1. Achsen

Achsentitel, Grenzen, Breaks, Labels und Abstaende anpassen

Dr. Paul Schmidt

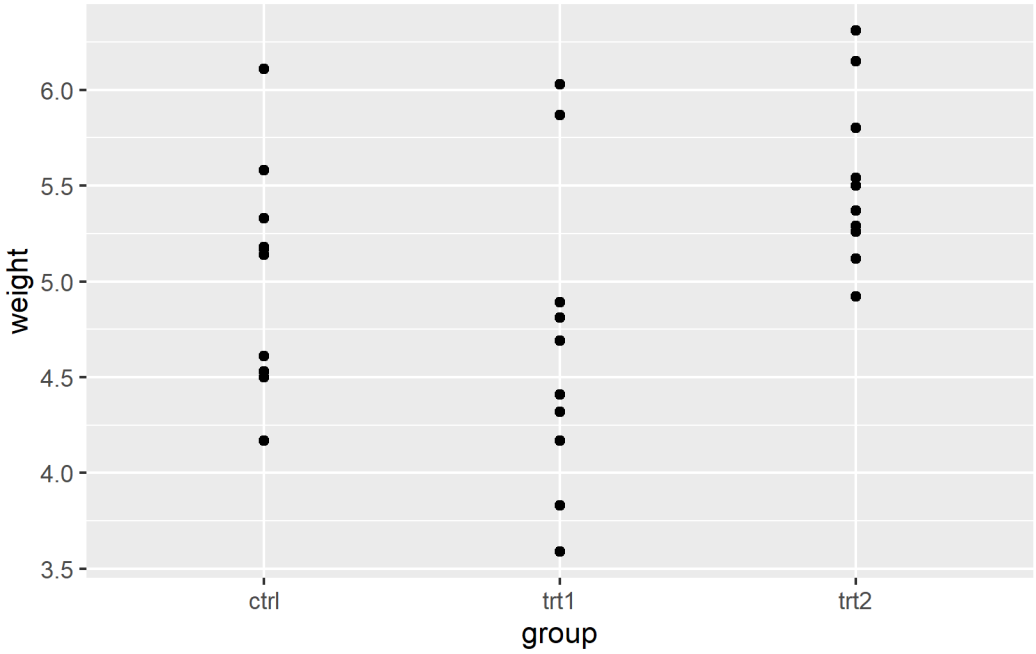
```
for (pkg in c("ggplot2", "scales")) {
  if (!require(pkg, character.only = TRUE)) install.packages(pkg)
}
```

> **Hinweis**
>
> Dieses Kapitel baut auf den ggplot2-Grundlagen aus Erster ggplot auf. Wenn man mit `ggplot()`, `aes()` und einfachen Geoms noch nicht vertraut ist, empfiehlt es sich, dort zu beginnen.

Wenn man einen Plot erstellt, sind die Standard-Achsenbeschriftungen einfach die Spaltennamen aus den Daten, der Achsenbereich wird automatisch bestimmt, und die Tick-Markierungen erscheinen an von ggplot2 gewaehlten Positionen. Fuer einen schnellen Blick reichen diese Defaults, aber sie genuegen selten den Anforderungen einer polierten Grafik - sei es fuer einen Bericht, eine Praesentation oder eine Publikation. Die `scale_x_*`- und `scale_y_*`-Funktionen geben uns die volle Kontrolle ueber diese Details.

In diesem Kapitel verwenden wir den eingebauten `PlantGrowth`-Datensatz und bauen unseren Plot Schritt fuer Schritt auf. Da `weight` eine stetige Variable ist, nutzen wir `scale_y_continuous()`. Da `group` kategorisch ist, nutzen wir `scale_x_discrete()`. Diese Unterscheidung ist wichtig: Stetige und diskrete Skalen teilen sich dieselben Argumente (`name`, `limits`, `breaks`, `labels`, `expand`), verhalten sich aber in wichtigen Punkten unterschiedlich, wie wir unten sehen werden.

