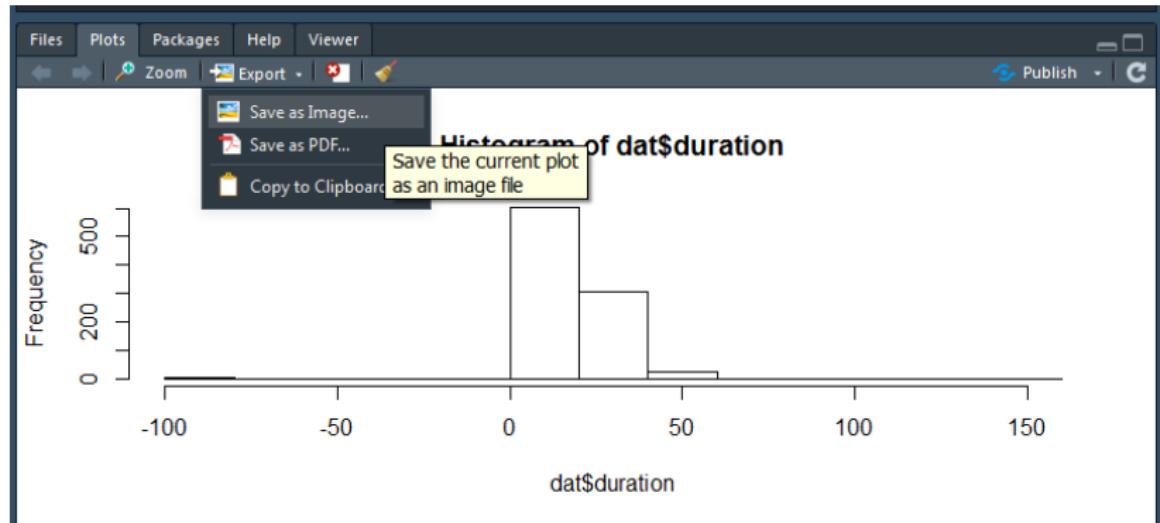
Wir erstellen ein Histogramm der Variablen Dauer:

```
?hist
```

```
hist(dat$Spielplaetze_100k)
```



# EXPORT MIT RSTUDIO



# BEFEHL ZUM SPEICHERN DER GRAFIK

- Alternativ auch mit den Befehlen png, pdf oder jpeg zum Beispiel.

```
png("Histogramm.png")
  hist(dat$Spielplaetze_100k)
dev.off()
```

```
pdf("Histogramm.pdf")
  hist(dat$Spielplaetze_100k)
dev.off()
```

```
jpeg("Histogramm.jpeg")
  hist(dat$Spielplaetze_100k)
dev.off()
```

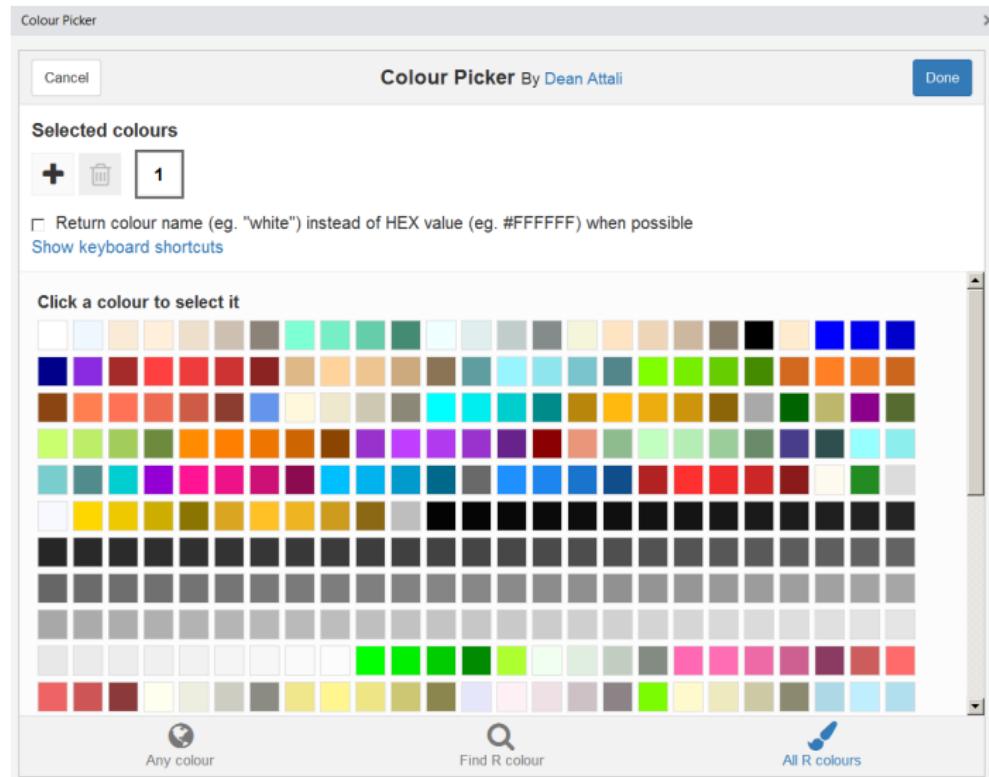
# HISTOGRAMM

- Der Befehl `hist()` zeichnet ein Histogramm.
- Mindestens ein Beobachtungsvektor muss an die Funktion übergeben werden.
- `hist()` hat viele weitere Argumente, die alle (sinnvolle) Standardwerte haben.

```
hist(dat$Spielplaetze_100k,col="blue",
      main="Spielplätze je 100 Kinder 2012",ylab="Frequency",
      xlab="Spielplätze")
```

# RSTUDIO ADDIN COLOURPICKER

```
install.packages("colourpicker")
```



# WEITERE ARGUMENTE:

```
?plot  
# or  
?par
```

## Graphical Parameters

adj

The value of `adj` determines the way in which text strings are justified in `text`, `mtext` and `title`. A value of 0 produces left-justified text, 0.5 (the default) centered text and 1 right-justified text. (Any value in [0, 1] is allowed, and on most devices values outside that interval will also work.)

Note that the `adj` argument of `text` also allows `adj = c(x, y)` for different adjustment in x- and y- directions. Note that whereas for `text` it refers to positioning of text about a point, for `mtext` and `title` it controls placement within the plot or device region.

ann

If set to `FALSE`, high-level plotting functions calling `plot.default` do not annotate the plots they produce with axis titles and overall titles. The default is to do annotation.

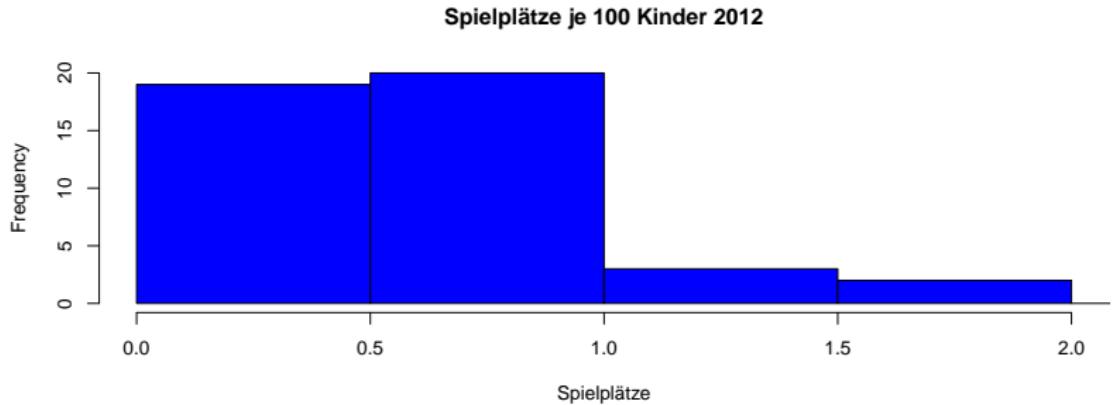
ask

logical. If `TRUE` (and the R session is interactive) the user is asked for input, before a new figure is drawn. As this applies to the device, it also affects output by packages `grid` and `lattice`. It can be set even on non-screen devices but may have no effect there.

This not really a graphics parameter, and its use is deprecated in favour of `devAskNewPage`.

