Plotten mit Ggplot2

Referentin: Anne Kunstmann

Betreuer: Jakob Lüttgau
Proseminar Programmieren in R
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

25. Mai 2016

Agenda

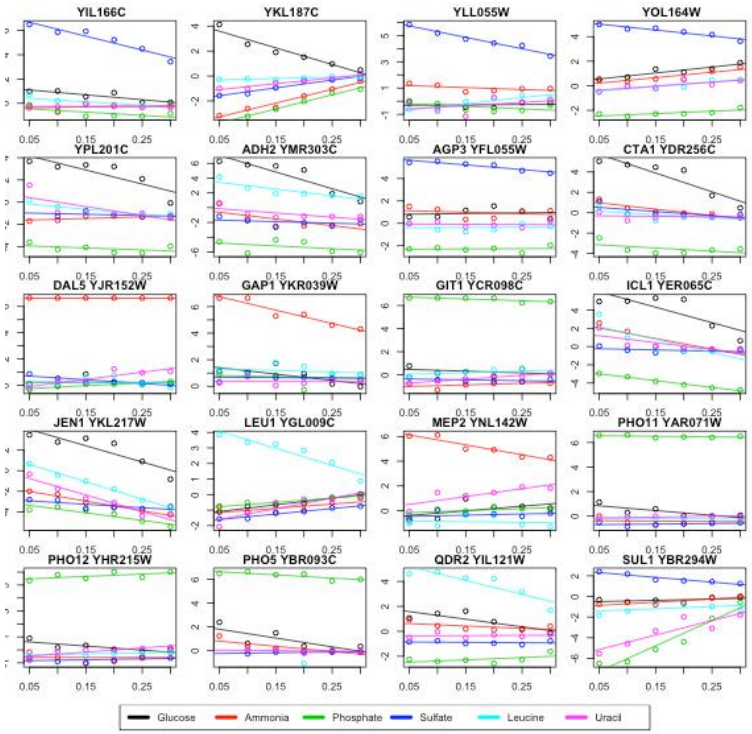
Gegenüberstellung: R vs GGPlot2

R native Plots

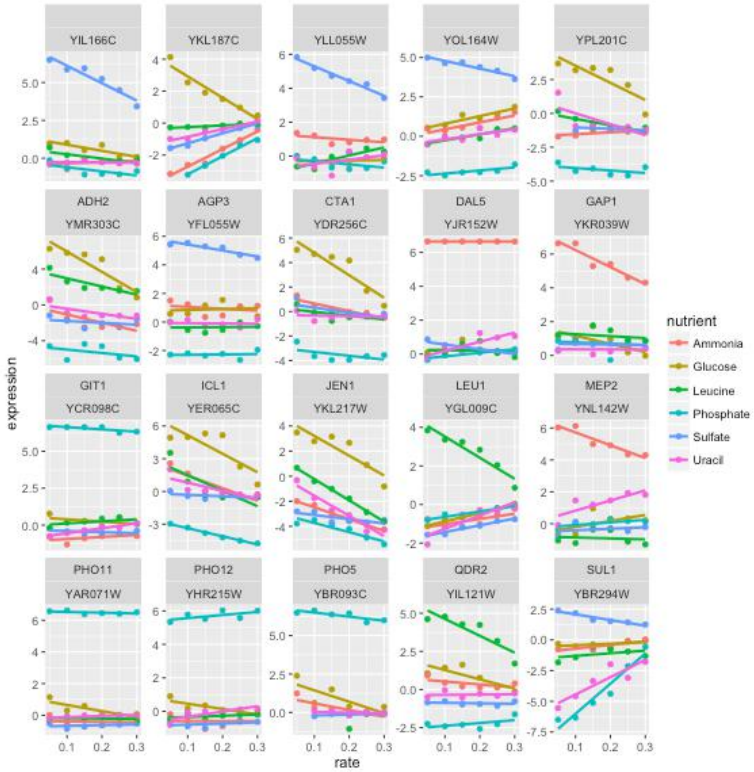
- grundlegende Plot-Funktionen
- wortreich für komplexe Plots
- keine automatische Legendenerstellung
- komplexe Codeänderungen nötig
- verbesserungswürdige Visualisierung

GGplot2

- grundlegende und fortgeschrittene Plot-Funktionen
- automatische Legendenerstellung
- robust bei Codeänderung
- feste Schemata bei Plot-Erstellung
- erwartungsgemäße Visualisierung



R nativer Plot
[http://varianceexplained.org/r/why-I-useggplot2/]



Plot mit GGPlot2
[http://varianceexplained.org/r/why-I-useggplot2/]

```r
par(mar = c(1.5, 1.5, 1.5, 1.5))

colors <- 1:6
names(colors) <- unique(top_data$nutrient)

# legend approach from http://stackoverflow.com/a/10391001/712603
m <- matrix(c(1:20, 21, 21, 21, 21), nrow = 6, ncol = 4, byrow = TRUE)
layout(mat = m, heights = c(.18, .18, .18, .18, .18, .1))

top_data$combined <- paste(top_data$name, top_data$systematic_name)
for (gene in unique(top_data$combined)) {
    sub_data <- filter(top_data, combined == gene)
    plot(expression ~ rate, sub_data, col = colors[sub_data$nutrient], main
    for (n in unique(sub_data$nutrient)) {
        m <- lm(expression ~ rate, filter(sub_data, nutrient == n))
        if (!is.na(m$coefficients[2])) {
            abline(m, col = colors[n])
        }
    }
}

# create a new plot for legend
plot(1, type = "n", axes = FALSE, xlab = "", ylab = "")
legend("top", names(colors), col = colors, horiz = TRUE, lwd = 4)
```

Code in R [http://varianceexplained.org/r/why-I-useggplot2/]

```r
ggplot(top_data, aes(rate, expression, color = nutrient)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
    facet_wrap(~name + systematic_name, scales = "free_y")
```

Code mit GGPlot2 [http://varianceexplained.org/r/why-I-useggplot2/]

Einführung

Geschichtlicher Hintergrund

- Paket zur Datenvisualisierung in R
- Entwickler Hadley Wickham
- Veröffentlichung im Jahr 2005
- Grundlage „The Grammar of Graphics“
  - Buch von Leland Wilkinson
  - Jahr 1999/2005
  - Thema statistische Datenvisualisierung
  - Fokus auf Ästhetik und Geometrie

Vorteile

- große Bandbreite an Funktionen
- hohe Abstraktion
- erweiterbar durch Pakete (Lattice,...)

Nachteile

- keine 3D-Grafiken
- keine Graphentheorie (Multigraphen, Kantenzüge,...)

Datenimport

- *read*-Funktion
- Aufruf des Dateiformats durch „ .“-Operator
- Namenszuweisung des importierten Dataframes

Parameter

- *title* Name der zu importierenden Datei
- *header = T/F* Übernahme der Spaltentitel

Codebeispiel 1: Datenimport einer csv-Datei

```
1 airquality <- read.csv("airquality.csv", header = T)
```

Datenarten

- Unterscheidung stetiger und diskreter Funktionen

- **stetig**
  - überabzählbar unendliche viele Merkmale
  - Größe von Personen, Temperatur in Grad Celsius,...
  - z.B. geom_point(), geom_curve(),...
- **diskret**
  - abzählbar unendlich viele Merkmale
  - Geschlecht, Religion, Kundenzufriedenheit, ...
  - z.B. geom_bar(), geom_hist(),...

Datentransformation

- Auslegung auf „long format “- Datensätze
- Pakete plyr und reshape2
- Schichten, Gruppieren, Zusammenfassen von Daten
- Transformation von „wide format “zu „long format “
- Erhöhen der Repräsentativität

```
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

Codebeispiel 2 : Ermitteln der Durchschnittstemperatur pro Monat

```
1 library(plyr)
  airquality_means <- ddply(airquality, .(Month), summarise,
      meanTemp = mean(Temp))
3 # Vereinfachen des Dataframes auf Monat und mittlere Temperatur
```

```
> head(airquality_means)
  Month meanTemp
1     5 65.54839
2     6 79.10000
3     7 83.90323
4     8 83.96774
5     9 76.90000
```

Grundfunktionen

qplot - Funktion

- „quick plot“
- Einstiegsfunktion
- kein typisches GGPlot2-Prinzip
- gleiche grafische Darstellung wie GGPlot2-Funktion

Codebeispiel 3: Die allgemeine qplot-Funktion

```
1 library(ggplot2)
  qplot(x, y, data =, color =, shape =, fill=, size=, alpha=,
3   geom=, method=, formula=, facets=, xlim=, ylim=, xlab=,
    ylab=, main=, sub=)
```

ggplot - Funktion

- „Grammar of Graphics “
- Daten ausschließlich als Dataframes
- *geom_...()*- Funktion
  - „geometric objetcs “
  - Implementation außerhalb ggplot()-Funktion
  - allgemeine Darstellung der Daten
  - Shortcuts der layer()-Funktion
  - *geom_point(), geom_boxplot(), geom_bar(),...*

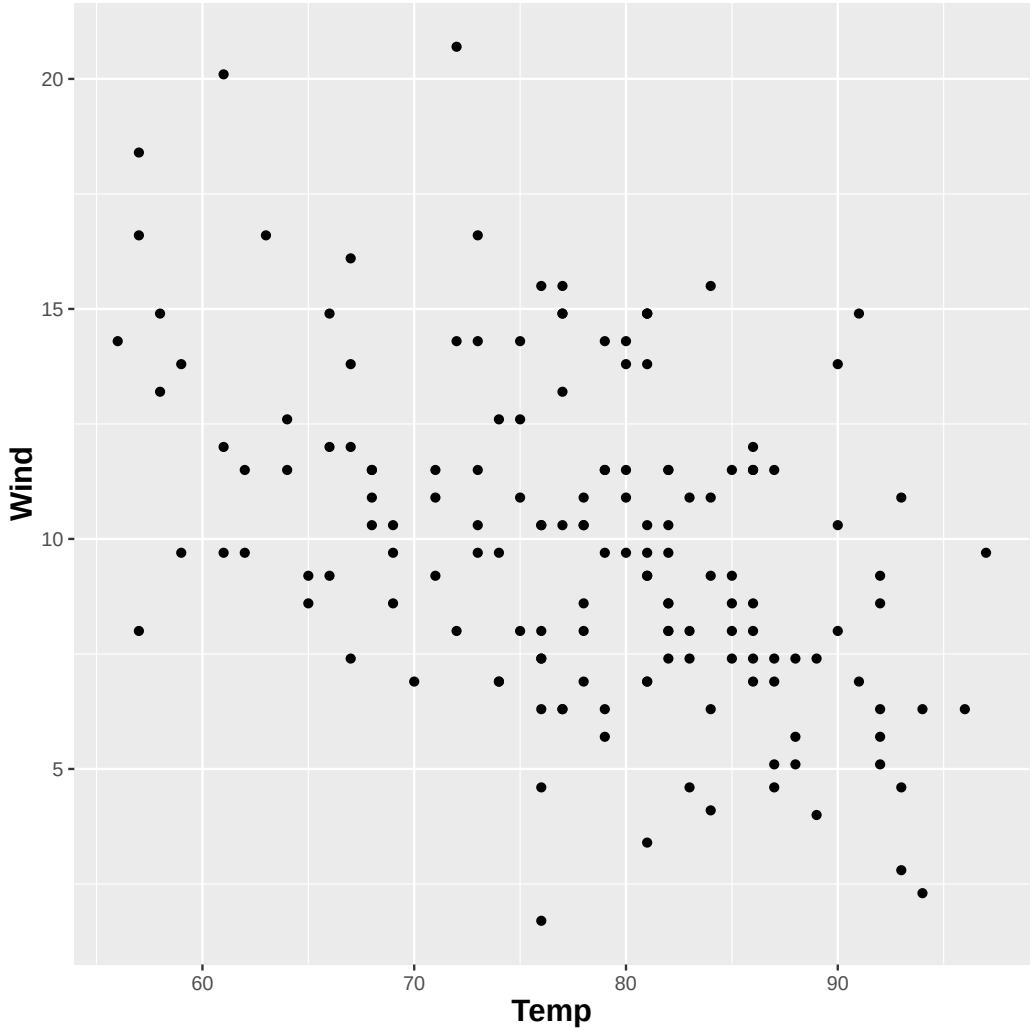
- aes()- Funktion
  - „aesthetics “
  - meist Implementation innerhalb ggplot()-Funktion
  - auch Implementation innerhalb geom_...-Funktionen
  - Unterteilung und Unterscheidung der Daten
  - Parameter abhängig von geom_...()-Funktion
  - *x, y, colour, fill, shape, size, ...*
- „+ “-Operator verbindet beide Funktionskomponenten

Codebeispiel 4 : Minimale ggplot-Funktion

