Bio Data Science

Prof. Dr. Jochen Kruppa

Inhaltsverzeichnis

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

1. Vorwort

1.1. Lernziel 1: Eine explorative Datananalyse durchführen

Gleich zu Beginn R Code zu zeigen und eine entsprechende Abbildung ist vielleicht ungewöhnlich, aber wir wollen zu dieser Abbildung 1.1 hin. In Abbildung 1.1 siehst du einen Boxplot. Und wie wir aus den Daten `flea_dog_cat.xlsx` einen Boxplot erstellen, das soll uns in den nächsten Kapitel beschäftigen. Dafür müssen wir nämlich eine Menge in dem Codeblock verstehen und dann auch Anwenden können. Und natürlich lernen was eigentlich ein Boxplot ist und was in einem Boxplot eigentlich dargestellt ist.

Hier ist der Codeblock der in R die Abbildung 1.1 erstellt.

```
## Einlesen von Daten aus Excel
data_tbl <- read_excel("data/flea_dog_cat.xlsx")

## Umformen der <chr> Spalte in einen Factor <fct>
data_tbl <- data_tbl %>%
  mutate(animal = as_factor(animal))

## Auswählen der wichtigen Spalten für den anschließenden Boxplot
data_tbl <- data_tbl %>%
  select(animal, jump_length)

## Generieren des Boxplots in ggplot()
ggplot(data_tbl, aes(x = animal, y = jump_length, fill = animal)) +
  geom_boxplot() +
  geom_jitter() +
  labs(x = "Tierart", y = "Sprungweite in [cm]", fill = "Tierart")
```

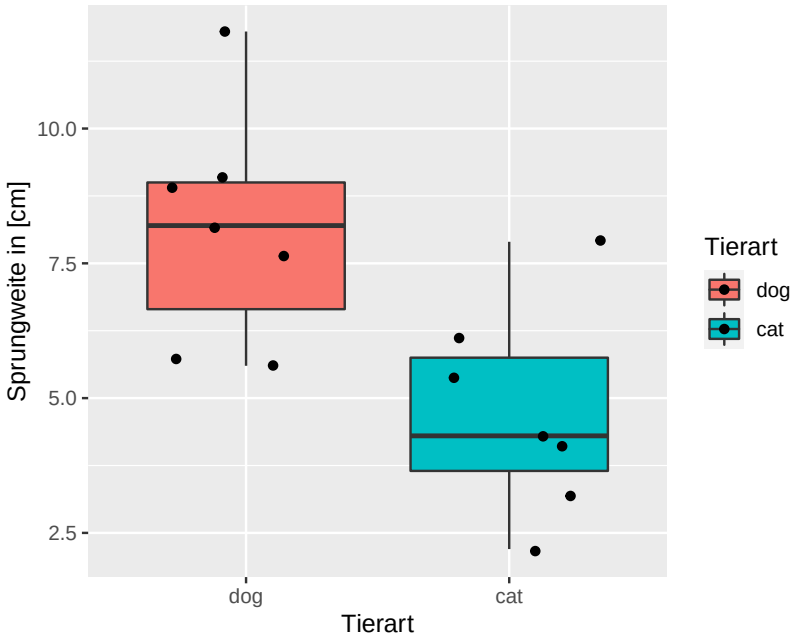


Abbildung 1.1.: Boxplot der Sprungweiten [cm] von Hunden und Katzen.

Wir müssen nun folgende Dinge lernen um den Codeblock zu verstehen:

- Wir müssen das Datenbeispiel verstehen. Was sind das eigentlich für Daten, die wir da abbilden? Was sind überhaupt Daten im Sinne der Statistik bzw. für R.
- Wir müssen den R Code verstehen. Von einzelnen wichtigen Opertatoren wie `->` und `%\>%` zu dem den Unterschieden von Worten und Objekten.
- Wie kriegen wir Daten aus Excel in R hinein? Wir können die Daten ja nicht einfach in R eintragen sondern haben die Daten ja meist in einer (Excel) Datei wie `flea_dog_cat.xlsx`.
- Was ist eigentlich ein Boxplot und welche statistsichen Maßzahlen werden hier eigentlich abgebildet?

All diese Fragen und weitere Fragen, die sich diesen Fragen anschließen, wollen wir uns in den nächsten Kapitel anschauen. Leider kann ich hier nur *linear* schreiben.

1.2. RStudio und R

Wir arbeiten in R und nutzen dafür das RStudio. Führe einfach folgende Schritte aus.

1. R installieren unter https://cran.rstudio.com/
2. RStudio installieren unter https://www.rstudio.com/products/rstudio/download/#download

Bitte die Reihenfolge bachten. Beide Schritte kannst du dir auch nochaml im Video anschauen oder aber du kommst in das R Tutorium was regelmäßig an der Hochschule Osnabrück von mir angeboten wird.

> Was ist eigentlich RStudio und woher kriege ich das?
>
> Du findest auf YouTube Einführung in R - Teil 01 - Installation von RStudio und R als Video. Ich gehe in dem Video einmal alle wichtigen Schritte durch und so kannst du dir Rstudio und R installieren.

1.3. Lernziel 2: Statistische Versuche verstehen

Wie funktioniert ein *statistischer* Versuch? Ich könnte auch wissenschaftliches Experiment schreiben, aber ein wissenschfatliches Experiment ist sehr abstrakt. Wir wollen ja einen Versuch durchführen und danach - ja was eigentlich? Was wollen wir nach dem Versuch haben? Meistens eine neue Erkenntnis. Um diese Erkenntnis zu validieren oder aber abzusichern nutzen wir Statistik. Dazu musst du noch wissen, dass wir eine spezielle Form der Statistik nutzen: die *frequentistische Statistik*.

Die *frequentistische Statistik* basiert - wie der Name andeutet - auf Wiederholungen in einem Versuch. Daher der Name frequentistisch. Also eine Frequenz von Beobachtungen. Ist ein wenig gewollt, aber daran gewöhnen wir uns schon mal. Konkret, ein Experiment welches wir frequentistisch Auswerten wollen besteht immer aus biologischen Wiederholungen. Wir müssen also ein Experiment planen in dem wir wiederholt ein Outcome

Eine **biologische Wiederholung** ist die gleiche Messung an einem anderen Tier, Pflanze oder Mensch. Eine **technische** Wiederholung ist die gleiche Messung an dem gleichen Tier, Pflanze oder Mensch.

an vielen Tieren, Pflanzen oder Menschen messen. Auf das Outcome gehen wir noch später ein. Im Weiteren konzentrieren wir uns hier auf die *parametrische* Statistik. Die parametrische Statistik beschäftigt sich mit Parametern von Verteilungen. Ein schwieriger Satz. Schauen wir uns einmal eine Verteilung an. Abbildung 1.2 zeigt eine Normalverteilung.

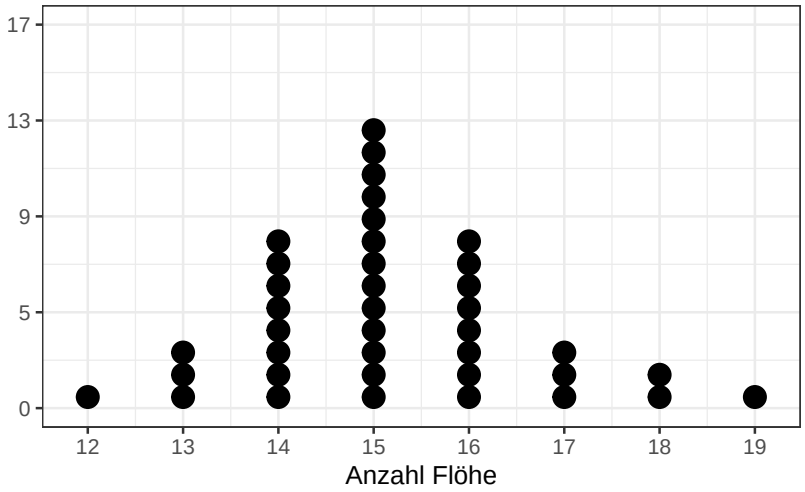


Abbildung 1.2.: An 39 Hunden wurde die Anzahl an Flöhen gezählt.

Parameter sind Zahlen, die eine Verteilungskurve beschreiben.

Eine Verteilung hat Parameter. Parameter sind die Eigenschaften einer Verteilung, die notwendig sind um eine Verteilung vollständig zu beschreiben. Im Falle der Normalverteilung brauchen wir einen Paramter für den höchsten Punkt der Kurve, sowie einen Parameter für die Ausbreitung, also wie weit geht die Kurve nach links und nach rechts.

1) Der Mittelwert beschriebt den höchsten Punkt einer Normalverteilung.
2) Die Standardabweichung beschreibt die Ausbreitung einer Normalverteilung.

- Wir messen wiederholt ein Outcome an verschiedenen Tieren, Pflanzen oder Menschen
- Wir überlegen uns aus welcher Verteilungsfamilie unser Outcome stammt

https://rstudio-pubs-static.s3.amazonaws.com/100906_8e3a32dd11c14b839468db756cee7400.html

Hundeffloh Beispiel in groß!!! Dormann (2013) Hurlbert (1984)

Mit der Poisson Verteilunt anfangen??? Nur ein Lageparameter und eine diskrete Verteilung. Dann rüber zur Normalverteilung und mit der dann zum t-Test…

Data Science for Agriculture in R

1.4. Literatur

2. 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.

> **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.

2.1. 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

2.1.1. 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.

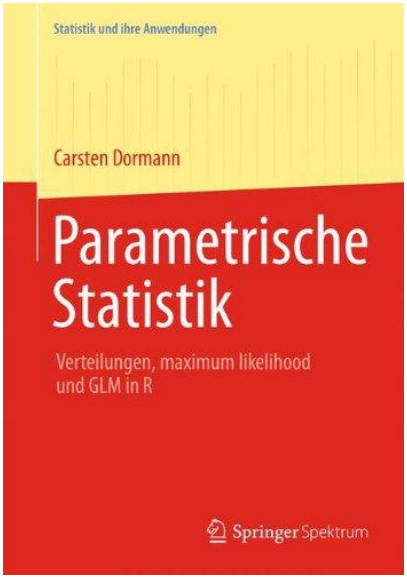
2.1.2. Auf GitHub

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.

2.2. 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.

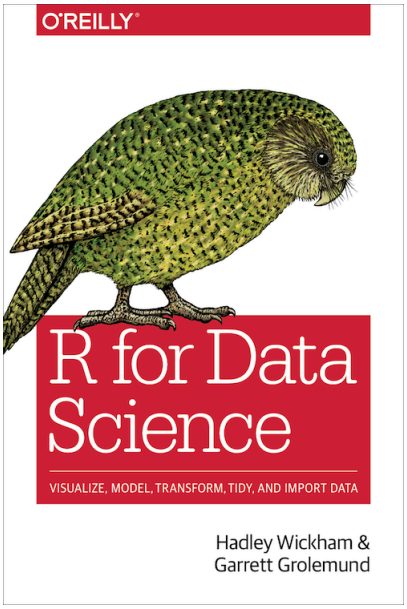
2.2.1. Parametrische Statistik



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.