# DAS `xlim` ARGUMENT

```
hist(dat$Spielplaetze_100k,col="blue",
     main="Spielplätze je 100 Kinder 2012",ylab="Frequency",
     xlab="Spielplätze",xlim=c(0,2))
```



# DAS BREAKS ARGUMENT

- Während die vorherigen Argumente für viele Grafikfunktionen gelten, gilt das Folgende hauptsächlich für Histogramme:

```
hist(dat$Spielplaetze_100k,col="red",
      main="Spielplätze je 100 Kinder 2012",ylab="Häufigkeit",
      xlab="Spielplätze",breaks=20)
```

- Mit `breaks` kann man die Zahl der Balken kontrollieren:

# TABELLIEREN UND BARPLOT

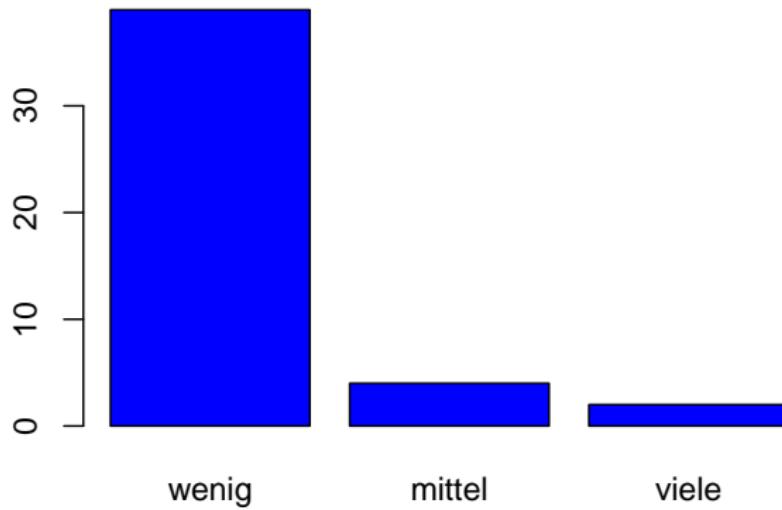
- Der Befehl `barplot()` erzeugt einen Barplot aus einer Frequenztabelle.
- Wir erhalten die Tabelle mit dem folgenden Befehl:

```
tab_spiel <- table(dat$Spiel100K)
```

```
barplot(tab_spiel)
```

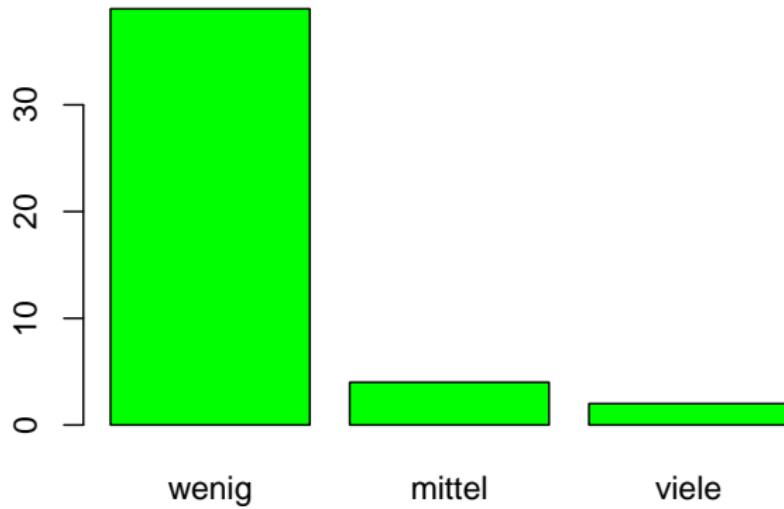
# MEHR FARBE:

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



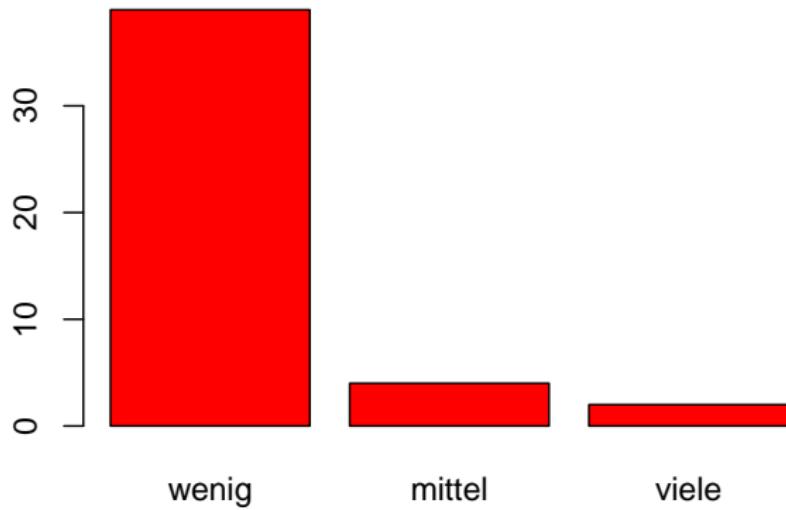
# GRÜNE FARBE

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



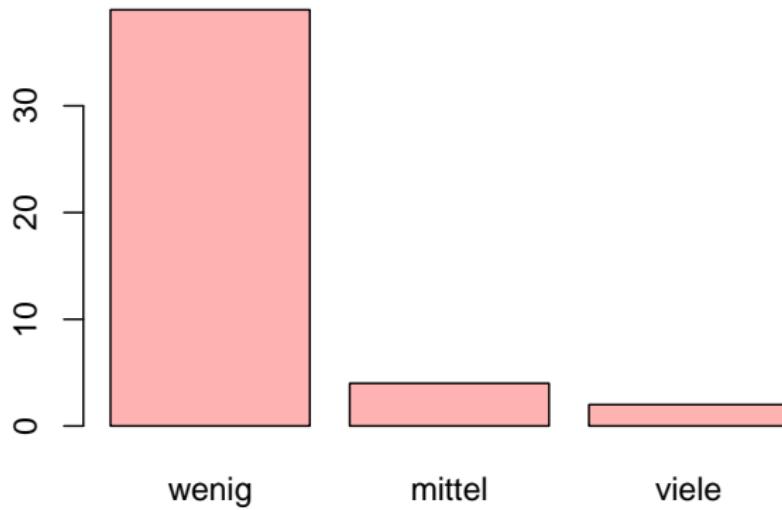
# ROTE FARBE

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



# TRANSPARENT

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



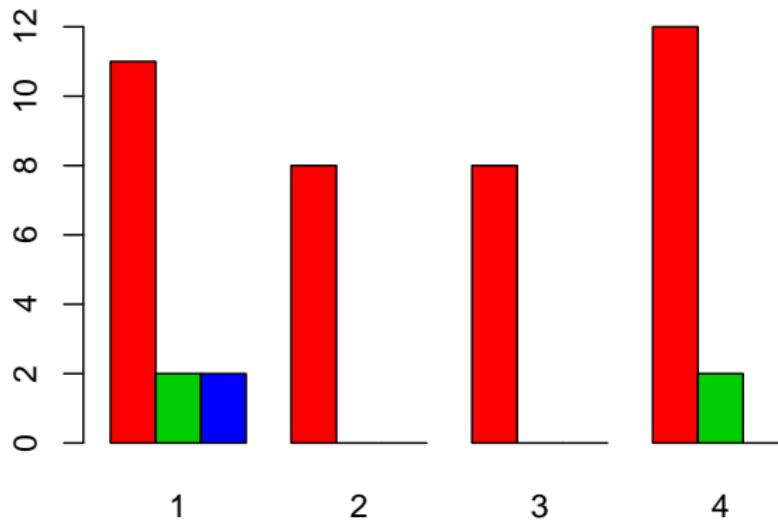
# EINE ZWEIDIMENSIONALE TABELLE

- Wenn das übergebene Tabellenobjekt zweidimensional ist, wird ein bedingter Barplot erstellt.

```
tab2dim <- table(dat$Spiel100K,dat$clust)
```

# BEDINGTER BARPLOT

```
barplot(tab2dim,col=2:4,beside=T)
```

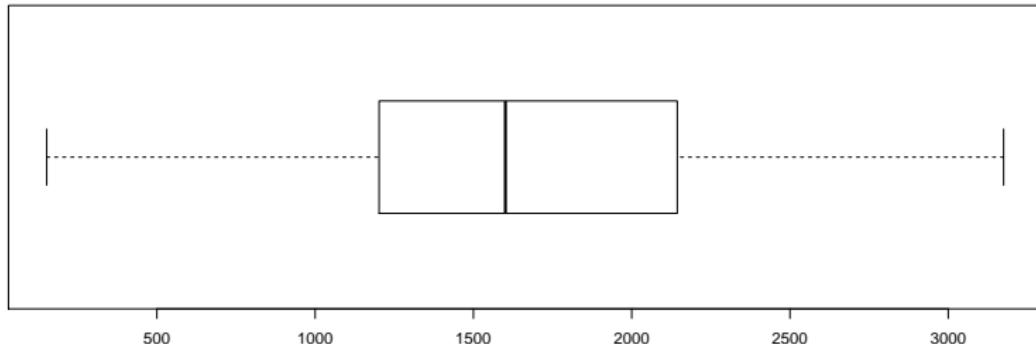


# HORIZONTALER BOXPLOT

- Ein einfacher **boxplot** kann mit `boxplot()` erstellt werden.
- Für den Befehl `boxplot()` muss mindestens ein Beobachtungsvektor übergeben werden.

```
?boxplot
```

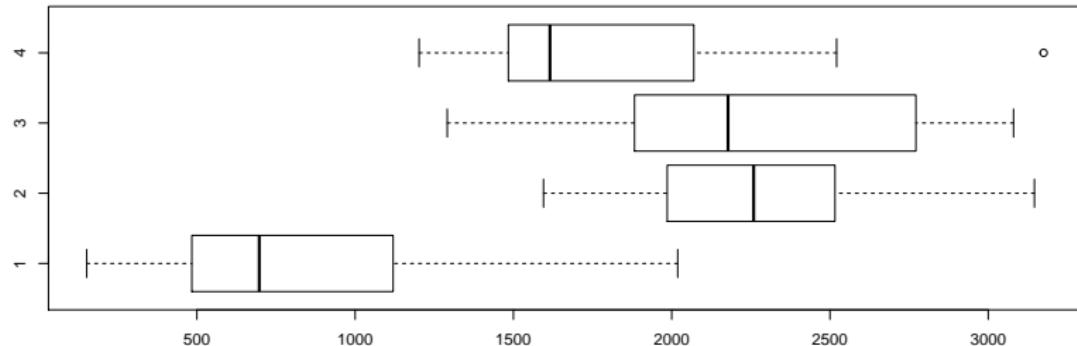
```
boxplot(dat$wohnungsbestand, horizontal=TRUE)
```



# GRUPPIERTE BOXPLOTS

- Ein sehr einfacher Weg, sich einen ersten Eindruck von bedingten Verteilungen zu verschaffen, ist über sogenannte gruppierte Boxplots.
- Dazu muss ein sogenanntes Formelobjekt an die Funktion `boxplot()` übergeben werden.
- Die bedingte Variable befindet sich auf der rechten Seite einer Tilde.