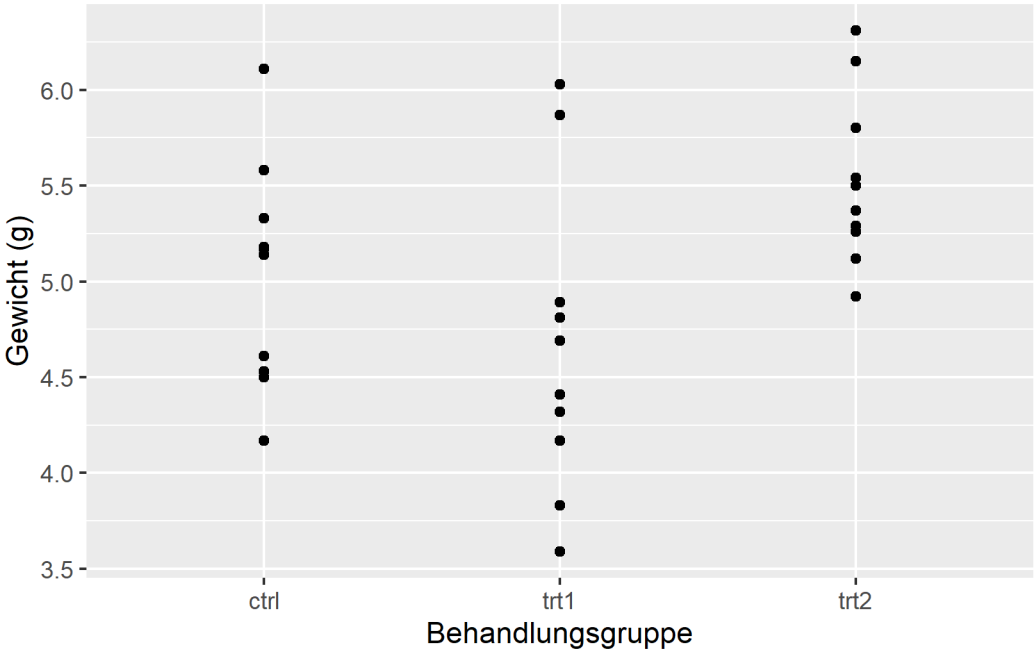
```
myplot <- ggplot(data = PlantGrowth) +
  aes(y = weight, x = group) +
  geom_point()

myplot
```



Name

Die einfachste Anpassung ist der Achsentitel. Standardmaessig verwendet ggplot2 den Spaltennamen aus den Daten - in unserem Fall `weight` und `group`. Fuer explorative Arbeit ist das in Ordnung, aber fuer jede geteilte Ausgabe sollte man aussagekraeftige Titel mit Einheiten angeben. Das Argument `name` erledigt genau das:

```
myplot +
  scale_y_continuous(name = "Gewicht (g)") +
  scale_x_discrete(name = "Behandlungsgruppe")
```

