Skript Bio Data Science

Prof. Dr. Jochen Kruppa

Table of contents

Ein herzliches Willkommen

Auf den folgenden Seiten wirst du eine Menge über Statistik oder Data Science lernen. Ich freue mich, dass du Lust hast hier etwas zu lernen… oder aber du *must* da bald eine Klausur ansteht.

> **i** Kleine Anmerkung

Vorwort

Dieses Skript dient...

Kontakt und Literatur

Was ist gute Literatur? Immer schwer zu beurteilen. Im folgenden liste ich einige Literaturquellen auf. Zum einen basiert eine Menge von dem R Code auf Wickham (2016) zum Anderen möchtest du dich vielleicht nochmal rechts oder links weiter bilden. Du musst aber nicht um die Klausur bestehen zu können. Siehe es eher als ein Angebot.

> **i** Die Frage nach der Klausur…
>
> Und daher hier nochmal gleich zu Anfang, es ist nicht notwendig mehr als das Skript durchzuarbeiten und bei den Übungen zu sein um die Klausur zu bestehen. Für deine Bachelorarbeit wirst du aber Programmieren in R können müssen.

Kontakt

Wie erreichst du mich? Am einfachsten über die gute, alte E-Mail.

Einfach an j.kruppa@hs-osnabrueck.de schreiben. Du findest hier auch eine kurze Formulierungshilfe. Einfach auf den Ausklapppfeil klicken.

E-Mailvorlage mit beispielhafter Anrede

Hallo Herr Kruppa,
ich belege gerade Ihr Modul `Modulname` und hätte eine Bitte/Frage/Anregung… benötige Hilfe bei der Planung/Auswertung meiner Bachelorarbeit…
Mit freundlichen Grüßen
M. Muster

Auf YouTube

Wenn du möchtest kannst du auf YouTube unter https://www.youtube.com/c/JochenKruppa noch einige Lehrvideos als Ergänzung schauen. In den Videos wiederhole ich Inhalte und du kannst auf Pause drücken um nochmal Programmierschritte nachverfolgen zu können.

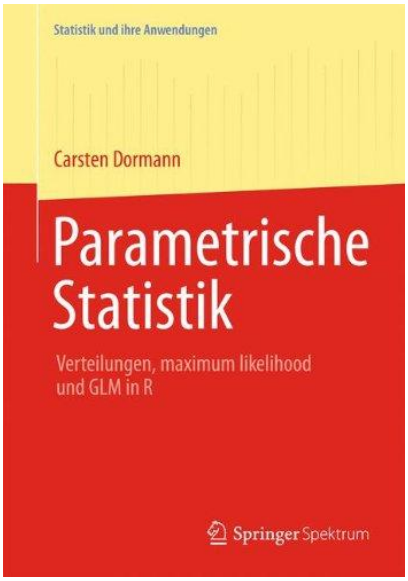
Auf GitHub

Alle Materialien von mir findest du immer auf GitHub unter https://github.com/jkruppa/teaching. Selebst wenn du nicht mehr in einem meiner Kurse bist, kannst du so auf die Lehrinhalte immer nochmal zugreifen und die aktuellen Versionen haben.

Literatur

Neben diesem Modul musst du vermutlich noch andere Module belegen. Deshalb hier eine Auswahl Literatur, die dir helfen mag. Zum einen ist die Literatur anders geschrieben und zum anderen sind dort andere Imhalte.

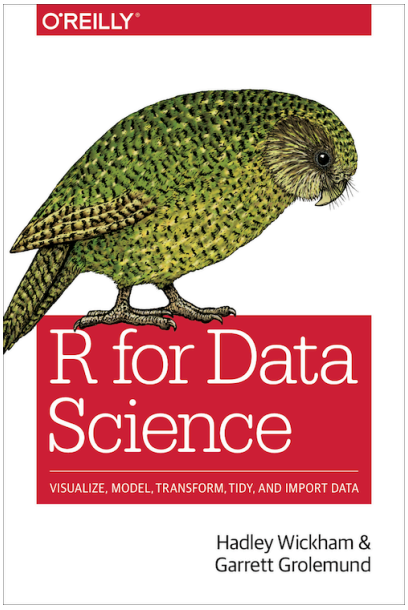
Parametrische Statistik

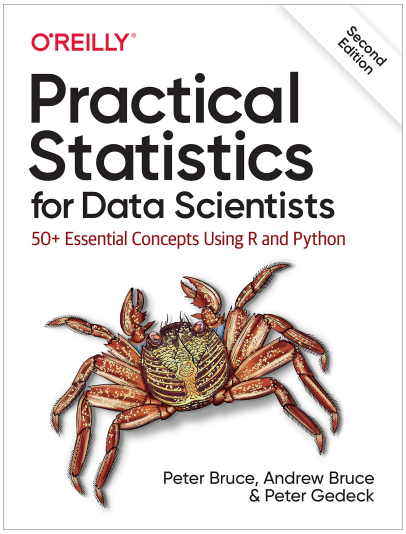


Dormann (2013) liefert ein tolles deutsches Buch für die Vertiefung in die Statistik. Insbesondere wenn du wissenschaftlich Arbeiten willst weit über die Bachelorarbeit hinaus. Dormann baut in seinem Buch eine hervorragende Grundlage auf. Das Buch ist an der Hochschule Osnabrück kostenlos über den Link
https://link.springer.com/book/10.1007/978-3-662-54684-0 zu erhalten.

R for Data Science

Wickham (2016) ist die Grundlage für die R Programmierung. Das Material von Wickahm findet sich kostenlos online unter https://r4ds.had.co.nz/ und https://www.tidyverse.org/. Wir werden uns hauptsächlich mit R wie es Wickham lehrt beschäftigen. Somit ist Wickham unsere Grundlage für R.


O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham &
Garrett Grolemund


O'REILLY
Second Edition
Practical Statistics
for Data Scientists
50+ Essential Concepts Using R and Python
Peter Bruce, Andrew Bruce
& Peter Gedeck

Practical Statistics for Data Scientists

Bruce (2020) schreibt ein Buch für den Anwender. Ohne Vorkenntnisse ist das Buch vermutlich etwas schwer zu lesen. Dafür bietet das Buch aber *nach* einem Statistikkurs sehr gute Anknüpfungspunkte Richtung maschinelles Lernen und somit der Klassifikation.

Data Science for Agriculture in R

Schmidt liefert auf der Webseite https://schmidtpaul.github.io/DSFAIR/index.html eine tolle Sammlung an experimentellen Designs bzw. Versuchsanlagen samt der Auswertung in R. Ohne Vorkenntnisse schwer zu verstehen. Sollte aber nach einem Kurs Statistik dann möglich sein. Gerne hier auch mich fragen, dann können wir gemeinsam das passende Design raussuchen und besprechen.

Odds & Ends

Am Ende dann noch eine Mathebuch von Weisberg zu finden unter https://jonathanweisberg.org/vip/. Eigentlich eher ein Buch über Wahrscheinlichkeiten und wenn ein Buch am Ende stehen muss, dann ist es dieses Buch. Ich finde es sehr spannend zu lesen, aber das ist dann vermutlich *special intrest*.

Literatur

Einführende Datenbeispiele

Wir brauchen am Anfang erstmal ein Beispiel. Konkrete Zahlen mit denen wir arbeiten können und Grundlagen aufbauen können. Was liegt da näher als sich einmal am Kopf zu kratzen und zu fragen, was juckt den da? Genau! Flöhe. Wir schauen uns einmal Flöhe auf Hunden und Katzen an. Daran können wir viel über Zahlen und Buchstaben in der Statistik und dann im Programmieren lernen.

> **Zahlen, Buchstaben und Wörter**
>
> Mir ist bewusst, dass du die Unterschiede kennst. Nur leider ist eine Zahl nicht nur eine Zahl und ein Wort nicht immer ein Wort. Das hat mit der eingeschränkten Kommunikationsfähigkeit von Computerprogrammen zu tun. R braucht da deine Mithilfe und dein *neues* Verständnis von Buchstaben und Zahlen. Eben wie ein Computer denkt.

Beispiel 1: Von Flöhen und Hunden

In unserem ersten Beispiel wollen wir uns verschiedene Daten D von Hunden und Hundeflöhen anschauen. Unter anderem sind dies die Sprungweite, die Anzahl an Flöhen, die Boniturnoten auf einer Hundemesse sowie der Infektionsstatus. Hier nochmal detailiert, was wir uns im folgenden im Kapitel einmal anschauen wollen.

- **Sprungweite** in [cm] von verschiedenen Flöhen

$$Y_{jump} = \{5.2, 4.9, 12.1, 8.2, 5.6, 9.1, 7.4\}.$$

- **Anzahl an Flöhen** auf verschiedenen Hunden

$$Y_{count} = \{18, 22, 17, 12, 23, 18, 21\}.$$

- **Boniturnoten** [1 = schlechteste bis 9 = beste Note] von verschiedenen Hunden

$$Y_{grade} = \{8, 7, 5, 6, 7, 7, 9\}.$$

- **Infektionstatus** [0 = gesund, 1 = infiziert] mit Flöhen von verschiedenen Hunden

$$Y_{infected} = \{0, 1, 1, 0, 1, 0, 0\}.$$

Je nachdem was wir messen, nimmt Y andere Zahlenräume an. Wir sagen, Y folgt einer Verteilung. Die Sprungweite ist normalverteilt, die Anzahl an Flöhen folgt einer Poisson Verteilung, die Boniturnoten sind multinominal/ordinal bzw. kategorial verteilt. Der Infektionsstatus ist binomial verteilt. Wir werden uns später die Verteilungen anschauen und visualisieren. Das können wir hier aber noch nicht. Wichtig ist, dass du schon mal gehört hast, dass Y unterschiedlich *verteilt* ist, je nachdem welche Dinge wir messen.