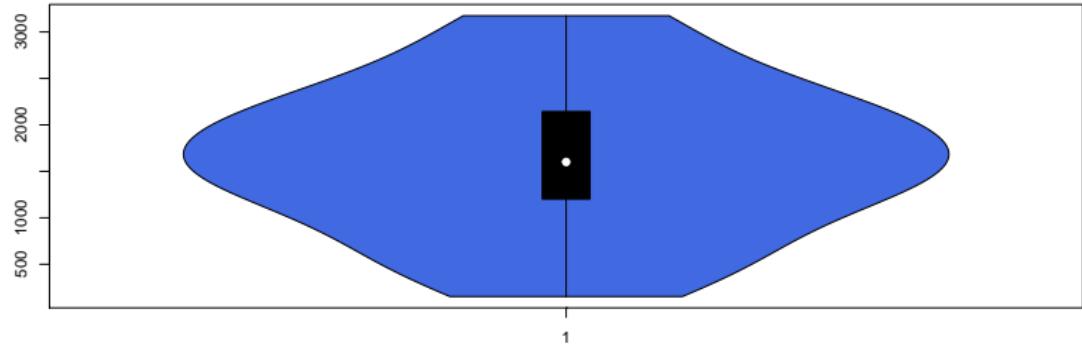
```
boxplot(wohnungsbestand~clust,data=dat,horizontal=TRUE)
```



# BOXPLOT ALTERNATIVEN - VIOPLOT

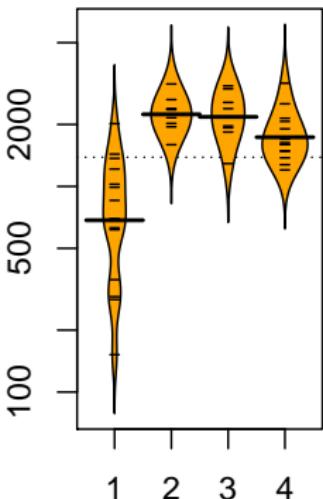
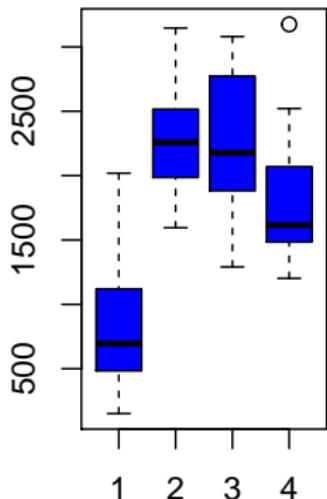
- Baut auf dem boxplot auf - Zusatzinformationen zur Dichte
- Die Dichte wird mit der Kernel-Methode berechnet.
- Je weiter die Ausdehnung, desto höher ist die Dichte an dieser Stelle.
- Weißer Punkt - Medianwert

```
library(vioplot)
vioplot(na.omit(dat$wohnungsbestand), col="royalblue")
```



# ALTERNATIVEN ZUM BOXPLOT()

```
library(beanplot)
par(mfrow = c(1,2))
boxplot(dat$wohnungsbestand~dat$clust,data=dat,col="blue")
beanplot(dat$wohnungsbestand~dat$clust,data=dat,col="orange")
```

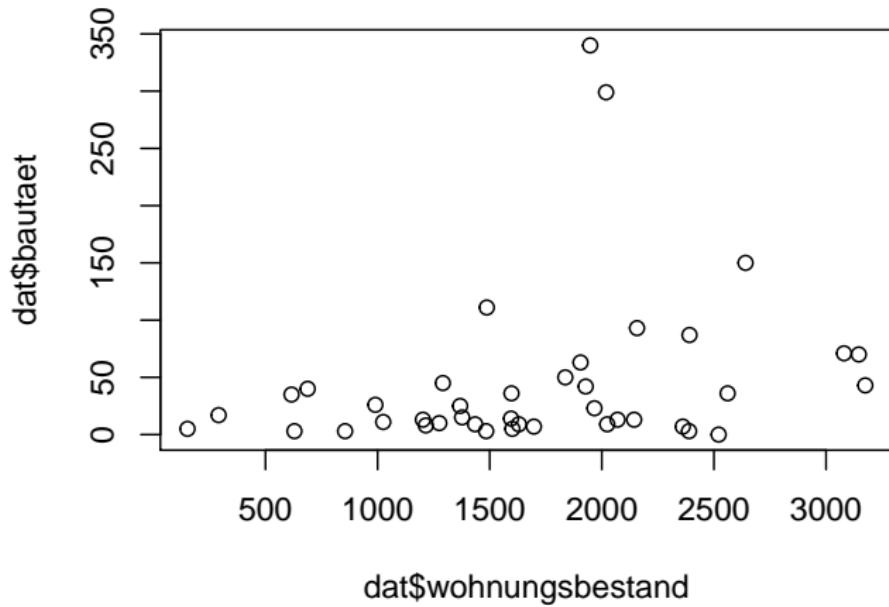


# BEDINGTE, BI- UND MULTIVARIATE GRAPHIKEN - SCATTERPLOTS

- Ein einfaches Streudiagramm kann mit der Funktion `plot()` erstellt werden.
- Um ein Scatterplot zu erstellen, müssen `x` und `y` als Beobachtungsvektoren übergeben werden.
- Argument `col` - Farbe als Zeichen oder numerisch
- Argument `pch` - Plotsymbol als Zeichen oder numerisch
- Achsenbeschriftung wird mit `xlab` und `ylab` definiert.

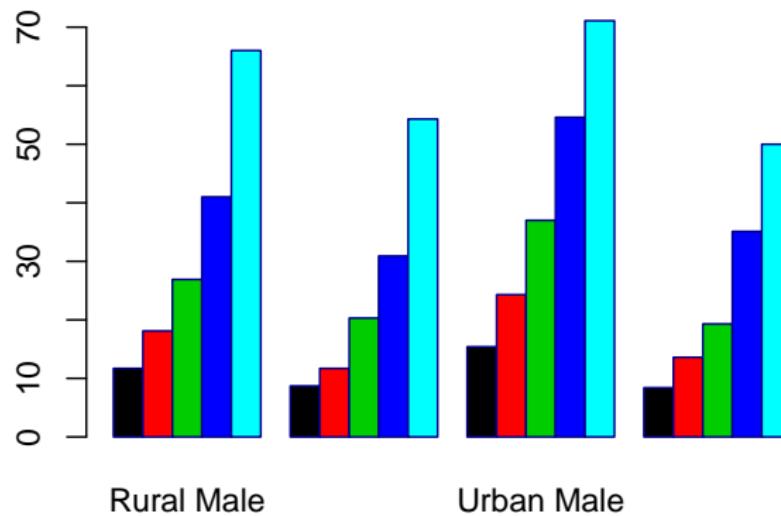
# SCATTERPLOT

```
plot(dat$wohnungsbestand,dat$bautaet)
```



# ÜBUNG - BALKENDIAGRAMM VADEATHS

- Laden Sie den Datensatz VADeaths und erstellen Sie die folgende Darstellung:

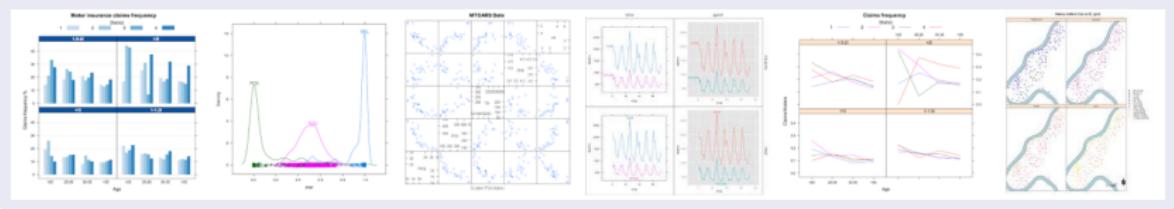


# DAS LATTICE-PAKET

## Definition einer lattice Graphik

*It is designed to meet most typical graphics needs with minimal tuning, but can also be easily extended to handle most nonstandard requirements.*