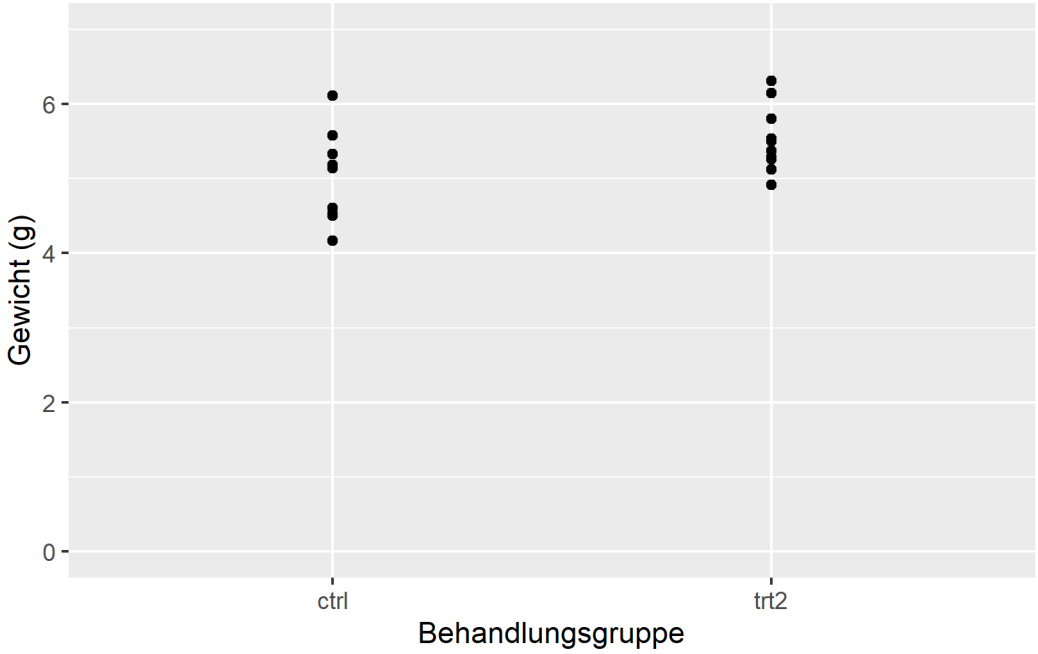


Limits

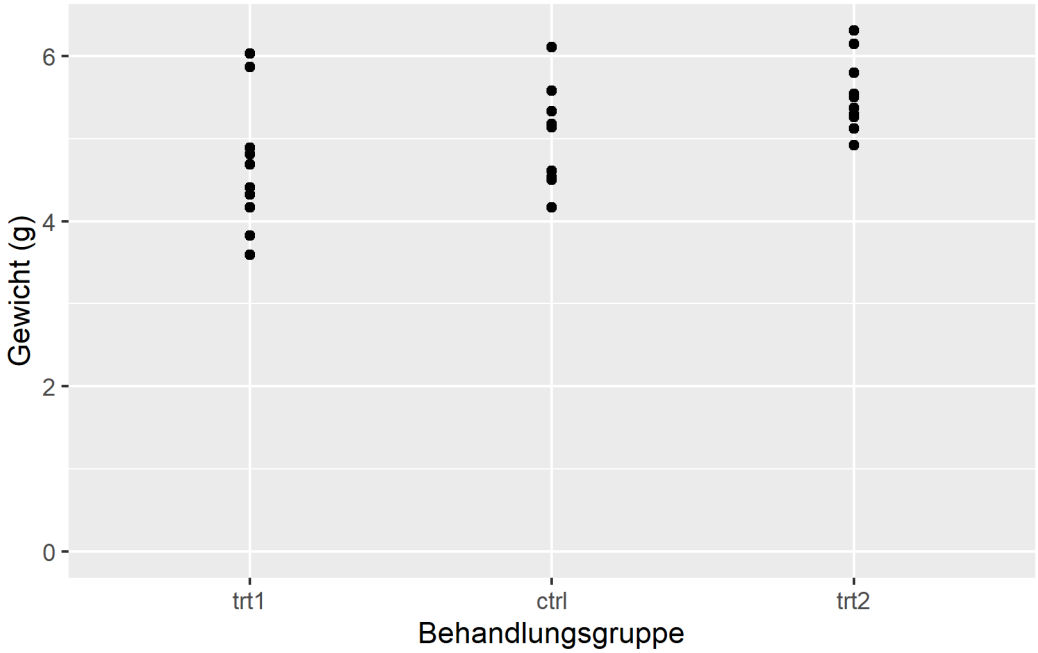
Das Argument `limits` steuert, welche Werte auf der Achse angezeigt werden. Hier verhalten sich stetige und diskrete Skalen deutlich unterschiedlich: Bei einer stetigen Skala definiert `limits` den numerischen Bereich (z.B. 0 bis 7). Bei einer diskreten Skala bestimmt es, welche Faktor-Levels erscheinen und - wichtig - deren Reihenfolge von links nach rechts.

```
myplot +
  scale_y_continuous(
    name = "Gewicht (g)",
    limits = c(0, 7)
  ) +
  scale_x_discrete(
    name = "Behandlungsgruppe",
    limits = c("ctrl", "trt2")
  )
```

```
Warning: Removed 10 rows containing missing values or values outside the scale
range
(`geom_point()`).
```




```
myplot +
  scale_y_continuous(
    name = "Gewicht (g)",
    limits = c(0, NA)
  ) +
  scale_x_discrete(
    name = "Behandlungsgruppe",
    limits = c("trt1", "ctrl", "trt2")
  )
```



Im linken Plot funktioniert `limits = c(0, 7)` auf der y-Achse wie erwartet. Werden jedoch bei der diskreten Skala nur “ctrl” und “trt2” angegeben, verschwindet “trt1” komplett - `limits` bei einer diskreten Skala muss alle anzuzeigenden Levels enthalten.