Beispiel 2: Von Flöhen, Hunden und Katzen

Wir wollen jetzt das Beispiel von den Hunden und Flöhen um eine Spezies erweitern. Wir nehmen noch die Katzen mit dazu und fragen uns, wie sieht es mit der Sprungfähigkeit von Katzen und Hundeflöhen aus? Konzentrieren wir uns hier einmal auf die Sprungweite. Wir können wie in dem Beispiel die Sprungweiten [cm] wieder aufschreiben:

$$Y_{jump} = \{3.2, 1.2, 6.6, 4.1, 4.3, 7.8, 6.2\}.$$

Wenn wir jetzt die Sprungweiten der Hundeflöhe mit den Katzenflöhen vergleichen wollen haben wir ein Problem. Beide Zahlenvektoren heißen gleich, nämlich Y_{jump}. Wir könnten jeweils in die Indizes noch *dog* und *cat* schreiben als $Y_{jump,\,dog}$ und $Y_{jump,\,cat}$ und erhalten folgende Vektoren.

$$Y_{jump,\,dog} = \{5.2, 4.9, 12.1, 8.2, 5.6, 9.1, 7.4\}$$

$$Y_{jump,\,cat} = \{3.2, 1.2, 6.6, 4.1, 4.3, 7.8, 6.2\}$$

Dadurch werden die Indizes immer länger und unübersichticher. Auch das Y einfach Y_{dog} oder Y_{cat} zu nennen ist keine Lösung - wir wollen uns vielleicht später nicht nur die Sprungweite vergleichen, sondern vielleicht auch die Anzahl an Flöhen oder den Infektionsstatus. Dann ständen wir wieder vor dem Problem die Y für die verschiedenen Outcomes zu unterscheiden. Daher erstellen wir uns die Tabelle 1. Wir haben jetzte eine *Daten*tabelle.

Tabelle 1: Sprunglängen [cm] für Hunde- und Katzenflöhe. Die Tabelle ist im Wide-Format dargestellt.

dog	cat
5.2	3.2
4.9	1.2
12.1	6.6
8.2	4.1

dog	cat
5.6	4.3
9.1	7.8
7.4	6.2

Intuitiv ist die Tabelle 1 übersichtlich und beinhaltet die Informationen die wir wollten. Dennoch haben wir das Probem, das wir in dieser Tabelle 1 nicht noch weitere Outcomes angeben können. Wir können die Anzahl an Flöhen auf den Hunde und Katzen nicht darstellen. Als Lösung ändern wir die Tabelle 1 in das Long-Format. Dargestellt in Tabelle 2. Jede Beobachtung belegt nun eine Zeile. Dies ist sehr wichtig im Kopf zu behalten, wenn du eigene Daten in z.B. Excel eingibts.

Tabelle 2: Sprunglängen [cm] für Hunde- und Katzenflöhe. Die Tabelle ist im Long-Format dargestellt.

animal	jump_length	flea_count	grade	infected
dog	5.2	18	8	0
dog	4.9	22	7	1
dog	12.1	17	5	1
dog	8.2	12	6	0
dog	5.6	23	7	1
dog	9.1	18	7	0
dog	7.4	21	9	0
cat	3.2	12	9	1
cat	1.2	13	5	0
cat	6.6	11	7	0
cat	4.1	12	8	0
cat	4.3	16	6	1
cat	7.8	9	6	0
cat	6.2	7	8	0

! Was war der Sinn der Reise?

Wir nutzen nur das Long-Format für die Erstellung einer Datentabelle! Nur eine Long-Format Tabelle können wir in R später weiterverarbeiten.

Nun haben wir Tabelle 2 mit Daten zu verschiedenen Oucomes, wie Sprungweite [cm], Anzahl an Flöhen auf Hunden und Katzen, die Boniturnoten oder aber den Infektionsstatus. Die Tabelle 2 ist zwar nicht groß aber auch nicht wirklich klein. Im nächsten Kapitel wollen wir uns damit beschäftigen, die Zahlen in der Tabelle sinnvoll zusammenzufassen.

Daten in R

Im vorherigen Kapitel haben wir die Datentabelle Tabelle 2 erschaffen. Bevor wir uns weiter mit statistischen Kennzahlen beschäftigen, wollen wir uns einmal die Realisierung der Tabelle Tabelle 2 in R anschauen. Dabei wollen wir auch Eigenschaften von Zahlen und Buchstaben lernen, die notwendig sind um mit einem Programm wie R kommunizieren zu können. Wir wollen später R nutzen um die explorative Datenanalyse anzuwenden. Über die explorative Datenanalyse lernen wir in späteren Kapiteln mehr.

> Einführung in R per Video
>
> Du findest auf YouTube Grundlagen in R als Video Reihe. Ich werde zwar alles nochmal hier als Text aufschreiben, aber manchmal ist das Sehen und Hören dann einfacher.

Genutzte R Pakete für das Kapitel

Wir wollen folgende R Pakete in diesem Kapitel nutzen.

```
pacman::p_load(tidyverse, magrittr)
```

Am Ende des Kapitels findest du nochmal den gesamten R Code in einem Rutsch zum selber durchführen oder aber kopieren.

Von Buchstaben und Zahlen

Nun haben wir Tabelle Tabelle 2 mit Daten zu verschiedenen Oucomes, wie Sprungweite [cm], Anzahl an Flöhen auf Hunden und Katzen, die Boniturnoten oder aber den Infektionsstatus. Die Tabelle Tabelle 2 ist zwar nicht groß aber auch nicht wirklich klein. Wir wollen uns nun damit beschäftigen, die Zahlen sinnvoll in R darzustellen.

Tabelle 1: Sprunglängen [cm] für Hunde- und Katzenflöhe. Die Tabelle ist im *Long*-Format dargestellt.

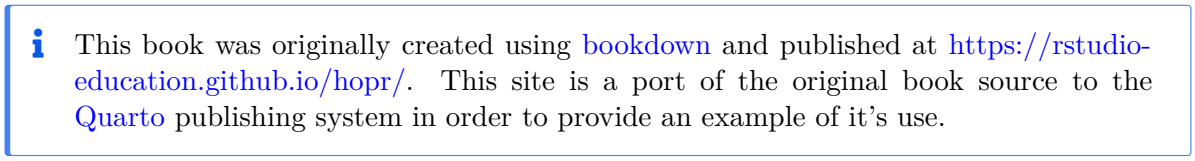
animal	jump_length	flea_count	grade	infected
dog	5.2	18	8	0
dog	4.9	22	7	1
dog	12.1	17	5	1
dog	8.2	12	6	0
dog	5.6	23	7	1
dog	9.1	18	7	0
dog	7.4	21	9	0
cat	3.2	12	9	1
cat	1.2	13	5	0
cat	6.6	11	7	0
cat	4.1	12	8	0
cat	4.3	16	6	1
cat	7.8	9	6	0
cat	6.2	7	8	0

Im folgenden sehen wir die Datentabelle Tabelle 2 in R als `tibble` dargestellt. Was ist nun ein `tibble`? Ein `tibble` ist zu aller erst ein Speicher für Daten in R. Das heist wir haben Spalten und Zeilen. Jede Spalte repräsentiert eine Messung oder Variable und die Zeilen jeweils eine Beobachtung.

```
# A tibble: 14 x 5
   animal jump_length flea_count grade infected
   <chr>        <dbl>      <int> <dbl> <lgl>
 1 dog            5.2         18     8 FALSE
 2 dog            4.9         22     7 TRUE
 3 dog           12.1         17     5 TRUE
 4 dog            8.2         12     6 FALSE
 5 dog            5.6         23     7 TRUE
 6 dog            9.1         18     7 FALSE
 7 dog            7.4         21     9 FALSE
 8 cat            3.2         12     9 TRUE
 9 cat            1.2         13     5 FALSE
10 cat            6.6         11     7 FALSE
11 cat            4.1         12     8 FALSE
12 cat            4.3         16     6 TRUE
13 cat            7.8          9     6 FALSE
14 cat            6.2          7     8 FALSE
```

Als erstes erfahren wir, dass wir einen `A tibble: 14 x 5` vorliegen haben. Das heist, wir haben 14 Zeile und 5 Spalten. In einem `tibble` wird immer in der ersten Zeile angezeigt wieviele Beobachtungen wir in dem Datensatz haben. Wenn das `tibble` zu groß wird, werden wir nicht mehr das ganze `tibble` sehen sondern nur noch einen Ausschnitt. Im Weiteren hat jede Spalte noch eine Eigenschaft unter dem Spaltennamen

- `<chr>` bedeutet `character`. Wir haben also hier Worte vorliegen.
- `<dbl>` bedeutet `double`. Ein `double` ist eine Zahl mit Kommastellen.
- `<int>` bedeutet `integer`. Ein `integer` ist eine ganze Zahl ohne Kommastellen.
- `<lgl>` bedeutet `logical` oder `boolean`. Hier gibt es nur die Ausprägung *wahr* oder *falsch*. Somit `TRUE` oder `FALSE`. Statt den Worten `TRUE` oder `FALSE` kann hier auch 0 oder 1 stehen.
- `<str>` bedeutet `string` der aus verschiedenen `character` besteht kann, getrennt durch Leerzeichen.



This book was originally created using bookdown and published at https://rstudio-education.github.io/hopr/. This site is a port of the original book source to the Quarto publishing system in order to provide an example of it's use.