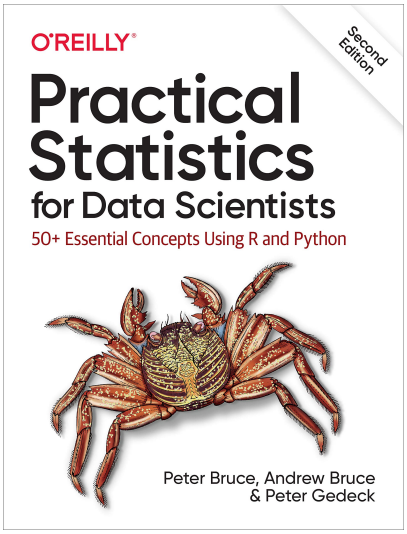
2.2.2. R for Data Science



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.

2.2.3. 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.



2.2.4. 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.

2.2.5. 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*.

2.3. Literatur

Teil I.

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.

Palmer Penguins

Palmer Penguins

3. 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

$$Y_{jump} = \{5.7, 8.9, 11.8, 8.2, 5.6, 9.1, 7.6\}.$$

- **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.

3.1. Palmer Penguins

Palmer Penguins

4. 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 3 die Sprungweiten [cm] wieder aufschreiben:

$$Y_{jump} = \{3.2, 2.2, 5.4, 4.1, 4.3, 7.9, 6.1\}.$$

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.

$$Y_{jump,\,dog} = \{5.7, 8.9, 11.8, 8.2, 5.6, 9.1, 7.6\}$$

$$Y_{jump,\,cat} = \{3.2, 2.2, 5.4, 4.1, 4.3, 7.9, 6.1\}$$

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 4.1. Wir haben jetzte eine *Daten*tabelle.

Tabelle 4.1.: Sprunglängen [cm] für Hunde- und Katzenflöhe. Die Tabelle ist im Wide-Format dargestellt.

dog	cat
5.7	3.2
8.9	2.2
11.8	5.4
8.2	4.1
5.6	4.3
9.1	7.9
7.6	6.1

Intuitiv ist die Tabelle 4.1 übersichtlich und beinhaltet die Informationen die wir wollten. Dennoch haben wir das Probem, das wir in dieser Tabelle 4.1 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 4.1 in das Long-Format. Dargestellt in Tabelle 4.2. Jede Beobachtung belegt nun eine Zeile. Dies ist sehr wichtig im Kopf zu behalten, wenn du eigene Daten in z.B. Excel eingibts.

```
Warning: Please use 'rowNames' instead of 'row.names'
```

Tabelle 4.2.: Sprunglängen [cm] für Hunde- und Katzenflöhe. Die Tabelle ist im Long-Format dargestellt.

animal	jump_length	flea_count	grade	infected
dog	5.7	18	8	0
dog	8.9	22	7	1
dog	11.8	17	5	1
dog	8.2	12	6	0
dog	5.6	23	7	1
dog	9.1	18	7	0
dog	7.6	21	9	0
cat	3.2	12	9	1
cat	2.2	13	5	0
cat	5.4	11	7	0
cat	4.1	12	8	0

animal	jump_length	flea_count	grade	infected
cat	4.3	16	6	1
cat	7.9	9	6	0
cat	6.1	7	8	0

4.1. Palmer Penguins

Palmer Penguins

5. 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.

> **!** 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 4.2 mit Daten zu verschiedenen Oucomes, wie Sprungweite [cm], Anzahl an Flöhen auf Hunden und Katzen, die Boniturnoten oder aber den Infektionsstatus. Die Tabelle 4.2 ist zwar nicht groß aber auch nicht wirklich klein. Im nächsten Kapitel wollen wir uns damit beschäftigen, die Zahlen in der Tabelle sinnvoll zusammenzufassen.

5.1. Palmer Penguins

Palmer Penguins

Teil II.

Daten in R

Im vorherigen Kapitel haben wir die Datentabelle Tabelle 4.2 erschaffen. Bevor wir uns weiter mit statistischen Kennzahlen beschäftigen, wollen wir uns einmal die Realisierung der Tabelle Tabelle 4.2 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.

6. Von Buchstaben und Zahlen

Im vorherigen Kapitel haben wir die Datentabelle Tabelle 4.2 erschaffen. Bevor wir uns weiter mit statistischen Kennzahlen beschäftigen, wollen wir uns einmal die Realisierung der Tabelle Tabelle 4.2 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. Nun haben wir Tabelle Tabelle 4.2 mit Daten zu verschiedenen Outcomes, wie Sprungweite [cm], Anzahl an Flöhen auf Hunden und Katzen, die Boniturnoten oder aber den Infektionsstatus. Die Tabelle Tabelle 4.2 ist zwar nicht groß aber auch nicht wirklich klein. Wir wollen uns nun damit beschäftigen, die Zahlen sinnvoll in R darzustellen.

Tabelle 6.1.: Sprunglängen [cm] für Hunde- und Katzenflöhe. Die Tabelle ist im *Long*-Format dargestellt.

animal	jump_length	flea_count	grade	infected
dog	5.7	18	8	0
dog	8.9	22	7	1
dog	11.8	17	5	1
dog	8.2	12	6	0
dog	5.6	23	7	1
dog	9.1	18	7	0
dog	7.6	21	9	0
cat	3.2	12	9	1
cat	2.2	13	5	0
cat	5.4	11	7	0
cat	4.1	12	8	0
cat	4.3	16	6	1
cat	7.9	9	6	0
cat	6.1	7	8	0

6.1. Daten in R sind `tibble()`

Im folgenden sehen wir die Datentabelle Tabelle 4.2 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            5.7         18     8 FALSE
 2 dog            8.9         22     7 TRUE
 3 dog           11.8         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            7.6         21     9 FALSE
 8 cat            3.2         12     9 TRUE
 9 cat            2.2         13     5 FALSE
10 cat            5.4         11     7 FALSE
11 cat            4.1         12     8 FALSE
12 cat            4.3         16     6 TRUE
13 cat            7.9          9     6 FALSE
14 cat            6.1          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.

> **i** 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.

6.2. Faktoren als Wörter zu Zahlen

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 4.2 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
```

6.3. 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"
```

6.4. 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.

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

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

```
"animal"
```

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

6.4.2. 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
```

6.5. 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.

```
animal_1_tbl <- select(data_tbl, animal, jump_length)
animal_2_tbl <- filter(animal_1_tbl, jump_length >= 4)
sort(animal_2_tbl$jump_length)
```

```
 [1]  4.1  4.3  5.4  5.6  5.7  6.1  7.6  7.9  8.2  8.9  9.1 11.8
```

```
data_tbl %>%
  select(animal, jump_length) %>%
  filter(jump_length >= 4) %>%
  pull(jump_length) %>%
  sort
```

```
 [1]  4.1  4.3  5.4  5.6  5.7  6.1  7.6  7.9  8.2  8.9  9.1 11.8
```

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.

6.6. 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

6.6.1. 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            5.7         18
 2 dog            8.9         22
 3 dog           11.8         17
 4 dog            8.2         12
 5 dog            5.6         23
 6 dog            9.1         18
 7 dog            7.6         21
 8 cat            3.2         12
 9 cat            2.2         13
10 cat            5.4         11
11 cat            4.1         12
12 cat            4.3         16
13 cat            7.9          9
14 cat            6.1          7
```

6.6.2. 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            5.7         18     8 FALSE
2 dog            8.9         22     7 TRUE
3 dog           11.8         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            7.6         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            5.7         18     8 FALSE
2 dog            8.9         22     7 TRUE
3 dog           11.8         17     5 TRUE
4 dog            5.6         23     7 TRUE
5 dog            9.1         18     7 FALSE
6 dog            7.6         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            8.9         22     7 TRUE
2 dog           11.8         17     5 TRUE
3 dog            5.6         23     7 TRUE
4 cat            3.2         12     9 TRUE
5 cat            4.3         16     6 TRUE
```

6.6.3. 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            5.7         18     8 FALSE
 2 dog            8.9         22     7 TRUE
 3 dog           11.8         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            7.6         21     9 FALSE
 8 cat            3.2         12     9 TRUE
 9 cat            2.2         13     5 FALSE
10 cat            5.4         11     7 FALSE
11 cat            4.1         12     8 FALSE
12 cat            4.3         16     6 TRUE
13 cat            7.9          9     6 FALSE
14 cat            6.1          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            5.7 FALSE
 2 dog            8.9 TRUE
 3 dog           11.8 TRUE
 4 dog            8.2 TRUE
 5 dog            5.6 FALSE
 6 dog            9.1 TRUE
 7 dog            7.6 TRUE
 8 cat            3.2 FALSE
 9 cat            2.2 FALSE
10 cat            5.4 FALSE
```

```
11 cat            4.1 FALSE
12 cat            4.3 FALSE
13 cat            7.9 TRUE
14 cat            6.1 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.

6.7. 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> 5.7, 8.9, 11.8, 8.2, 5.6, 9.1, 7.6, 3.2, 2.2, 5.4, 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] 5.7 8.9 11.8 8.2 5.6 9.1 7.6 3.2 2.2 5.4 ...
 $ 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 ...
```

6.8. Pakete und `library()`

In der Vanilla[1] Variante hat R sehr wenige Funktionen. Ohne zusätzliche Pakete ist R mehr ein sehr potenter Taschenrechner. Leider mit der Funktionalität aus den 90'zigern, was die Programmierumgebeung und die Funktionen angeht. Das wollen wir aber nicht. Wir wollen auf den aktuellen Stand der Technik und auch Sprache programmieren. Daher nutzen wir zusätzliche R Pakete.

[1] Als Vanilla beschreibt man in der Informatikerwelt ein Programm, was keine zusätzlichen Pakete geladen hat. Also die reinst Form ohne zusätzlichen Geschmack.

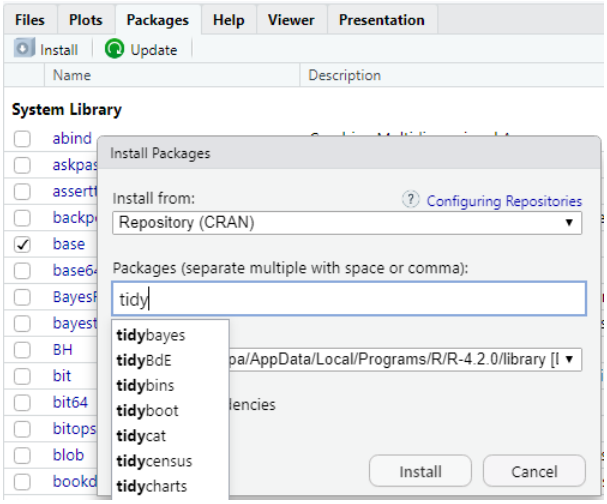


Abbildung 6.1.: Auf den Reiter *Packages* klicken und dann *Install*. In der deutschen version vom RStudio mögen die Begriffe leicht anders sein.

