Einführung in R

Clemens Brunner

14.-15.2.2019

Grafiken

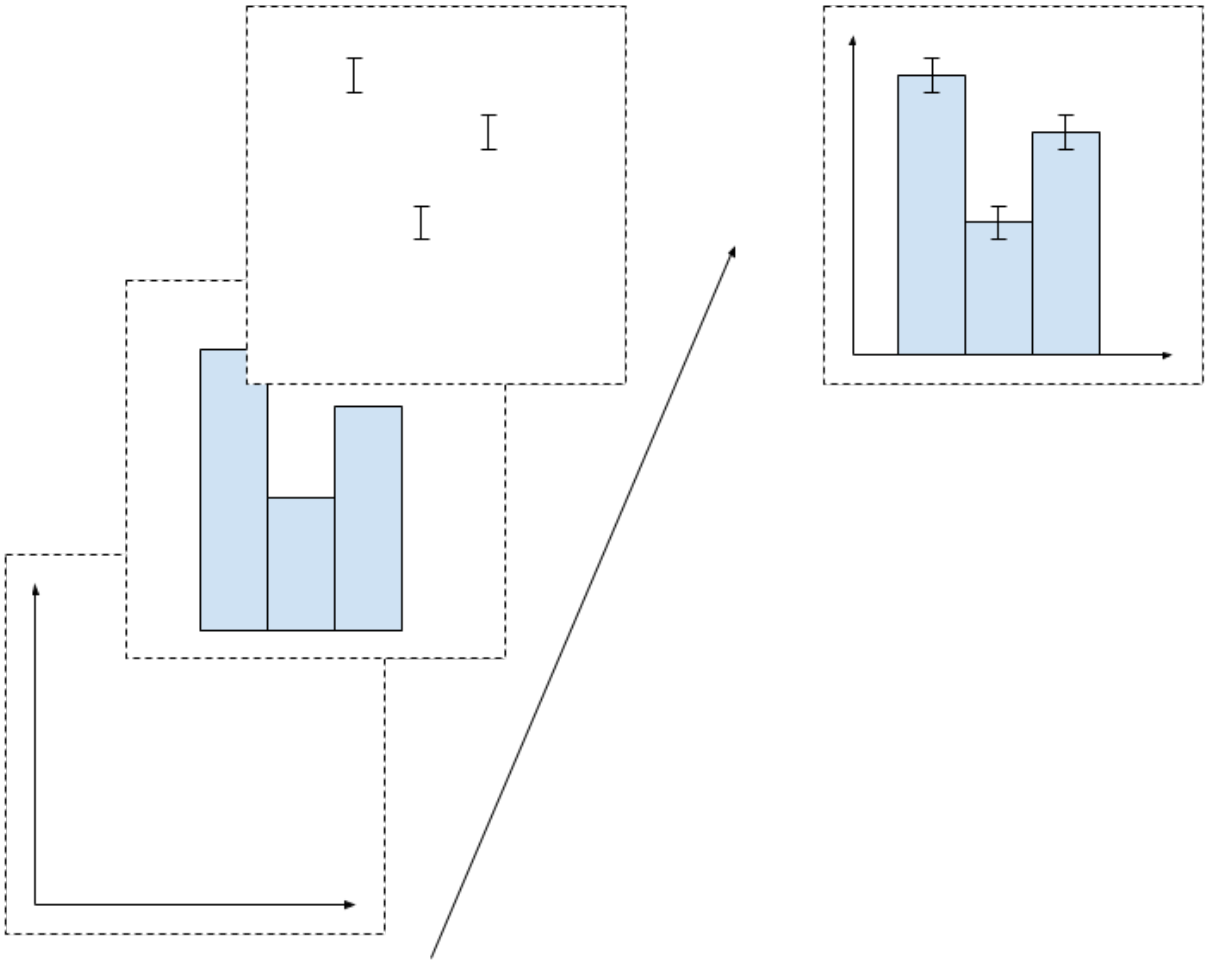
In R gibt es mehrere Pakete, mit denen man Daten grafisch darstellen kann. Die bekanntesten Pakete sind `graphics` (standardmäßig bei R dabei, wird deshalb oft als Base Graphics bezeichnet), `ggplot2` und `lattice`. Grafiken, welche mit unterschiedlichen Paketen erstellt wurden, lassen sich aber nicht miteinander kombinieren. Deshalb entscheidet man sich typischerweise vor der Erstellung einer Grafik für das zu verwendende Grafikpaket. In dieser Veranstaltung werden wir uns nur mit `ggplot2` beschäftigen (welches Teil des Tidyverse ist). Im Gegensatz zum Standard-Grafikpaket `graphics` muss das Paket `ggplot2` einmalig installiert und danach aktiviert werden.

```
library(ggplot2)
```

Das Paket ist eine Implementierung der sogenannten Grammar of Graphics, welche statistische Grafiken mit einheitlichen grundlegenden Elementen zu beschreiben versucht. Dies hat den Vorteil, dass man mit dieser Grammatik die unterschiedlichsten Grafiken zusammenbauen kann und nicht für jede Darstellung einen anderen Befehl benötigt.

Aufbau einer Grafik

Eine Grafik in `ggplot2` besteht aus verschiedenen Ebenen. Jede Ebene beinhaltet geometrische Elemente (Geoms genannt) - z.B. Balken, Linien oder Text. Geoms haben ästhetische Eigenschaften (kurz Aes genannt), welche das Aussehen von Geoms bestimmen (beispielsweise Farben oder Linienstile). Diese ästhetischen Eigenschaften können global für den gesamten Plot oder für individuelle Ebenen bzw. Geoms definiert werden. Das folgende Bild veranschaulicht den Aufbau einer Grafik in `ggplot2`.