Buchstaben zu Zahlen - Faktoren

Ein Computer und somit auch eine Programmsprache wie R kann keine Buchstaben **verrechnen**. Ein Programm kann nur mit Zahlen rechnen. Wir haben aber in der Datentabelle Tabelle 2 in der Spalte `animal` Buchstaben stehen. Da wir hier einen Kompromiss eingehen müssen führen wir Faktoren ein. Ein Faktor kombiniert Buchstaben mit Zahlen. Wir als Anwender sehen die Buchstaben, die Wörter bilden. Intern steht aber jedes Wort für eine Zahl, so dass R mit den Zahlen rechnen kann. Klingt ein wenig kryptisch, aber wir schauen uns einen `factor` einmal an.

```
as_factor(data_tbl$animal[1:8])
```

```
[1] dog dog dog dog dog dog dog cat
Levels: dog cat
```

Mit dem `$` Zeichen können wir uns eine einzelne Zeile aus dem Datensatz `data_tbl` rausziehen. Du kannst dir das `$` wie einen Kleiderbügel und das `data_tbl` als einen Schrank für Kleiderbügel verstellen. An dem Kleiderbügel hängen dann die einzelnen Zahlen und Worte. Im Weiteren nehmen wir nicht den ganzen Vektor `animal` mit vierzehn Einträgen sondern nur die ersten acht. Das machen wir mit `[1-8]` hinter dem `animal`. Schauen wir auf das Ergebnis,

so erhalten wir sieben Mal `dog` und einmal `cat`. Insgesamt die ersten acht Einträge der Datentabelle. Darüber hinaus sehen wir auch, dass die der Faktor jetzt `Levels` hat. Exakt zwei Stück. Jeweils einen für `dog` und einen für `cat`.

```
  animal <- c("dog", "dog", "dog", "cat", "cat", "cat")

  as.factor(animal)
```

```
[1] dog dog dog cat cat cat
Levels: cat dog
```

```
  factor(animal, levels = c("dog", "cat"))
```

```
[1] dog dog dog cat cat cat
Levels: dog cat
```

```
  factor(animal, labels = c("katze", "hund"))
```

```
[1] hund  hund  hund  katze katze katze
Levels: katze hund
```

```
  as_factor(animal)
```

```
[1] dog dog dog cat cat cat
Levels: dog cat
```

```
  dose <- c("low", "low", "mid", "mid", "high", "high")

  as.factor(dose)
```

```
[1] low  low  mid  mid  high high
Levels: high low mid
```

```
  factor(dose, levels = c("low", "mid", "high"))
```

```
[1] low  low  mid  mid  high high
Levels: low mid high
```

Der Zuweisungspfeil <-

Mit dem Zuweisungspfeil speichern wir *Dinge* in Objekte in R. Das heist wir speichern damit intern in R Datensätze und viele andere Sachen, die wir dan später wieder verwenden wollen. Schauen wir uns das einmal im Beispiel an. Schrieben wir nur den Vektor `c()` mit Hunden und Katzen darin, so erscheint eine Ausgabe in R.

```
c("dog", "dog", "cat", "cat", "fox", "fox")
```

```
[1] "dog" "dog" "cat" "cat" "fox" "fox"
```

Schreiben wir den gleichen Vektor und nutzen den Zuweisungspfeil, dann wird der Vektor in dem Objekt `animal` gespeichert.

```
animal <- c("dog", "dog", "cat", "cat", "fox", "fox")
```

Wie kommen wir jetzt an die Sachen, die in `animal` drin sind? Wir können einfach `animal` in R schreiben und dann wird uns der Inhalt von `animal` ausgegeben.

```
animal
```

```
[1] "dog" "dog" "cat" "cat" "fox" "fox"
```

Von Wörtern und Objekten

Das mag etwas verwirrend sein, denn es gibt in R Wörter `string <str>` oder `character <chr>`. Wörter sind was anderes als Objekte. Streng genommen sind beides Wörter, aber in Objekten werden Dinge gespeichert wohin gegen das Wort einfach ein Wort ist. Deshalb kennezeichnen wir Wörter auch mit `"wort"` und zeigen damit, dass es sich hier um einen String handelt.

Ein `string <str>` oder `character <chr>`

Wir tippen `"animal"` in R und erhalten `"animal"` zurück.

```
"animal"
```

```
[1] "animal"
```

Ein Objekt

Wir tippen `animal` ohne die Anführungszeichen in R und erhalten den Inhalt von `animal` ausgegeben.

```
animal
```

```
[1] "dog" "dog" "cat" "cat" "fox" "fox"
```

Sollte es das Objekt `animal` nicht geben, also nicht über den Zuweisungspfeil `<-` erschaffen worden, dann wird eine Fehlermeldung von R ausgegeben:

```
Fehler in eval(expr, envir, enclos) : Objekt 'animal' nicht gefunden
```

Die Pipe %>%

Im Weiteren nutzen wir den Pipe Operator dargestellt als `%\>%`. Du kannst dir den Pipe Operator als eine Art Röhre vorstellen in dem die Daten verändert werden und dann an die nächste Funktion weitergeleitet werden. Nehmen wir nochmal das Beispiel von weiter oben. Wir wollen die `character` Spalte aus dem Datensatz `data_tbl` extrahieren und dann in einen Faktor umwandeln.

```
as_factor(data_tbl$animal[1:5])
```

```
[1] dog dog dog dog dog
Levels: dog
```

```
data_tbl %>%
  pull(animal) %>%
  extract(1:5)
```

```
[1] "dog" "dog" "dog" "dog" "dog"
```

Zuerst siehst du das alte Beispiel und dann die Nutzung des Pipe Operators `%>%`. Das Ergebnis ist das gleiche, aber der Code ist einfacher zu lesen. Wir nehmen den Datensatz `data_tbl` leiten den Datensatz in den Funktion `pull()` und ziehen uns damit den Vektor `animal` aus dem Datensatz. Den Vektor leiten wir dann weiter in die Funktion `extract()` und nehmen nur die ersten 5 Werte aus dem Vektor.

Daten bearbeiten

Im folgenden wollen wir den Datensatz `data_tbl` in R bearbeiten. Das heist wir wollen Spalten auswählen mit `select()` oder Zeilen auswählen mit `filter()`. Schlussendlich wollen wir auch die Eigenschaften von Spalten mit der Funktion `mutate` ändern

Spalten wählen mit `select()`

https://dplyr.tidyverse.org/reference/select.html

```
data_tbl %>%
  select(animal, jump_length, flea_count)
```

```
# A tibble: 14 x 3
   animal jump_length flea_count
   <chr>        <dbl>      <int>
 1 dog            5.2         18
 2 dog            4.9         22
 3 dog           12.1         17
 4 dog            8.2         12
 5 dog            5.6         23
 6 dog            9.1         18
 7 dog            7.4         21
 8 cat            3.2         12
 9 cat            1.2         13
10 cat            6.6         11
11 cat            4.1         12
12 cat            4.3         16
13 cat            7.8          9
14 cat            6.2          7
```

Zeilen wählen mit `filter()`

https://dplyr.tidyverse.org/reference/filter.html

```
data_tbl %>%
  filter(animal %in% c("dog"))
```

```
# A tibble: 7 x 5
  animal jump_length flea_count grade infected
  <chr>        <dbl>      <int> <dbl> <lgl>
1 dog            5.2         18     8 FALSE
2 dog            4.9         22     7 TRUE
3 dog           12.1         17     5 TRUE
4 dog            8.2         12     6 FALSE
5 dog            5.6         23     7 TRUE
6 dog            9.1         18     7 FALSE
7 dog            7.4         21     9 FALSE
```

```
data_tbl %>%
  filter(flea_count > 15)
```

```
# A tibble: 7 x 5
  animal jump_length flea_count grade infected
  <chr>        <dbl>      <int> <dbl> <lgl>
1 dog            5.2         18     8 FALSE
2 dog            4.9         22     7 TRUE
3 dog           12.1         17     5 TRUE
4 dog            5.6         23     7 TRUE
5 dog            9.1         18     7 FALSE
6 dog            7.4         21     9 FALSE
7 cat            4.3         16     6 TRUE
```

```
data_tbl %>%
  filter(infected == TRUE)
```

```
# A tibble: 5 x 5
  animal jump_length flea_count grade infected
  <chr>        <dbl>      <int> <dbl> <lgl>
1 dog            4.9         22     7 TRUE
2 dog           12.1         17     5 TRUE
3 dog            5.6         23     7 TRUE
4 cat            3.2         12     9 TRUE
5 cat            4.3         16     6 TRUE
```

Spalten ändern mit `mutate()`

https://dplyr.tidyverse.org/reference/mutate.html

```r
data_tbl %>%
  mutate(animal = as_factor(animal))
```

```
# A tibble: 14 x 5
   animal jump_length flea_count grade infected
   <fct>        <dbl>      <int> <dbl> <lgl>
 1 dog            5.2         18     8 FALSE
 2 dog            4.9         22     7 TRUE
 3 dog           12.1         17     5 TRUE
 4 dog            8.2         12     6 FALSE
 5 dog            5.6         23     7 TRUE
 6 dog            9.1         18     7 FALSE
 7 dog            7.4         21     9 FALSE
 8 cat            3.2         12     9 TRUE
 9 cat            1.2         13     5 FALSE
10 cat            6.6         11     7 FALSE
11 cat            4.1         12     8 FALSE
12 cat            4.3         16     6 TRUE
13 cat            7.8          9     6 FALSE
14 cat            6.2          7     8 FALSE
```

```r
data_tbl %>%
  mutate(long_jump = if_else(jump_length > 7, TRUE, FALSE)) %>%
  select(animal, jump_length, long_jump)
```

```
# A tibble: 14 x 3
   animal jump_length long_jump
   <chr>        <dbl> <lgl>
 1 dog            5.2 FALSE
 2 dog            4.9 FALSE
 3 dog           12.1 TRUE
 4 dog            8.2 TRUE
 5 dog            5.6 FALSE
 6 dog            9.1 TRUE
 7 dog            7.4 TRUE
 8 cat            3.2 FALSE
 9 cat            1.2 FALSE
10 cat            6.6 FALSE
11 cat            4.1 FALSE
12 cat            4.3 FALSE
```

```
13 cat            7.8 TRUE
14 cat            6.2 FALSE
```

> **Die Funktionen select(), filter() und mutate() in R**
>
> Bitte schaue dir auch die Hilfeseiten der Funktionen an. In diesem Skript kann ich nicht alle Funktionalitäten der Funktionen zeigen. Oder du kommst in das R Tutorium welches ich anbiete und fragst dort nach den Möglichkeiten Daten in R zu verändern.