```
install.packages("ggplot2") #Paket installieren
library(ggplot2)            #Bibliothek einbinden
head(airquality)            #Einblick in Dataframe

#Frage: Existiert irgendeine Art von Relation zwischen der
    Temperatur und der Windstaerke?

ggplot(airquality, aes(x = Temp, y = Wind)) + geom_point()
#airquality als Dataframe, Temp und Wind als Vektoren aus
    airquality, geom_point() fuer Streudiagramm
```



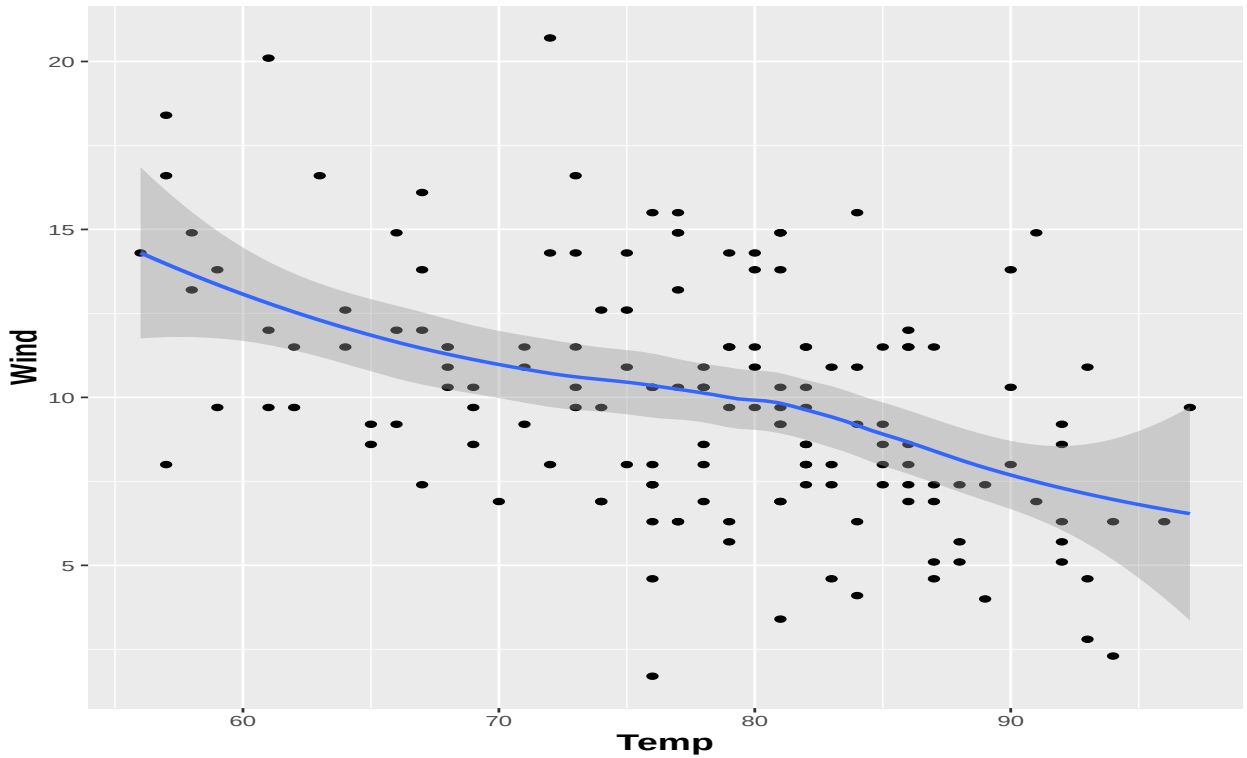
zu Codebeispiel 4: einfaches Streudiagramm

Regression

- Verdeutlichen von Abhängigkeiten und Trends
- Implementation durch geom_smooth()
- Regressionslinie mitsamt Konfidenzintervall
- Erweiterung durch viele Pakete möglich
  - *mgcv, splines, MASS,...*

Codebeispiel 5: default-Regressionskurve

```
ggplot(airquality, aes(x = Temp, y = Wind)) + geom_point() +
    geom_smooth()
```

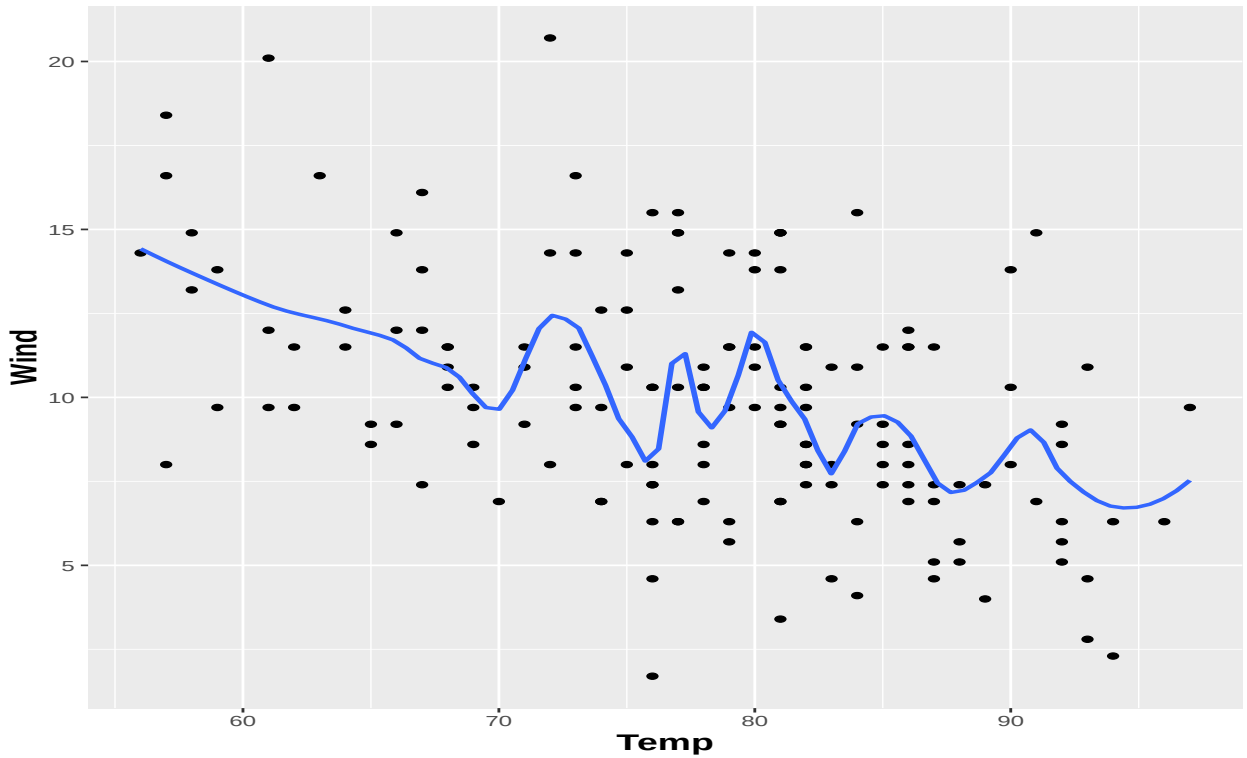


Parameter

- *method* Regressionsarten
  - lineare Regression „*lm*“
  - lokale Regression „*loess*“
- *se* Konfidenzintervall
  - mit Konfidenzintervall *TRUE*
  - ohne Konfidenzintervall *FALSE*
- *span* Empfindlichkeit der Regression
  - Werte *0...1*

Codebeispiel 6: lokale empfindliche Regressionskurve ohne Konfidenzintervall

```
1 ggplot(airquality, aes(x = Temp, y = Wind)) + geom_point() +
    geom_smooth(span = 0.2, se = F, method = "loess")
```

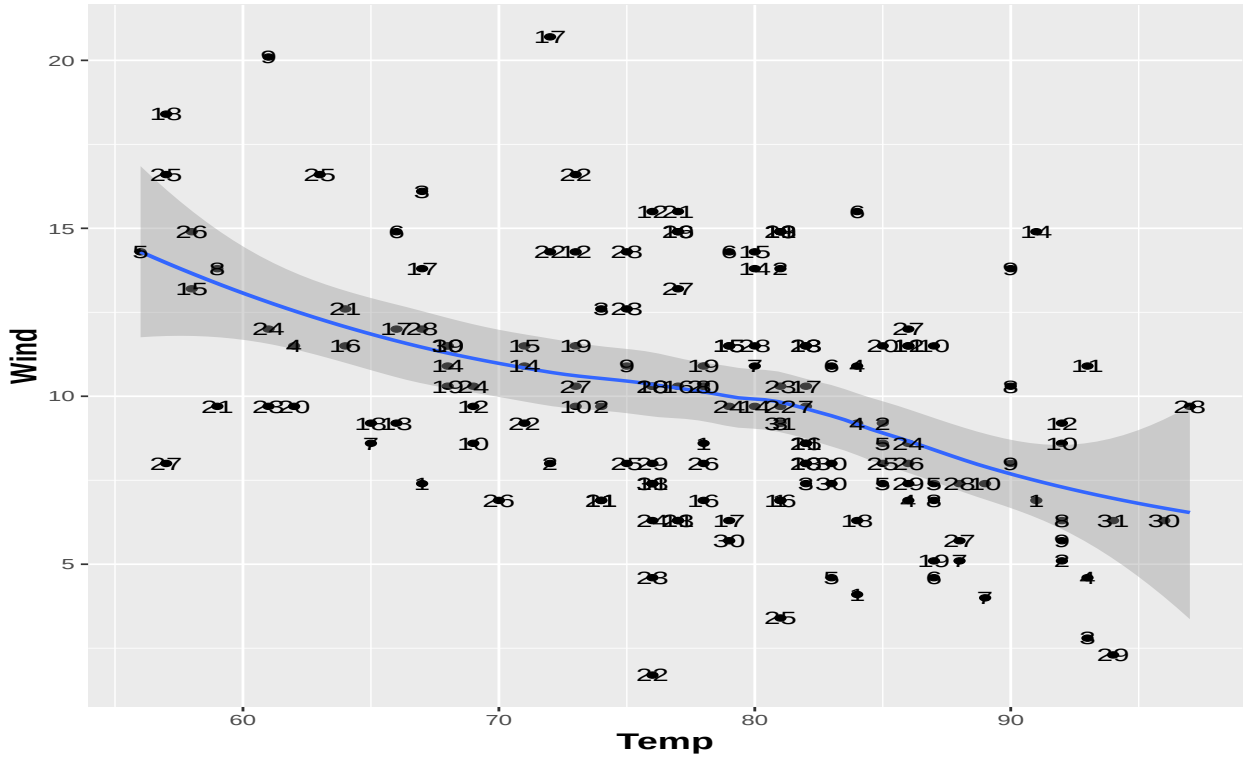


Labels

- Hinzufügen von punktegebundenen Labels
- Implementation durch *geom_text()*
  - Textlabels überschreiben Punkte
  - Nachteil: gegenseitiges Überschreiben der Labels
- Implementation durch geom_text_repel()
  - Textlabels neben Punkte
  - Labels ziehen Verbindungslinien zu Punkten
  - Nachteil: Einbinden des Pakets ggrepel nötig

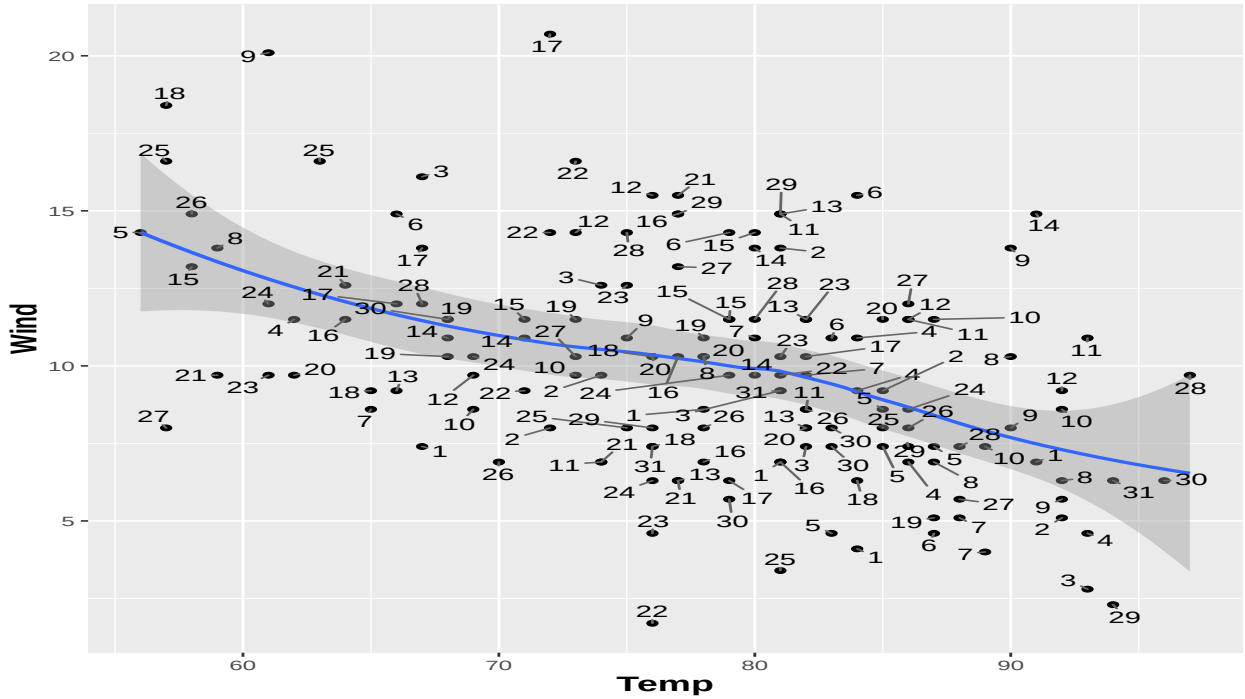
Codebeispiel 7: geom_text()-Funktion