In Abbildung 6.1 wird gezeigt wie du ein zusätzliches Paket installieren kannst. Hierbei ist nochmal wichtig den semantischen Unterschied zu wissen. Es gibt das Paket `tidyverse` was wir viel nutzen. Wir isnatllieren *einmalig* Pakete der Funktion `install.packages()` oder eben wie in Abbildung 6.1 gezeigt. Wir nutzen die Funktion `library()` um ein Paket in R zu laden. Ja, es müsste anders heisen, tut es aber nicht.

```
## Das Paket tidyverse installieren - einmalig
install.packages(tidyverse)

## Das Paket tidyverse laden - jedes Mal
```

```
library(tidyverse)
```

Nun muss man sich immer merken, ob das Paket schon installiert ist oder man schreibt relativ viele `library()` untereinander. Das passiert schnell, wenn du viele Pakete laden willst. Dafür erlaubt dir das Paket `pacman` eine Vereinfachung. Die Funktion `p_load()` installiert Pakete, wenn die Pakete nicht installiert sind. Sollten die Pakete installiert sein, so werden die Pakete geladen. Du musst nur einmal `install.packages(pacman)` ausführen um das Paket `pacman` zu installieren.

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

Schlussendlich gibt es noch die Möglichkeit sich alles nochmal bei YoTube anzuschauen.

> Unterschied von Packages und Libraries in R
>
> Du findest auf YouTube Einführung in R - Teil 03 - Unterschied Packages und Libraries in R als Video. Hier erkläre ich nochmal den Ablauf zwischen Installieren eines Paketes und dem Laden eines Paketes.

7. Daten in R einlesen

Gängige Fehler beim Einlesen von Dateien in R

- der Pfad zur Datei ist falsch (Kapitel 7.3)
- in der Datei sind komische Zeichen, wie Umlaute und Co. (Kapitel 7.4)
- in der Datei sind Leerzeichen in den Spaltennamen (Kapitel 7.5)

7.1. Genutzte R Pakete für das Kapitel

Wir wollen folgende R Pakete in diesem Kapitel nutzen.

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

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

7.2. Importieren mit RStudio

Wir können das RStudio nutzen um Daten mit Point-and-Klick rein zuladen und dann den Code wieder in den Editor kopieren. Im Prinzip ist dieser Weg der einfachste um einmal zu sehen, wie ein pfad funktioniert und der Code lautet. Später benötigt man diese 'Krücke' nicht mehr. Wir nutzen dann direkt den Pfad zu der Datei. Abbildung 7.1 zeigt einen Ausschnitt, wo wir im RStudio die *Import Dataset* Funktionalität finden.

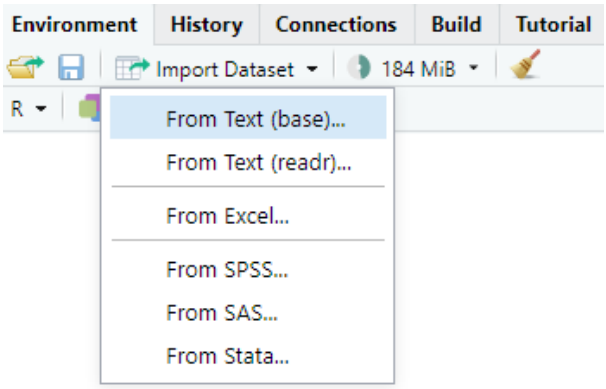


Abbildung 7.1.: Auf den Reiter *Einviroment* klicken und dann *Import Dataset*. In der deutschen version vom RStudio mögen die Begriffe leicht anders sein.

> **Importieren mit RStudio als Video**
>
> Du findest auf YouTube Einführung in R - Teil 21.0 - Daten importieren mit RStudio - Point and Klick als Video. Point and Klick ist als Video einfacher nachzuvollziehen als Screenshots in einem Fließtext.

7.3. Importieren per Pfad

In Abbildung 7.2 können wir sehen wie wir den Pfad zu unserer Excel Datei `flea_dog_cat.xlsx` finden. Natürlich kannst du den Pfad auch anders herausfinden bzw. aus dem Explorer oder Finder kopieren.

Nachdem wir den Pfad gefunden haben, können wir den Pfad in die Funktion `read_excel()` kopieren und die Datei in das Objekt `data_tbl` einlesen. Ja, es wird nichts in der R Console ausgegeben, da sich die Daten jetzt in dem Object `data_tbl` befinden.

```
## Ganzer Pfad zur Datei flea_dog_cat.xlsx
data_tbl <- read_excel("data/flea_dog_cat.xlsx")
```

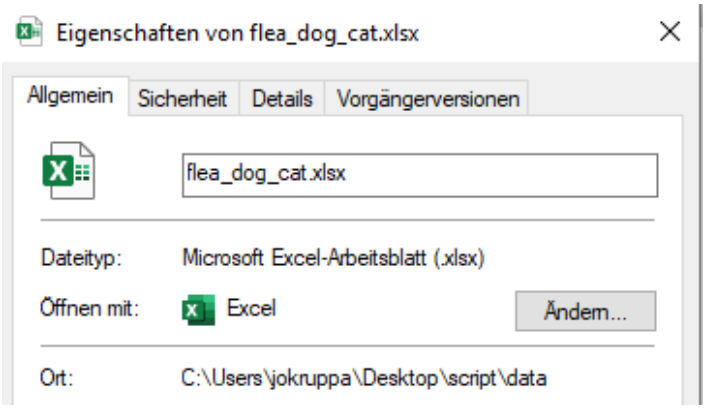


Abbildung 7.2.: Durch den Rechts-Klick auf die Eigenschaften einer Datei kann man sich den Pfad zur Datei anzeigen lassen. **Achtung!** Unter Windows muss der Slash \ noch in den Backslash / gedreht werden.

! Unterschied zwischen \ in Windows und / in R

Achte einmal auf den Slash im Pfad in R und einem im Pfsd in Windows. Einmal ist es der Slash \ im Dateipfad und einmal der Backslash /. Das ist sehr ärgerlich, aber dieses Problem geht zurück in die 80'ziger. Bill hat entschieden für sein Windows / zu nutzen und Steve (und Unix) eben /. Und mit dieser Entscheidung müssen wir jetzt leben...

7.4. Auf ein englisches Wort in Dateien

Ein großes Problem in Datein sind Umlaute (ä,ö,ü) oder aber andere (Sonder)zeichen (ß, ?, oder #). Als dies sollte vermieden werden. Eine gute Datei für R beinhaltet nur *ganze* Wörter, Zahlen oder aber leere Felder. Ein leeres Feld ist ein fehlender Wert. Abbildung 7.3 zeigt eine gute Exceldatentablle. Wir schreiben `jump_length` mit Unterstrich um den Namen besser zu lesen zu können. Sonst ist auch alles in Englisch geschrieben. Wir vermeiden durch die neglische Schreibweise *aus ver-*

sehen einen Umlaut oder anderweitig problematische Zeichen zu verwenden. Später können wir alles noch für Abbildungen anpassen.

	A	B	C	D	E
1	animal	jump_length	flea_count	grade	infected
2	dog	5,2	18	8	0
3	dog	4,9	22	7	1
4	dog	12,1	17	5	1
5	dog	8,2	12	6	0
6	dog	5,6	23	7	1
7	dog	9,1	18	7	0
8	dog	7,4	21	9	0
9	cat	3,2	12	9	1
10	cat	1,2	13	5	0
11	cat	6,6	11	7	0
12	cat	4,1	12	8	0
13	cat	4,3	16	6	1
14	cat	7,8	9	6	0
15	cat	6,2	7	8	0

Abbildung 7.3.: Beispiel für eine gute (Excel)Datentabelle. Keine Umlaute sind vorhanden und die Spaltennamen haben keine Leerzeichen oder Sonderzeichen.

7.5. Spaltennamen in der (Excel)-Datei

Die Funktion `clean_names()` aus dem R Paket `janitor` erlaubt es die Spaltennamen einer eingelesenen Datei in eine für R gute Form zu bringen.

- Keine Leerzeichen in den Spaltennamen.
- Alle Spaltennamen sind klein geschrieben.

```
data_tbl %>%
  clean_names()
```

```
# A tibble: 14 x 5
   animal jump_length flea_count grade infected
   <chr>        <dbl>      <dbl> <dbl>    <dbl>
 1 dog            5.7         18     8        0
 2 dog            8.9         22     7        1
 3 dog           11.8         17     5        1
 4 dog            8.2         12     6        0
 5 dog            5.6         23     7        1
 6 dog            9.1         18     7        0
 7 dog            7.6         21     9        0
 8 cat            3.2         12     9        1
 9 cat            2.2         13     5        0
10 cat            5.4         11     7        0
11 cat            4.1         12     8        0
12 cat            4.3         16     6        1
13 cat            7.9          9     6        0
14 cat            6.1          7     8        0
```

7.5.1. Datenbeispiel

7.6. 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
```

7.7. 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
```

```
data_tbl <- tibble(treatment = gl(4, 4, labels = c("A", "B", "C", "D")),
                   block = rep(1:4, 4),
```

```
                    rep_1 = round(rnorm(16, 10, 4), 2),
                    rep_2 = round(rnorm(16, 10, 4), 2),
                    rep_3 = round(rnorm(16, 10, 4), 2),
                    rep_4 = round(rnorm(16, 10, 4), 2),
                    rep_5 = round(rnorm(16, 10, 4), 2),
                    rep_6 = round(rnorm(16, 10, 4), 2),
                    rep_7 = round(rnorm(16, 10, 4), 2),
                    rep_8 = round(rnorm(16, 10, 4), 2),
                    rep_9 = round(rnorm(16, 10, 4), 2))

  data_tbl %>%
    gather(rep, value, rep_1:rep_9) %>%
    arrange(treatment, block)
```

```
# A tibble: 144 x 4
   treatment block rep   value
   <fct>     <int> <chr> <dbl>
 1 A             1 rep_1  6.74
 2 A             1 rep_2  3.93
 3 A             1 rep_3  7.72
 4 A             1 rep_4  9.22
 5 A             1 rep_5  9.89
 6 A             1 rep_6 10.2
 7 A             1 rep_7 14.1
 8 A             1 rep_8  9.22
 9 A             1 rep_9 16.2
10 A             2 rep_1 10.5
# ... with 134 more rows
```

8. 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.

> **!** 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 4.2 mit Daten zu verschiedenen Oucomes, wie Sprungweite [cm], Anzahl an Flöhen auf Hunden und Katzen, die Boniturnoten oder aber den Infektionsstatus. Die Tabelle 4.2 ist zwar nicht groß aber auch nicht wirklich klein. Im nächsten Kapitel wollen wir uns damit beschäftigen, die Zahlen in der Tabelle sinnvoll zusammenzufassen.