Mehr Informationen durch `glimpse()` und `str()`

```
glimpse(data_tbl)
```

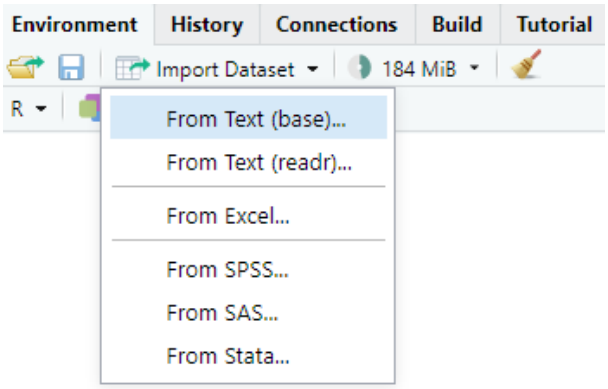
```
Rows: 14
Columns: 5
$ animal      <chr> "dog", "dog", "dog", "dog", "dog", "dog", "dog", "cat", "c~
$ jump_length <dbl> 5.2, 4.9, 12.1, 8.2, 5.6, 9.1, 7.4, 3.2, 1.2, 6.6, 4.1, 4.~
$ flea_count  <int> 18, 22, 17, 12, 23, 18, 21, 12, 13, 11, 12, 16, 9, 7
$ grade       <dbl> 8, 7, 5, 6, 7, 7, 9, 9, 5, 7, 8, 6, 6, 8
$ infected    <lgl> FALSE, TRUE, TRUE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE,~
```

```
str(data_tbl)
```

```
tibble [14 x 5] (S3: tbl_df/tbl/data.frame)
 $ animal     : chr [1:14] "dog" "dog" "dog" "dog" ...
 $ jump_length: num [1:14] 5.2 4.9 12.1 8.2 5.6 9.1 7.4 3.2 1.2 6.6 ...
 $ flea_count : int [1:14] 18 22 17 12 23 18 21 12 13 11 ...
 $ grade      : num [1:14] 8 7 5 6 7 7 9 9 5 7 ...
 $ infected   : logi [1:14] FALSE TRUE TRUE FALSE TRUE FALSE ...
```

Cadiergues, Joubert, and Franc (2000)

Daten in R einlesen



Formeln

Das ist ein Text

$$T_{\alpha=5\%} = \frac{\bar{x}_1 - \bar{x}_2}{s_{pooled} \cdot \sqrt{\frac{2}{n_{group}}}}$$

$$T = \frac{\text{signal}}{\text{noise}}$$

$$T = \sqrt{n}\frac{\bar{d}}{s_d}$$

$$T_{calc} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s^2_{x_1}}{n_{x_1}} \cdot \frac{s^2_{x_2}}{n_{x_2}}}}$$

$$T = \frac{\Delta \cdot n}{s}$$

$$\Pr(D|H_0)$$

$$\begin{aligned} T_{calc} &= \frac{7.24 - 9.71}{6.45 \cdot \sqrt{\frac{2}{7.5}}} \\ &= \frac{-2.47}{6.45 \cdot 0.52} \\ &= \frac{-2.47}{3.354} = -0.73 \end{aligned}$$

$$H_0 : \bar{x}_{dog} = \bar{x}_{cat}$$
$$H_A : \bar{x}_{dog} \neq \bar{x}_{cat}$$

$$\chi^2 = \frac{(O-E)^2}{E}$$

$$\left[(\bar{x}_{dog} - \bar{x}_{cat}) - T_{\left(1-\frac{\alpha}{2}\right)} \cdot \frac{s_p}{\sqrt{n_g}};\ (\bar{x}_{dog} - \bar{x}_{cat}) + T_{\left(1-\frac{\alpha}{2}\right)} \cdot \frac{s_p}{\sqrt{n_g}}; \right]$$

Explorative Datenanalyse

Im folgenden Kapitel wollen wir uns mit der explorativen Datenanalyse beschäftigen. Die explorative Datenanlalyse hat das Ziel Daten D zusammenzufassen und/oder zu visualisieren. Damit stell die explorative Datenanalyse den ersten Schritt zum Erkenntnisgewinn über ein Experiment dar.

Genutzte R Pakete für das Kapitel

Wir wollen folgende R Pakete in diesem Kapitel nutzen.

```
pacman::p_load(tidyverse, readxl)
```

Am Ende des Kapitels findest du nochmal den gesamten R Code in einem Rutsch zum selber durchführen oder aber kopieren.

Beispiel 1: Von Hunden und Flöhen

In unserem ersten Beispiel wollen wir uns verschiedene Daten D von Hunden und Hundeflöhen anschauen. Unter anderem sind dies die Sprungweite, die Anzahl an Flöhen, die Boniturnoten auf einer Hundemesse sowie der Infektionsstatus. Hier nochmal detailiert, was wir uns im folgenden im Kapitel einmal anschauen wollen.

- **Sprungweite** in [cm] von verschiedenen Flöhen

$$Y_{jump} = \{5.2, 4.9, 12.1, 8.2, 5.6, 9.1, 7.4\}.$$

- **Anzahl an Flöhen** auf verschiedenen Hunden

$$Y_{count} = \{18, 22, 17, 12, 23, 18, 21\}.$$

- **Boniturnoten** [11 = schlechteste bis 9 = beste Note] von verschiedenen Hunden

$$Y_{grade} = \{8, 7, 5, 6, 7, 7, 9\}.$$

- **Infektionstatus** [0 = gesund, 1 = infiziert] mit Flöhen von verschiedenen Hunden

$$Y_{infected} = \{0, 1, 1, 0, 1, 0, 0\}.$$

Je nachdem was wir messen, nimmt Y andere Zahlenräume an. Wir sagen, Y folgt einer Verteilung. Die Sprungweite ist normalverteilt, die Anzahl an Flöhen folgt einer Poisson Verteilung, die Boniturnoten sind multinominal/ordinal bzw. kategorial verteilt. Der Infektionsstatus ist binomial verteilt. Wir werden uns später die Verteilungen anschauen und visualisieren. Das können wir hier aber noch nicht. Wichtig ist, dass du schon mal gehört hast, dass Y unterschiedlich *verteilt* ist, je nachdem welche Dinge wir messen.

Absolutes Verhältnis

Wir schreiben, dass 3 von 4 Hunden von Flöhen befallen sind.

Relatives Verhältnis oder Risk Ratio

Wir schreiben, dass $3/7 = 0.43 = 43\%$ der Hunden einen Flohbefall haben.

Chancenverhältnis oder Odds Ratio

Wir schreiben, dass die Chance von Flöhen infiziert zu sein $4 : 3 = 4/3 = 1.33 = 133\%$ ist.

https://en.wikipedia.org/wiki/Categorical_distribution https://search.r-project.org/CRAN/refmans/LaplacesI

```
rmultinom(10, size = 12, prob = c(0.1,0.2,0.8))
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    0    0    0    3    3    2    1    1     1
[2,]    0    1    4    2    3    5    4    1    5     1
[3,]   11   11    8   10    6    4    6   10    6    10
```

Deskriptive Statistik

Wir messen sieben Sprungweiten von sieben Hundeflöhen und messen dabei folgende Werte in [cm]: 5.2, 4.9, 12.1, 8.2, 5.6, 9.1 und 7.4. Wir schreiben nun y als einen Vektor in der Form

$$y = \{5.2, 4.9, 12.1, 8.2, 5.6, 9.1, 7.4\}.$$

In R würde der Vektor wie etwas anders aussehen.

```
y <- c(5.2, 4.9, 12.1, 8.2, 5.6, 9.1, 7.4)
```

Mittelwert

$$\bar{y} = \sum_{i=1}^{n} \frac{x_i}{n} = \frac{5.2 + 4.9 + 12.1 + 8.2 + 5.6 + 9.1 + 7.4}{7} = 7.5$$

```
y %>% mean
```

```
[1] 7.5
```

Spannweite oder range

$$y_{range} = y_{max} - y_{min} = 12.1 - 4.9 = 7.2$$

```
range(y)
```

```
[1]  4.9 12.1
```

Varianz

$$s^2 = \sum_{i=1}^{n} \frac{(y_i - \bar{y})^2}{n-1} = \frac{(5.2 - 7.5)^2 + (4.9 - 7.5)^2 + ... + (7.4 - 7.5^2)}{7-1} = 6.65$$

```
y %>% var %>% round(2)
```

```
[1] 6.65
```

Standardabweichung

$$s = \sqrt{s^2} = \sqrt{6.65} = 2.58$$

```
y %>% sd %>% round(2)
```

```
[1] 2.58
```

Standardfehler oder Standard Error (SE)

$$SE = \frac{s}{\sqrt{n}} = \frac{2.58}{2.65} = 0.97$$

```
se <- sd(y)/sqrt(length(y))
se %>% round(2)
```

```
[1] 0.97
```

Median

$$4.9, 5.2, 5.6, \underbrace{7.4,}_{Median} 8.2, 9.1, 12.1$$

```
median(y)
```

```
[1] 7.4
```

Quartile

$$4.9, \underbrace{5.2,}_{1st\ Quartile} 5.6, 7.4, 8.2, 9.1, 12.1$$

$$4.9, 5.2, 5.6, 7.4, 8.2, \underbrace{9.1,}_{3rd\ Quartile} 12.1$$

```
quantile(y, probs = c(0.25, 0.5, 0.75))
```

```
 25%  50%  75%
5.40 7.40 8.65
```

Warum unterscheiden sich die händisch berechneten Quartile von den Quartilen aus R? Es gibt verschiedene Arten der Berechnung. In der Klausur nutzen wir die Art und Weise wie die händische Berechnung hier beschrieben ist. Später in der Anwendung nehmen wir die Werte, die R ausgibt. Die Abweichungen sind so maginal, dass wir diese Abweichungen in der praktischen Anwendung ignorieren wollen.