```
1 ggplot(airquality, aes(x = Temp, y = Wind)) + geom_point() +
      geom_smooth() + geom_text(aes(label = Day))
  #Nummerierung anhand der Messungstage auf den Punkten
```



Codebeispiel 8: geom_text_repel()

```
install.packages("ggrepel")
library(ggrepel)
ggplot(airquality, aes(x = Temp, y = Wind)) + geom_point() +
    geom_smooth() + geom_text_repel(aes(label = Day))
#Nummerierung anhand der Messungstage neben den Punkten
```

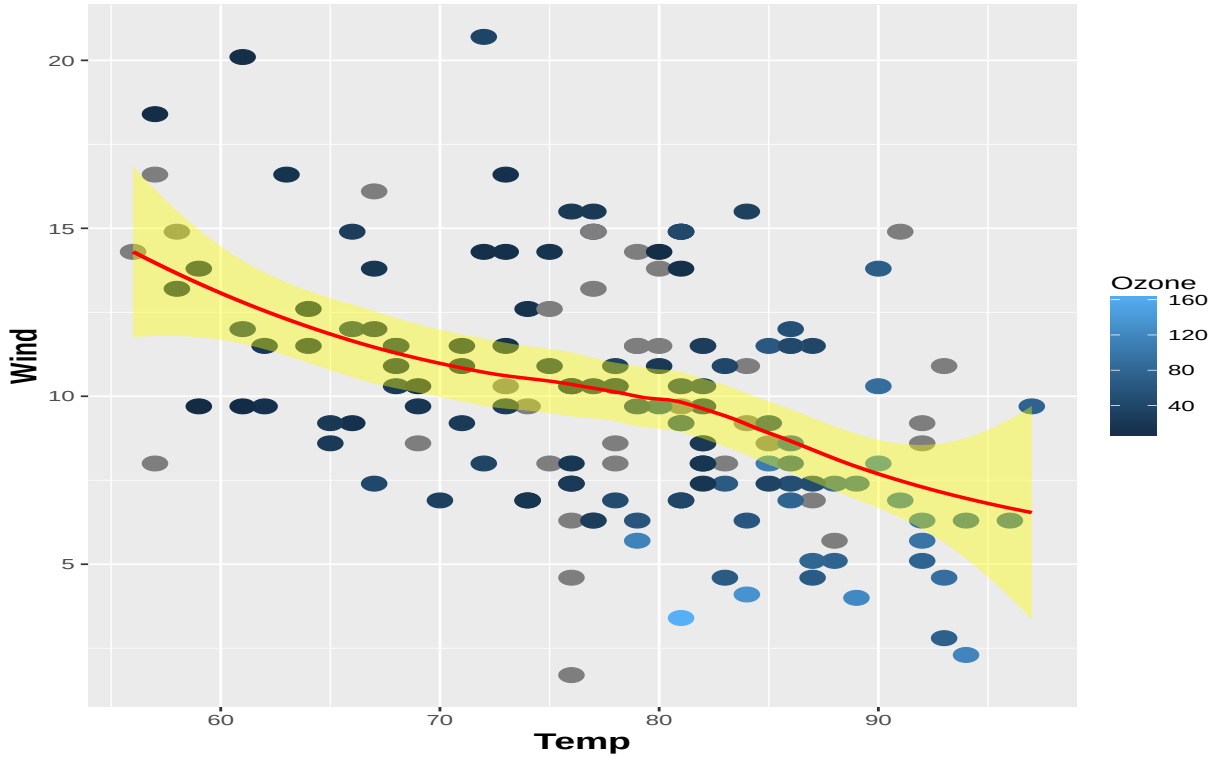


Mapping vs Setting

- möglich in ggplot()- und geom()-Funktionen
- „Setten“ Konstanten als Werte
- „Mappen“ Variablen als Werte
  - gegenseitiges Erweitern
  - gegenseitiges Überschreiben
  - gegenseitiges Löschen
- Einsparen von Code
- Verdeutlichen der Grafikstruktur
- automatische Legendenerstellung

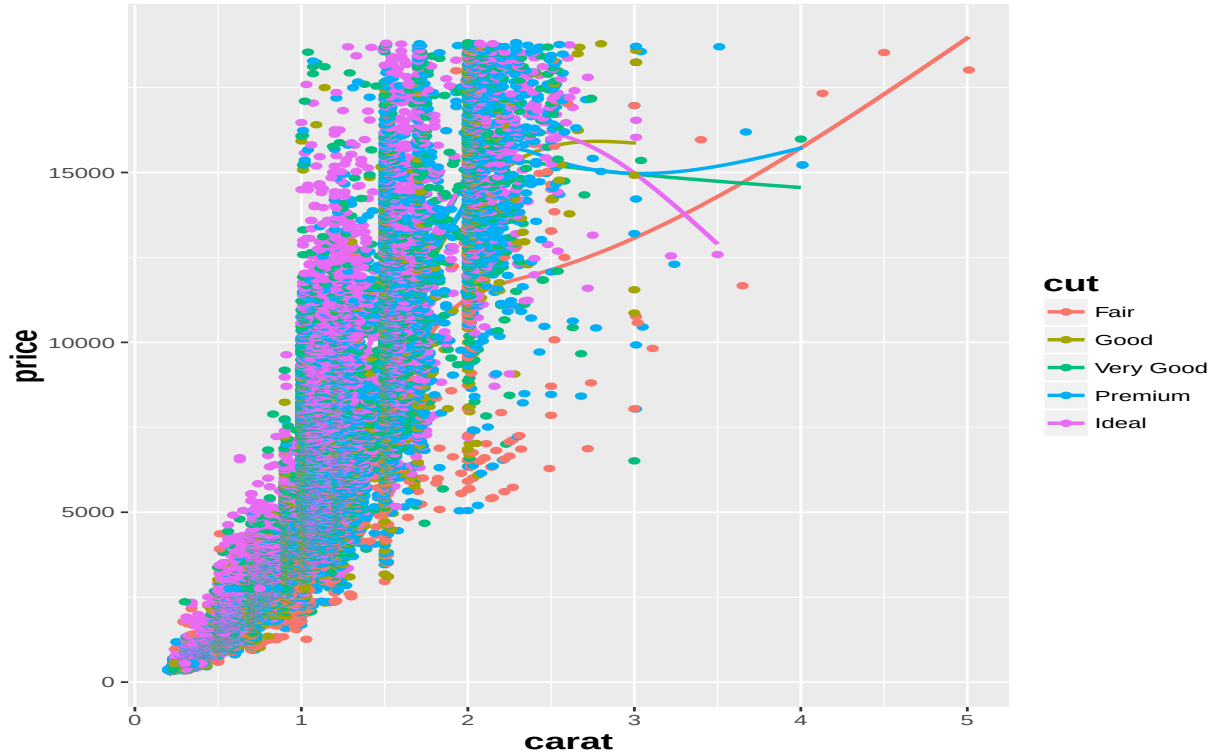
Codebeispiel 9: Mapping und Setting

```
ggplot(airquality, aes(x = Temp, y = Wind, colour = Ozone)) +
    geom_point(size = 4) + geom_smooth(fill = "yellow", colour
    = "red") #Mapping: Ozone, Setting: yellow, red, 4
```



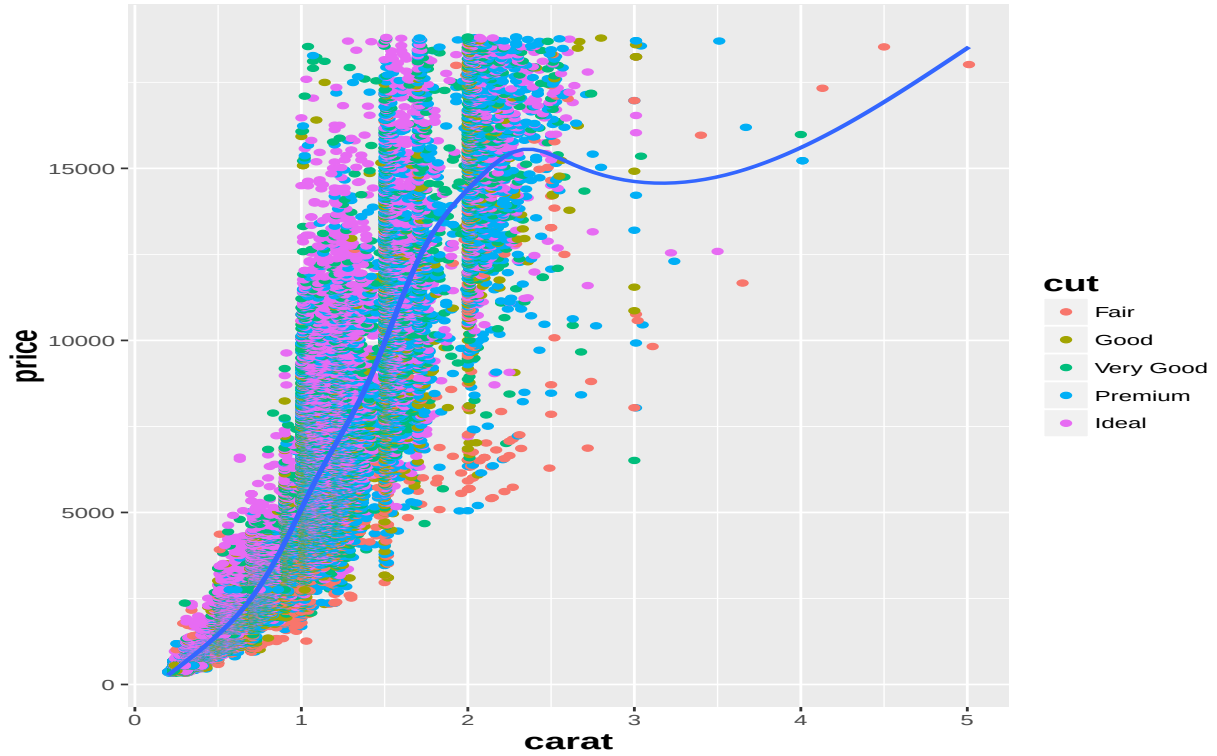
Codebeispiel 10 : Mapping innerhalb ggplot