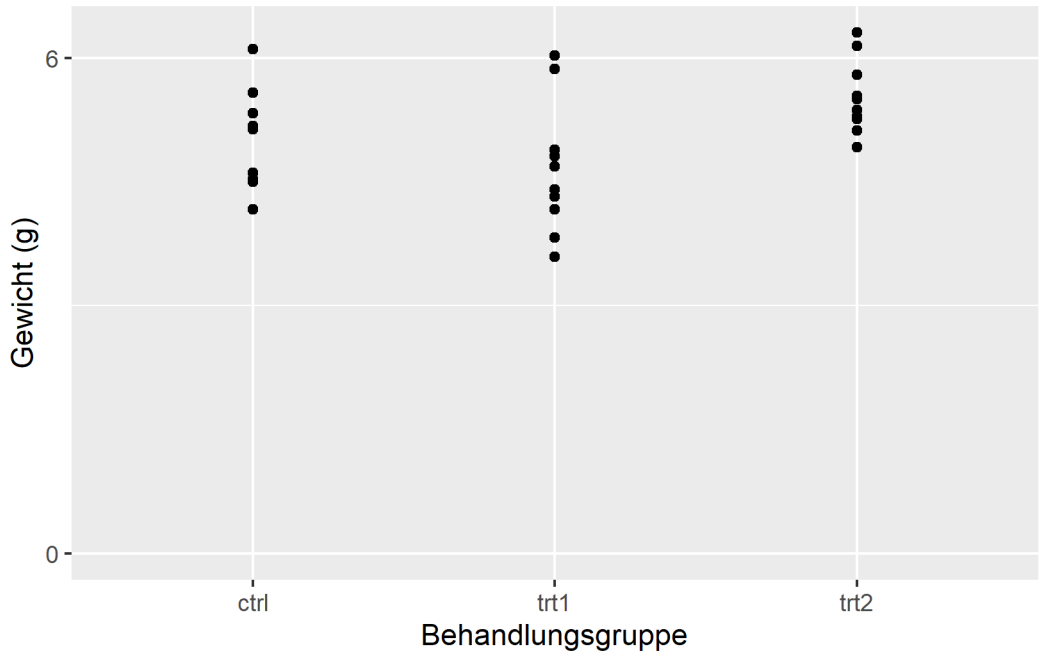
Im rechten Plot wird `NA` fuer die obere y-Grenze verwendet, sodass ggplot2 diese aus den Daten ermittelt. Die Auflistung aller drei Gruppen in einer bestimmten Reihenfolge steuert deren Anordnung auf der x-Achse.

> ! Wichtig
>
> Das Setzen von Limits kann Daten ausserhalb des angegebenen Bereichs ausschliessen. Diese ausgeschlossenen Daten werden weder fuer statistische Berechnungen noch fuer die Darstellung der Geoms beruecksichtigt. Das ist ein haeufiger Stolperstein bei der Kombination von `limits` mit statistischen Layern wie `geom_smooth()` - die Regressionslinie basiert dann nur auf den sichtbaren Daten.