Interquartilesabstand (IQR)

$$IQR = \text{3rd Quartile} - \text{1st Quartile} = 9.1 - 5.2 = 3.9$$

Datenbeispiel

Wide format

dog	cat
5.2	10.1
4.9	9.4
12.1	11.8
8.2	6.7
5.6	8.2
9.1	9.1
7.4	7.1

```
jump_wide_tbl <- tibble(dog = c(5.2, 4.9, 12.1, 8.2, 5.6, 9.1, 7.4),
                        cat = c(10.1, 9.4, 11.8, 6.7, 8.2, 9.1, 7.1))

jump_wide_tbl
```

```
# A tibble: 7 x 2
    dog   cat
  <dbl> <dbl>
1   5.2  10.1
2   4.9   9.4
3  12.1  11.8
4   8.2   6.7
```

```
5   5.6   8.2
6   9.1   9.1
7   7.4   7.1
```

Long format

```
jump_tbl <- tibble(dog = c(5.2, 4.9, 12.1, 8.2, 5.6, 9.1, 7.4),
                   cat = c(10.1, 9.4, 11.8, 6.7, 8.2, 9.1, 7.1)) %>%
  gather(key = "animal", value = "jump_length")
jump_tbl
```

```
# A tibble: 14 x 2
   animal jump_length
   <chr>        <dbl>
 1 dog            5.2
 2 dog            4.9
 3 dog           12.1
 4 dog            8.2
 5 dog            5.6
 6 dog            9.1
 7 dog            7.4
 8 cat           10.1
 9 cat            9.4
10 cat           11.8
11 cat            6.7
12 cat            8.2
13 cat            9.1
14 cat            7.1
```

Zusammenfassen von Daten per Faktor

```
jump_tbl %>%
  gather(key = "animal", value = "jump_length") %>%
  mutate(animal = as_factor(animal)) %>%
  group_by(animal) %>%
  summarise(mean(jump_length),
            sd(jump_length))
```

```
# A tibble: 2 x 3
  animal `mean(jump_length)` `sd(jump_length)`
  <fct>                <dbl>             <dbl>
1 dog                   7.5               2.58
2 cat                   8.91              1.77
```

Mehr Daten oder zwei Gruppen

Bis jetzt haben wir uns die Sprungweite [cm] nur für Hunde angeschaut.

$$y_{dog} = \{5.2, 4.9, 12.1, 8.2, 5.6, 9.1, 7.4\}.$$

Wenn wir nun noch die Sprungweite von Katzen hinzunehmen, dann erhalten wir einen weiteren Vektor y_{cat} an Sprungweiten [cm].

$$y_{cat} = \{3.2, 1.2, 6.6, 4.1, 4.3, 7.8, 6.2\}.$$

$$y_{dog,\,cat} = \{5.2, 4.9, 12.1, 8.2, 5.6, 9.1, 7.4, 3.2, 1.2, 6.6, 4.1, 4.3, 7.8, 6.2\}.$$

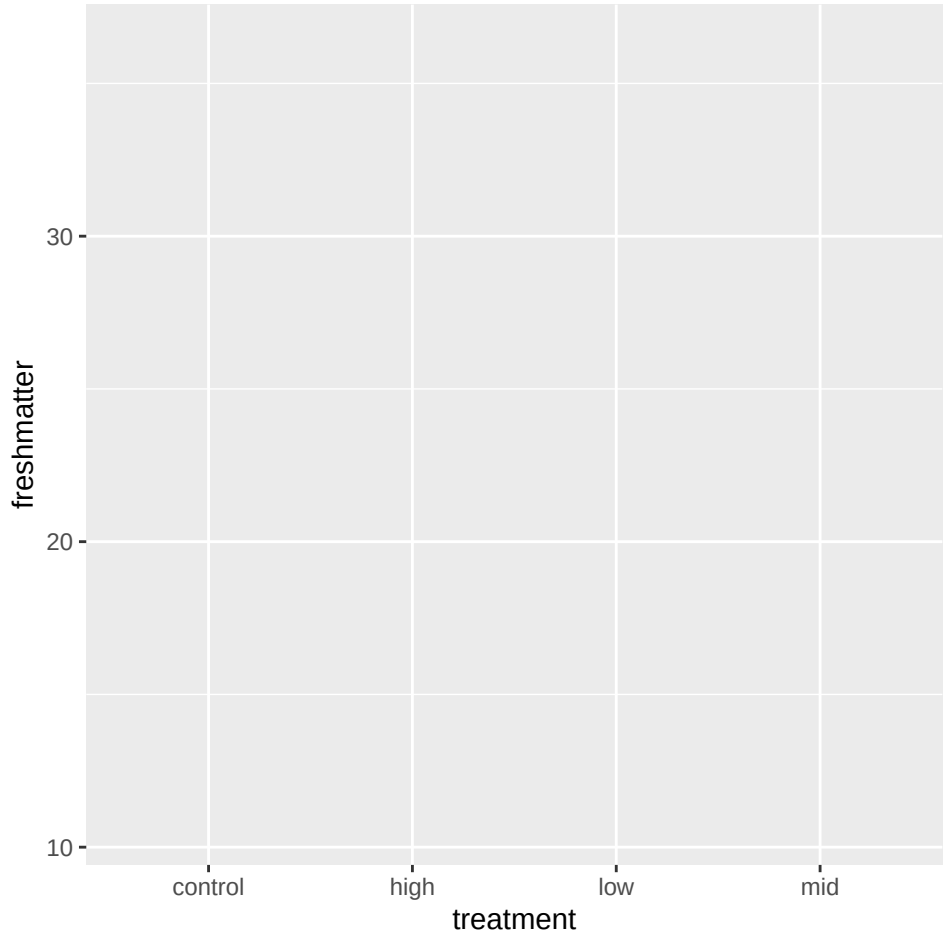
$$y = \begin{pmatrix} 5.2 \\ 4.9 \\ 12.1 \\ 8.2 \\ 5.6 \\ 9.1 \\ 7.4 \\ 3.2 \\ 1.2 \\ 6.6 \\ 4.1 \\ 4.3 \\ 7.8 \\ 6.2 \end{pmatrix} \quad x = \begin{pmatrix} dog \\ dog \\ dog \\ dog \\ dog \\ dog \\ dog \\ cat \\ cat \\ cat \\ cat \\ cat \\ cat \\ cat \end{pmatrix}$$

```r
tibble(y = c(y_dog, y_cat),
       x = rep(c("dog", "cat"), each = 7))
```

```
# A tibble: 14 x 2
       y x
   <dbl> <chr>
 1   5.2 dog
 2   4.9 dog
 3  12.1 dog
 4   8.2 dog
 5   5.6 dog
 6   9.1 dog
 7   7.4 dog
 8   3.2 cat
 9   1.2 cat
10   6.6 cat
11   4.1 cat
12   4.3 cat
13   7.8 cat
14   6.2 cat
```

Grundlagen in ggplot()

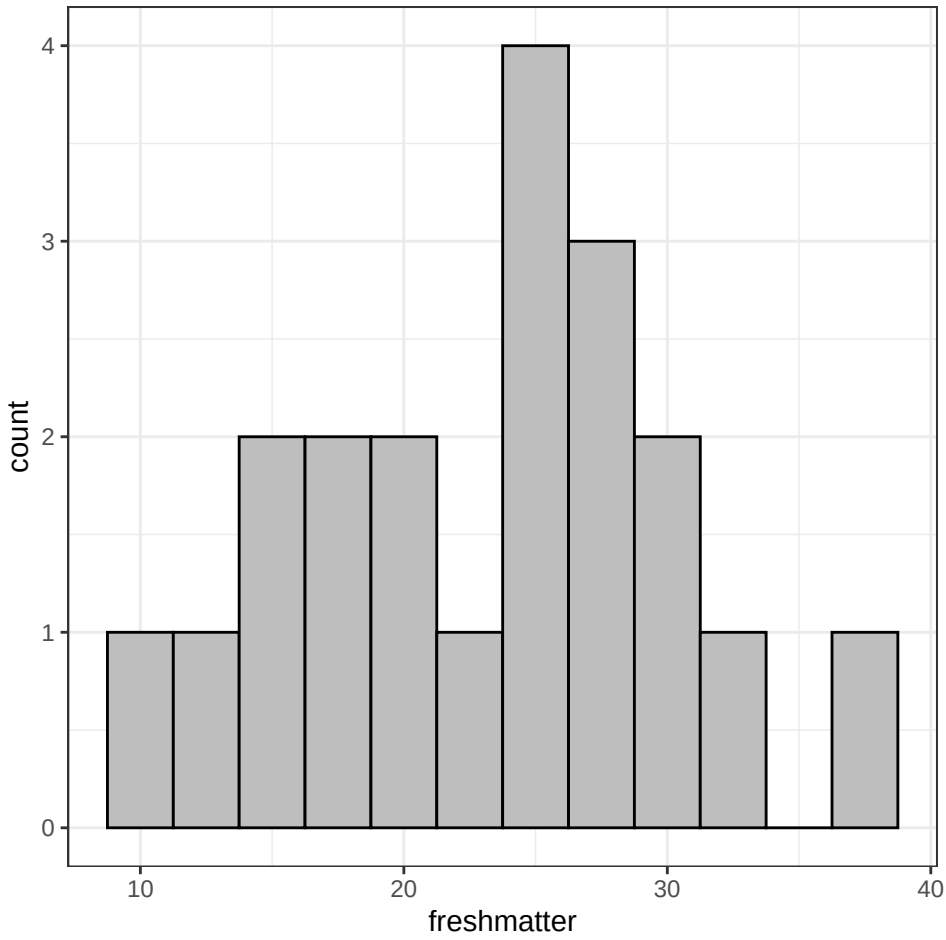
```
data_tbl <- read_excel(file.path("data/germination_data.xlsx"))

ggplot(data = data_tbl, aes(x = treatment, y = freshmatter))
```