```
1 ggplot(diamonds, aes(x=carat, y=price, colour=cut))
    + geom_point() + geom_smooth(se=F) #Regression für Gruppen
    von Punkten mit gleichem Schliff
```



Codebeispiel 11 : Mapping innerhalb ggplot und geom_point

```
ggplot(diamonds, aes(x=carat, y=price)) + geom_point(aes(colour=
    cut)) + geom_smooth(se=F) #Regression für Gruppen von
    Punkten mit gleichem Schliff
```



Scaling

- Erstellen einer benutzerdefinierten Legende

Stetige Plots

- Legendenposition theme(legend.position = ...)
  - „top“, „bottom“, „left“, „right“
- Achsenlabels
  - x-Achse scale_x_continuous = *
  - y-Achse scale_y_continuous = *

* Einfügen eines Strings
- Einfügen eines Zahlenwertes

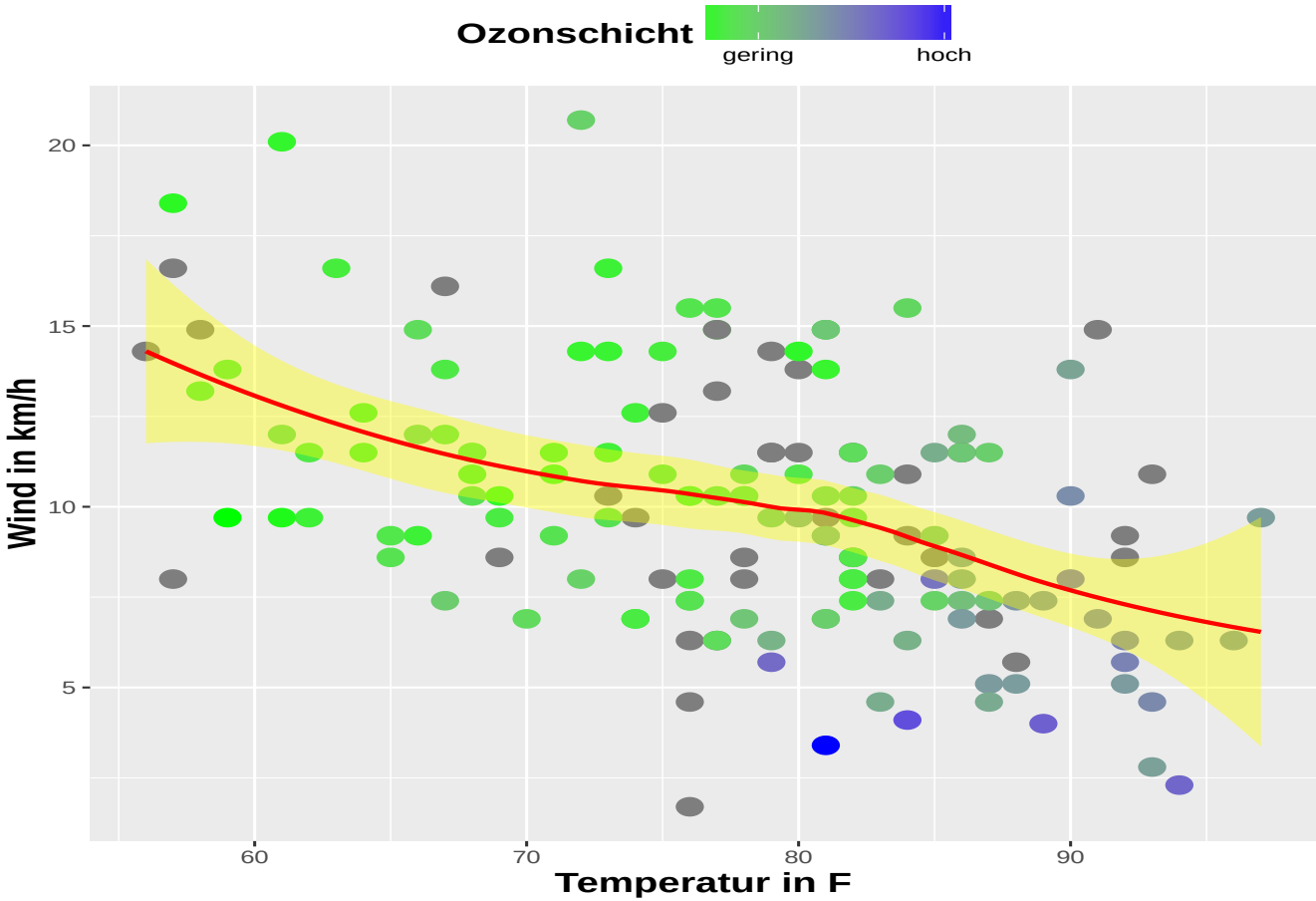
- Gestaltung der Legende *scale_colour_continuous()*
  - Name der Legende *name* = *
  - Abschnitte der Legende *breaks* = c(-,-,-)
  - Abschnittsbezeichnungen *labels* = c(*,*,*)
  - Farbverlauf *low* = *, *high* = *
- Erweiterung *scale_colour_gradient2()*
  - dreifarbige Erweiterung *low*, *mid*, *high*

Diskrete Plots

- *scale_colour_discrete()*
- *scale_x_discrete*, *scale_y_discrete*
- identische Funktionen

Codebeispiel 12 : benutzerdefinierte Legende