8.1. Palmer Penguins

Palmer Penguins

Teil III.

Explorative Datenanalyse

Im vorherigen Kapitel haben wir die Datentabelle Tabelle 4.2 erschaffen. Bevor wir uns weiter mit statistischen Kennzahlen beschäftigen, wollen wir uns einmal die Realisierung der Tabelle Tabelle 4.2 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.

9. Formeln

9.1. Effektschätzer

9.1.1. Unterschied zweier Mittelwerte

Wir berechnen zwei Mittelwerte $\bar{y}_1$ und $\bar{y}_2$. Wenn wir wissen wollen wie groß der Effekt zwischen den beiden Mittelwerten ist, dann bilden wir die Differenz. Wir berechnen das Δ von y_1 und y_2 indem wir die Diffenz bilden.

$$\Delta_{y_1-y_2} = \bar{y}_1 - \bar{y}_2$$

Wenn es keinen Unterschied zwischen den beiden Mittelwerten $\bar{y}_1$ und $\bar{y}_2$ gibt, dann ist die Differenz $\Delta_{y_1-y_2} = \bar{y}_1 - \bar{y}_2$ gleich 0.

$$H_0 : \Delta_{y_1-y_2} = \bar{y}_1 - \bar{y}_2 = 0$$

In Tabelle 9.1 ist eine Datenbeispiel gegeben.

Tabelle 9.1.: Beispiel für die Berechnung von einem Mittelwertseffekt an der Sprunglänge [cm] von Hunde und Katzenflöhen.

animal	jump_length
cat	8.0
cat	7.9
cat	8.3
cat	9.1
dog	8.0
dog	7.8
dog	9.2

animal	jump_length
dog	7.7

Nehmen wir an, wir berechnen für die Sprungweite [cm] der Hundeflöhe einen Mittelwert von $\bar{y}_{dog} = 8.2$ und für die Sprungweite [cm] der Katzenflöhe einen Mittelwert von $\bar{y}_{cat} = 8.3$. Wie große ist nun der Effekt? Oder anders gesprochen, welchen Unterschied in der Sprungweite macht es aus ein Hund oder eine Katze zu sein? Was ist also der Effekt von `animal`? Wir rechnen $\bar{y}_{dog} - \bar{y}_{cat} = 8.2 - 8.3 = -0.1$. Zum einen wissen wir jetzt "die Richtung". Da wir ein Minus vor dem Mittelwertsunterschied haben, müssen die Katzenflöhe weiter springen als die Hundeflöhe, nämlich 0,1cm. Dennoch ist der Effekt sehr klein.

9.1.2. Unterschied zweier Anteile

Neben den Unterschied zweier Mittelwerte ist auch häufig das Interesse an dem Unterschied zwischen zwei Wahrscheinlichkeiten oder auch Anteilen. Ebenso kann die Chance berechnet werden. Hier tritt häufig Verwirrung auf, daher hier zuerst ein Beispiel.

Tabelle 9.2.: Eine 2x2 Tabelle als Beispiel für unterschiedliche Flohinfektionen bei Hunden und Katzen für die Berechnung von Effektschätzern eines Anteils.

		Infected	
		Yes (1)	*No (0)*
Animal	*Dog*	23_a	10_b
	Cat	18_c	14_d

Aus der Tabelle 9.2 können wir entnehmen, dass 23 Hunde mit Flöhen infiziert sind und 10 Hunde keine Infektion aufweisen. Bei den Katzen haben wir 18 infizierte und 14 gesunde Tiere beobachtet. Wir können nun zwei Arten von Anteilen berechnen. Das bekanntere ist die Frequenz oder Wahrscheinlichkeit oder Risk Ratio (RR). Das andere ist das Chancenverhältnis oder Odds Ratio (OR). Beide kommen in der Statistik vor und

sind unterschiedlich zu interpretieren.

9.1.2.1. Wahrscheinlichkeitsverhältnis oder Risk Ratio (RR)

$$Pr(\text{dog}|\text{infected}) = \frac{a}{a+c} = \frac{23}{23+10} \approx 0.67$$

$$Pr(\text{cat}|\text{infected}) = \frac{b}{b+d} = \frac{18}{18+14} \approx 0.56$$

$$\Delta_{y_1/y_2} = RR = \frac{Pr(\text{dog}|\text{infected})}{Pr(\text{cat}|\text{infected})} = \frac{0.67}{0.56} \approx 1.2$$

9.1.2.2. Chancenverhältnis oder Odds Ratio (OR)

$$Odds(\text{dog}|\text{infected}) = a : b = 23 : 10 = \frac{23}{10} = 2.3$$

$$Odds(\text{cat}|\text{infected}) = c : d = 18 : 14 = \frac{18}{14} \approx 1.29$$

$$\Delta_{y_1/y_2} = OR = \frac{Odds(\text{dog}|\text{infected})}{Odds(\text{cat}|\text{infected})} = \frac{a \cdot d}{b \cdot c} = \frac{2.30}{1.29} \approx 1.78$$

Wann liegt nun kein Effekt bei einem Anteil wie dem RR oder OR vor? Wenn der Anteil in der einen Gruppe genauso groß ist wie der Anteil der anderen Gruppe. Dies gilt sowohl fürdas RR als auch das OR.

$$H_0 : RR = \frac{Pr(\text{dog}|\text{infected})}{Pr(\text{cat}|\text{infected})} = 1$$

$$H_0 : OR = \frac{Odds(\text{dog}|\text{infected})}{Odds(\text{cat}|\text{infected})} = 1$$

! Stärke eines Effektes

Du musst immer den Effekt, hier den Mittelwertsunterschied, im Kontext der Fragestellung bzw. des Outcomes y bewerten. Der numerische Unterschied von 0,1cm kann in einem Kontext viel sein. Das Wachstum von Bakterienkolonien kann ein Unterschied von 0,1cm sehr viel sein. Oder aber sehr wenig, wenn wir uns das Wachstum von Bambus pro Tag anschauen. Hier bist du gefragt, den Effekt in den Kontext richtig einzuordnen. Ebenso stellt sich die Frage, ob ein Unterschied von 6% viel oder wenig ist...

Effektschätzer

Wenn wir uns einen Unterschied eines **Mittelwerts** anschauen, dann haben wir *keinen* Effekt vorliegen, wenn das Δ zwischen A und B gleich 0 ist. Die Nullhypothese gilt.

$$\Delta_{A-B} = A - B = 0$$

Wenn wir uns einen Unterschied eines **Anteils** anschauen, dann haben wir *keinen* Effekt vorliegen, wenn das Δ zwischen A und B gleich 1 ist. Die Nullhypothese gilt.

$$\Delta_{A/B} = \frac{A}{B} = 1$$

Dieses Wissen brauchen wir um später die Signifikanzschwelle bei einem 95% Konfidenzintervall richtig zu setzen und interpretieren zu können.

9.2. Diagnostisches Testen

Tabelle 9.3.: Eine 2x2 Tabelle oder Vierfeldertafel

		Infected		
		Yes (1)	*No (0)*	
Test	*+ (1)*	a	b	**a + b**
	- (0)	c	d	**c + d**

Tabelle 9.3.: Eine 2x2 Tabelle oder Vierfeldertafel

	a + c	**b + d**	**n**

Das ist ein Text Tabelle 9.3

9.3. Formelsammlung

Tabelle 9.4.: test

	Teststatistik T_{calc}	**p-Wert** $Pr(\geq T_{\alpha}\|H_0)$	**95% Konfidenzintervall** $\text{KI}_{1-\alpha}$
H_0 ablehnen	$T_{calc} \geq T_{\alpha=5\%}$	$Pr(\geq T_{\alpha}\|H_0) \leq \alpha$	Bei Δ_{A-B}: enthält *nicht* 0 **oder** bei $\Delta_{A/B}$: enthält *nicht* 1

Das ist ein Text Tabelle 9.4

! Entscheidung mit dem p-Wert

Wenn der p-Wert $\leq \alpha$ dann wird die Nullhypothese (H_0) abgelehnt. Das Signifikanzniveau α wird als Kulturkonstante auf 5% oder 0.05 gesetzt. Die Nullhypothese (H_0) kann auch Gleichheitshypothese gesehen werden. Wenn die H_0 gilt, liegt kein Unterschied zwischen z.B. den Behandlungen vor.

! Entscheidung mit der berechneten Teststatistik

Bei der Entscheidung mit der Teststatistik müssen wir zwei Fälle unterschieden.

(1) Bei einem t-Test und einem $\mathcal{X}^2$-Test gilt, wenn $T_{calc} \geq T_{\alpha=5\%}$ wird die Nullhypothese (H_0) abgelehnt.

(2) Bei einem Wilcoxon-Mann-Whitney-Test gilt, wenn $T_{calc} < T_{\alpha=5\%}$ wird die Nullhypothese (H_0) abgelehnt.

Achtung – Wir nutzen die Entscheidung mit der Teststatistik nur und ausschließlich in der Klausur. In der praktischen Anwendung hat die Bertrachtung der berechneten Teststatistik *keine* Verwendung mehr.

! Entscheidung mit dem 95% Konfidenzintervall

Bei der Entscheidung mit dem 95% Konfidenzinterval müssen wir zwei Fälle unterscheiden.

(1) Entweder schauen wir uns einen Mittelwertsunterschied (Δ_{A-B}) an, dann können wir die Nullhypothese (H_0) *nicht* ablehnen, wenn die 0 im 95% Konfidenzinterval ist.

(2) Oder wir schauen uns einen Anteilsunterschied ($\Delta_{A/B}$) an, dann können wir die Nullhypothese (H_0) *nicht* ablehnen, wenn die 1 im 95% Konfidenzinterval ist.

9.4. $\mathcal{X}^2$-Test

Tabelle 9.5.: Eine 2x2 Tabelle als Beispiel für unterschiedliche Flohinfektionen bei Hunden und Katzen für die Berechnung von Effektschätzern eines Anteils.

		Infected		
		Yes (1)	*No (0)*	
Animal	*Dog*	$23_{\ a}$	$10_{\ b}$	$\mathbf{a+b=33}$
	Cat	$18_{\ c}$	$14_{\ d}$	$\mathbf{c+d=32}$
		$\mathbf{a+c=41}$	$\mathbf{b+d=24}$	$n=65$

Text Tabelle 9.5

Tabelle 9.6.: Eine 2x2 Tabelle als Beispiel für unterschiedliche Flohinfektionen bei Hunden und Katzen für die Berechnung von Effektschätzern eines Anteils.