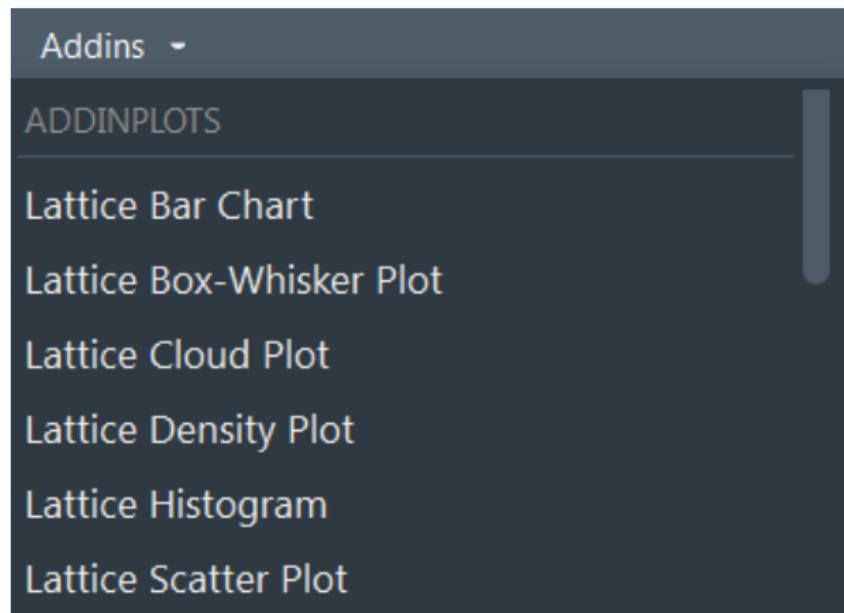
## BEISPIELE FÜR LATTICE GRAPHIKEN



# EIN WEITERES ADDIN FÜR RSTUDIO

- das addinplots-Paket installieren - den Datensatz markieren, der visualisiert werden soll, und einen Plottyp wählen:

```
devtools::install_github("homerhanumat/addinplots")
```



# BENUTZER INTERFACE FÜR ADDINPLOTS

Cancel Histogram Code-Helper

Data  
iris

Choose the numerical variable.

X  
Sepal.Length

Group Facet Other

The Plot

The Code

```
lattice::histogram(~ Sepal.Length | Species,  
                   data = iris,  
                   layout = c(1,3),  
                   type = "percent")
```

Facet by:  
Species

Also facet by:

Rows in Layout  
3

Columns in Layout  
1

Show Facet-Variable Names

```
iris # Beispieldatensatz
```

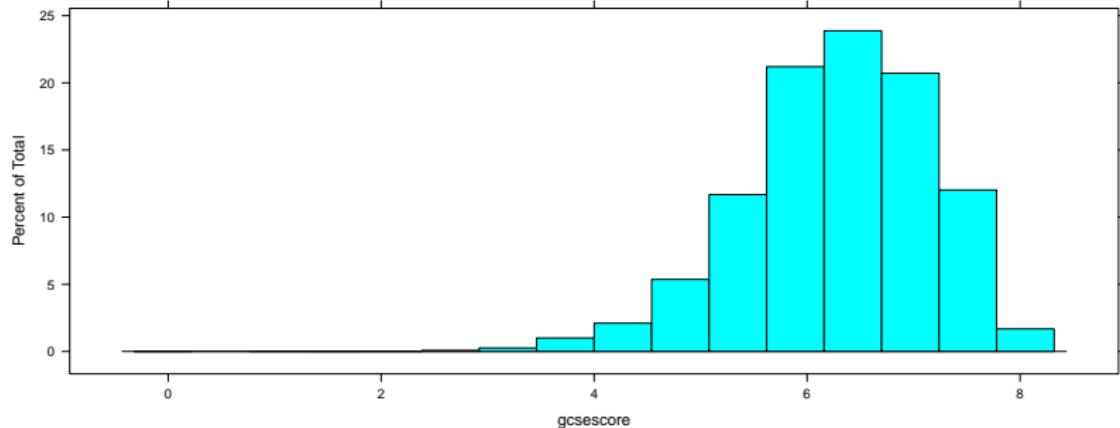
# EIN BEISPIELDATENSATZ - TESTERGEBNISSE BEI A-LEVEL CHEMIE TEST AUS DEM JAHR 1997

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

variables	categories
lea	Local Education Authority
school	School identifier
student	Student identifier
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

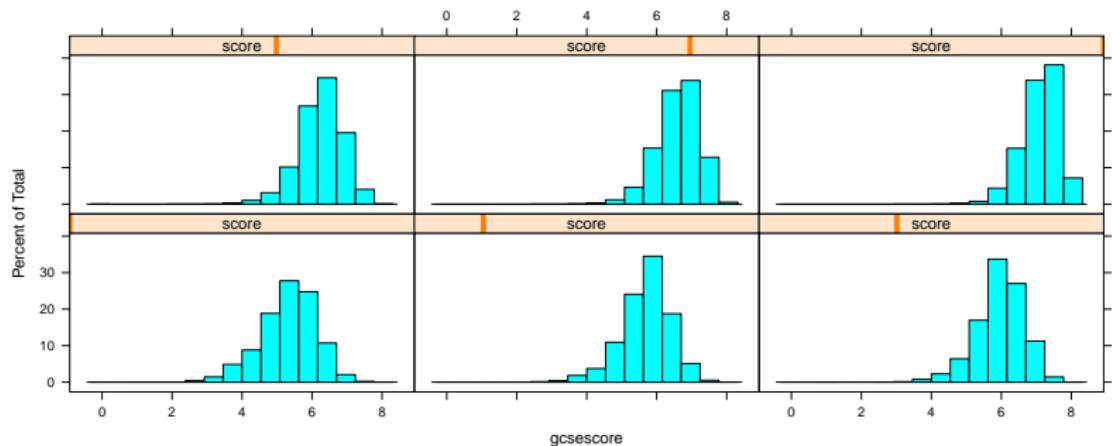
# HISTOGRAMM MIT LATTICE

```
library("lattice")
histogram(~ gcsescore, data = Chem97)
```



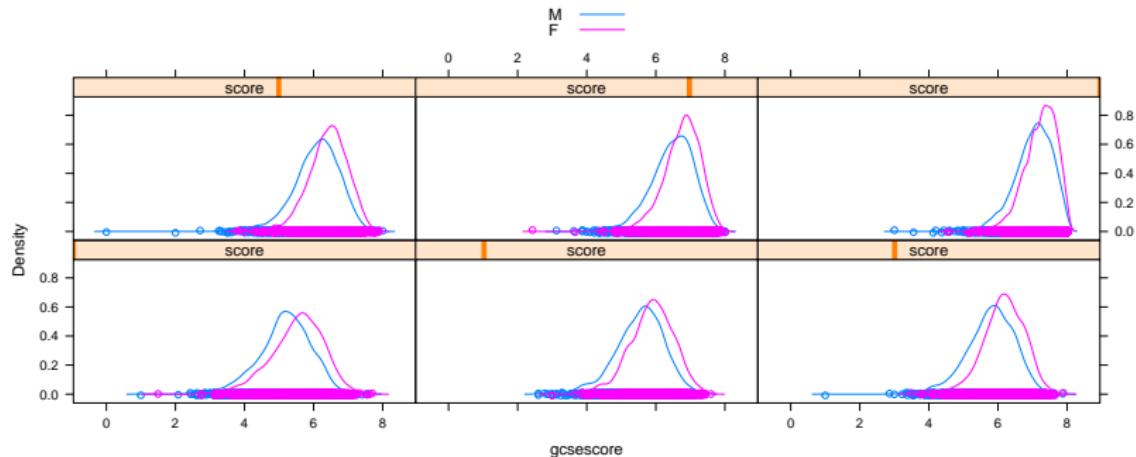
# MEHR HISTOGRAMME MIT LATTICE

```
histogram(~ gcsescore | score,data = Chem97)
```