Häufig verwendete Abbildungen

Histogramm

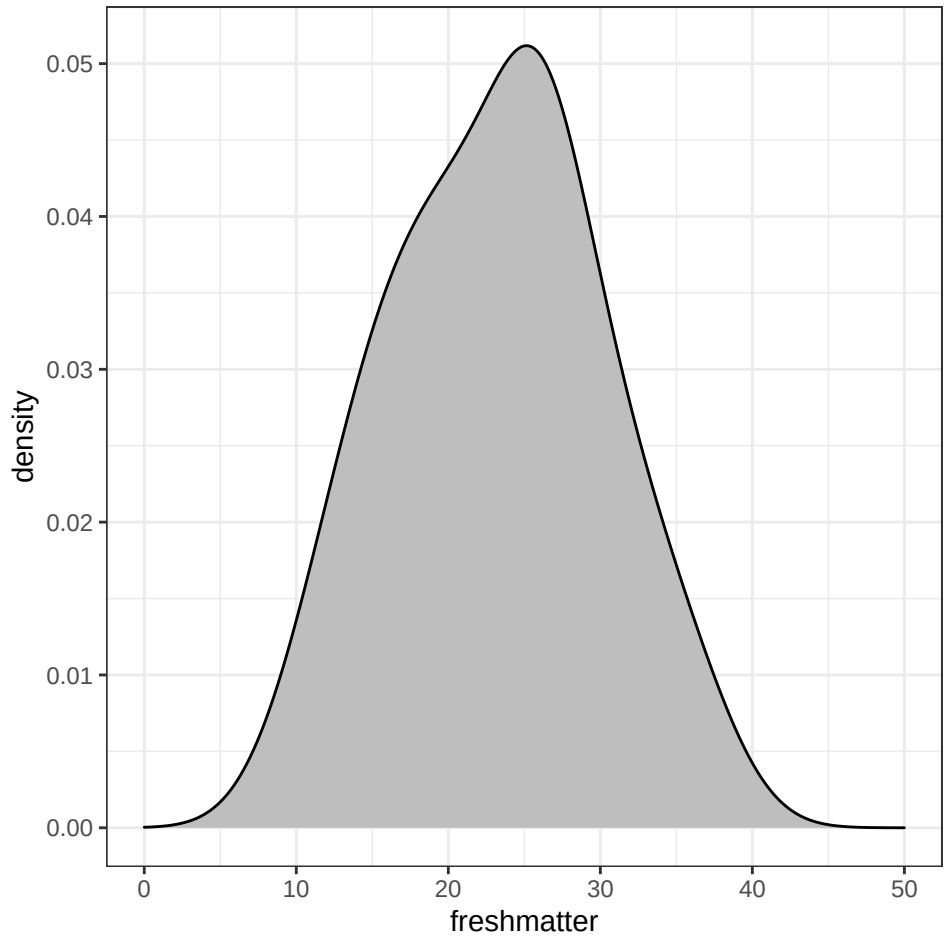
```
ggplot(data = data_tbl, aes(x = freshmatter)) +
  geom_histogram(binwidth = 2.5, fill = "gray", color = "black") +
  theme_bw()
```



Wenn wir viele Beobachtungen haben. Viele meint mehr als zwanzig Beobachtungen.

Density Plot

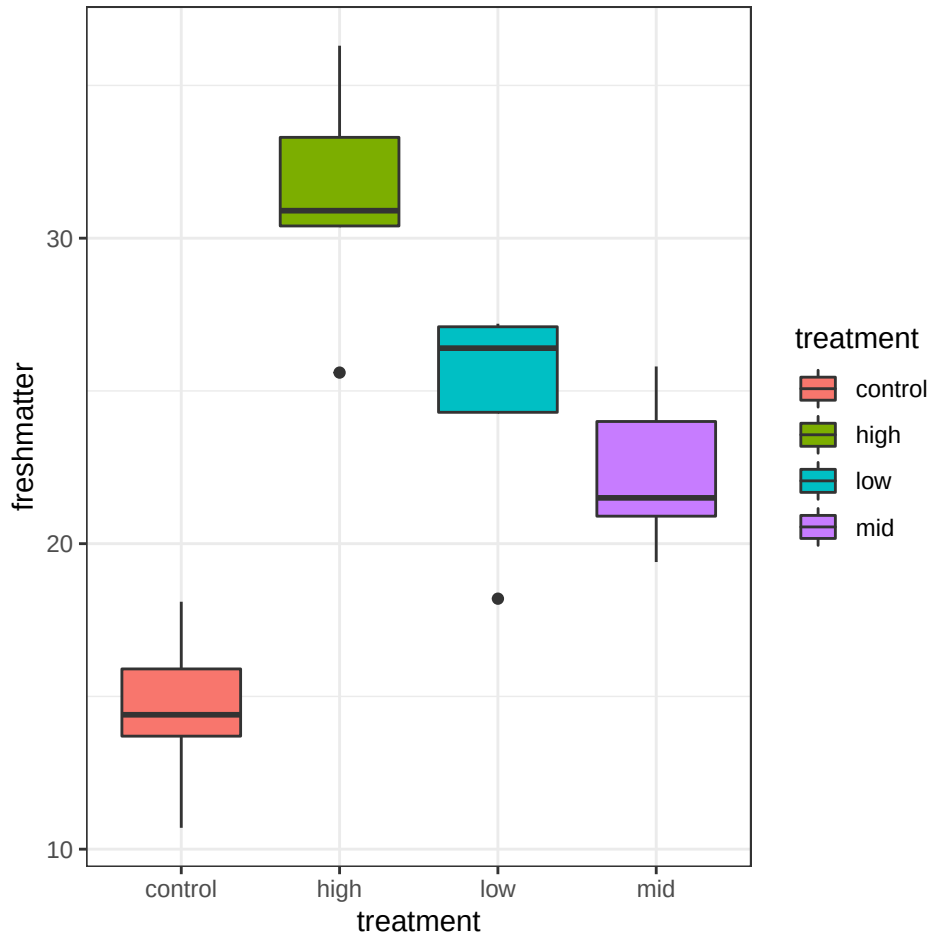
```
ggplot(data = data_tbl, aes(x = freshmatter)) +
  geom_density(fill = "gray", color = "black") +
  xlim(0, 50) +
  theme_bw()
```



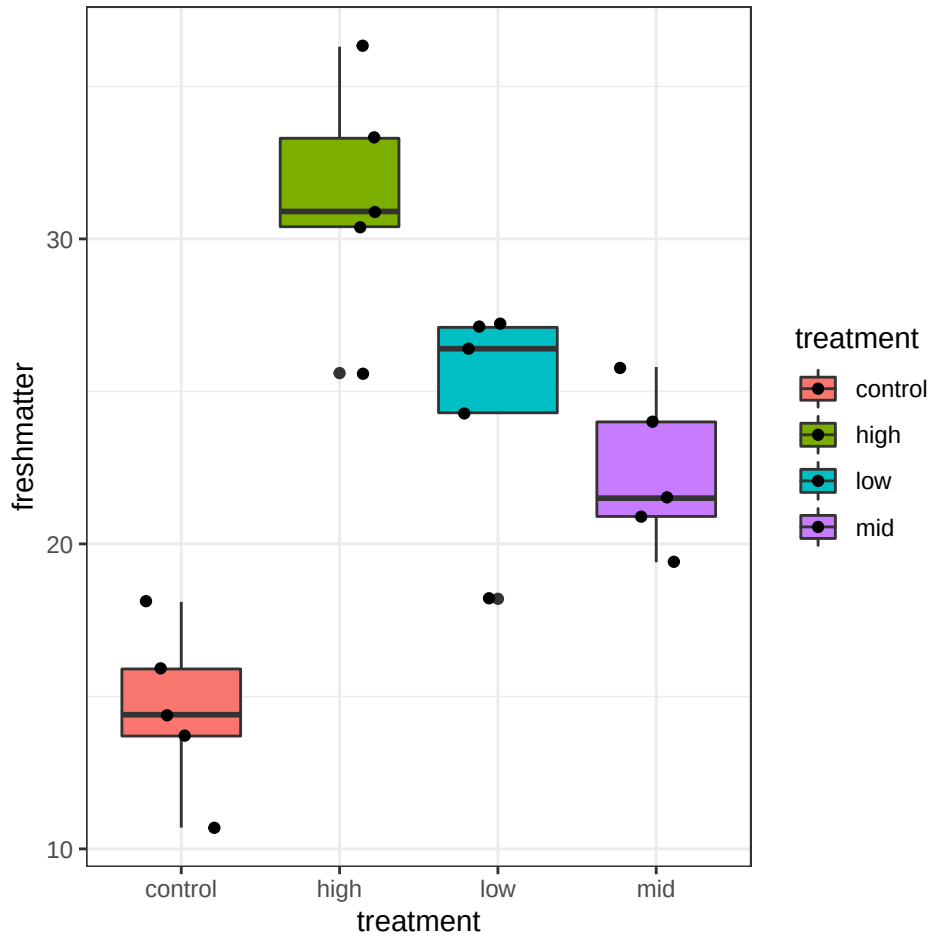
Wenn wir viele Beobachtungen.