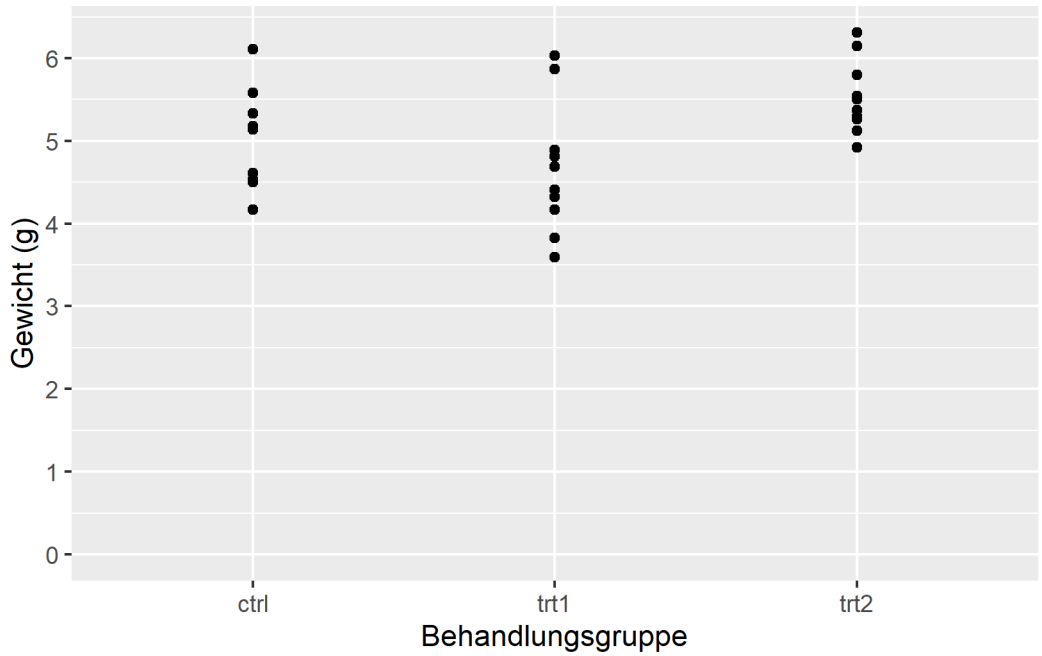
Breaks

Waehrend `limits` den Bereich der Achse definiert, steuert `breaks`, wo innerhalb dieses Bereichs die Tick-Markierungen erscheinen. Gut gewaehlte Break-Positionen machen einen Plot leichter lesbar - zu wenige und man kann keine Werte abschaetzen, zu viele und die Achse wirkt ueberladen.

```
myplot +
  scale_y_continuous(
    name = "Gewicht (g)",
    limits = c(0, NA),
    breaks = c(0, 6)
  ) +
  scale_x_discrete(
    name = "Behandlungsgruppe"
  )
```




```
myplot +
  scale_y_continuous(
    name = "Gewicht (g)",
    limits = c(0, NA),
    breaks = seq(0, 6)
  ) +
  scale_x_discrete(
    name = "Behandlungsgruppe"
  )
```



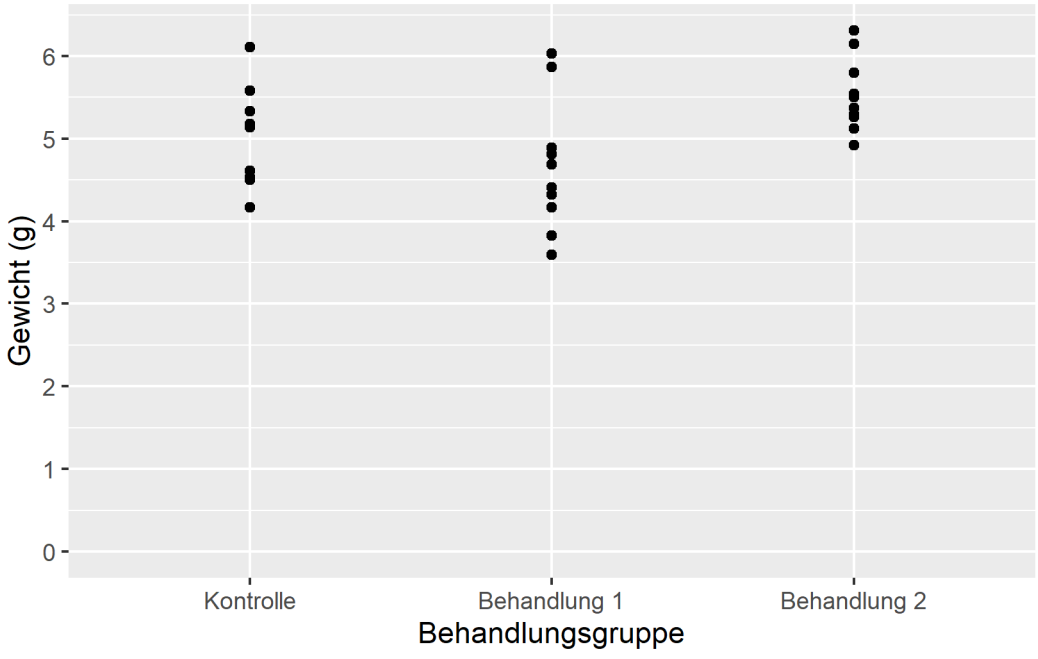
Im linken Plot erscheinen nur zwei Tick-Markierungen bei 0 und 6. Im rechten Plot setzt `seq(0, 6)` Markierungen bei jeder ganzen Zahl von 0 bis 6.

> **Tipp**
>
> Anstatt den Maximalwert manuell anzugeben, gibt es flexiblere Alternativen:
>
> - `breaks = seq(0, max(PlantGrowth$weight))` findet das Datenmaximum automatisch
> - `breaks = scales::breaks_width(1)` definiert einfach den Abstand zwischen den Breaks