Die `qplot`-Funktion

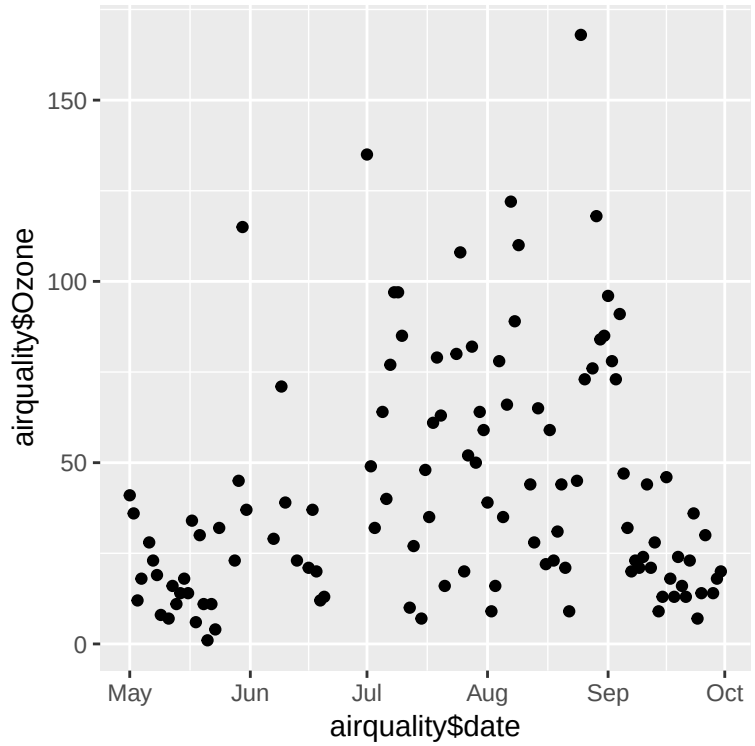
Wenn man bereits mit dem Base-Plotting-System vertraut ist, kann man mit der `qplot`-Funktion sehr ähnlich Grafiken mit dem `ggplot2`-Paket erstellen. Dies hat allerdings den Nachteil, dass man das volle Potenzial von `ggplot2` nicht ausschöpfen kann - für viele Standardgrafiken ist `qplot` aber eine gute Wahl.

Erstellen wir mit `qplot` einige beispielhafte Grafiken. Dazu verwenden wir den Datensatz `airquality`, welcher Teil von R ist. Diese Daten beinhalten diverse Luftgütemessungen in New York im Zeitraum Mai bis September 1973 (siehe dazu `?airquality`). Es ist insbesondere beim Erstellen von Grafiken sinnvoll, wenn wir zunächst aus den beiden Spalten `Month` und `Day` eine neue Spalte namens `date` erzeugen:

```
airquality$date <- as.Date(paste(airquality$Month, airquality$Day, "1973"), format="%m %d %Y")
```

Nun verwenden wir `qplot`, um das Datum auf der x-Achse und die Ozonwerte auf der y-Achse grafisch darzustellen. Diese beiden Spalten kann man direkt als Argumente übergeben.

```
qplot(airquality$date, airquality$Ozone)
```

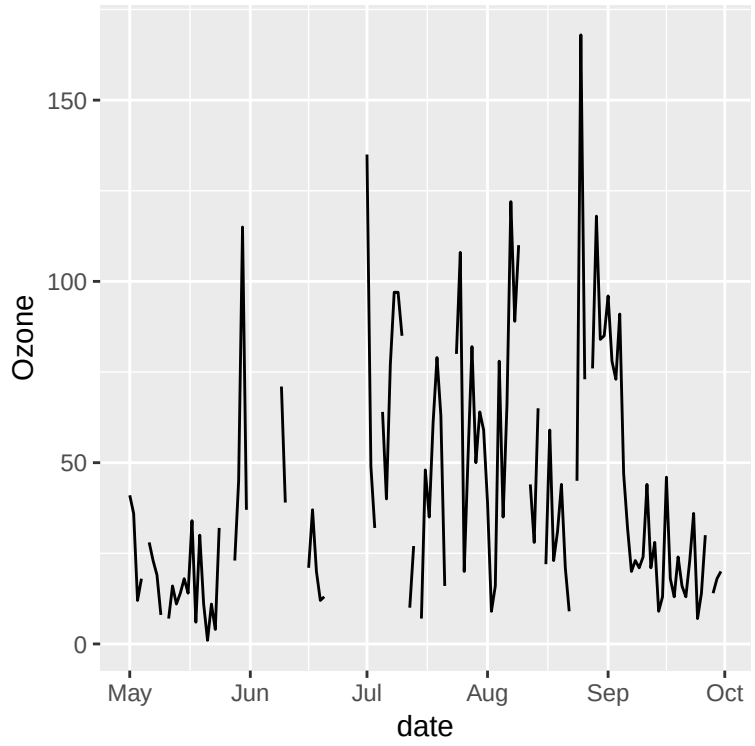


Alternativ kann man, wenn sich die Daten als Spalten in einem Data Frame befinden, die Funktion auch wie folgt aufrufen:

```
qplot(date, Ozone, data=airquality)
```

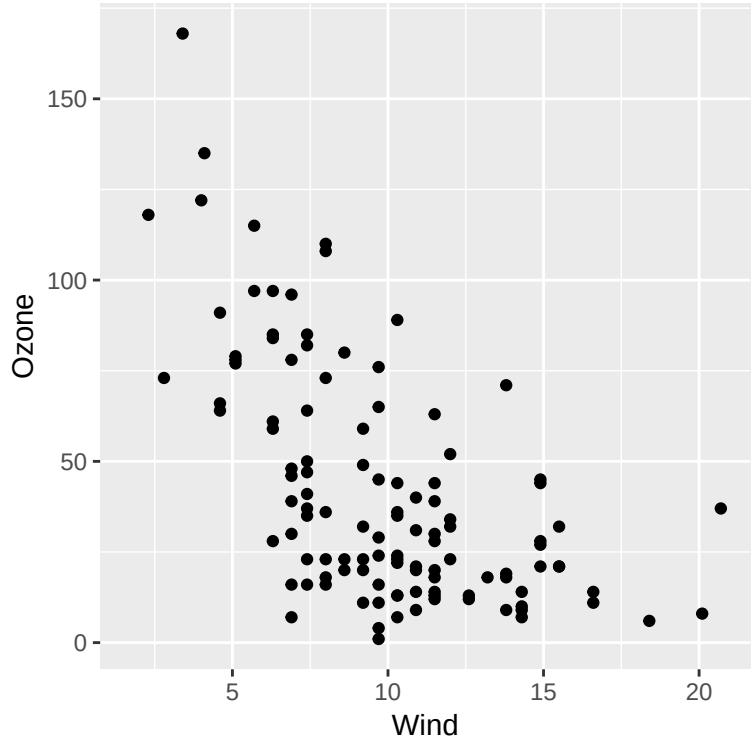
Mit dem zusätzlichen Argument `geom="line"` werden die Daten durch eine Linie verbunden:

```
qplot(date, Ozone, data=airquality, geom="line")
```