# DIE DICHTE PLOTSEN MIT EINER LEGENDE

```
densityplot(~ gcse score | score, Chem97,  
groups=gender,auto.key=TRUE)
```

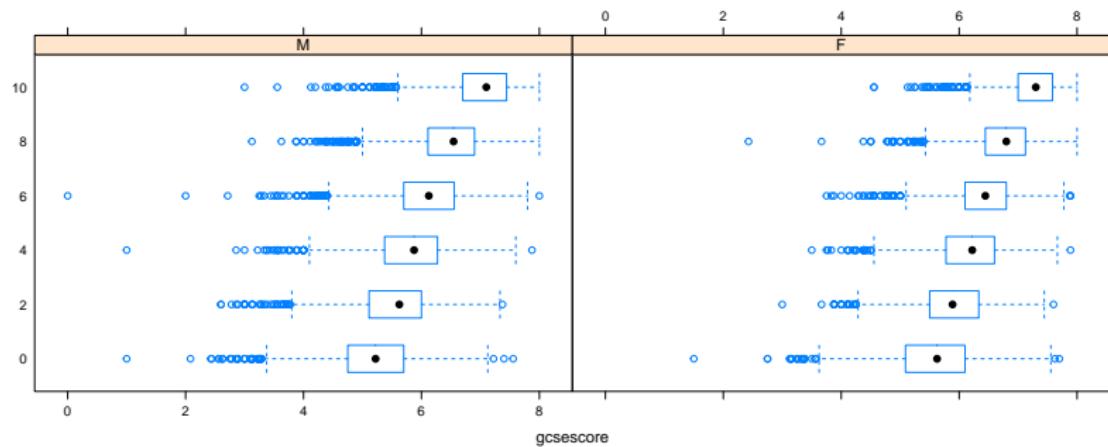


## Einführung in das lattice Paket

# EINEN BOXPLOT MIT LATTICE ERZEUGEN

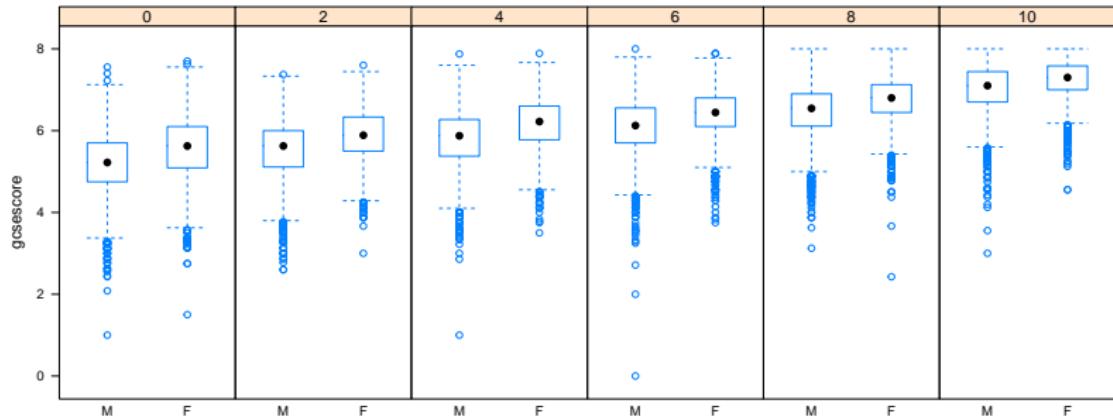
```
Chem97$score <- as.factor(Chem97$score)
```

```
bwplot(score ~ gcsescore | gender, Chem97)
```



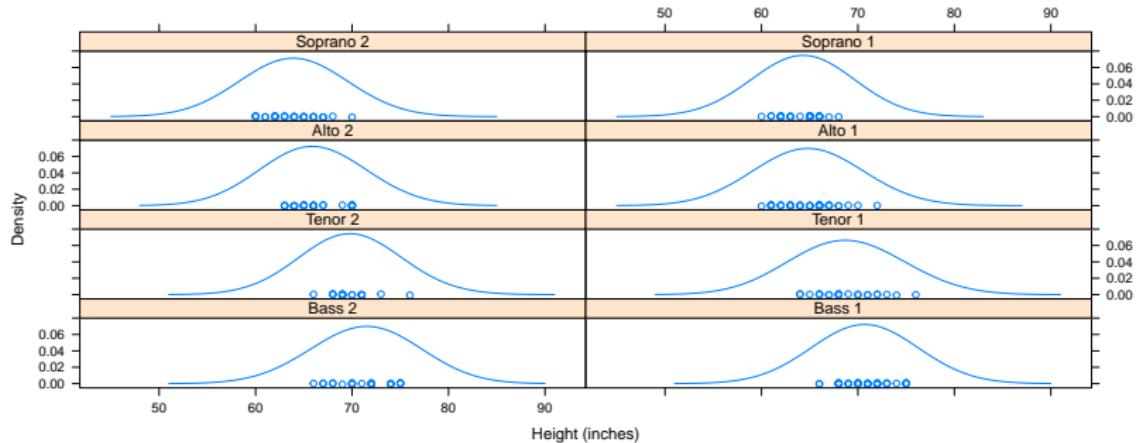
# BEDINGTE BOXPLOTS MIT LATTICE ERZEUGEN

```
bwplot(gcsescore ~ gender | score, Chem97,  
       layout = c(6, 1))
```



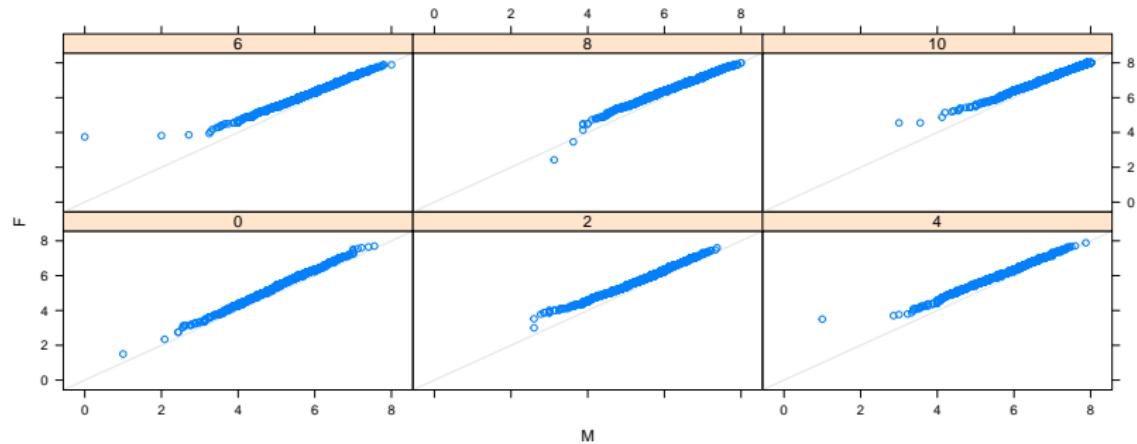
# EIN DENSITYPLOT

```
densityplot(~height|voice.part,data=singer,layout = c(2,4),  
           xlab = "Height (inches)",bw = 5)
```



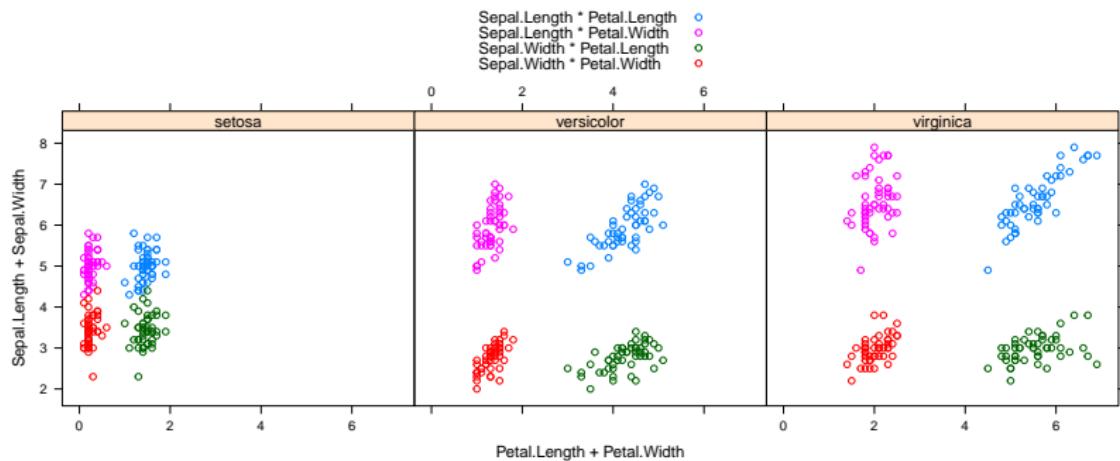
# BIVARIATE PLOTS - QUANTILE-QUANTILE PLOT

```
qq(gender ~ gcse score | score, Chem97)
```



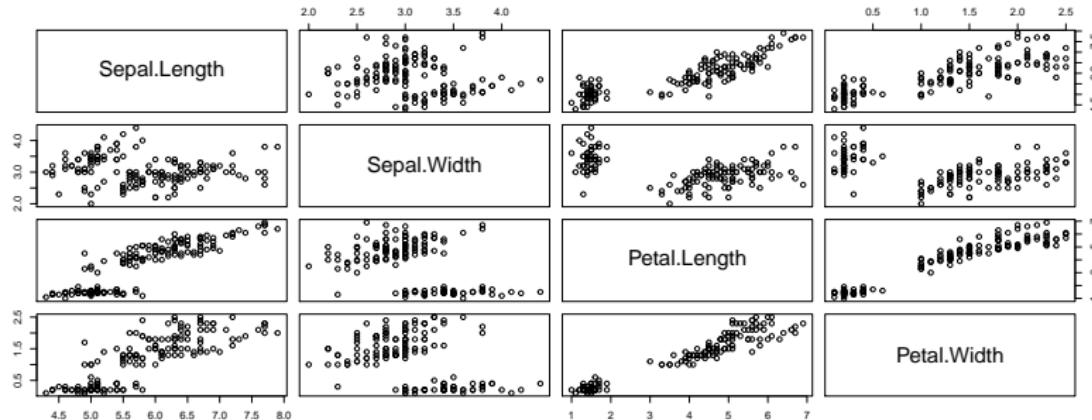
# SCATTERPLOT MIT LATTICE - XYPLOT

```
xyplot(Sepal.Length+Sepal.Width~Petal.Length+Petal.Width  
| Species,data = iris, auto.key = T)
```