		Infected		
		Yes (1)	*No (0)*	
Animal	*Dog*	$\frac{41 \cdot 33}{65} = 20.82$	$\frac{24 \cdot 33}{65} = 12.18$	**33**
	Cat	$\frac{41 \cdot 32}{65} = 20.18$	$\frac{24 \cdot 32}{65} = 11.82$	**32**
		41	**24**	$n = 65$

Text Tabelle 9.6

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

$$\chi^2 = \frac{(23-20.82)^2}{20.82} + \frac{(10-12.18)^2}{12.18} + \frac{(18-20.18)^2}{20.18} + \frac{(14-11.82)^2}{11.82} = 1.25$$

Test

```
mat <- matrix(c(23, 10, 18, 14), byrow = TRUE, nrow = 2)
chisq.test(mat, correct = FALSE)
```

```
    Pearson's Chi-squared test

data:  mat
X-squared = 1.26134, df = 1, p-value = 0.2614
```

$\chi^2_{\alpha=5\%} = 3.84$

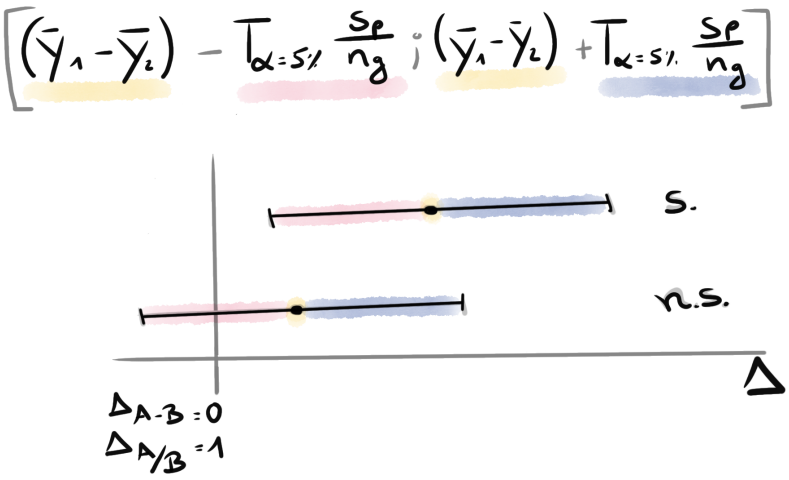


Abbildung 9.1.: kj

9.5. Konfidenzintervalle

Abbildung 9.1

- $(\bar{y}_1 - \bar{y}_2)$ ist der Effekt. In diesem Fall der Mittelwertsunterschied. Wir finden den Effekt als Punkt in der Mitte des Intervals.
- $T_{\alpha=5\%} \cdot \frac{s_p}{\sqrt{n_g}}$ ist ein fester Wert, der die Arme des Intervals bildet. Wir vereinfachen die Formel mit s_p für die gepoolte Standardabweichung und n_g für die Fallzahl der beiden Gruppen. Wir nehmen an das beide Gruppen die gleiche Fallzahl $n_1 = n_2$ haben.

Abbildung 9.2

(a) Nicht signifikant und nicht relevant
(b) Signifikant und nicht relevant
(c) Signifikant und relevant
(d) Signifikant und nicht relevant

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

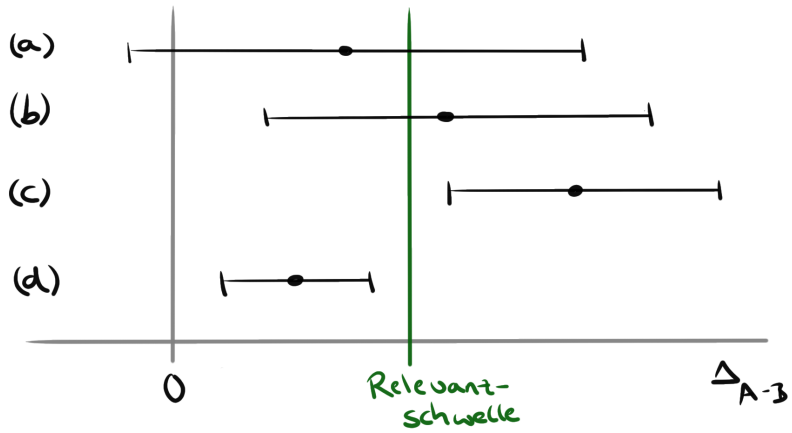


Abbildung 9.2.: kjasdsa

9.6. Testverteilung

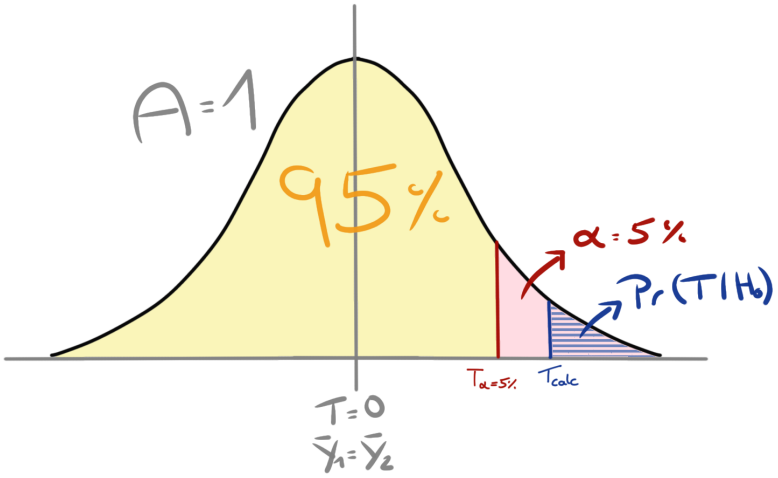


Abbildung 9.3.: kjk

Abbildung 9.3

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)
```

10.0.1. 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
```

10.0.2. Spannweite oder range

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

```
range(y)
```

```
[1]  4.9 12.1
```

10.0.3. 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
```

10.0.4. Standardabweichung

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

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

```
[1] 2.58
```

10.0.5. 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
```

10.0.6. Median

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

```
median(y)
```

```
[1] 7.4
```

10.0.7. 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.

10.0.8. Interquartilesabstand (IQR)

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

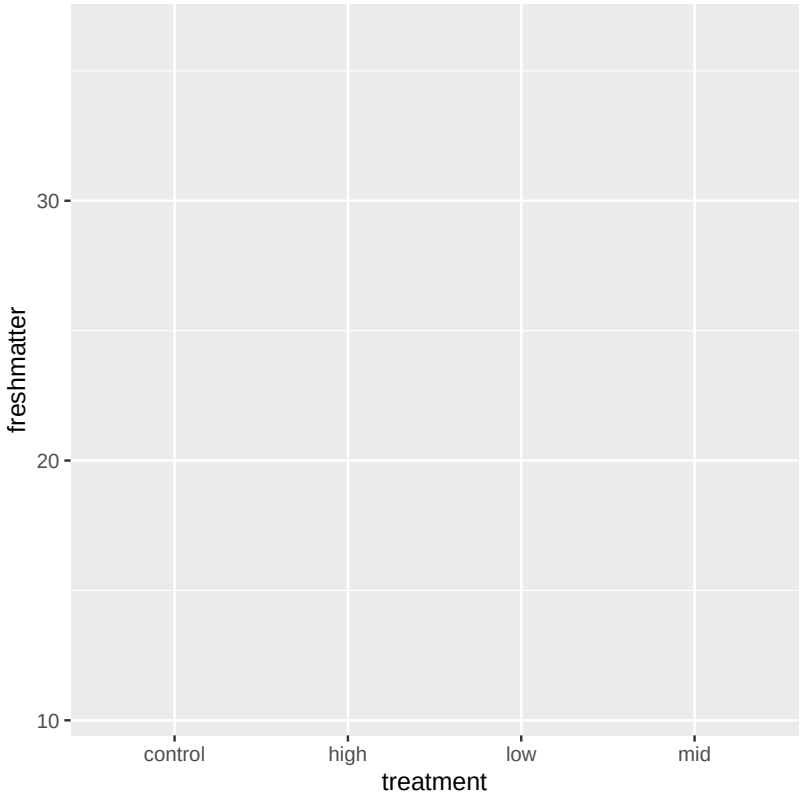
10.1. Zusammenfassen von Daten per Faktor

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

```
# A tibble: 3 x 3
  animal     `mean(jump_length)` `sd(jump_length)`
  <fct>                    <dbl>             <dbl>
1 flea_count              15.1               4.95
2 grade                    7                 1.30
3 infected                 0.357             0.497
```