Dementsprechend kann man zwei beliebige Spalten gegeneinander auftragen und man erhält einen Scatterplot:

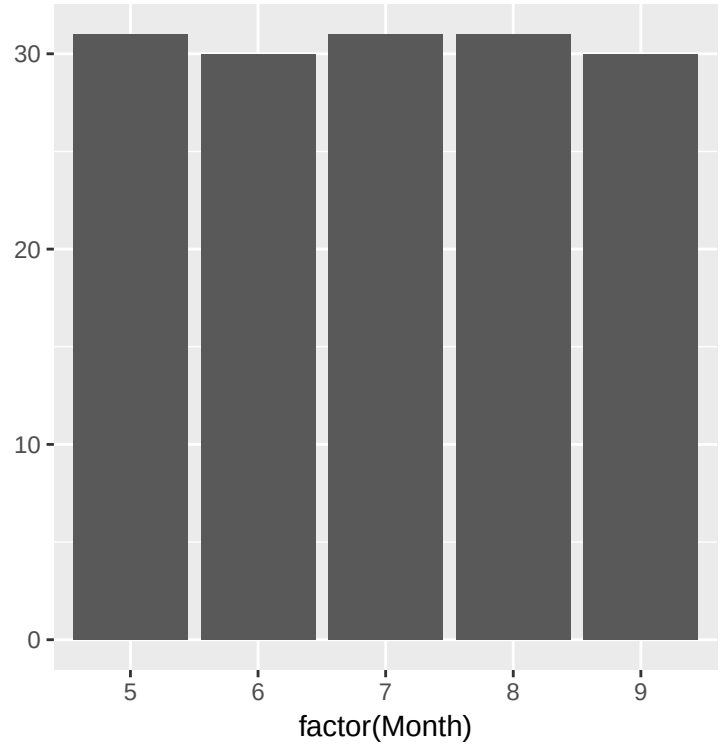
```
qplot(Wind, Ozone, data=airquality)
```



Wenn man einen Faktor übergibt (und das y-Argument weglässt), erhält man eine Balkengrafik mit den

absoluten Häufigkeiten für jede Faktorstufe:

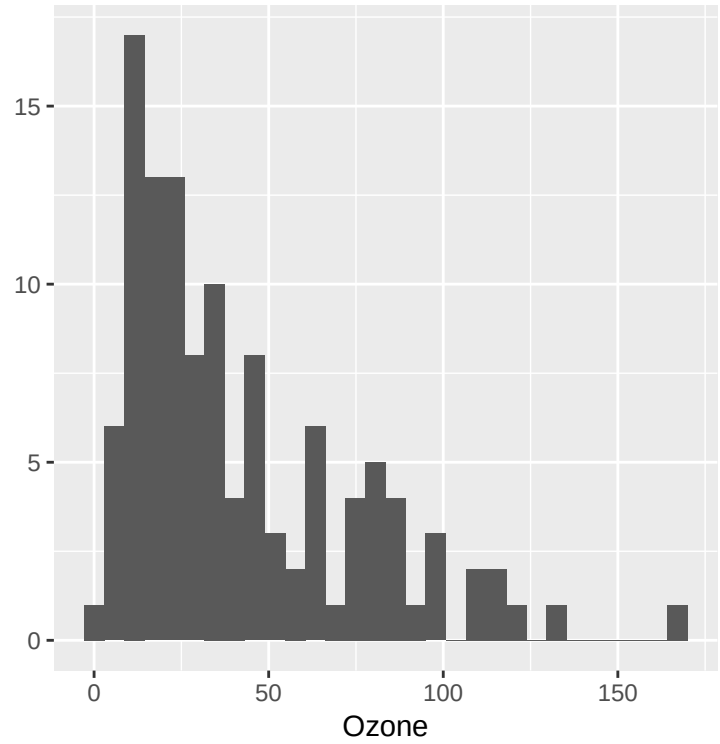
```
qplot(factor(Month), data=airquality)
```



Übergibt man einen numerischen Vektor für das erste Argument (und lässt das `y`-Argument weg), erhält man ein Histogramm:

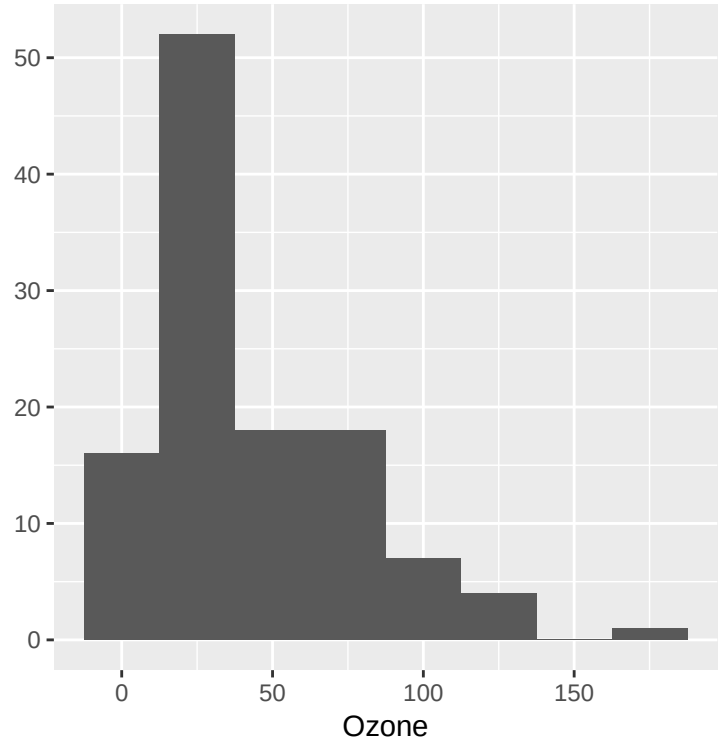
```
qplot(Ozone, data=airquality)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



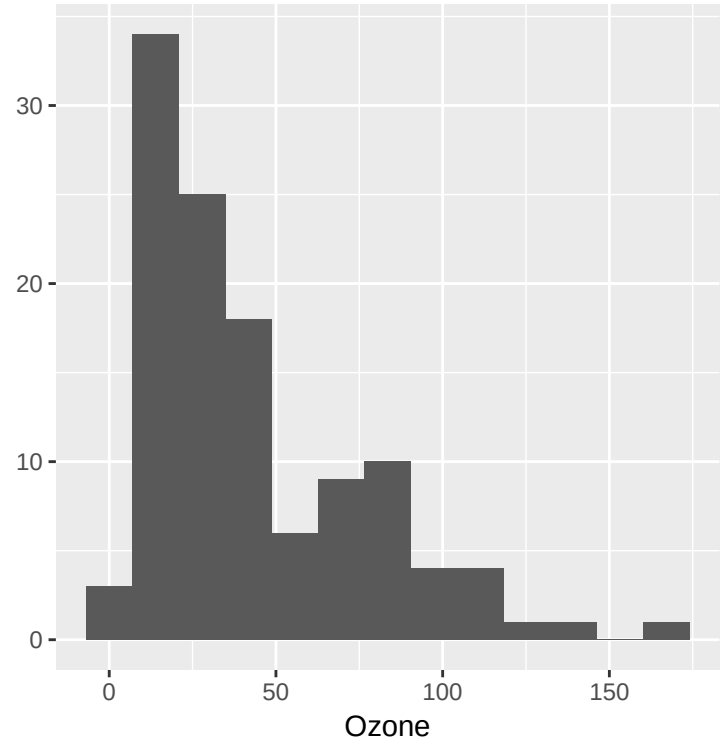
Wie die Meldung bereits andeutet, kann man die Anzahl der Bins manuell anpassen, z.B. indem man die Breite eines Bins angibt:

```
qplot(Ozone, data=airquality, binwidth=25)
```



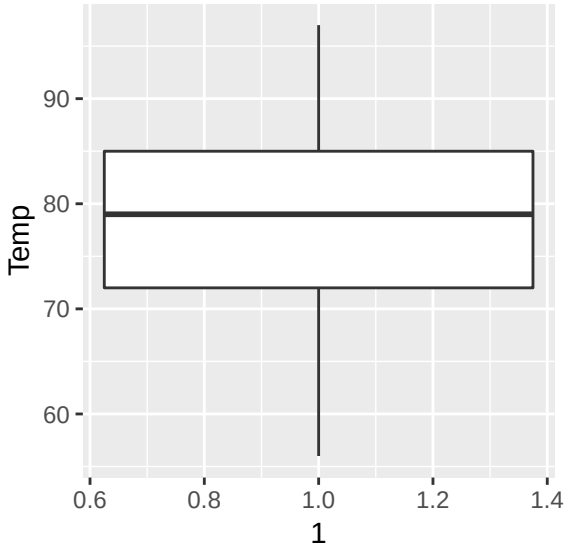
Man kann aber auch die Anzahl der Bins angeben:

```
qplot(Ozone, data=airquality, bins=13)
```



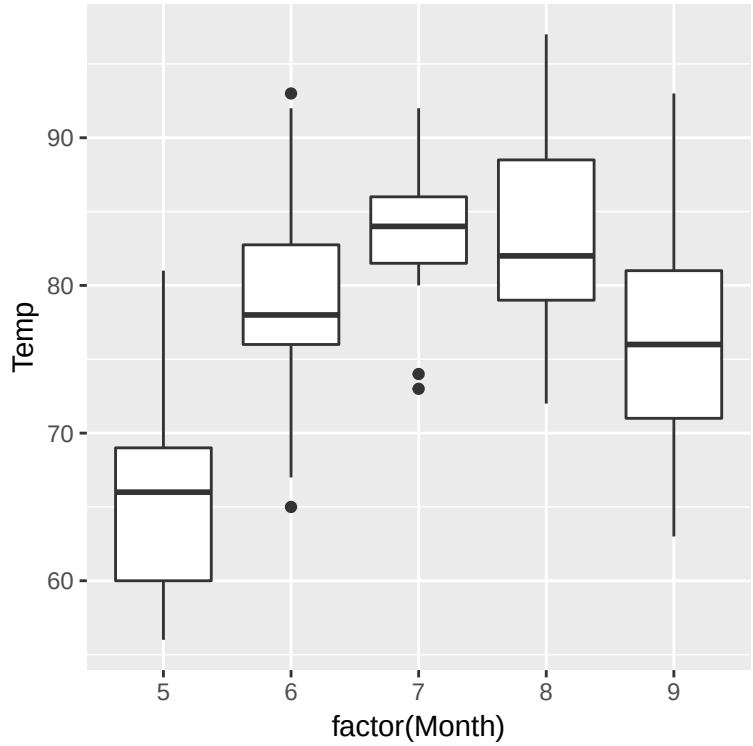
Mit dem Argument `geom` kann man die Art der Grafik festlegen. Beispielsweise kann man mit `geom="boxplot"` einen Boxplot erzeugen.

```
qplot(x=1, y=Temp, data=airquality, geom="boxplot")
```



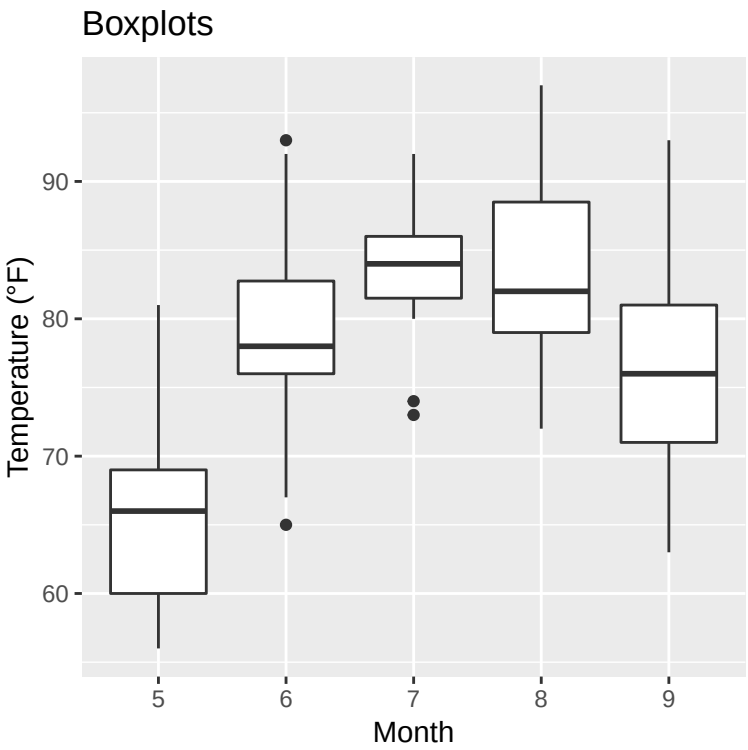
Boxplots sind besonders sinnvoll, wenn man mehrere gleichzeitig darstellt:

```
qplot(x=factor(Month), y=Temp, data=airquality, geom="boxplot")
```



Mit den Argumenten `main`, `xlab` und `ylab` kann man den Titel sowie die Achsenbeschriftungen festlegen:

```
qplot(x=factor(Month), y=Temp, data=airquality, geom="boxplot",
      main="Boxplots", xlab="Month", ylab="Temperature (°F)")
```



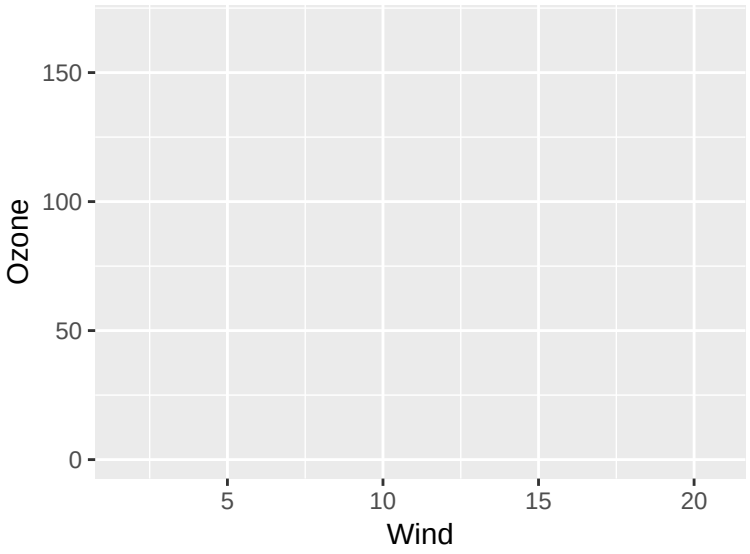
Die `ggplot`-Funktion

Das eigentliche `ggplot2`-System wird durch die `qplot`-Funktion größtenteils versteckt. Stößt man mit `qplot` an die Grenzen, kann man mit der Grammar of Graphics das volle Potenzial von `ggplot2` nutzen - dazu verwendet man dann die `ggplot`-Funktion.

Hier ist nun der Aufbau einer Grafik von Bedeutung - d.h. es ist wichtig zu wissen, dass sich eine Grafik aus mehreren Ebenen zusammensetzt. Man beginnt immer mit der Spezifikation der Grafik, d.h. man legt das Data Frame fest, aus dem die darzustellenden Spalten stammen. Außerdem gibt man die Spalten aus diesem Data Frame an, die man plotten möchte.

Im ersten Schritt legen wir also das Data Frame und die Spalten fest:

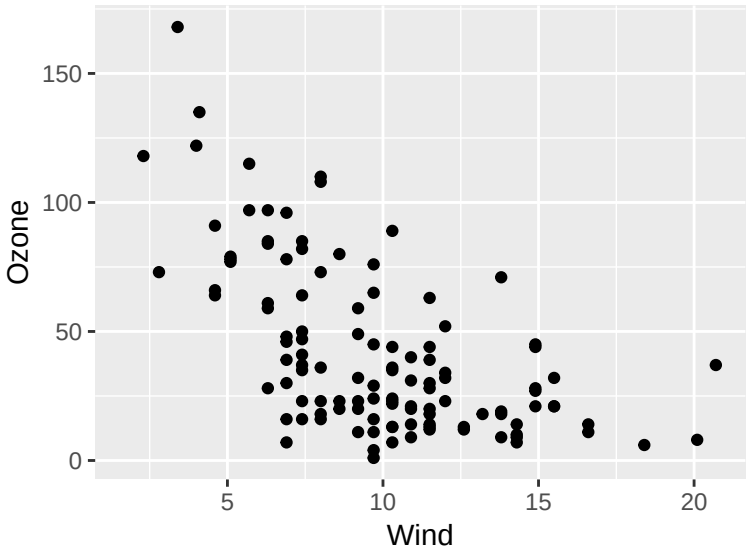
```
ggplot(airquality, aes(Wind, Ozone))
```



Hier ist anzumerken, dass die darzustellenden Spalten innerhalb der Funktion `aes` angegeben werden müssen.

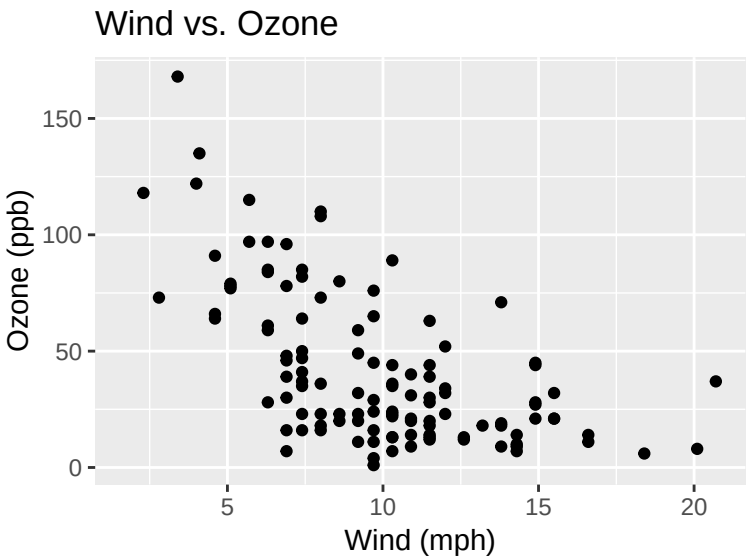
Die eigentlichen Daten werden noch nicht dargestellt - dafür benötigt man eine neue Ebene mit den gewünschten Geoms. In unserem Beispiel möchten wir Punkte zeichnen, d.h. wir verwenden `geom_point` dafür und addieren diese zur ersten Ebene (der Zeilenumbruch ist hier nicht notwendig, erhöht aber die Übersicht):