Labels

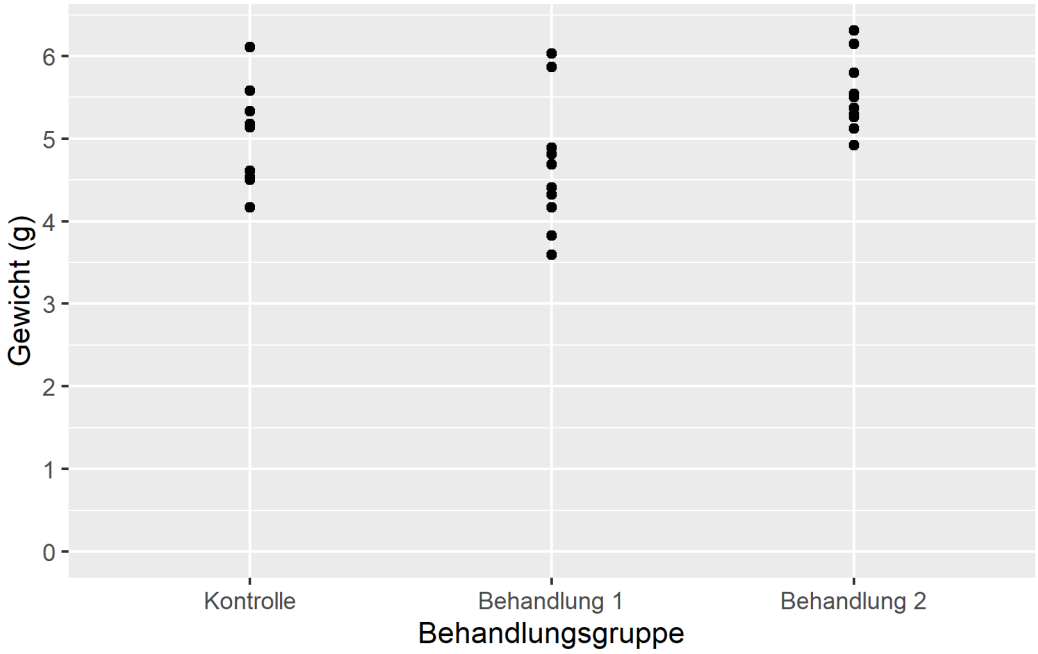
Bisher haben wir gesteuert, *wo* Tick-Markierungen erscheinen. Das Argument `labels` kontrolliert, *welcher Text* an jeder Markierung angezeigt wird. Das ist besonders bei diskreten Achsen nuetzlich, wo die rohen Faktor-Levels (wie `ctrl`, `trt1`, `trt2`) oft Abkuerzungen sind, die fuer einen mit den Daten unvertrauten Leser nichts bedeuten.

```
myplot +
  scale_y_continuous(
    name = "Gewicht (g)",
    limits = c(0, NA),
    breaks = seq(0, 6)
  ) +
  scale_x_discrete(
    name = "Behandlungsgruppe",
    labels = c("Kontrolle", "Behandlung 1", "Behandlung 2")
  )
```




```
myplot +
  scale_y_continuous(
    name = "Gewicht (g)",
    limits = c(0, NA),
    breaks = seq(0, 6)
  ) +
  scale_x_discrete(
    name = "Behandlungsgruppe",
```

```
    labels = c(
      ctrl = "Kontrolle",
      trt1 = "Behandlung 1",
      trt2 = "Behandlung 2"
    )
  )
```



Im ersten Beispiel funktioniert ein einfacher Vektor mit Labels, solange die Levels in der erwarteten Reihenfolge vorliegen. Im zweiten Beispiel stellt ein benannter Vektor sicher, dass jedes Label korrekt seinem Level zugeordnet wird, unabhaengig von der Reihenfolge. Benannte Vektoren sind generell die sicherere Option.