```
g1 + theme(legend.position="top")
+ scale_colour_continuous(name="Ozonschicht", breaks=c(40, 160),
      labels=c("gering", "hoch"), low ="green", high="blue")
+ scale_x_continuous("Temperatur in F")
+ scale_y_continuous("Wind in km/h")
```



zu Codebeispiel 12: Legendenerstellung

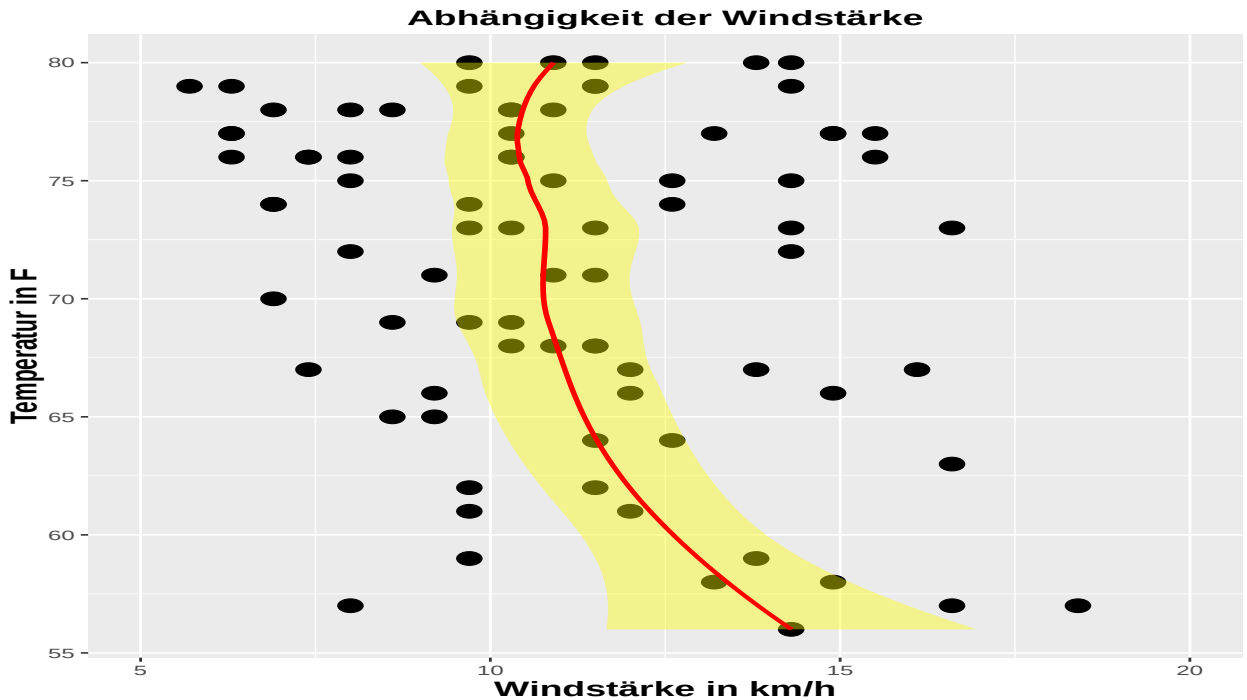
Stetige und diskrete Plots

- *ggtitle(*)* Titel des Koordinatensystems
- *xlab(*)*, *ylab(*)* Achsenbeschriftung
- *xlim =(-,-)*, *ylim =(-,-)* Achsenbegrenzung
- *coord_cartesian()* Zoom eines Achsenabschnitts
  - *xlim = c(-,-)*, *ylim=c(-,-)*, *expand = T/F*
- Zusammenfassung *labs(list(title = *, x = *, y = *))*
- Zusammenfassung *lims(x = c(-,-), y = x(-,-))*

- *coord_flip()* Vertauschen der Achsen
- *coord_fixed(ratio = -)* Achsenlänge im Verhältnis $\frac{y}{x}$
- *coord_trans(*)* Achsentransformation
  - *log2*, *sqrt*, ...
- Umkehren der x-Achsenrichtung *scale_x_reverse()*
- Umkehren der y-Achsenrichtung *scale_y_reverse()*
- keine Achsenabschnittslabels
  - *theme(axis.ticks = element_blank(), axis.ticks.y = element_blank()*

Codebeispiel 13 : benutzerdefiniertes Koordinatensystem

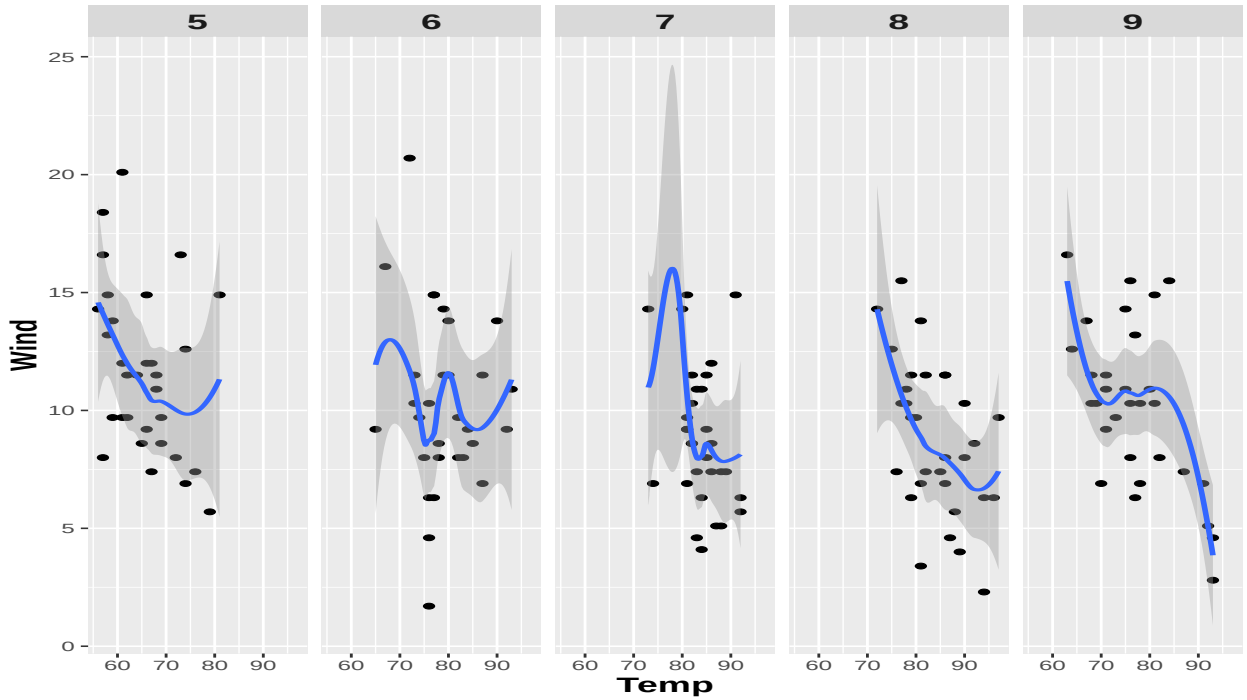
```
g1 + labs(title="Abhängigkeit der Windstärke" ,x="Temperatur in
    F", y="Windstärke in km/h")
+ lims(x=c(NA, 80), y=c(5, 20))
+ coord_flip()
```



- Splitten der Visualisierung in Variablen
- Implementation durch *facet_grid()*
  - Unterscheidung in horizontale oder vertikale Richtung
  - Abhängigkeit von maximal zwei weiteren Variablen
  - Notation horizontaler Aufbau *. ~ data*
  - Notation vertikaler Aufbau *data ~.*
- Implementation durch *facet_wrap()*
  - Eingabe der Plot-Anzahl in einer Zeile/Spalte
  - Abhängigkeit von einer Variable
  - Notation *data*, *ncol= -*/*nrow= -*

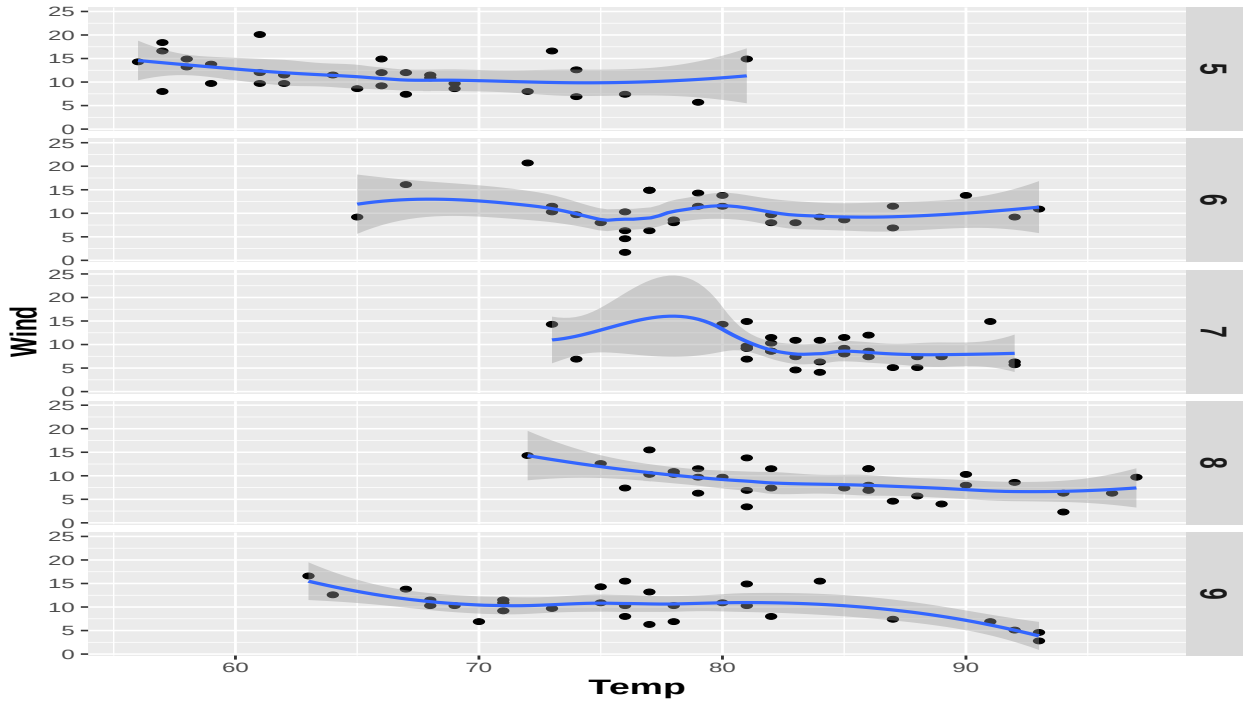
Codebeispiel 14 : horizontale Anordnung

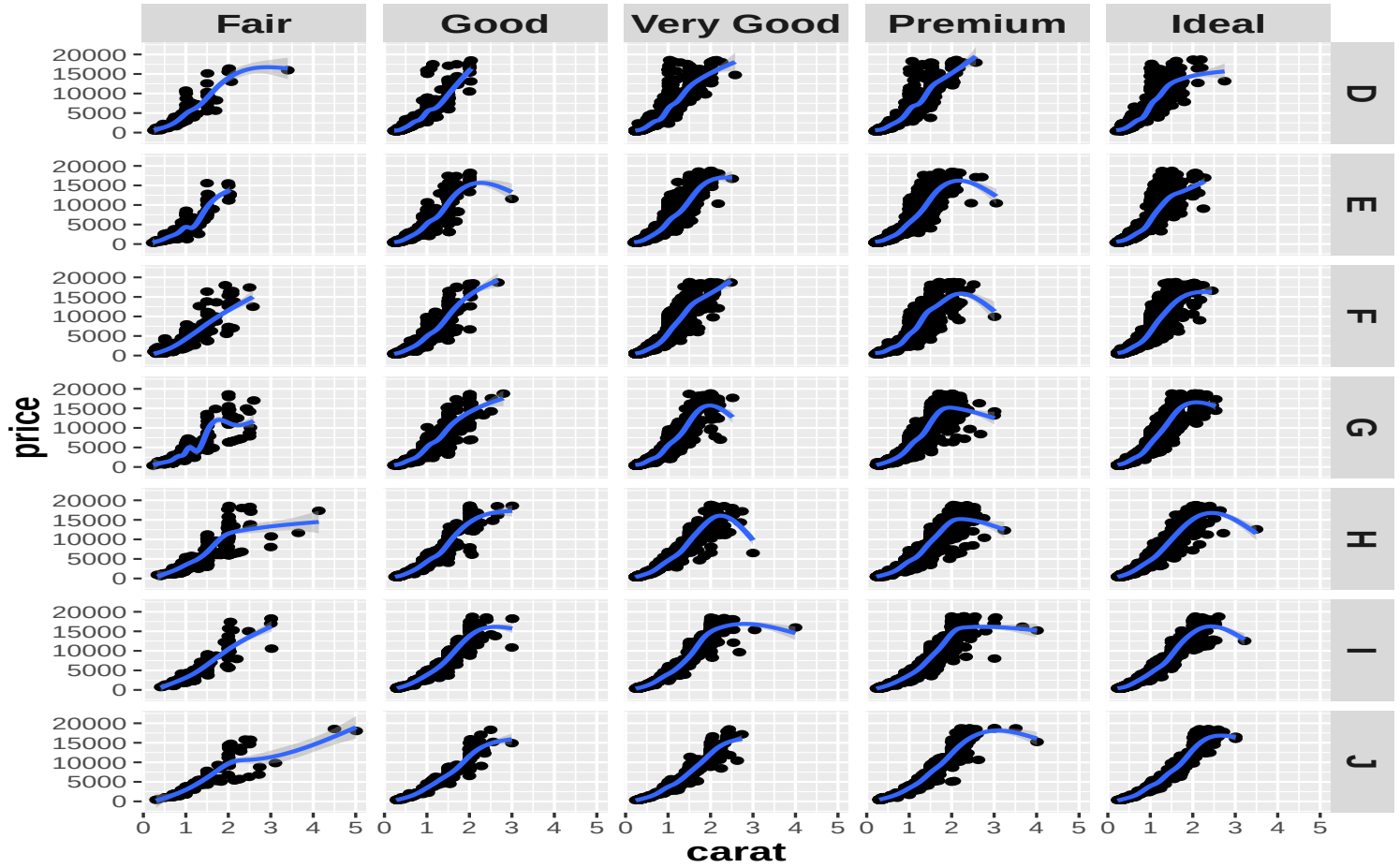
```
1 ggplot(airquality, aes(x=Temp, y=Wind)) + geom_point() +
geom_smooth() + facet_grid(.~Month)
```



Codebeispiel 15 : vertikale Anordnung

```
ggplot(airquality, aes(x=Temp, y=Wind)) + geom_point() +
geom_smooth() + facet_grid(Month~.)
```

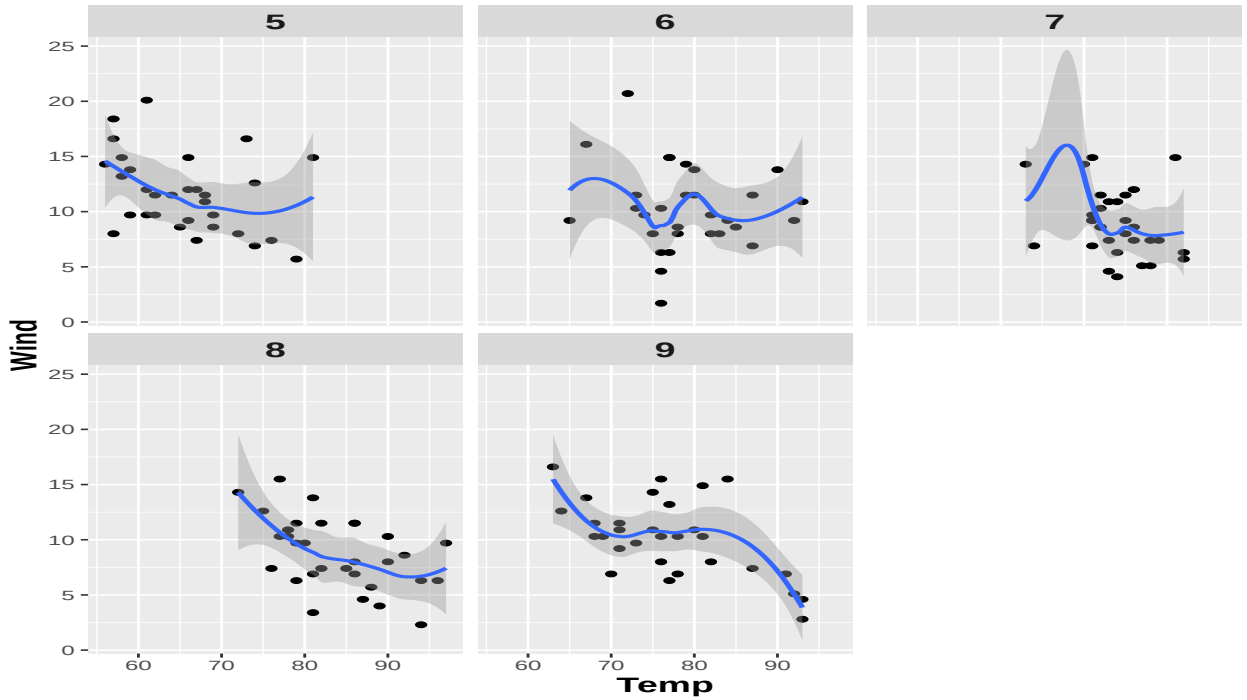




Diamanten unterteilt in Preis, Carat, Farbe und Schliff

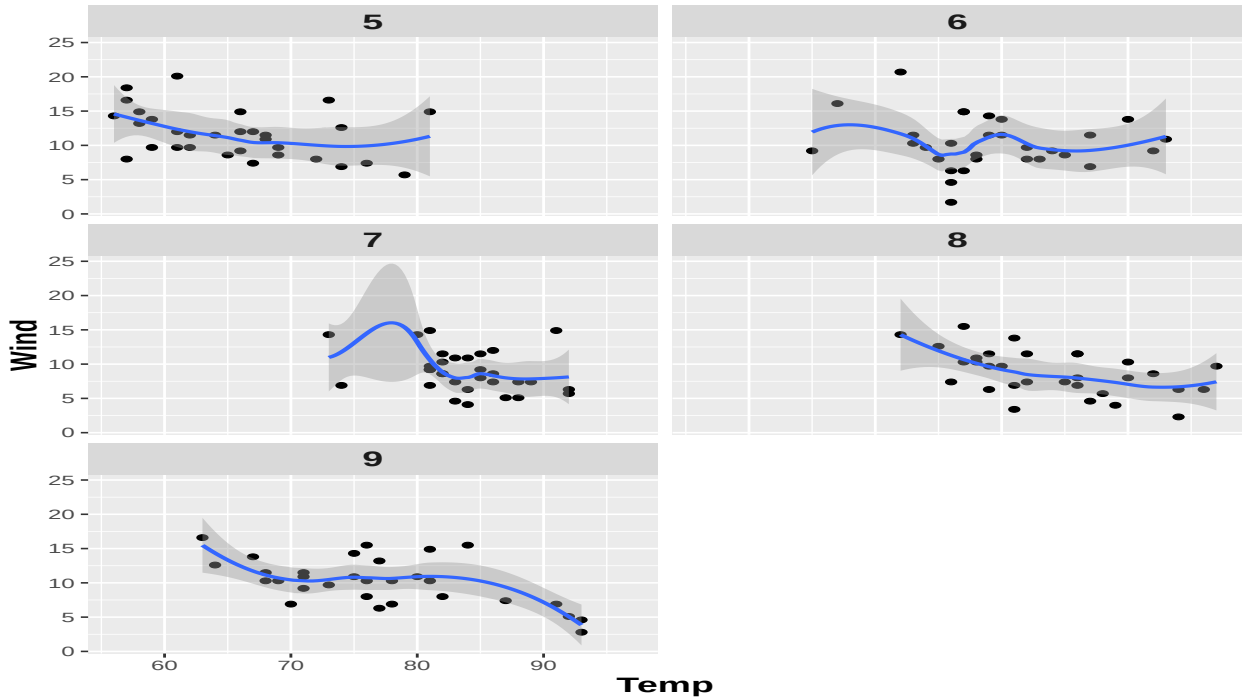
Codebeispiel 16 : 3 Plots pro Zeile