```
ggplot(airquality, aes(Wind, Ozone)) +
    geom_point()
```



In weiteren Ebenen kann man dann Titel und Achsenbeschriftungen ändern/hinzufügen:

```
ggplot(airquality, aes(Wind, Ozone)) +
    geom_point() +
    ggtitle("Wind vs. Ozone") +
    xlab("Wind (mph)") +
    ylab("Ozone (ppb)")
```

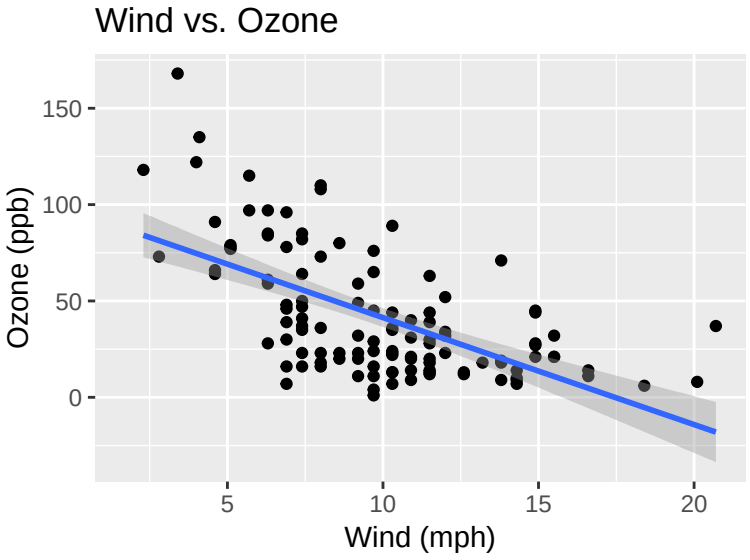


Alternativ kann man alle drei Beschriftungen mit der Funktion `labs` hinzufügen:

```
ggplot(airquality, aes(Wind, Ozone)) +
    geom_point() +
    labs(title="Wind vs. Ozone", x="Wind (mph)", y="Ozone (ppb)")
```

In einer weiteren Ebene kann man dann eine Regressionsgerade (einen sogenannten Smoother) hinzufügen.

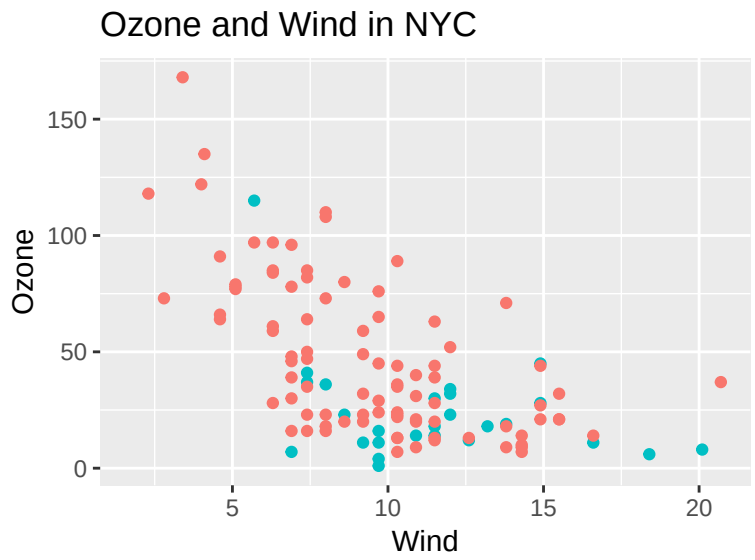
```
ggplot(airquality, aes(Wind, Ozone)) +
    geom_point() +
    labs(title="Wind vs. Ozone", x="Wind (mph)", y="Ozone (ppb)") +
    geom_smooth(method="lm")
```



Möchte man z.B. den Monat Mai (entspricht dem Wert 5) hervorheben, kann man eine globale Ästhetik `color` definieren.

```
ggplot(airquality, aes(Wind, Ozone, color=Month==5)) +
    geom_point() +
```