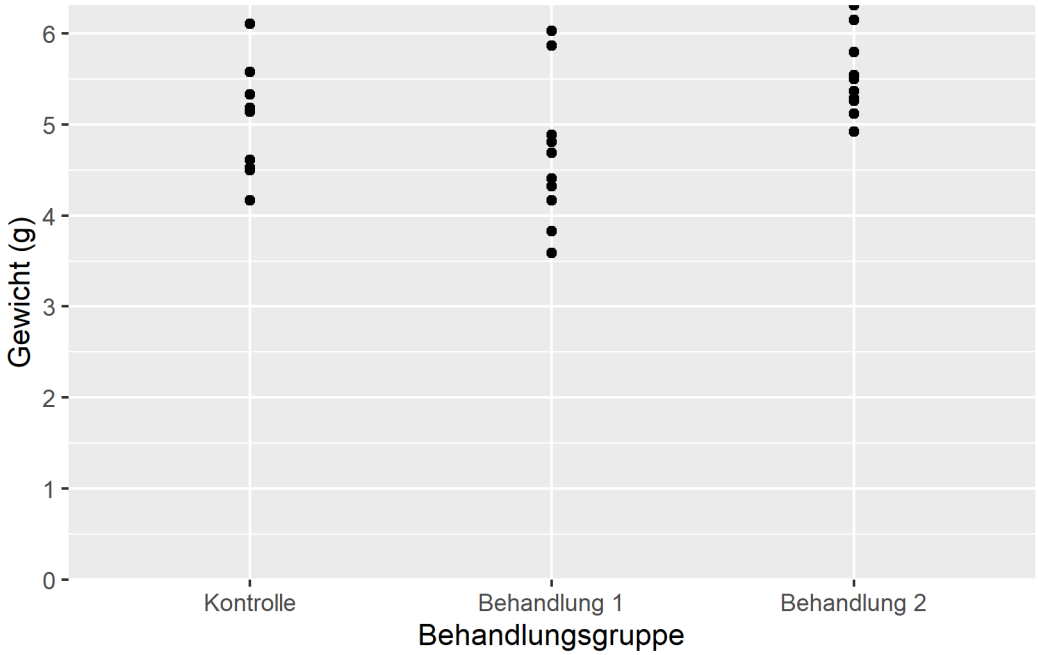
> **Tipp**
>
> Fuer stetige Achsen bietet das {scales}-Paket praktische `label_*()` Funktionen:
>
> - `labels = label_number()` - Zahlen mit benutzerdefinierten Dezimaltrennern, Suffixen etc. formatieren
> - `labels = label_percent()` - 0.05 als 5%, 0.4 als 40% anzeigen
> - `labels = label_log()` - Logarithmische Labels wie 10^1, 10^2, 10^3 anzeigen

Expand

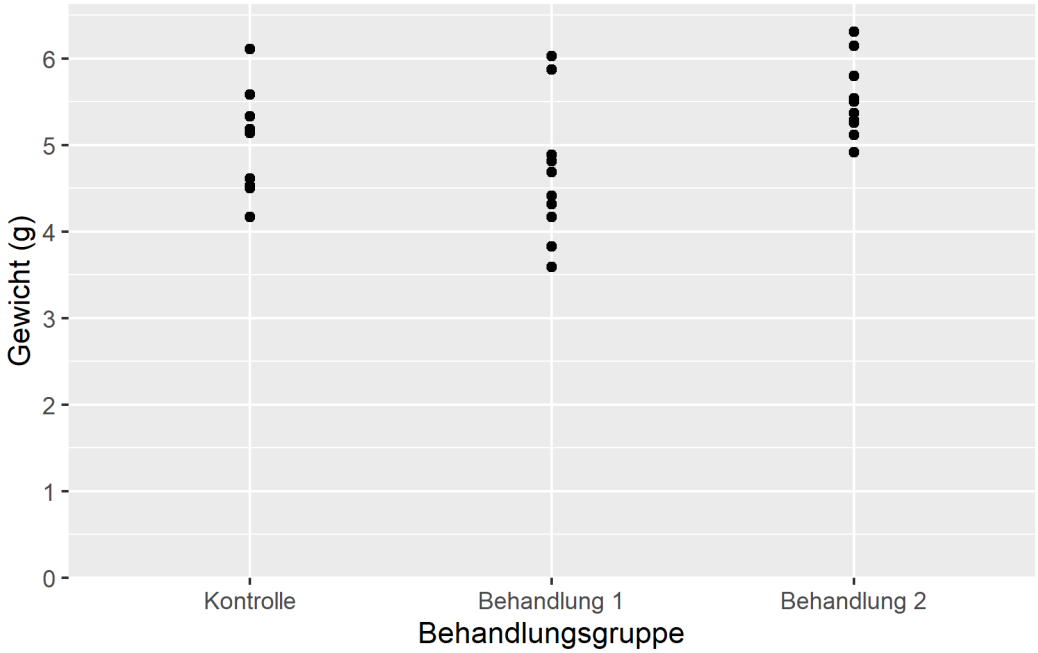
Standardmaessig fuegt ggplot2 einen kleinen Puffer leeren Raums zwischen den Daten und den Achsengrenzen hinzu. Das verhindert, dass Datenpunkte direkt auf den Achsenlinien liegen. In den meisten Faellen ist dieser Standard in Ordnung, aber es gibt haeufige Situationen, in denen man ihn anpassen moechte - beispielsweise wenn ein Balkendiagramm genau bei Null beginnen soll, ohne Luecke darunter. Das Argument `expand` und die Hilfsfunktion `expansion()` bieten diese Kontrolle.

```
myplot +
  scale_y_continuous(
```

```
    name = "Gewicht (g)",
    limits = c(0, NA),
    breaks = seq(0, 6),
    expand = c(0, 0)
  ) +
  scale_x_discrete(
    name = "Behandlungsgruppe",
    labels = c(
      ctrl = "Kontrolle",
      trt1 = "Behandlung 1",
      trt2 = "Behandlung 2"
    )
  )
```




```
myplot +
  scale_y_continuous(
    name = "Gewicht (g)",
    limits = c(0, NA),
    breaks = seq(0, 6),
    expand = expansion(mult = c(0, 0.05))
  ) +
  scale_x_discrete(
    name = "Behandlungsgruppe",
    labels = c(
      ctrl = "Kontrolle",
      trt1 = "Behandlung 1",
      trt2 = "Behandlung 2"
    )
  )
```