Boxplot

```
ggplot(data = data_tbl, aes(x = treatment, y = freshmatter,
                            fill = treatment)) +
  geom_boxplot() +
  theme_bw()
```




```
ggplot(data = data_tbl, aes(x = treatment, y = freshmatter,
                            fill = treatment)) +
  geom_boxplot() +
  geom_jitter(width = 0.25) +
  theme_bw()
```



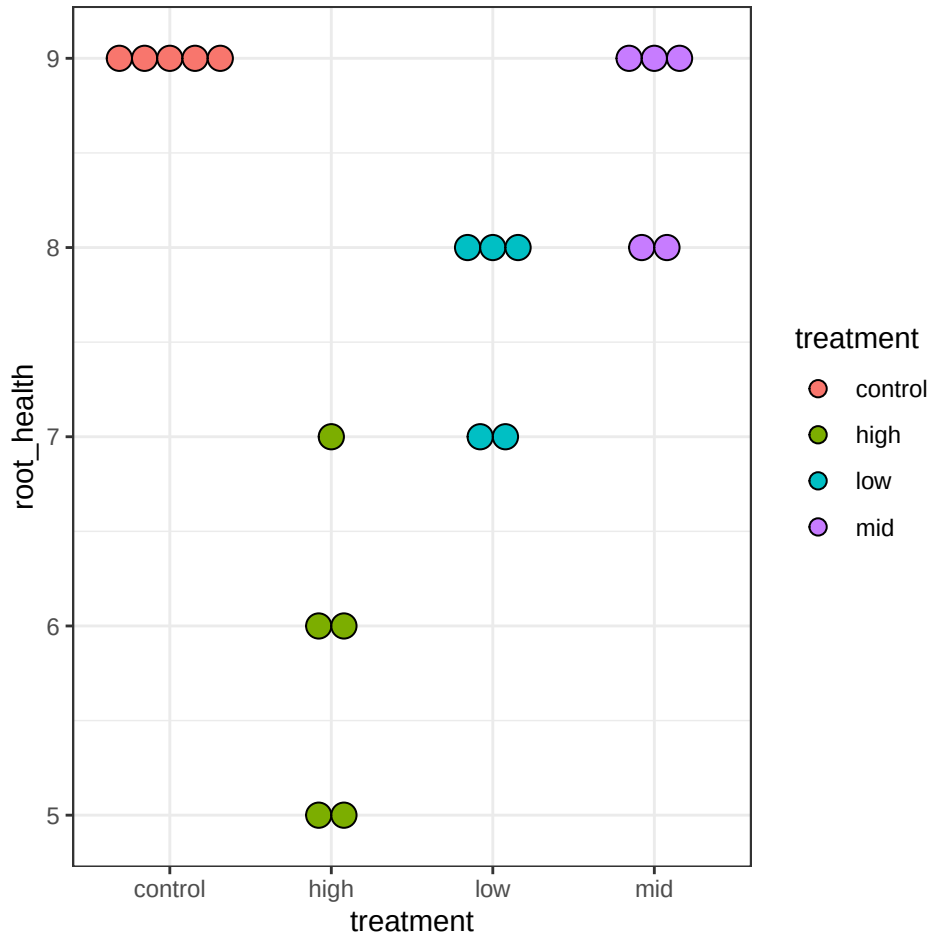
Wenn wir wenige Beobachtungen haben.

Dotplot

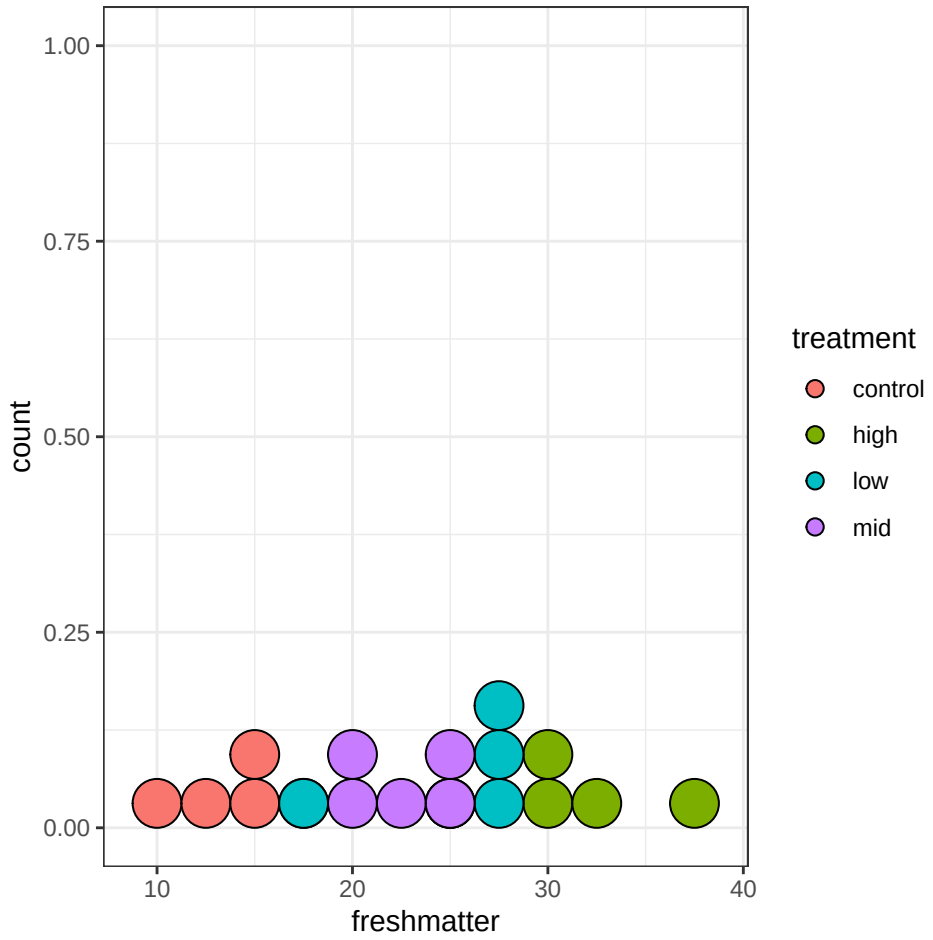
Wenn wir ganz wenige Beobachtungen haben.

```
ggplot(data = data_tbl, aes(x = treatment, y = root_health,
                            fill = treatment)) +
  geom_dotplot(binaxis = "y", stackdir = "center") +
  theme_bw()
```

```
Bin width defaults to 1/30 of the range of the data. Pick better value with `binwidth`.
```

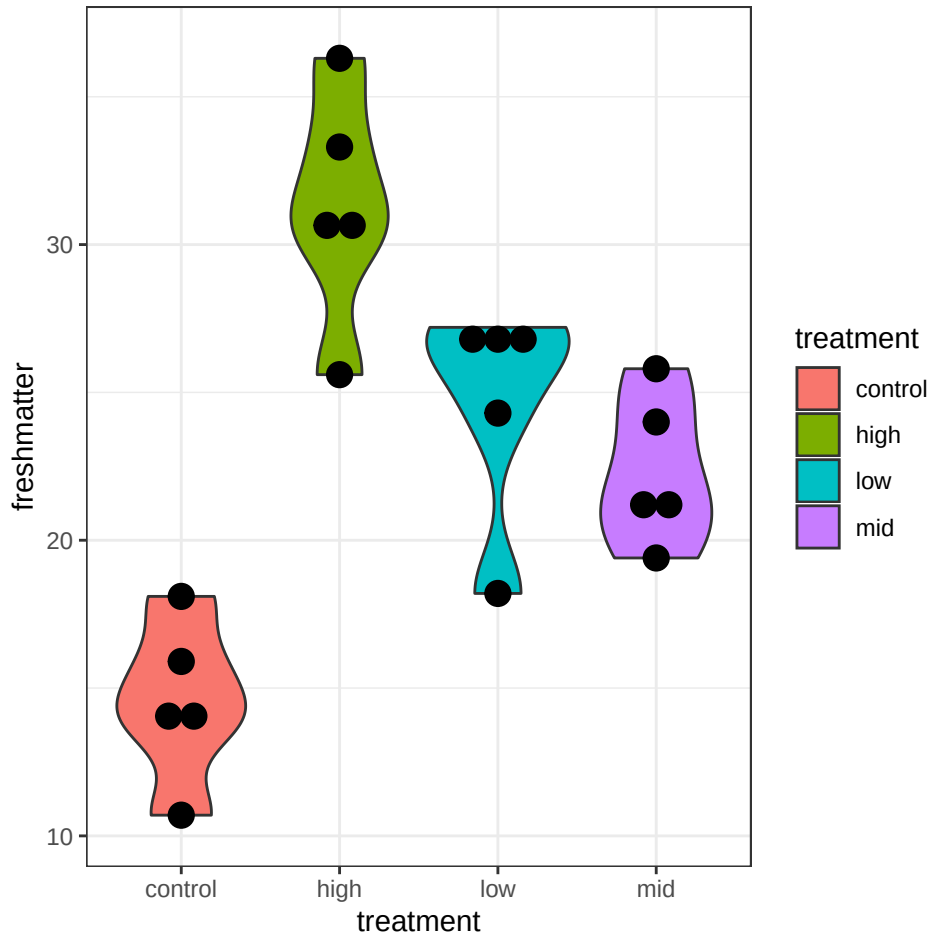



```
ggplot(data = data_tbl, aes(x = freshmatter, fill = treatment)) +
  geom_dotplot(method="histodot", binwidth = 2.5) +
  theme_bw()
```