```
ggplot(airquality, aes(x=Temp, y=Wind)) + geom_point() +
geom_smooth() + facet_wrap(~Month, ncol=3)
```



Codebeispiel 17 : 3 Plots pro Spalte

```
ggplot(airquality, aes(x=Temp, y=Wind)) + geom_point() +
geom_smooth() + facet_wrap(~Month, nrow=3)
```

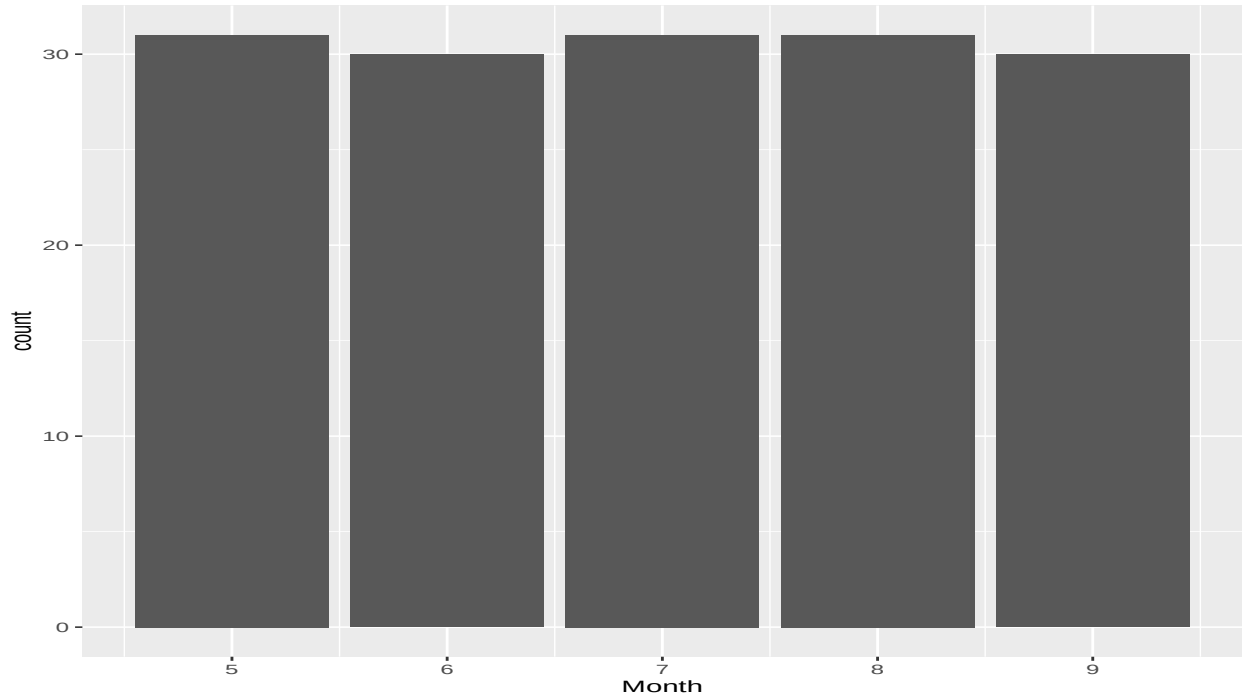


Statistische Transformation

- Überschreiben des Defaults der *geom*-Funktionen
- Umgestalten der Daten
  - *stat_smooth()*, *stat_bin()*,...
- Integrieren eigener Funktionen
  - *stat_summary()*, *stat_function()*,...

Codebeispiel 18 : default-Balkendiagramm

```
ggplot(airquality, aes(x=Month)) + geom_bar()
#Balkendiagramm fordert nur Parameter für x-Achse
```



Codebeispiel 19 : Error-Funktion

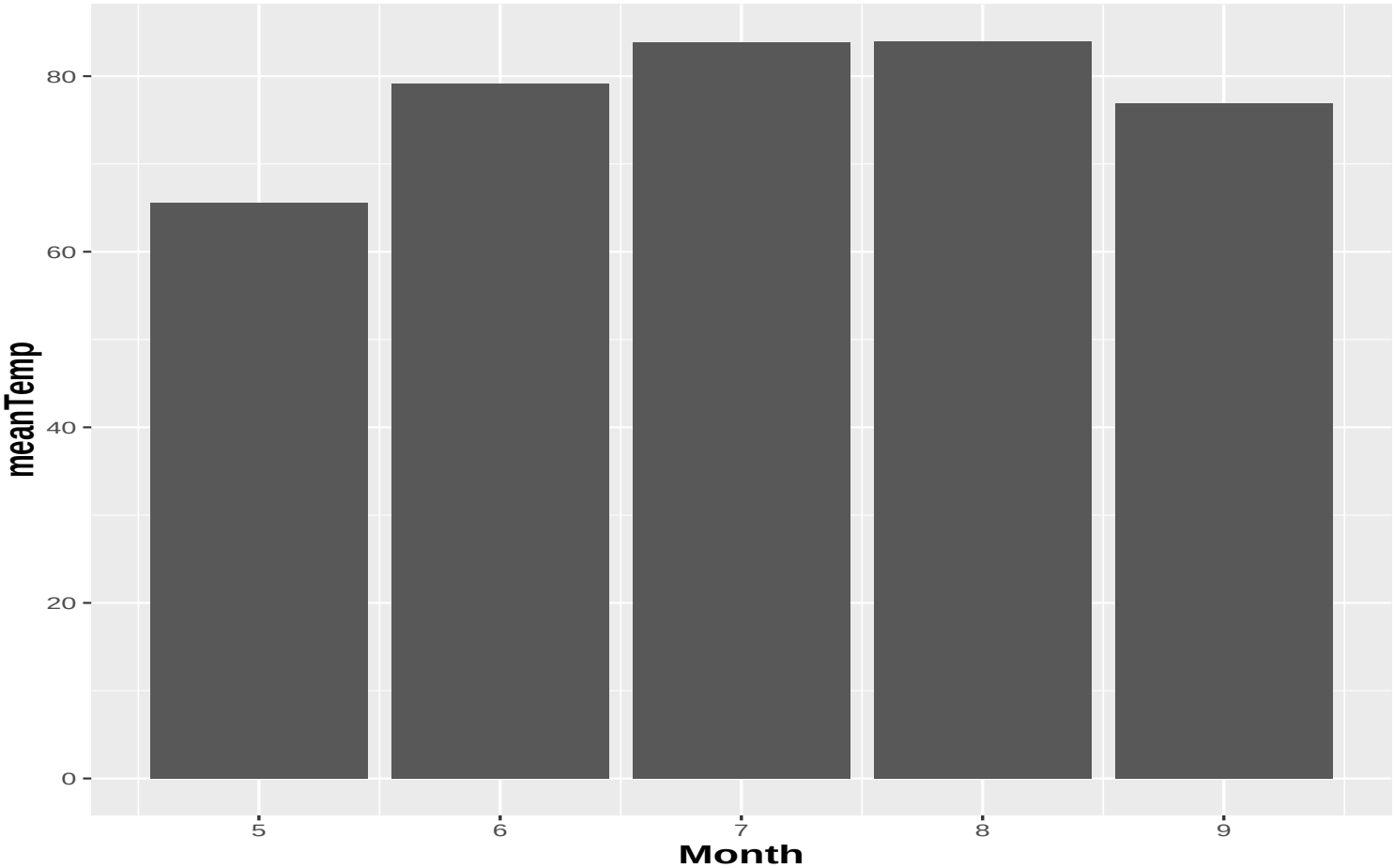
```
ggplot(airquality_means, aes(x=Month, y=meanTemp)) + geom_bar()
#Fehler bei Übergabe eines y-Parameters
```

```
Error: stat_count() must not be used with a y aesthetic.
```

- default-Parameter y = count
- benutzerdefinierte Eingabe des y-Parameters führt zu Übergabe zweier Werte eines Parameters

Codebeispiel 20: Änderung der statistischen Eigenschaften

```
ggplot(airquality_means, aes(x=Month, y=meanTemp)) + geom_bar(
    stat="identity")
#stat="identity" ermöglicht das Überschreiben des y-Parameters
```



zu Codebeispiel 20: Ein Balkendiagramm mit benutzerdefinierter x- und y-Achse

Export von Plots

- Export und Speicherung von Plots
- Implementation über *ggsave()*
  - *filename* Auswahl des Namens und Formats
  - *plot* Auswahl des Plots
    - default Abspeichern des letzten Plots
  - *device* Angabe bei unbekanntem Format
  - *path* Auswahl des Speicherpfads
    - default-Ordner „Dokumente“
  - *scale*, *width*, *height* Größenanpassungen
  - *unit* Einheit der Größenanpassung (cm, inch,...)

Codebeispiel 21: Automatische Speicherung

```
ggplot(airquality, aes(x=Temp, y=Wind)) + geom_point()
ggsave("meingraph.pdf")
#Abspeichern als PDF unter "meingraph" im Ordner "Dokumente"
```

Codebeispiel 22: Speicherung mit Pfadangabe