11. Grundlagen in ggplot()

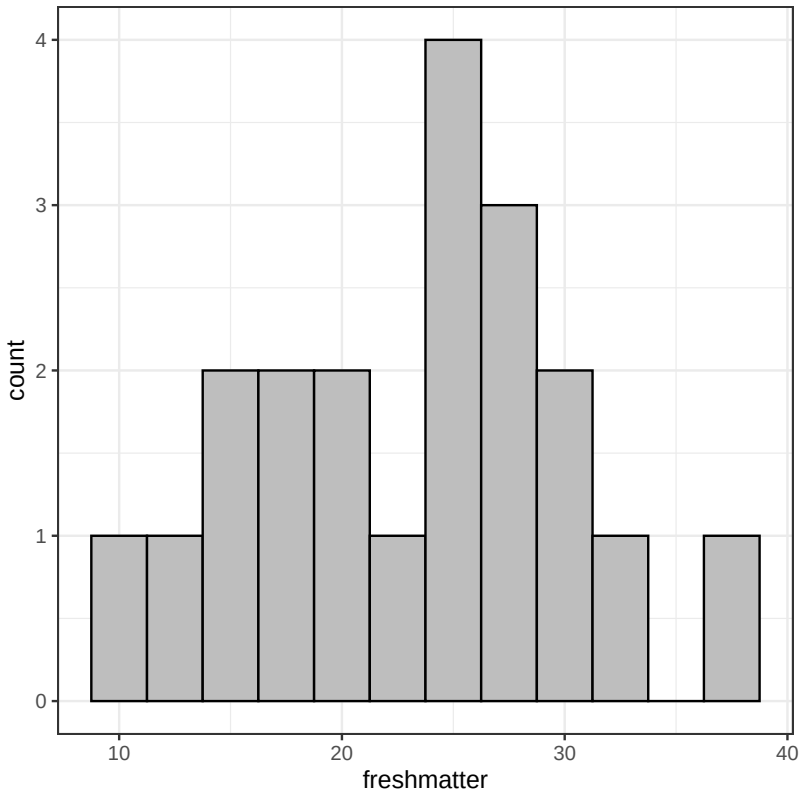
```r
data_tbl <- read_excel(file.path("data/germination_data.xlsx"))

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



11.1. Häufig verwendete Abbildungen

11.1.1. Histogramm

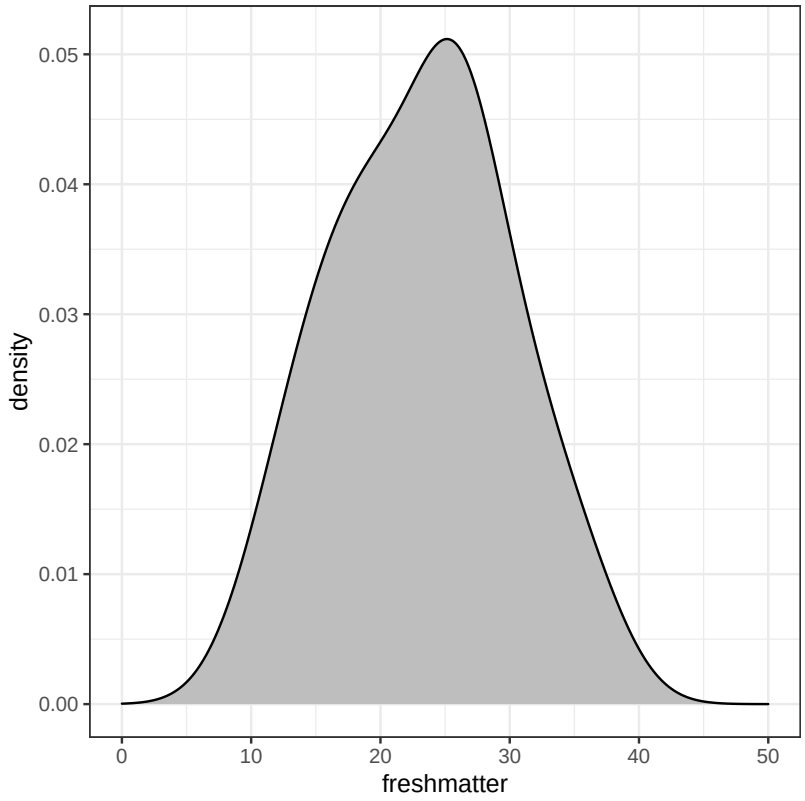
```
ggplot(data = data_tbl, aes(x = freshmatter)) +
  geom_histogram(binwidth = 2.5, fill = "gray", color = "black") +
  theme_bw()
```



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

11.1.2. Density Plot

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



Wenn wir viele Beobachtungen.

11.1.3. Boxplot

Abbildung 11.1

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

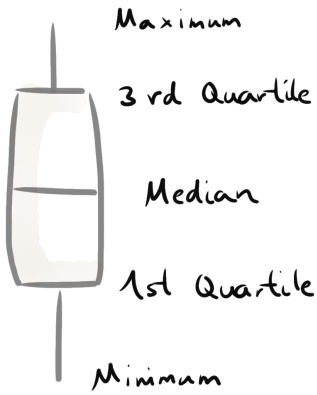
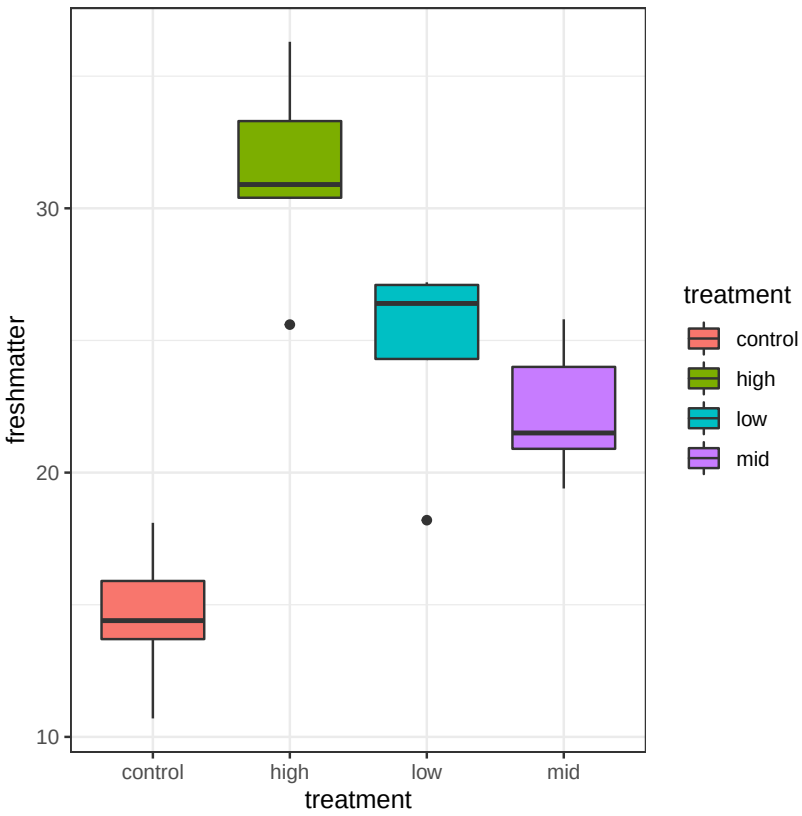
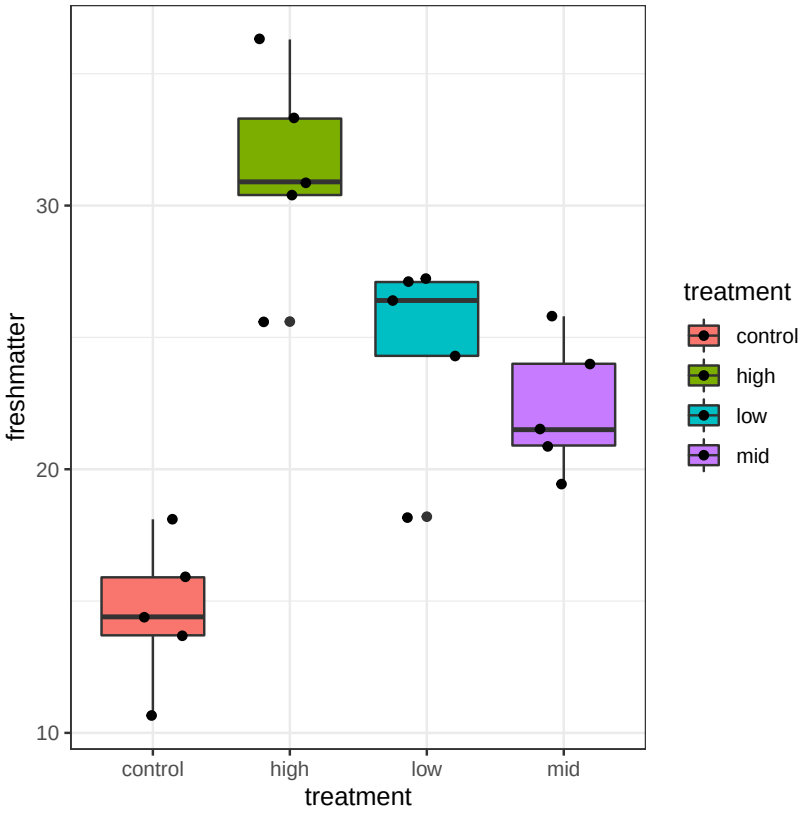


Abbildung 11.1.: kjasdsaddssd

```
  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.

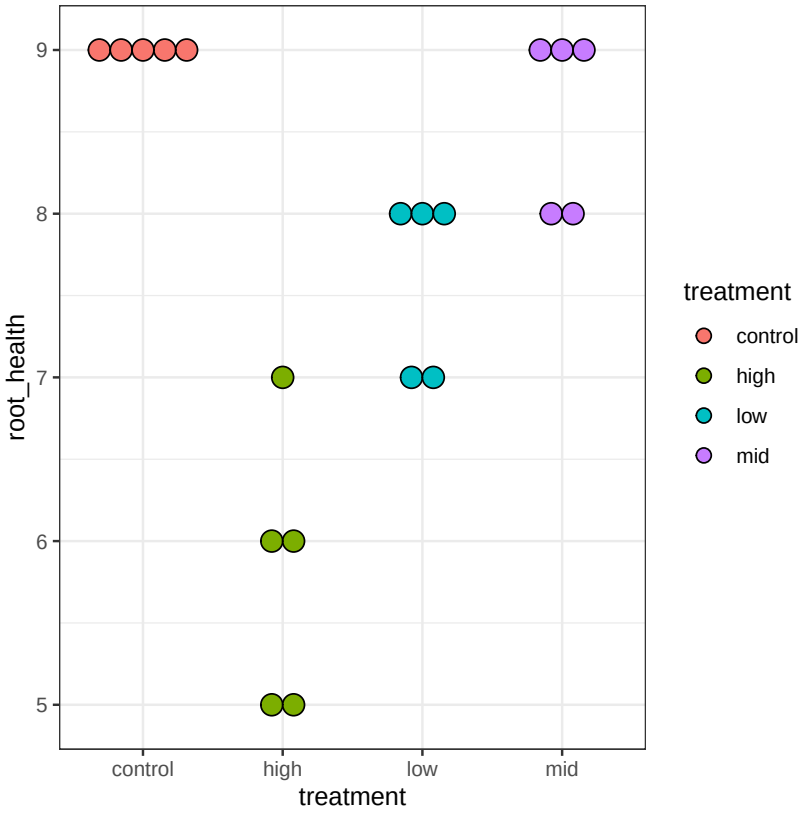
11.1.4. Dotplot

Wenn wir ganz wenige Beobachtungen haben.

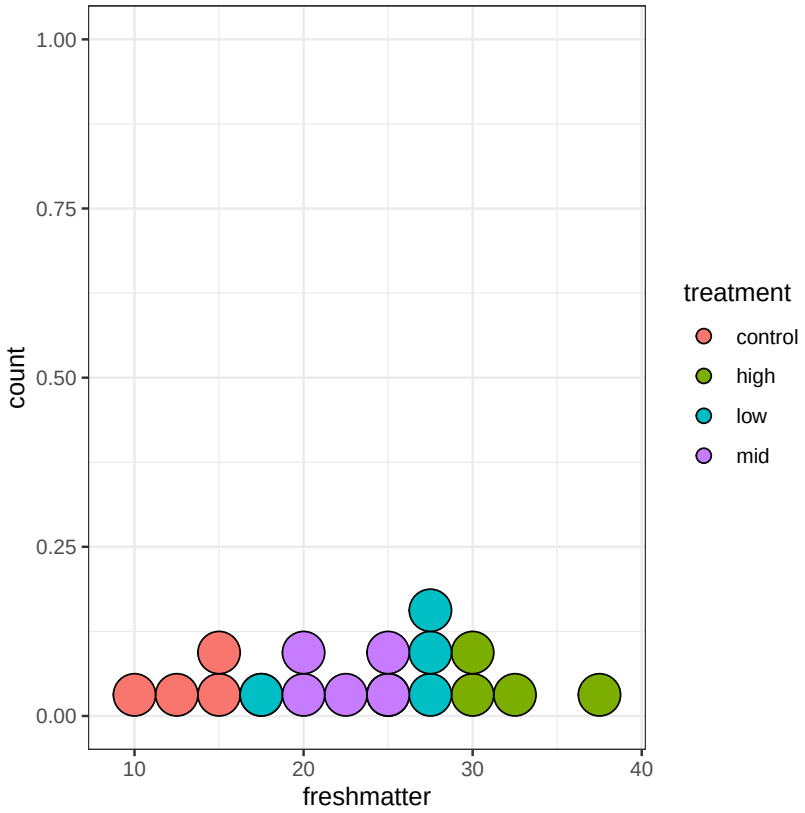
```
ggplot(data = data_tbl, aes(x = treatment, y = root_health,
                            fill = treatment)) +
  geom_dotplot(binaxis = "y", stackdir = "center") +
```

```
  theme_bw()
```