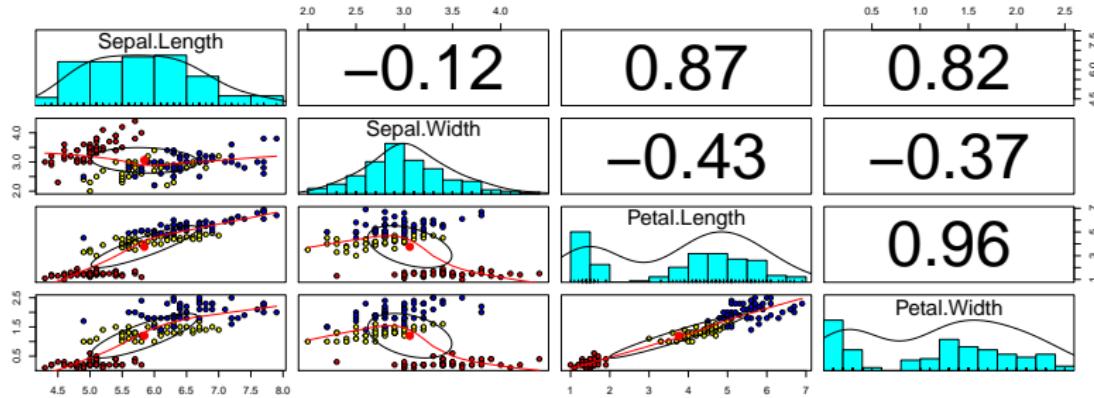
# ZUSAMMENHANG ZWISCHEN VARIABLEN - PAIRS PLOT

```
pairs(iris[,1:4])
```



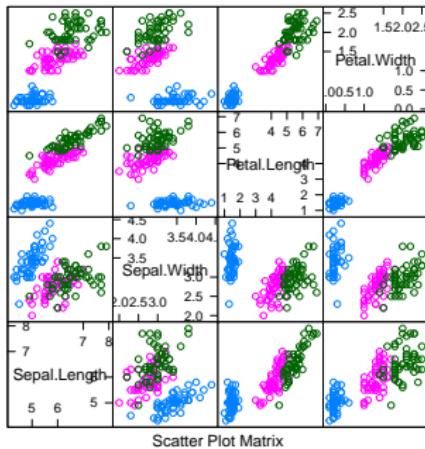
# DEN PAIRS PLOT ERWEITERT

```
library("psych")
pairs.panels(iris[,1:4],bg=c("red","yellow","blue")[iris$Species
                                         pch=21,main=""])
```



# MULTIVARIATE PLOTS - SPLOM

```
splom(~iris[,1:4], groups = Species, data = iris)
```



# MEHR ARGUMENTE IM **SPLOM** BEFEHL

```
super.sym <- trellis.par.get("superpose.symbol")
splom(~iris[1:4], groups = Species, data = iris,
      panel = panel.superpose,
      key = list(title = "Three Varieties of Iris",
                 columns = 3,
                 points = list(pch = super.sym$pch[1:3],
                               col = super.sym$col[1:3]),
                 text = list(c("Setosa", "Versicolor",
                               "Virginica"))))
```

# DER BEISPIELDATENSATZ BANKWAGES

```
install.packages("AER")
```

```
library("AER")
data(BankWages)
```

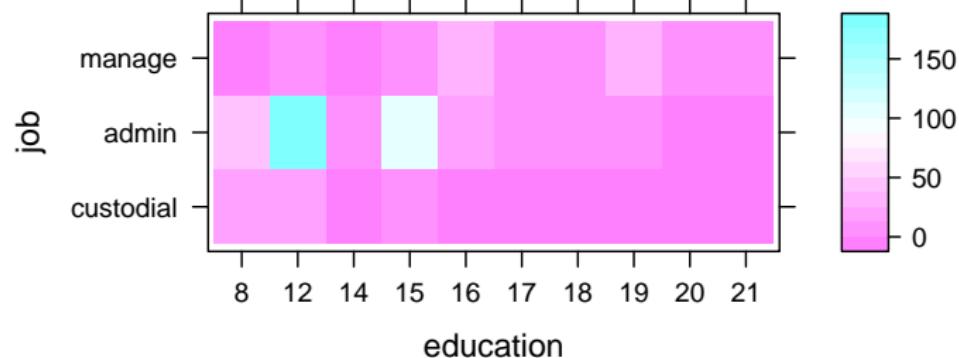
```
head(BankWages)
```

```
##      job education gender minority
## 1 manage         15   male      no
## 2 admin          16   male      no
## 3 admin          12 female     no
## 4 admin          8  female     no
## 5 admin          15   male      no
## 6 admin          15   male      no
```

# LEVELPLOT

- education in Jahren

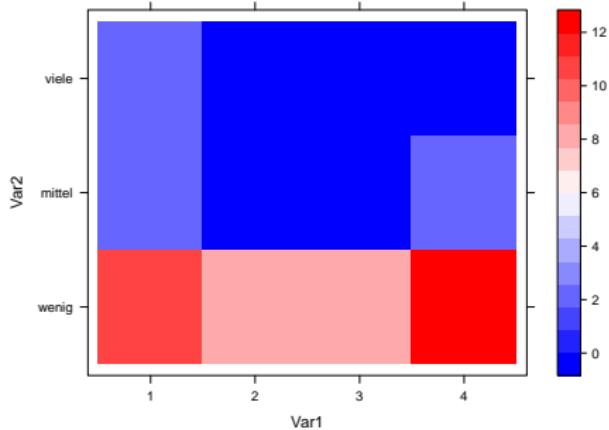
```
library("lattice")
levelplot(table(BankWages$education,BankWages$job),
          xlab="education",ylab="job")
```



# LEVELPLOT

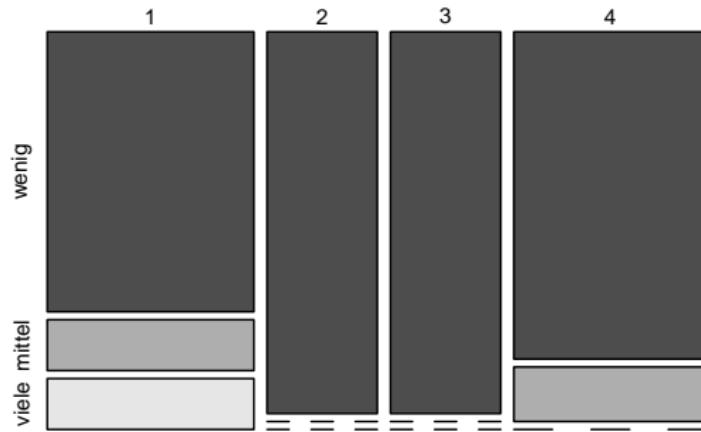
```
tab2dim2 <- table(dat$clust,dat$Spiel100K)
```

```
levelplot(tab2dim2,  
col.regions=colorRampPalette(c("blue","white","red")))
```



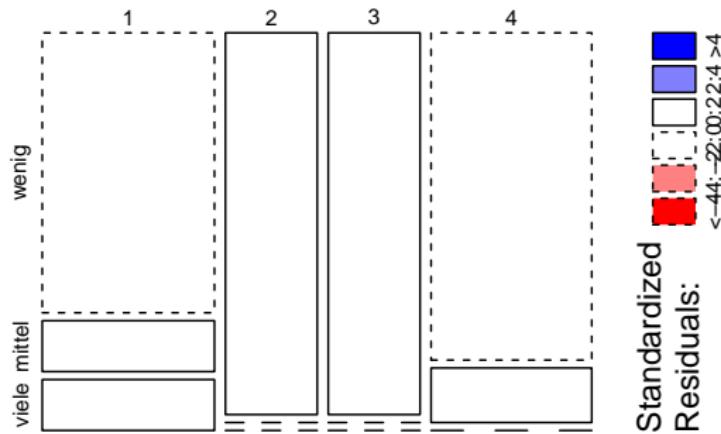
# ZUSAMMENHANG - KATEGORIALE VARIABLEN

```
mosaicplot(tab2dim2, color = TRUE, main="")
```