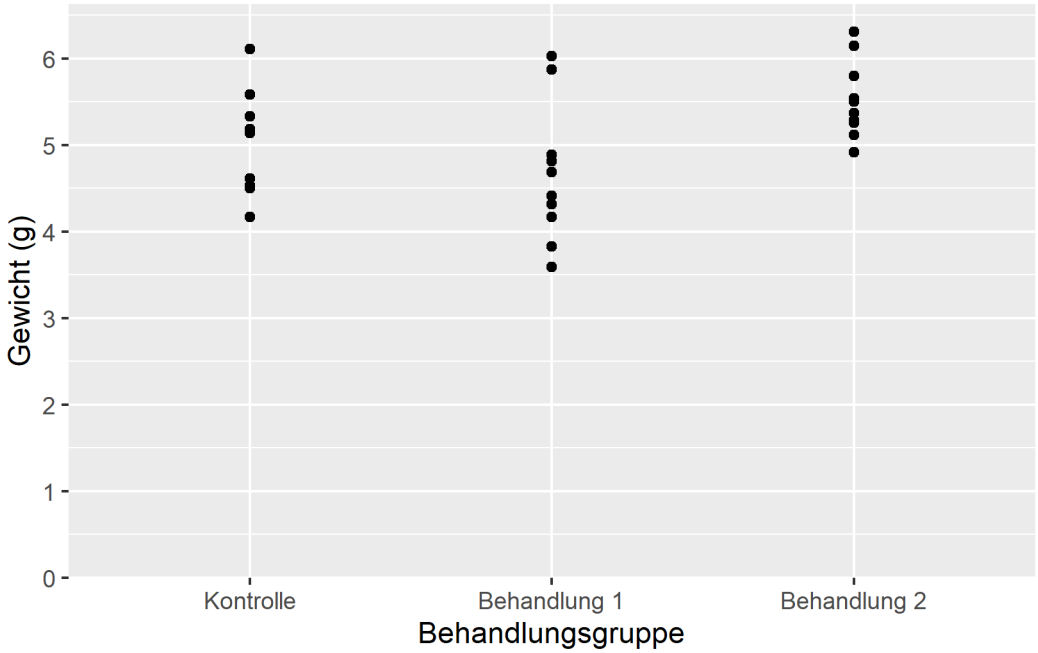


Im linken Plot entfernt `expand = c(0, 0)` den gesamten zusaetzlichen Raum, aber der Plot wird direkt an der hoechsten Beobachtung abgeschnitten. Im rechten Plot entfernt `expansion(mult = c(0, 0.05))` den Raum unterhalb von 0, behaelt aber 5% Puffer oberhalb der Daten - eine deutlich praktischere Einstellung fuer Plots, bei denen die y-Achse bei Null beginnen soll.

Zusammenfassung

Nachdem wir alle fuenf Schluesselargumente - `name`, `limits`, `breaks`, `labels` und `expand` - behandelt haben, setzen wir sie alle in einem polierten Plot zusammen:

```
myplot <- ggplot(data = PlantGrowth) +
  aes(y = weight, x = group) +
  geom_point() +
  scale_y_continuous(
    name = "Gewicht (g)",
    limits = c(0, NA),
    breaks = seq(0, 6),
    expand = expansion(mult = c(0, 0.05))
  ) +
  scale_x_discrete(
    name = "Behandlungsgruppe",
    labels = c(
      ctrl = "Kontrolle",
      trt1 = "Behandlung 1",
      trt2 = "Behandlung 2"
    )
  )

myplot
```



Dieses `myplot`-Objekt dient als Ausgangspunkt fuer die naechsten Kapitel, in denen wir Farben und Themes auf diese Achseneinstellungen aufbauen.

Bibliography