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

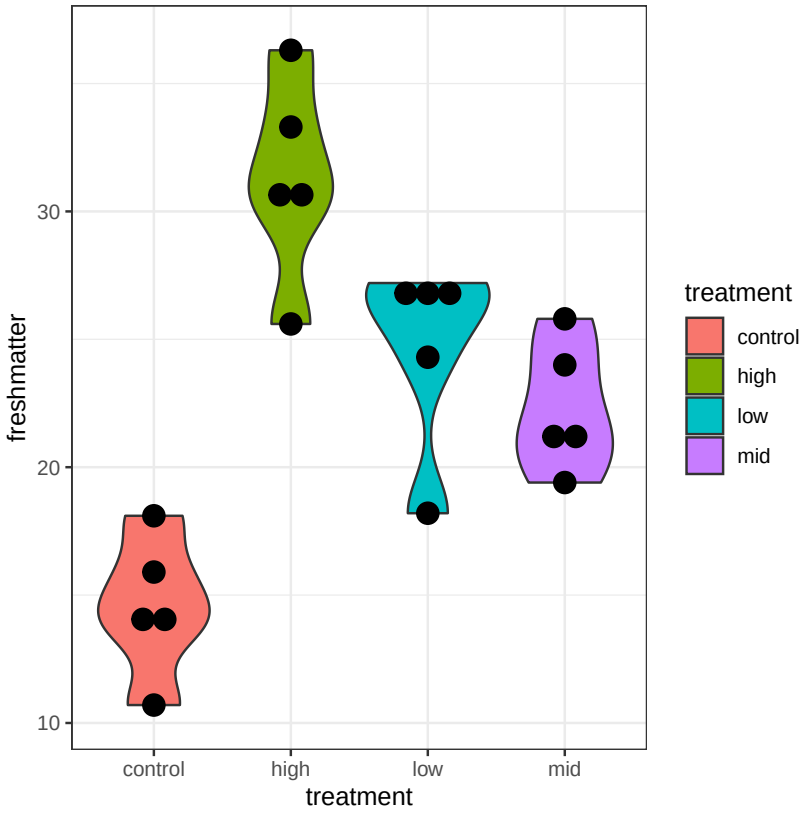



```
ggplot(data = data_tbl, aes(x = freshmatter, fill = treatment)) +
  geom_dotplot(method="histodot", binwidth = 2.5) +
  theme_bw()
```



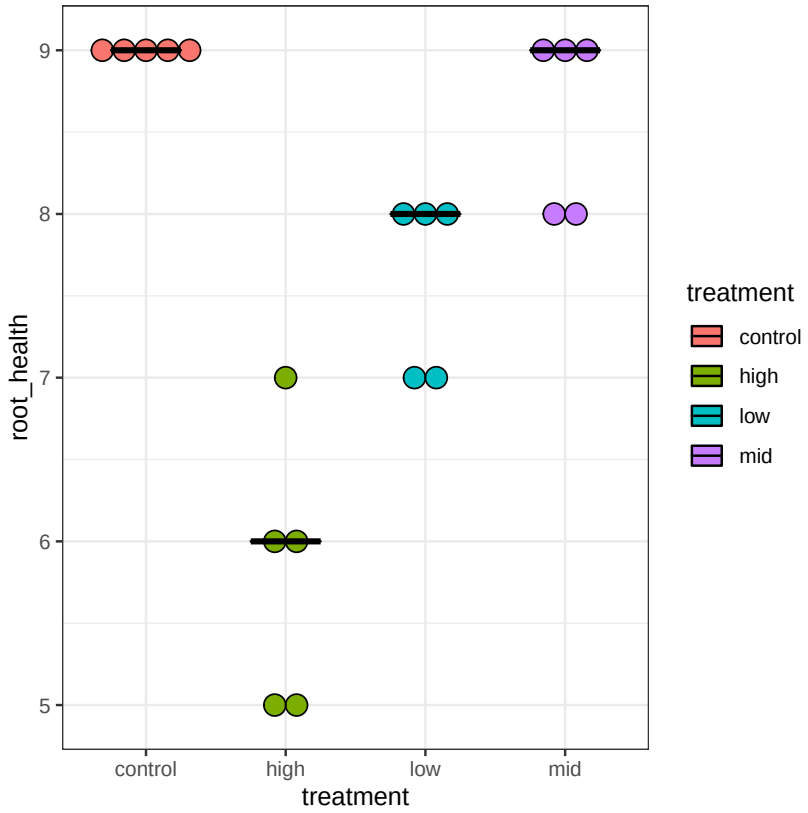

```
ggplot(data = data_tbl, aes(x = treatment, y = freshmatter,
                            fill = treatment)) +
  geom_violin() +
  geom_dotplot(binaxis = "y", stackdir = "center", fill = "black") +
  theme_bw()
```

```
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()
```

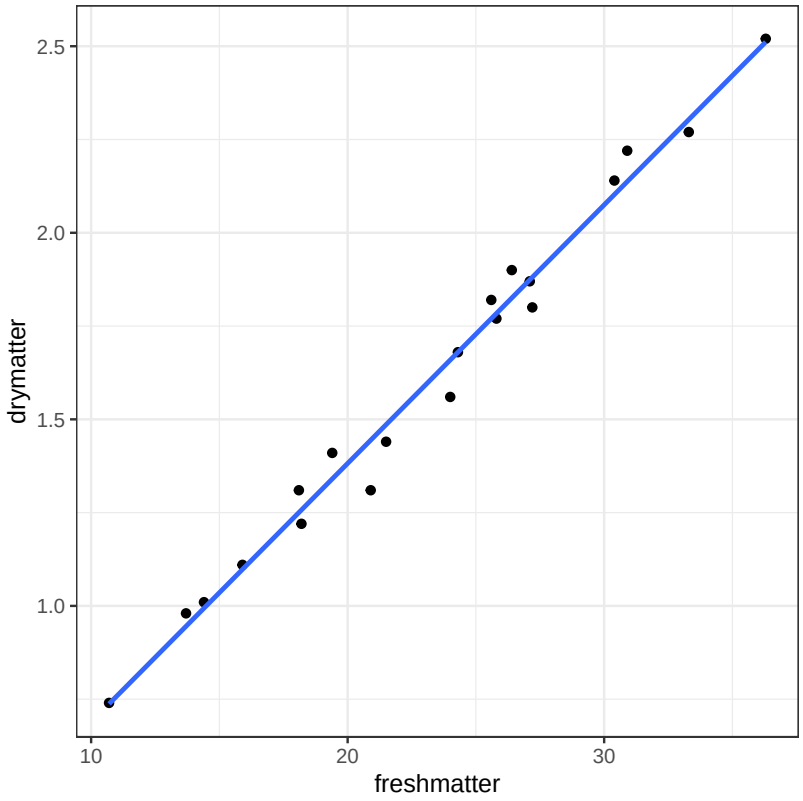
```
Bin width defaults to 1/30 of the range of the data. Pick better value with `binwidth`.
```



11.1.5. Scatterplot

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

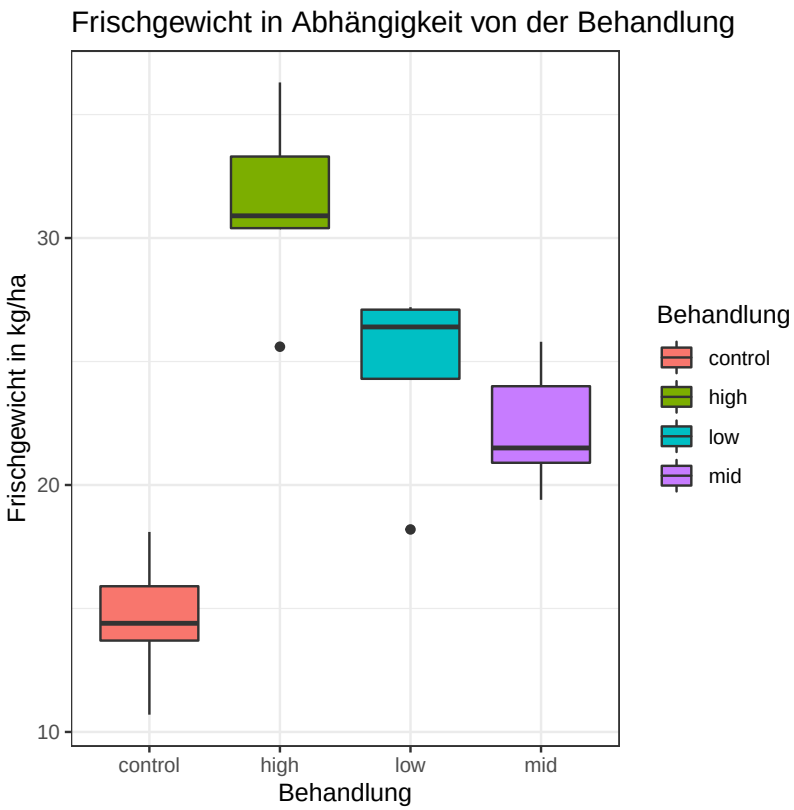
```
`geom_smooth()` using formula 'y ~ x'
```