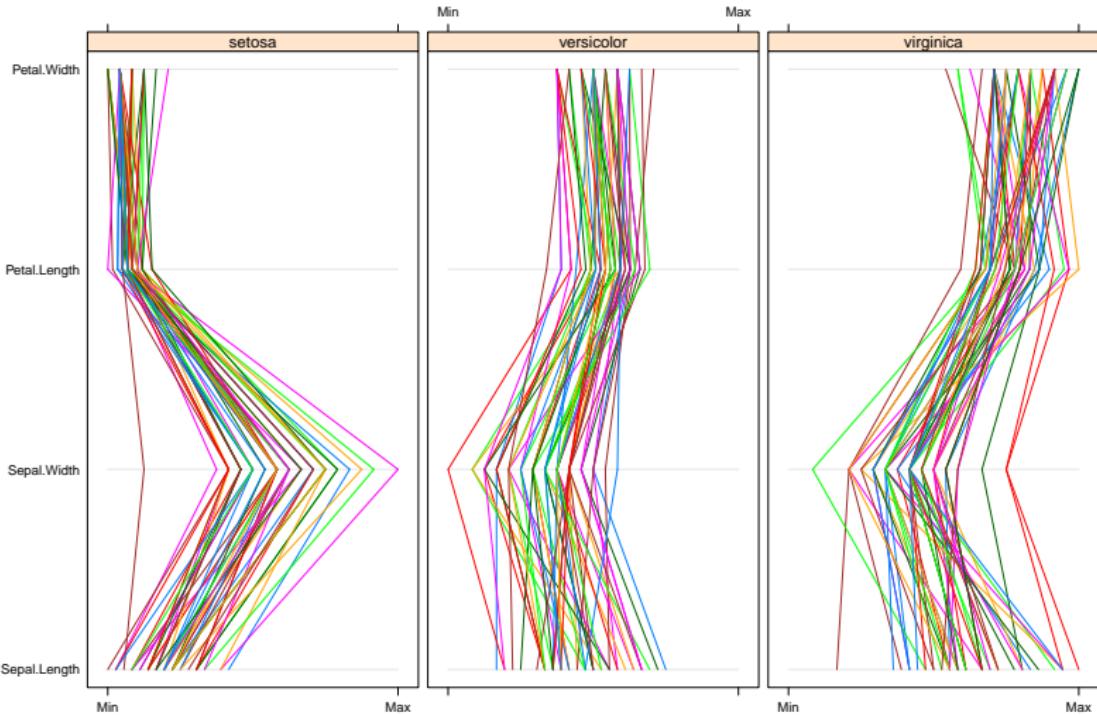
# EINFÄRBUNG NACH RESIDUEN:

```
mosaicplot(tab2dim2, main="", shade = TRUE)
```



# PARALLEL PLOT

```
parallelplot(~iris[,1:4] ~ Species, iris, layout=c(3,1))
```



# DAS GGPLOT2 PAKET

## Einführung ggplot2

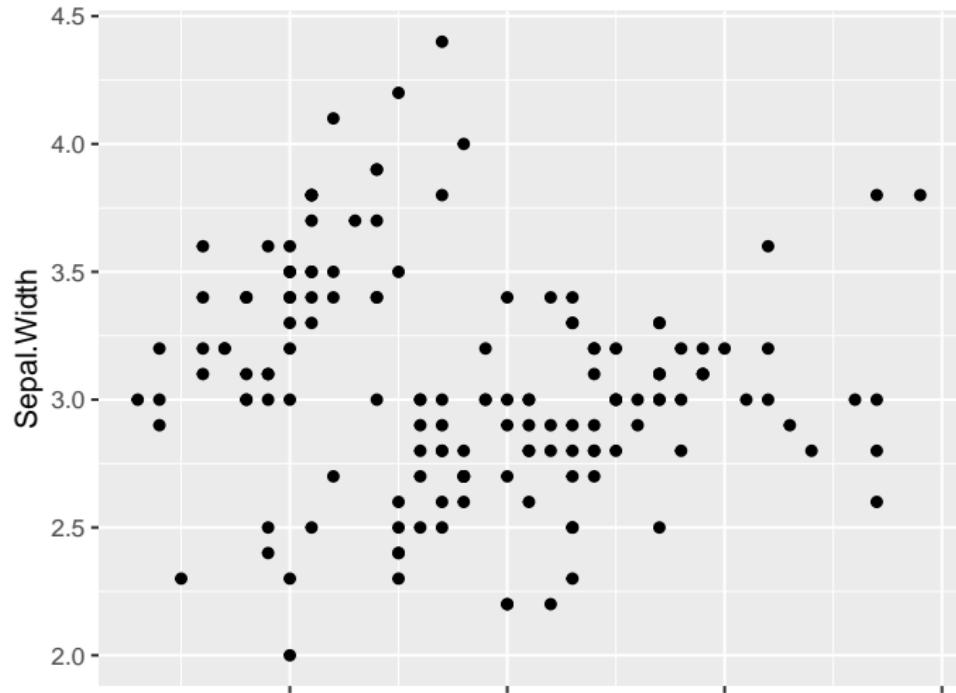
*The ggplot2 package, created by Hadley Wickham, offers a powerful graphics language for creating elegant and complex plots. Its popularity in the R community has exploded in recent years. Originally based on Leland Wilkinson's The Grammar of Graphics, ggplot2 allows you to create graphs that represent both univariate and multivariate numerical and categorical data in a straightforward manner.*

## BEISPIELE GGPLOT2 GRAPHIKEN



# EIN ERSTES BEISPIEL GGPLOT2

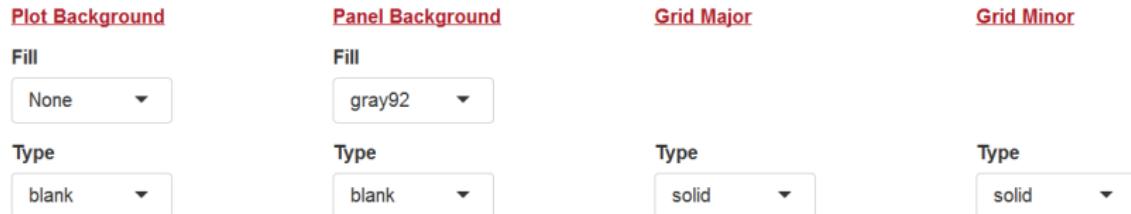
```
library(ggplot2)
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width)) +
  geom_point()
```



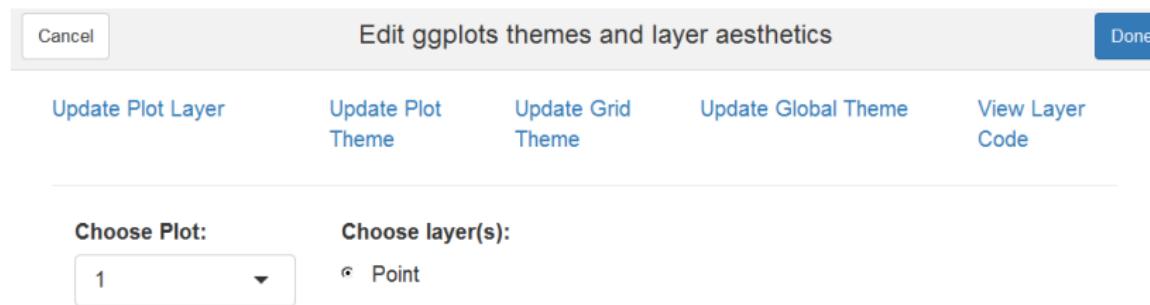
# EINIGE RSTUDIO ADDINS

- Eine ggplot Grafik muss im Quellcode markiert werden, um die folgenden Addins zu verwenden

```
install.packages("ggThemeAssist")
```