```
    theme(legend.position="none") +
    labs(title="Ozone and Wind in NYC")
```

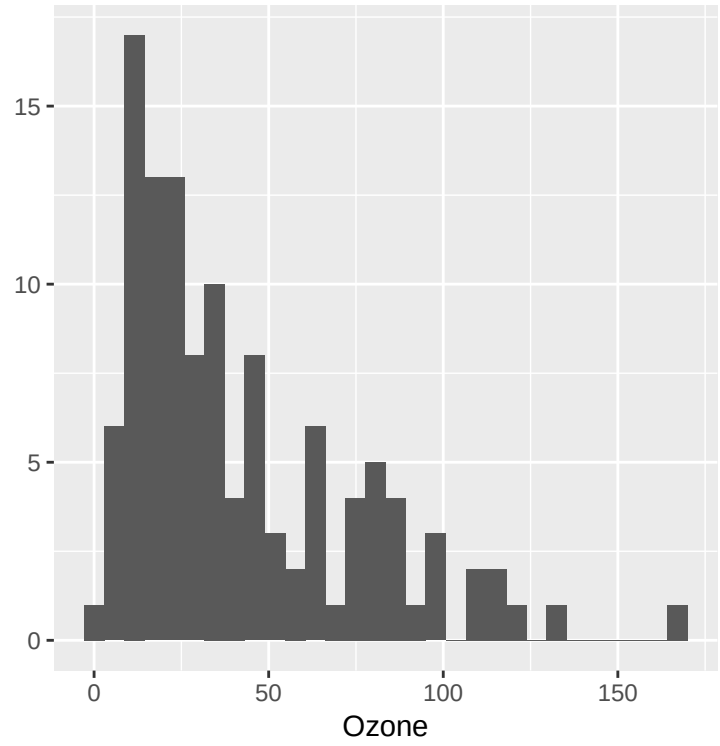


Es gibt eine große Anzahl an Elementen, aus denen man die unterschiedlichsten Grafiken zusammenbauen kann. Die offizielle ggplot2-Website hat eine vollständige Liste aller Geoms, Aesthetics und anderer Befehle. Im Folgenden sehen wir uns noch an, wie man Histogramme, Boxplots und Fehlerbalkenplots erzeugen kann.

Ein Histogramm erstellt man mit `geom_histogram`:

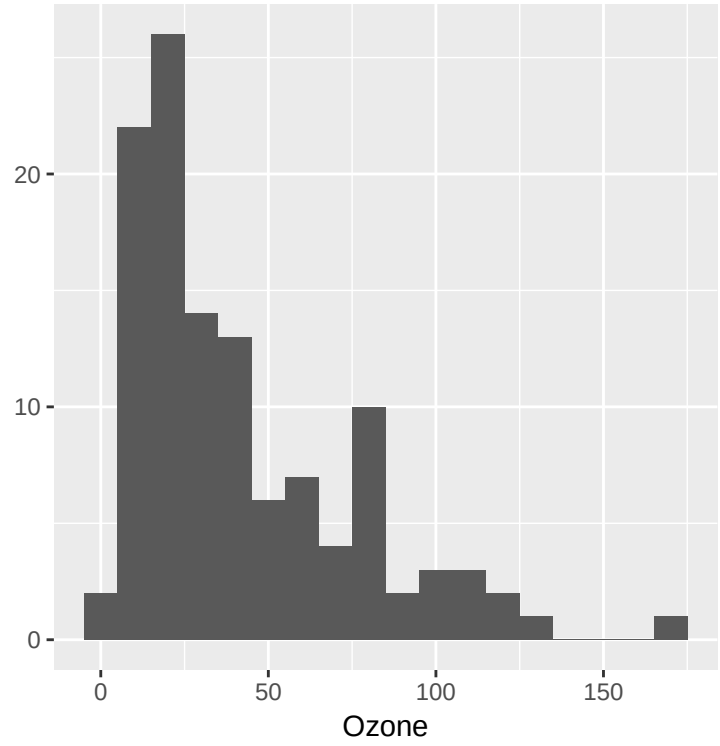
```
ggplot(airquality, aes(Ozone)) +
    geom_histogram() + ylab("")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



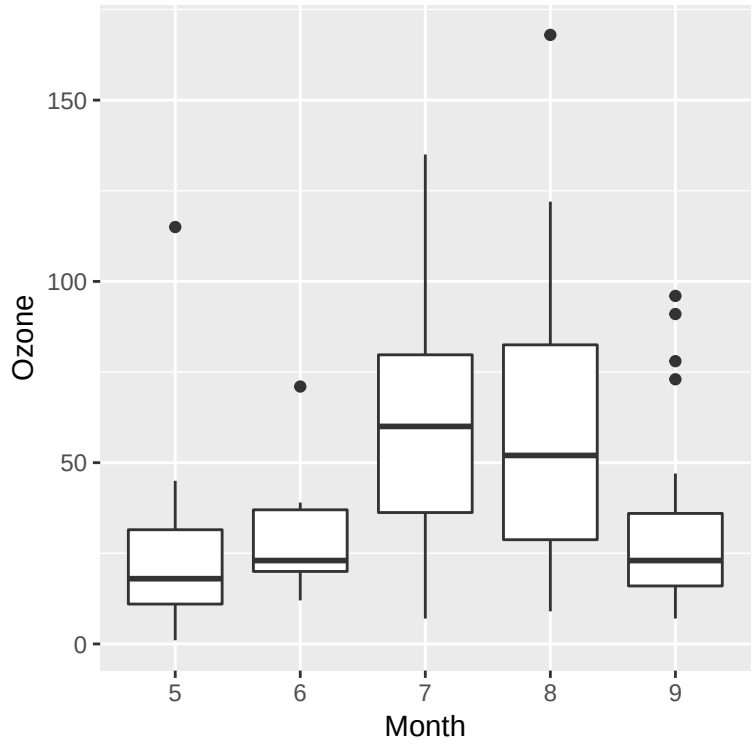
Auch hier kann man die Bingröße manuell angeben:

```
ggplot(airquality, aes(Ozone)) +
    geom_histogram(binwidth=10) + ylab("")
```



Boxplots können mit `geom_boxplot` erzeugt werden. Um Boxplots gleichzeitig für alle Stufen eines Faktors darzustellen, muss man dies wie folgt spezifizieren:

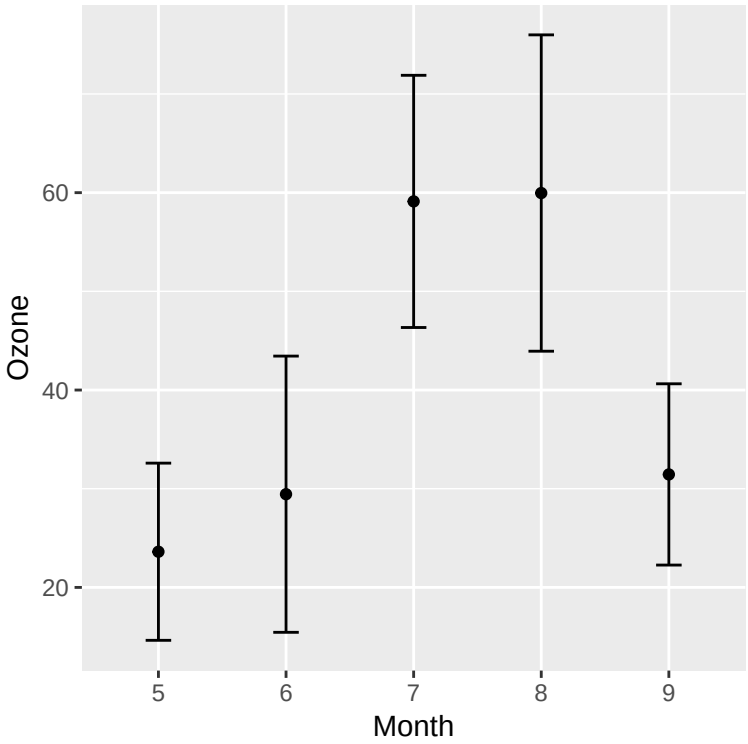
```
ggplot(airquality, aes(factor(Month), Ozone)) +
    geom_boxplot() + xlab("Month")
```