11.1.6. Mosaic Plot

11.1.7. 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()
```


Frischgewicht in Abhängigkeit von der Behandlung
Frischgewicht in kg/ha
30
20
10
control
high
low
mid
Behandlung
Behandlung
control
high
low
mid

Teil IV.

Übersicht statistische Tests

Der t-Test

Kapitel 12

Was macht der t-Test?

Der t-Test vergleicht die Parameter zweier Normalverteilungen miteinander.
Die Parameter einer Normalverteilung sind der Mittelwert und die Standardabweichung.
$\mathcal{N}(0, 1)$

Einführung in den t-Test 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.

Die ANOVA

Was macht die ANOVA?

Die ANOVA vergleicht die Parameter mehrerer Normalverteilungen miteinander.
Die Parameter einer Normalverteilung sind der Mittelwert und die Standardabweichung.
$\mathcal{N}(0, 1)$

Einführung in die ANOVA 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.

Der Wilcoxon-Mann-Whitney-Test

Was macht der Wilcoxon-Mann-Whitney-Test?

Der Wilcoxon-Mann-Whitney-Test vergleicht die Mediane zweier beliebiger Verteilungen miteinander.

Einführung in den Wilcoxon-Mann-Whitney-Test 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.

Der Kruskal-Wallis-Test

Was macht der Kruskal-Wallis-Test?

Der Kruskal-Wallis-Test vergleicht die Mediane mehrerer beliebiger Verteilungen miteinander.

Einführung in den Kruskal-Wallis-Test 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.

Lineare Regression

Was macht die lineare Regression?

Eine Regression ist der Mittelwert als Linie durch eine Punktwolke.

Einführung in die lineare Regression 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.

Der $\mathcal{X}^2$-Test

Was macht der $\mathcal{X}^2$-Test?

Ein $\mathcal{X}^2$-Test vergleicht die Anteile zweier Gruppen.

Einführung in den $\mathcal{X}^2$-Test 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.

12. Der t-Test

12.1. Genutzte R Pakete für das Kapitel

Wir wollen folgende R Pakete in diesem Kapitel nutzen.

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

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

12.2. Die Wichtigkeit des t-Tests

$$\text{Teststatistik} = \frac{\text{Signal}}{\text{Noise}}$$

Tabelle 12.1.: test caption

animal	jump_length	flea_count	grade	infected
dog	5.7	18	8	0
dog	8.9	22	7	1
dog	11.8	17	5	1
dog	8.2	12	6	0
dog	5.6	23	7	1
dog	9.1	18	7	0
dog	7.6	21	9	0
cat	3.2	12	9	1
cat	2.2	13	5	0
cat	5.4	11	7	0
cat	4.1	12	8	0
cat	4.3	16	6	1
cat	7.9	9	6	0

animal	jump_length	flea_count	grade	infected
cat	6.1	7	8	0

Beispieldaten sind in Tabelle 12.1 abgebildet.

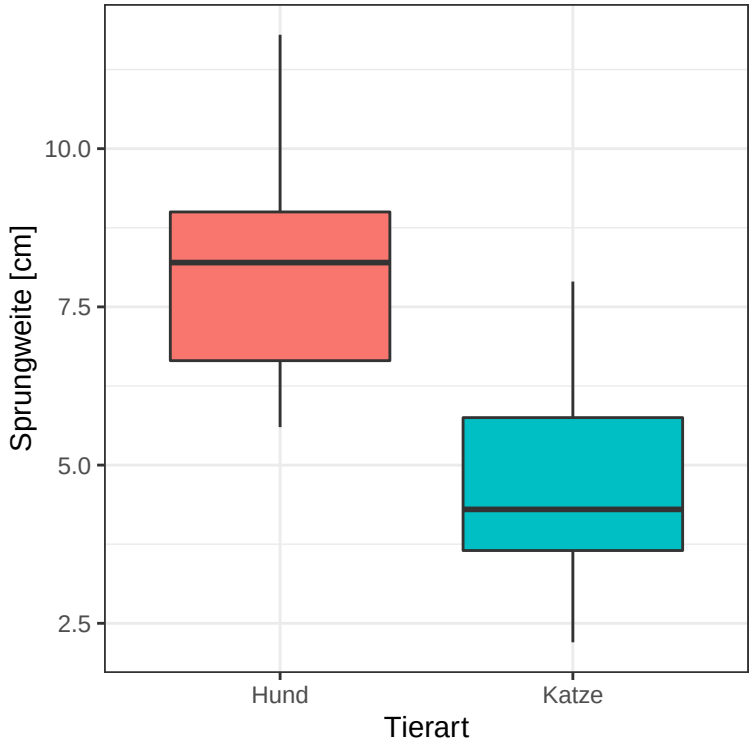


Abbildung 12.1.: Boxplot der Sprungweiten [cm] von Hunden und Katzen.

Das ist das Beispiel Abbildung 12.1

Das ist das Beispiel Abbildung 12.2

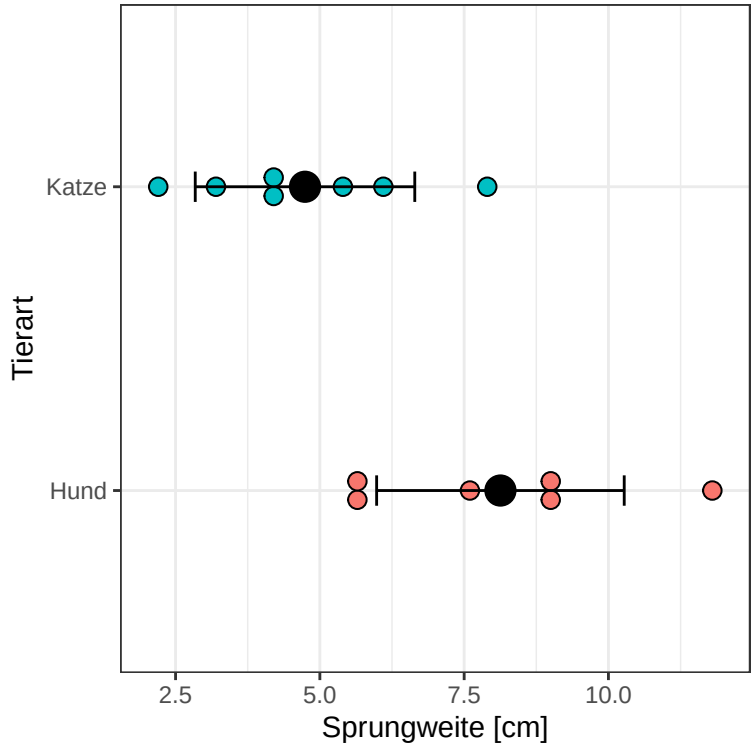


Abbildung 12.2.: Dotplot der Sprungweiten [cm] von Hunden und Katzen.

12.3. Student t-Test

```
sum_tbl <- data_tbl %>%
  group_by(animal) %>%
  summarise(mean = round(mean(jump_length), 2), sd = round(sd(jump_length), 2))

sd_pool <- (sum_tbl$sd[1] + sum_tbl$sd[2])/2
t_student <- round((sum_tbl$mean[1] - sum_tbl$mean[2])/(sd_pool * sqrt(2/7)), 2)
```

$$T_{calc} = \frac{\bar{y}_1 - \bar{y}_2}{s_{pooled} \cdot \sqrt{\frac{2}{n_{group}}}}$$

Foo[2]

[2] So dann hier

$$s_{pooled} = \sqrt{\frac{1}{2}(s_{y_1}^2 + s_{y_2}^2)}$$

$$s_{pooled} = \frac{s_{y_1} + s_{y_2}}{2}$$

$$s_{pooled} = \frac{2.14 + 1.9}{2} = 2.02$$

$$T_{calc} = \frac{8.13 - 4.74}{2.02 \cdot \sqrt{\frac{2}{7}}} = 3.14$$

```
t.test(jump_length ~ animal,
       data = data_tbl, var.equal = TRUE)
```

```
    Two Sample t-test

data:  jump_length by animal
t = 3.12528, df = 12, p-value = 0.0087684
alternative hypothesis: true difference in means between group dog and group cat is not equal t
95 percent confidence interval:
 1.0253394 5.7460892
```

```
sample estimates:
mean in group dog mean in group cat
        8.1285714         4.7428571
```

```
t.test(jump_length ~ animal,
       data = data_tbl, var.equal = TRUE) %>%
  tidy()
```

```
# A tibble: 1 x 10
  estimate estimate1 estimate2 statistic p.value parameter conf.low conf.high
     <dbl>     <dbl>     <dbl>     <dbl>   <dbl>     <dbl>    <dbl>     <dbl>
1     3.39      8.13      4.74      3.13 0.00877        12     1.03      5.75
# ... with 2 more variables: method <chr>, alternative <chr>
```

12.4. Welch t-Test

$$T_{calc} = \frac{\bar{y_1} - \bar{y_2}}{\sqrt{\frac{s^2_{y_1}}{n} + \frac{s^2_{y_2}}{m}}}$$

Hier muss man noch bedenken, dass die Freiheitsgrade anders berechnte werden Die Freiheitsgrade werden mit[3]

[3]

$$df = \frac{\left(\frac{s^2_{y_1}}{n} + \frac{s^2_{y_2}}{m}\right)^2}{\frac{\left(\frac{s^2_{y_1}}{n}\right)^2}{n-1} + \frac{\left(\frac{s^2_{y_2}}{m}\right)^2}{m-1}}$$

```
t.test(jump_length ~ animal,
       data = data_tbl, var.equal = FALSE)
```

```
    Welch Two Sample t-test

data:  jump_length by animal
t = 3.12528, df = 11.8307, p-value = 0.008906
alternative hypothesis: true difference in means between group dog and group cat is not equal t
95 percent confidence interval:
 1.0215869 5.7498416
sample estimates:
mean in group dog mean in group cat
        8.1285714         4.7428571
```

```r
t.test(jump_length ~ animal,
       data = data_tbl, var.equal = FALSE) %>%
  tidy()
```

```
# A tibble: 1 x 10
  estimate estimate1 estimate2 statistic p.value parameter conf.low conf.high
     <dbl>     <dbl>     <dbl>     <dbl>   <dbl>     <dbl>    <dbl>     <dbl>
1     3.39      8.13      4.74      3.13 0.00891      11.8     1.02      5.75
# ... with 2 more variables: method <chr>, alternative <chr>
```

12.5. Verbundener t-Test (Paired t-Test)

$$T_{calc} = \sqrt{n}\frac{\bar{d}}{s_d}$$

```r
t.test(jump_length ~ animal,
       data = data_tbl, paired = TRUE)
```

```
    Paired t-test

data:  jump_length by animal
t = 3.76033, df = 6, p-value = 0.0093949
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 1.1825691 5.5888595
sample estimates:
mean difference
      3.3857143
```

```r
t.test(jump_length ~ animal,
       data = data_tbl, paired = TRUE) %>%
  tidy()
```

```
# A tibble: 1 x 8
  estimate statistic p.value parameter conf.low conf.high method     alternative
     <dbl>     <dbl>   <dbl>     <dbl>    <dbl>     <dbl> <chr>      <chr>
1     3.39      3.76 0.00939         6     1.18      5.59 Paired t-~ two.sided
```

12.6. Rest Formeln

$$Pr\left(\left| \frac{\bar{y}_1 - \bar{y}_2}{s_{y_1, y_2} \cdot \sqrt{2/n_g}} \right| \bar{y}_1 - \bar{y}_2 = 0 \right)$$

$$Pr\left(\left| \frac{\Delta_{y_1, y_2}}{s_{y_1, y_2} \cdot \sqrt{2/n_g}} \right| \Delta_{y_1, y_2} = 0 \right)$$

$$Pr\left(T_{\alpha=5\%} | \, \Delta_{y_1, y_2} = 0 \right)$$

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

$$\Pr(D|H_0)$$

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

$$\begin{aligned} H_0 &: \bar{y}_{dog} = \bar{y}_{cat} \\ H_A &: \bar{y}_{dog} \neq \bar{y}_{cat} \end{aligned}$$

Teil V.

Verteilungen

13. Die Normalverteilung

Die Normalverteilung

Bruce, Peter, Andrew Bruce, und Peter Gedeck. 2020. *Practical statistics for data scientists: 50+ essential concepts using R and Python*. O'Reilly Media.

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

Hurlbert, Stuart H. 1984. „Pseudoreplication and the design of ecological field experiments". *Ecological monographs* 54 (2): 187–211.

Wickham, Hadley, und Garrett Grolemund. 2016. *R for data science: import, tidy, transform, visualize, and model data*. O'Reilly Media, Inc.