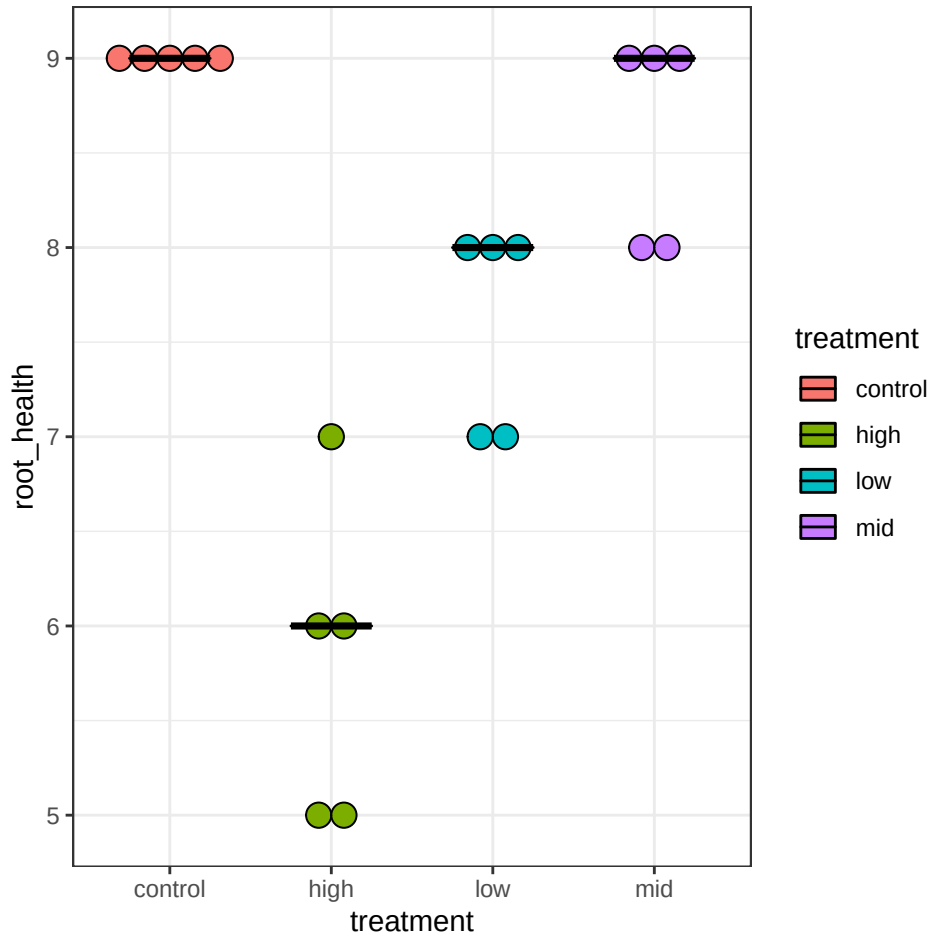

```
ggplot(data = data_tbl, aes(x = treatment, y = freshmatter,
                            fill = treatment)) +
  geom_violin() +
  geom_dotplot(binaxis = "y", stackdir = "center", fill = "black") +
  theme_bw()
```

```
Bin width defaults to 1/30 of the range of the data. Pick better value with `binwidth`.
```




```
ggplot(data = data_tbl, aes(x = treatment, y = root_health,
                            fill = treatment)) +
  geom_dotplot(binaxis = "y", stackdir = "center") +
  stat_summary(fun = median, fun.min = median, fun.max = median,
               geom = "crossbar", width = 0.5) +
  theme_bw()
```

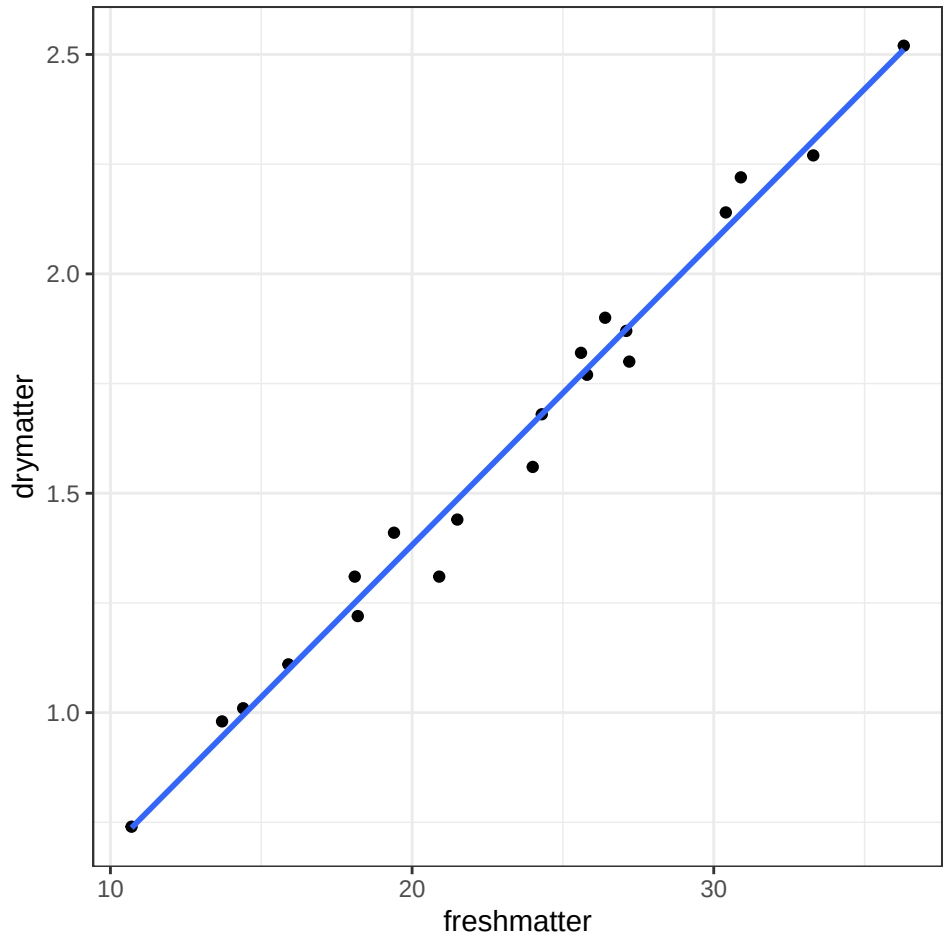
```
Bin width defaults to 1/30 of the range of the data. Pick better value with `binwidth`.
```



Scatterplot

```
ggplot(data = data_tbl, aes(x = freshmatter, y = drymatter)) +
  geom_point() +
  stat_smooth(method = "lm", se = FALSE) +
  theme_bw()
```

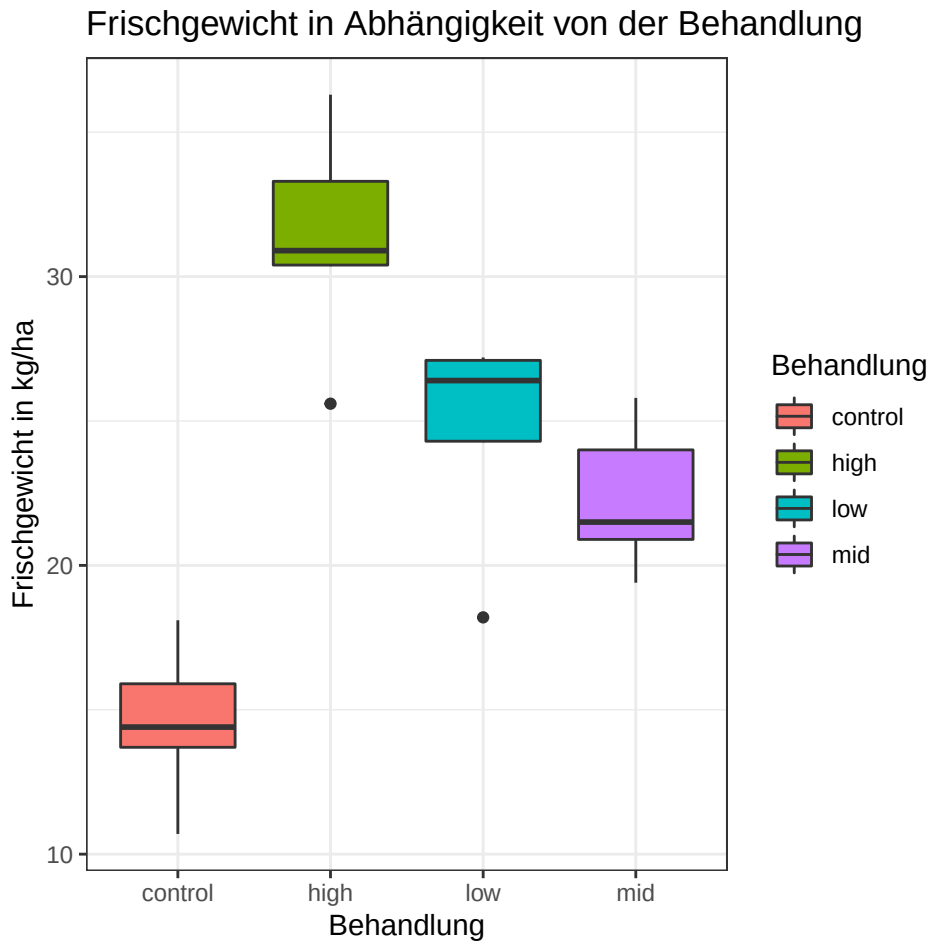
```
`geom_smooth()` using formula 'y ~ x'
```



Mosaic Plot

Abbildungen beschriften

```
ggplot(data = data_tbl, aes(x = treatment, y = freshmatter,
                            fill = treatment)) +
  geom_boxplot() +
  labs(title = "Frischgewicht in Abhängigkeit von der Behandlung",
       x = "Behandlung", y = "Frischgewicht in kg/ha",
       fill = "Behandlung") +
  theme_bw()
```



Methods

We describe our methods in this chapter.

Math can be added in body using usual syntax like this

math example

p is unknown but expected to be around 1/3. Standard error will be approximated

$$SE = \sqrt{(\frac{p(1-p)}{n})} \approx \sqrt{\frac{1/3(1-1/3)}{300}} = 0.027$$

You can also use math in footnotes like this[1].

We will approximate standard error to 0.027[2]

(Cadiergues, Joubert, and Franc 2000)

Literatur

Bruce, Peter, Andrew Bruce, and Peter Gedeck. 2020. *Practical Statistics for Data Scientists: 50+ Essential Concepts Using r and Python*. O'Reilly Media.

Cadiergues, Marie-Christine, Christel Joubert, and Michel Franc. 2000. "A Comparison of Jump Performances of the Dog Flea, Ctenocephalides Canis (Curtis, 1826) and the Cat Flea, Ctenocephalides Felis Felis (Bouché, 1835)." *Veterinary Parasitology* 92 (3): 239–41.

Dormann, Carsten F. 2013. *Parametrische Statistik*. Springer.

Wickham, Hadley, and Garrett Grolemund. 2016. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc.

[1] where we mention $p = \frac{a}{b}$

[2] p is unknown but expected to be around 1/3. Standard error will be approximated

$$SE = \sqrt{(\frac{p(1-p)}{n})} \approx \sqrt{\frac{1/3(1-1/3)}{300}} = 0.027$$