Um einen Fehlerbalkenplot zu erstellen (der die Mittelwerte und deren 95%-Konfidenzintervalle zeigt), kann man die Funktion `stat_summary` benutzen. Dafür muss das Paket `Hmisc` installiert sein.

```
library(Hmisc)
```

```
ggplot(airquality, aes(factor(Month), Ozone)) +
    stat_summary(fun.y=mean, geom="point") +
    stat_summary(fun.data=mean_cl_normal, geom="errorbar", width=0.2) +
    xlab("Month")
```



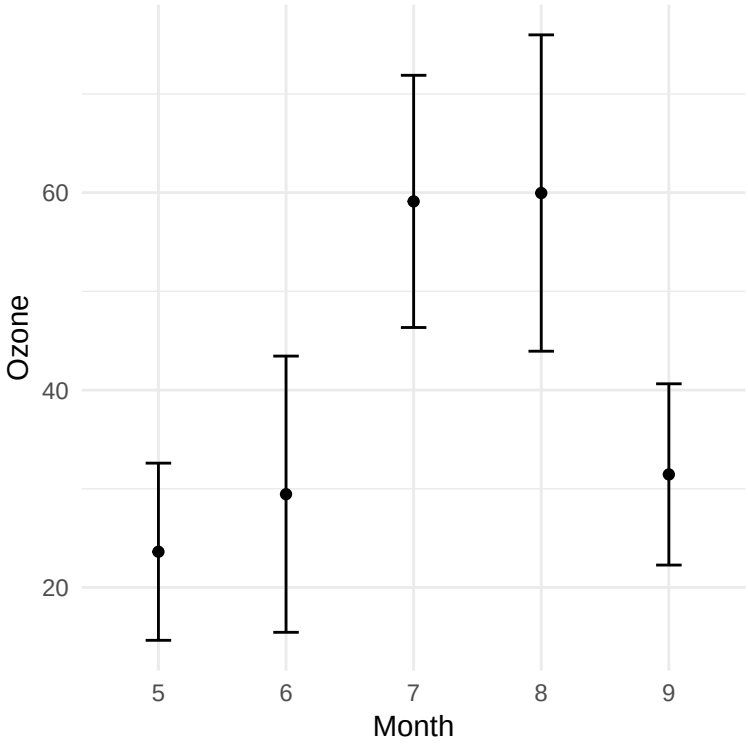
Themes

Wenn man mit dem generellen Stil der Grafiken nicht zufrieden ist, kann man ein anderes vordefiniertes Theme verwenden. Folgende Themes stehen zur Auswahl:

- `theme_gray`
- `theme_bw`
- `theme_linedraw`
- `theme_light`
- `theme_dark`
- `theme_minimal`
- `theme_classic`
- `theme_void`

Man aktiviert ein Theme indem man es als eigene Ebene zu einer vorhandenen Grafik hinzufügt:

```
ggplot(airquality, aes(factor(Month), Ozone)) +
    stat_summary(fun.y=mean, geom="point") +
    stat_summary(fun.data=mean_cl_normal, geom="errorbar", width=0.2) +
    xlab("Month") +
    theme_minimal()
```



Alternativ kann man mit der Funktion `theme_set` ein Theme auch global für alle Grafiken setzen. Alle Grafiken, die danach erzeugt werden, verwenden diesen Stil. Das folgende Beispiel setzt `theme_minimal` als Standard-Theme:

```
theme_set(theme_minimal())
```

Übungen

Übung 1

Laden Sie den in R integrierten Datensatz `iris` und erstellen Sie einen Scatterplot der Spalten `Sepal.Length` auf der x-Achse und `Petal.Length` auf der y-Achse. Stellen Sie die Punkte der drei Spezies in unterschiedlichen Farben dar. Beschriften Sie die Achsen mit aussagekräftigen Bezeichnungen.

Übung 2

Verwenden Sie wieder den Datensatz `iris` und erstellen Sie eine Grafik, die einen Boxplot der Sepal Length für jede Species (also insgesamt drei Boxplots) zeigt.

Übung 3

Laden Sie den in R integrierten Datensatz `Orange` und stellen Sie die Abhängigkeit des Stammumfanges `circumference` (y-Achse) vom Alter `age` (x-Achse) grafisch dar. Stellen Sie die Daten als Punkte dar und verbinden Sie außerdem die Daten eines jeden Baumes (d.h. Sie sollten dann 5 Linien gleichzeitig darstellen, am besten in unterschiedlichen Farben indem Sie das Mapping `color=Tree` verwenden).

Übung 4

Verwenden Sie den Datensatz `mpg`, welcher automatisch mit `ggplot2` geladen wird. Wie hängt der Hubraum `displ` mit dem Kraftstoffverbrauch `hwy` zusammen? Beantworten Sie diese Frage mit einem Scatterplot und überlagerter Regressionsgerade.

Übung 5

Verwenden Sie den Datensatz `mpg` und erstellen Sie einen Boxplot, in dem Sie den Kraftstoffverbrauch in Liter pro 100 Kilometer (l/100km) der Zylinderanzahl gegenüberstellen. *Hinweis:* Erstellen Sie eine neue Spalte `l100km` im Data Frame `mpg`, die den Kraftstoffverbrauch in l/100km aus der Spalte `hwy` (in Meilen pro Gallone, MPG) berechnet. Sie erhalten diesen Verbrauch mit folgender Formel:

$$\text{Verbrauch (in l/100km)} = \frac{235}{\text{Verbrauch (in MPG)}}$$

Übung 6

Welches Problem hat die Grafik, die Sie in Übung 4 erstellt haben? Wie können Sie dieses Problem mit `ggplot2` umgehen? *Hinweis:* Sehen Sie sich die Werte der `displ`-Spalte an. In diesem Zusammenhang könnte `geom_jitter` hilfreich sein.