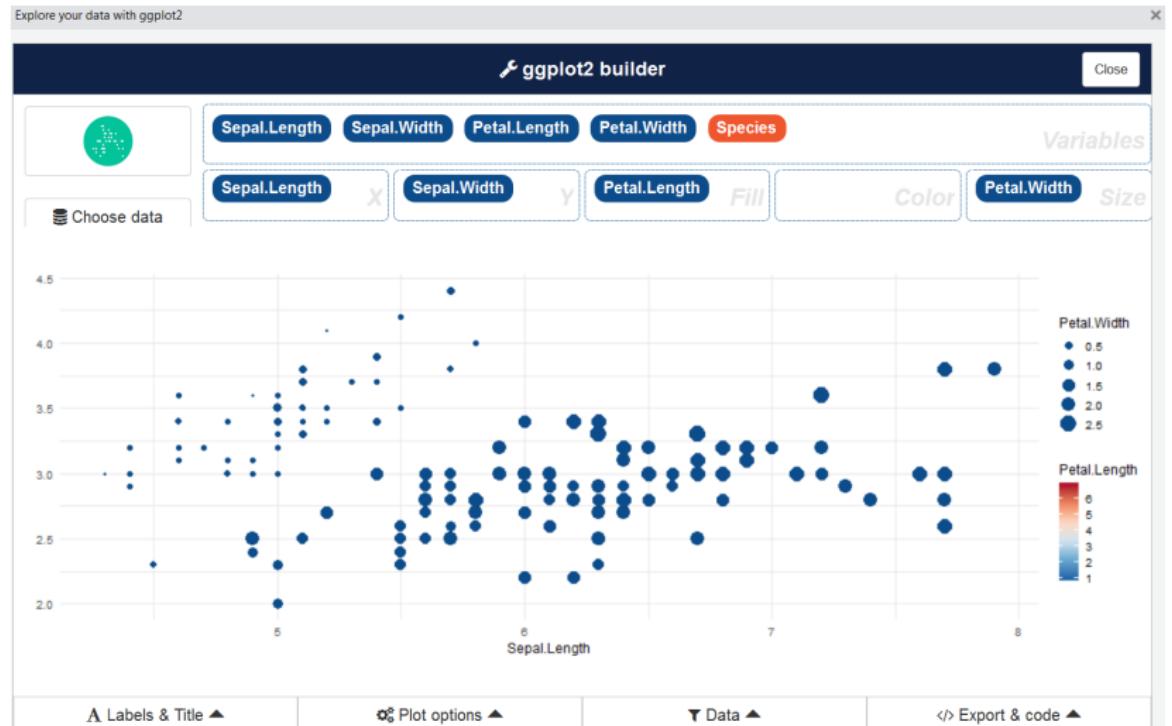


```
install.packages('ggedit')
```



# RSTUDIO ADDIN ZUM ERZEUGEN VON GG PLOT2 GRAPHIKEN

```
devtools::install_github("dreamRs/esquisse")
```



# SHINY APP - R GRAPHIK KATALOG

<http://shinyapps.stat.ubc.ca/r-graph-catalog/>

## R Graph Catalog

### About

This catalog is a complement to "Creating More Effective Graphs" by Naomi Robbins. All graphs were produced using the `r` language and the add-on package `ggplot2`, written by Hadley Wickham. The gallery is maintained by Joanna Zhao and Jennifer Bryan.

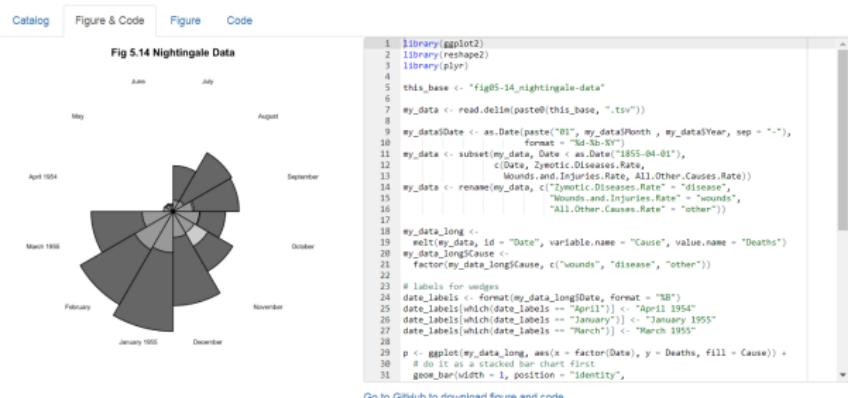
[More...](#)

**Filter by type**

- Good
- Not Recommended

**Filter by tags**

- Dot Plot
- Line Graph
- Pie Chart
- Bar Chart
- Scatterplot
- Trellis Display
- Histogram



# EIN BEISPIELDATENSATZ ZU DIAMANTEN

*A dataset containing the prices and other attributes of almost 54,000 diamonds.*

- price - price in US dollars (\$326–\$18,823)
- carat - weight of the diamond (0.2–5.01)
- cut - quality of the cut (Fair, Good, Very Good, Premium, Ideal)
- color - diamond colour, from J (worst) to D (best)
- clarity - a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))

```
data(diamonds)
```

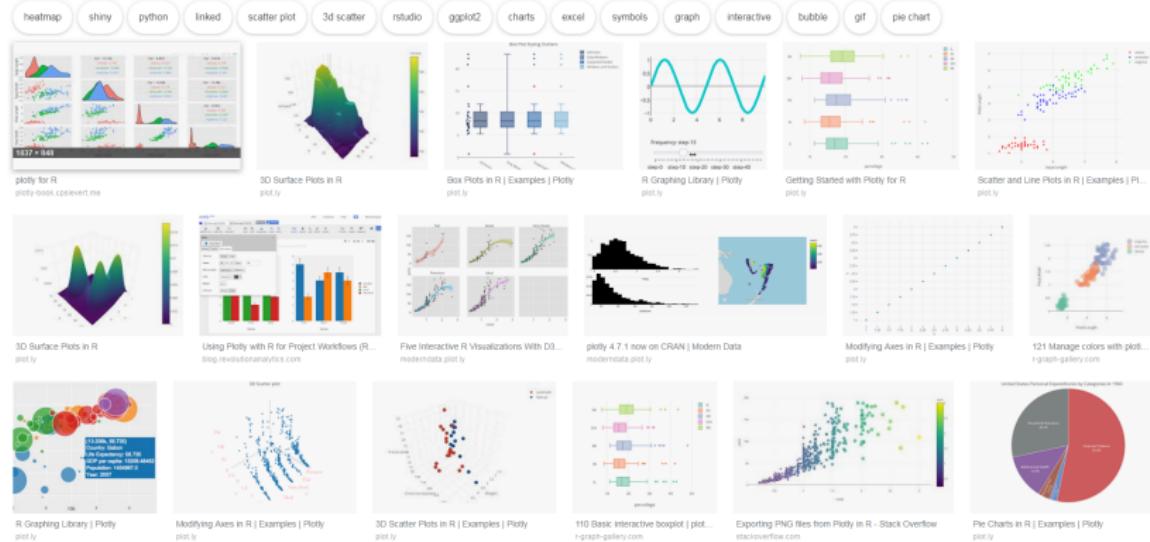
- Der Datensatz ist zu groß für unsere Anwendungszwecke:

```
d <- diamonds[sample(nrow(diamonds), 1000), ]
```

# DAS PAKET PLOTLY

*Create Interactive Web Graphics via 'plotly.js'*

```
library(plotly)
```



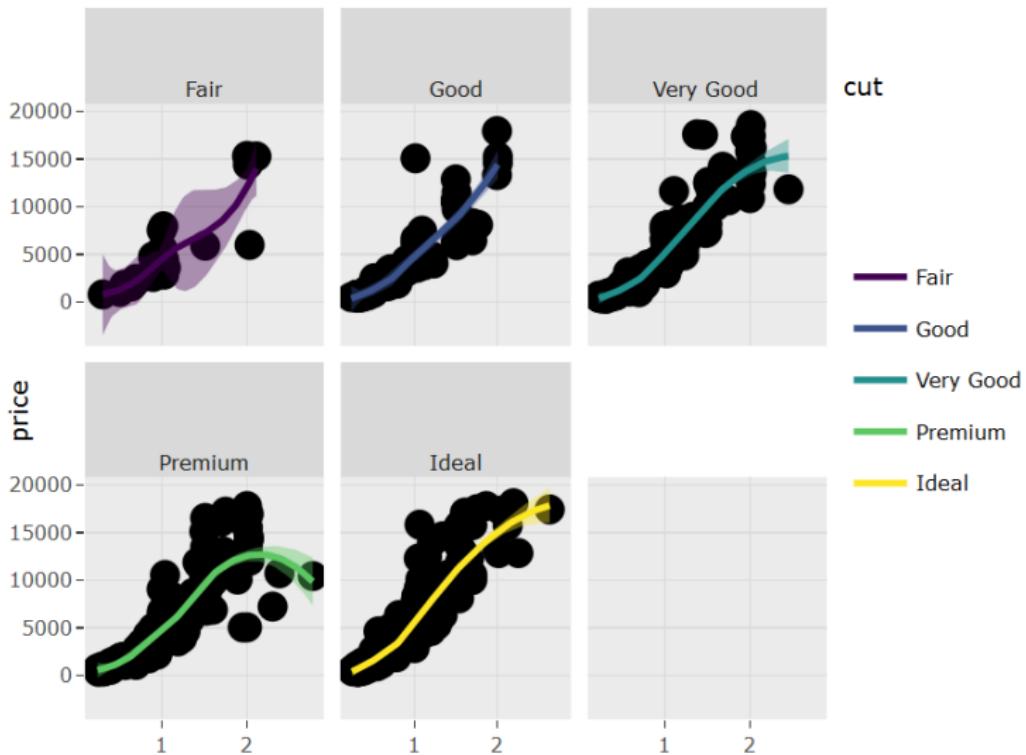
# INTERAKTIVITÄT HINZUFÜGEN

```
p <- ggplot(data = d, aes(x = carat, y = price)) +  
  geom_point(aes(text = clarity, size = 4)) +  
  geom_smooth(aes(colour = cut, fill = cut)) +  
  facet_wrap(~ cut)
```

ES WIRD EINE GGPLOT GRAPHIK ERZEUGT

# DAS ERGEBNIS - EINE INTERAKTIVE GRAPHIK

```
ggplotly(p)
```



- J H Maindonald - **Lattice and Other Graphics in R**
- Deepayan Sarkar - **An introduction to R - lattice lab**
- Flowingdata - **Comparing ggplot2 and R Base Graphics**
- Quick R - **ggplot2**
- **Top 50 ggplot2 Visualizations**
- **Bioconductor R manual** with an extensive part on graphics
- Shiny app to visualize **ggplot2 internals**
- Shiny app for **interactive plot editing**