```
g1 <- ggplot(airquality, aes(x=Temp, y=Wind))
+ geom_point()
g2 <- ggplot(airquality, aes(x=Ozone, y=Temp))
+ geom_point()
ggsave("meingraph.png", g1, path="C:/Users/Max Mustermann/
    Desktop/Graphen")
#Abspeichern des Graphen g1 als PNG unter "meingraph" im Ordner
    "Graphen"
```

essentielle Fragen

- Welcher Art sind die Daten? *stetig*, *diskret*
- Womit stelle ich die Daten dar? *geom*
- Wie stelle ich die Daten dar? *aes*
- Passe ich die Graphenumgebung des Plots an? *coord*, *theme*, *scale*, *facets*, *ggtheme*
- Muss ich die default-Funktion überschreiben? *stat*
- Wie exportiere und speichere ich Plots? *ggsave*

Fragen?

ggthemes

- Paket *ggthemes*
- zusätzliche Gestaltungs- und geom-Funktionen

Funktionen

- *geom_rangeframe()* Achsenbeginn bei erstem Funktionswert, Achsenende bei letztem Wert
- *geom_tufteboxplot()* Boxplots nach Tufte
  - Punkt als Median
  - Lücken als Interquartilabstand
  - Linien als Whiskers

Themes

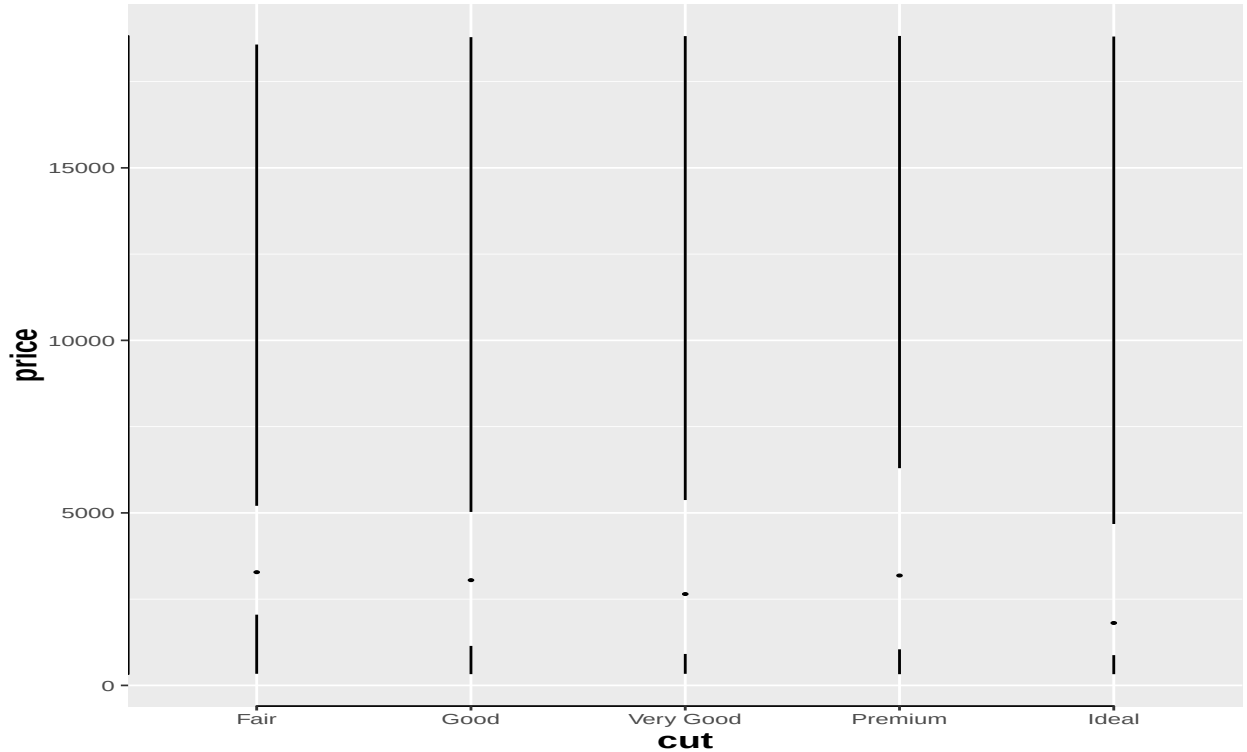
- *theme_economist()* Plots des Economist Magazins
- *theme_solarized()* Plots der solarized Farbpalette
- *theme_solid()* benutzerdefinierte Hintergrundfarbe

Scales

- *scale_colour_economist()* Achsen des Economist
- *scale_colour_solarized()* Achsen der solarized Palette
- *scale_colour_excel()* Achsen aus Excel
- *scale_colour_colorblind()* Achsen für Farbblinde

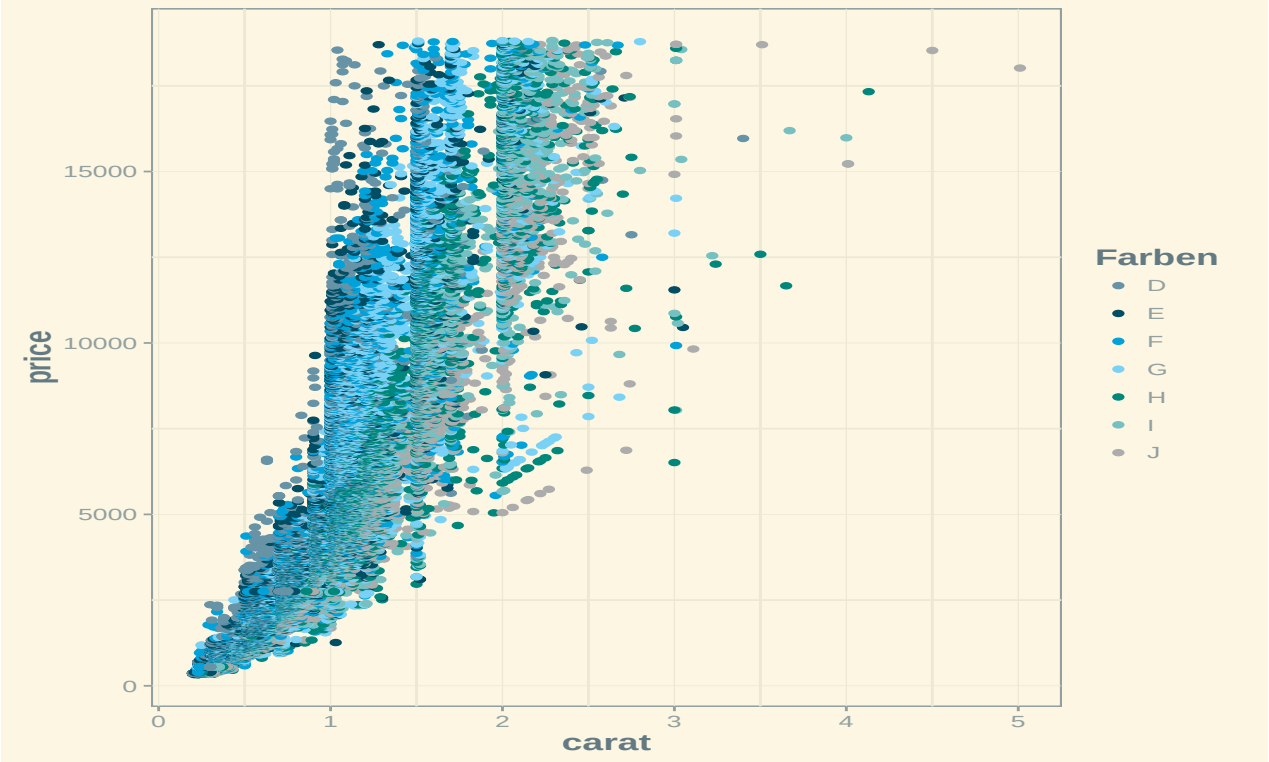
Codebeispiel 23: Tufteboxplot samt Rangeframe-Achse

```
install.packages("ggthemes")
library(ggthemes)
ggplot(diamonds, aes(x=cut, y=price)) + geom_tufteboxplot()
+ geom_rangeframe()
```



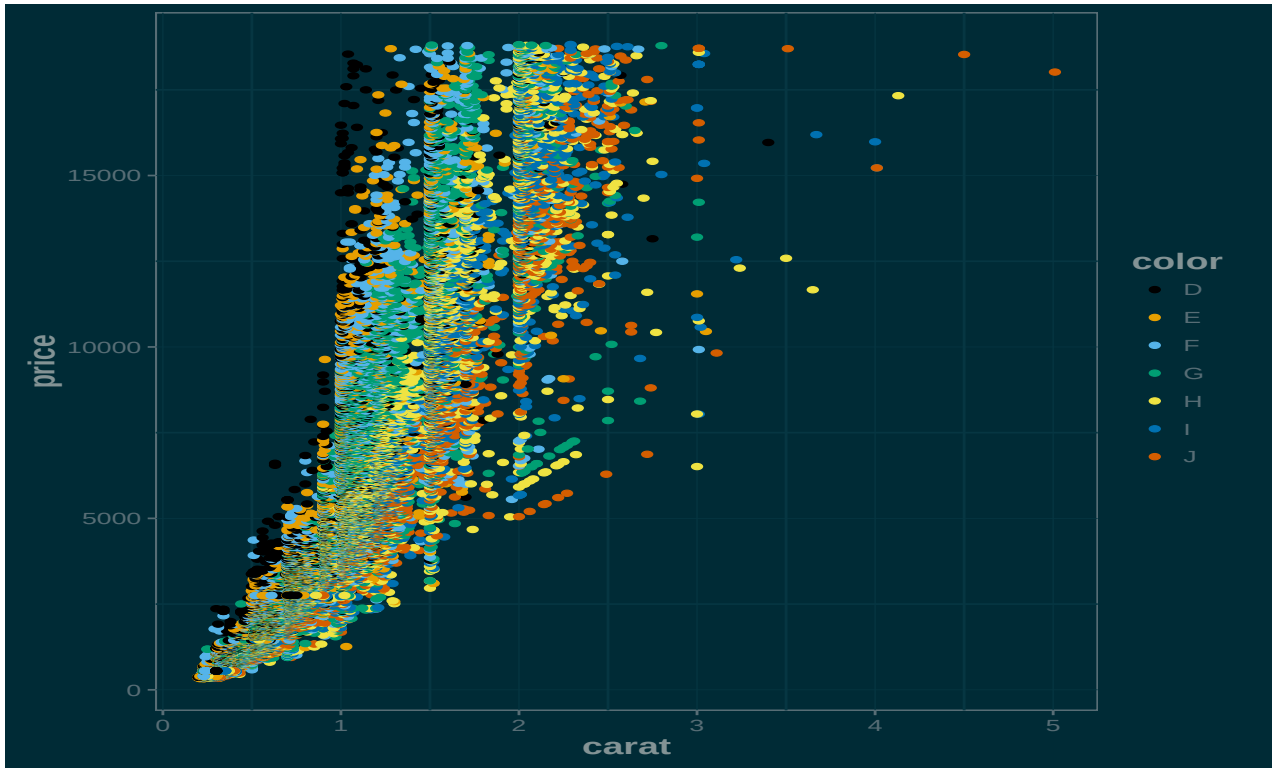
Codebeispiel 24: solarized-Theme mit Economist Legende und Parameter

```
ggplot(diamonds, aes(x=carat, y=price, colour=color)) + geom_
    point() + theme_solarized() + scale_colour_economist(name="
    Farben")
```



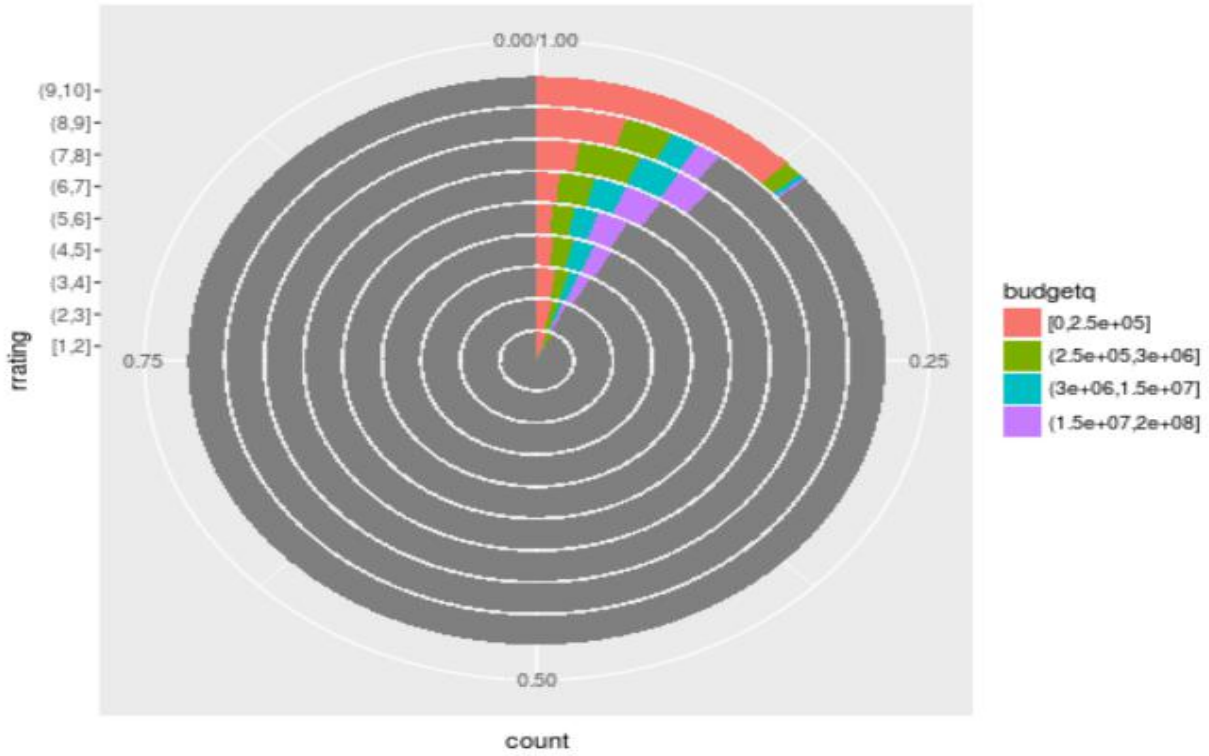
Codebeispiel 25: solarized-Theme mit Parameter und Legende für Farbblinde

```
1 ggplot(diamonds, aes(x=carat, y=price, colour=color)) + geom_
      point() + theme_solarized(light=F) + scale_colour_
      colorblind()
```



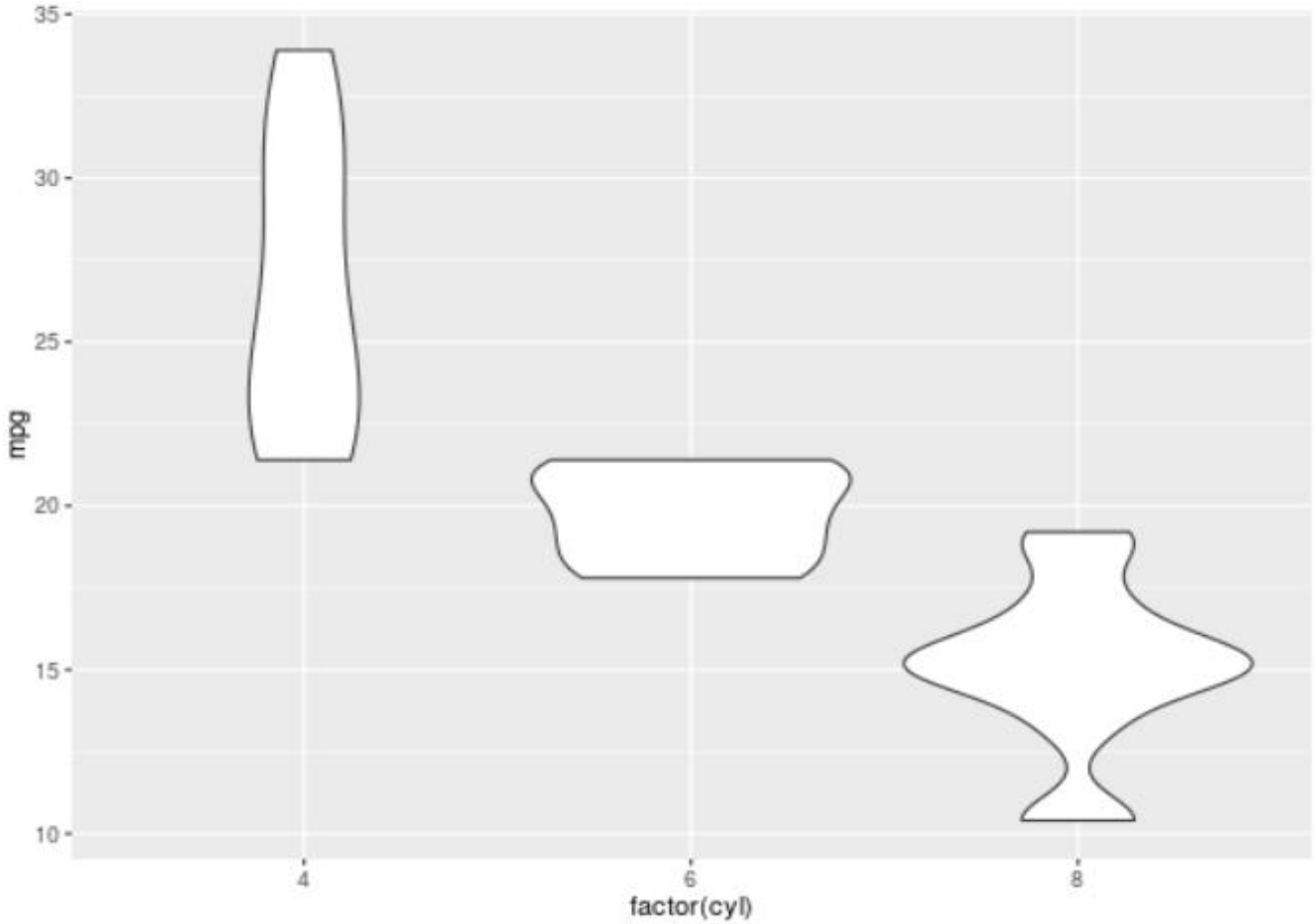
Gallerie

coord_polar



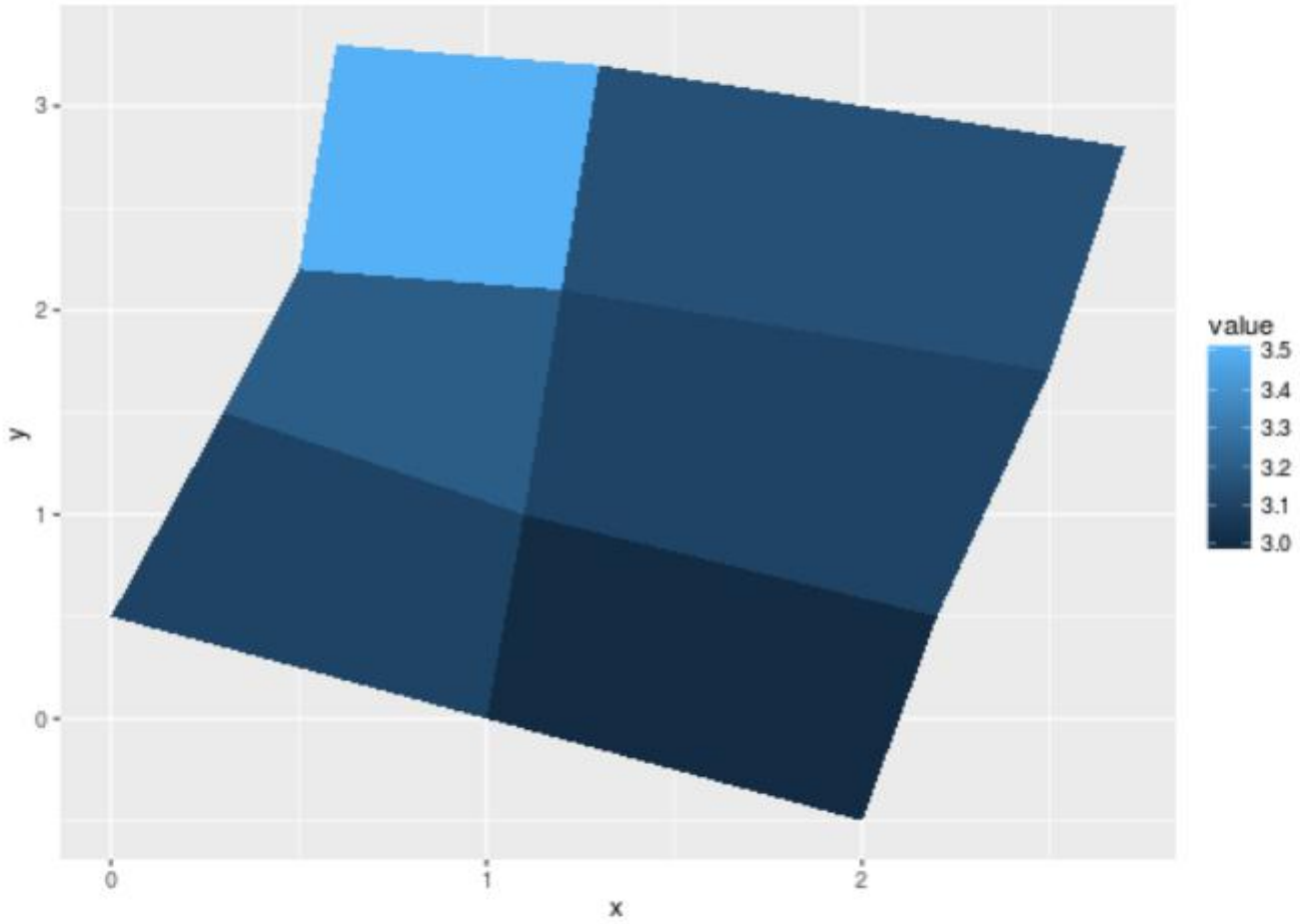
http://docs.ggplot2.org/current/coord_polar.html

geom_violin



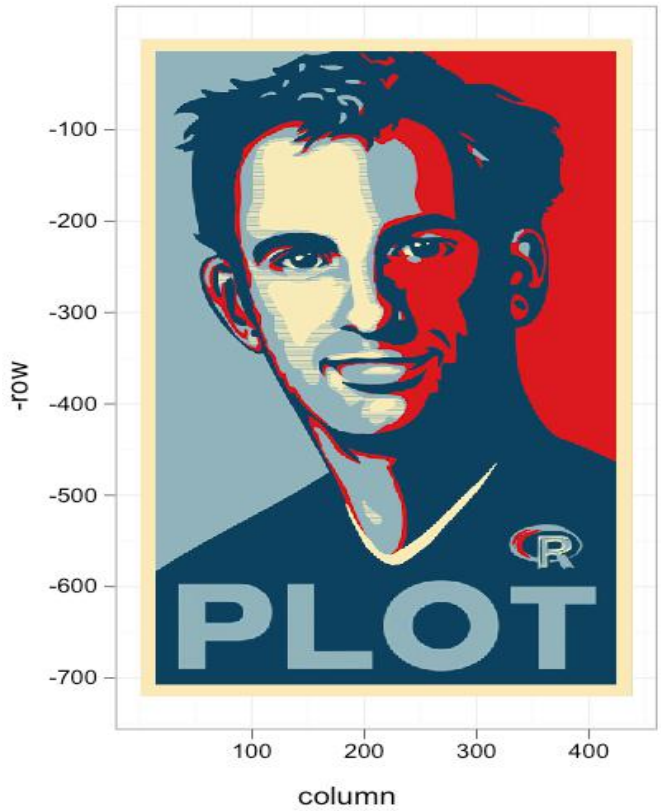
http://docs.ggplot2.org/current/geom_violin.html

geom_map



http://docs.ggplot2.org/current/geom_map.html

Ein Porträt



Porträt von Hadley Wickham erstellt mit GGPlot2 von David Kahle und Garrett Grolemund
http://priceonomics.com/hadley-wickham-the-man-who-revolutionized-r/

Quellenverzeichnis

- http://ggplot2.org/resources/2007-past-present-future.pdf
- http://en.wikipedia.org/wiki/Ggplot2
- http://varianceexplained.org/r/why-I-use-ggplot2/
- http://www.sthda.com/english/wiki/ggplot2-essentials
- http://ggplot2.org/book/qplot.pdf
- http://varianceexplained.org/RData/lessons/lesson2/
- http://www.aridhia.com/technical-tutorials/the-fundamentals-of-ggplot-explained/
- https://rpubs.com/hadley/ggplot2-layers
- http://www.cookbook-r.com/Graphs/Facets_%28ggplot2%29/
- http://www.cookbook-r.com/Graphs/Axes_%28ggplot2%29/
- http://www.cookbook-r.com/Graphs/Legends_%28ggplot2%29/
- https://cran.r-project.org/web/packages/ggthemes/vignettes/ggthemes.html
- http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html

- https://cran.r-project.org/web/packages/ggthemes/ggthemes.pdf
- http://docs.ggplot2.org/current/geom_text.html
- https://cran.r-project.org/web/packages/ggrepel/vignettes/ggrepel.html
- http://sape.inf.usi.ch/quick-reference/ggplot2/coord
- http://docs.ggplot2.org/current/theme.html
- http://docs.ggplot2.org/current/scale_continuous.html
- http://docs.ggplot2.org/current/ggsave.html
- http://blog.echen.me/2012/01/17/quick-introduction-to-ggplot2/
- http://www.cookbook-r.com/Manipulating_data/Summarizing_data/
- https://cran.r-project.org/web/packages/plyr/index.html
- http://seananderson.ca/2013/10/19/reshape.html
- Statistik für Betriebswirte I, A. Johannssen