1 Einlesen

1.1 read_csv - read_csv2

Erzeugt Tibble
read_csv: Komma zum Zeilentrennen und Punkt für Dezimalzahlen
read_csv2: Semicolon zum Zeilentrennen und Komma für Dezimalzahlen

```
read_csv(file = "", col_types = "")
read_csv2(file = "", col_types = "")
```

Mit col_types können wir als String wo jede Position für die jeweilige Spalten steht den Typ bestimmen
Typen:

- **c** = character
- **i** = integer
- **n** = number
- **d** = double
- **l** = logical
- **f** = factor
- **D** = date
- **T** = date time
- **t** = time
- **?** = guess
- **_ oder -** = skip

```
# Csv with ";" separator and "." as decimal point
read.csv("europe.data.csv", sep = ";", dec=".")

# Csv without first line as header
read.csv2("mpg.csv", header=F)

# Csv with 4 integer columns
read_csv2("magnets_pain.csv", col_types = "iiii")
```

1.2 read.csv - read.csv2

Erzeugt Dataframe

```
data <- read.csv(file="", sep = ";", dec = ".")
↪ %>% as_tibble()
```

- **sep** = Das Zeichen welches die Spalten trennt.
- **dec** = Gibt den trenner für Dezimalzahlen an

1.3 read_delim - read_delim2

Erzeugt Tibble
Der Rest wie bei _

```
read_delim(file = , delim = ,col_types = "")
```

Mit der Option **delim** können wir festlegen, mit welchem Zeichen die Zeilen getrennt werden.

1.4 Tidy Data

Typen:

- Jede Spalte muss eine Variable sein,
- Eine Observation ist eine Zeile,
- Eine Variable ist Z.b das Alter,
- Die einzelnen Daten sind die Beobachtungen

2 Tidying Data

2.1 pivot_longer()

Wir haben eine Tabelle wo es Spalten gibt die als Variablen selber Observationen haben. Wir wollen diese Observationen auch als Observationen hinschreiben

student <chr>	algebra <dbl>	analysis <dbl>	diskrete.math <dbl>
Adam	NA	2	3
Bernd	5	NA	NA
Cristian	3	1	2
Doris	4	3	4

Wir sehen, dass algebra etc eigentlich Observations sind.

```
student1 %>% pivot_longer(cols =
↪ algebra:diskrete.math, names_to = "classes",
↪ values_to = "grade", values_drop_na = T)
```

- **cols** = ein c() mit allen spalten oder spalte_1 : spalte_n.,
- **names_to** = In welche Spalte die Namens aus cols.,
- **values_to** = In Welche Spalte die Werte die in den Spalten aus cols waren,
- **values_drop_na** = T falls wir NAs droppen wollen

student <chr>	classes <chr>	grade <dbl>
Adam	algebra	NA
Adam	analysis	2
Adam	diskrete.math	3
Bernd	algebra	5
Bernd	analysis	NA
Bernd	diskrete.math	NA
Cristian	algebra	3

Hier haben wir jetzt in jeder Spalte eine Variable

2.2 pivot_longer

Wir haben in einer Spalte für jede Observation zwei Variablen und in einer anderen Spalte die Observation für jede Variable

name <chr>	type <chr>	measure <dbl>
Adam	height	1.83
Adam	weight	81.00
Bernd	height	1.75
Bernd	weight	71.00

Jede Variable in Type soll eine eigene Spalte bekommen

```
student2 %>% pivot_wider(names_from = type,
  values_from = measure)
```

- **names_from** = Die Spalte in der die Variablen stehen.,
- **values_from** = Die Spalte wo die Observationen drinne stehen.

name <chr>	height <dbl>	weight <dbl>
Adam	1.83	81
Bernd	1.75	71
Christian	1.69	55

Jetzt hat jede Variable eine Spalte

3 separate

Wir haben in einer Spalte Zwei Observations in einer Zelle

name <chr>	reatio <chr>
Adam	81/1.83
Bernd	71/1.75
Christian	55/1.69
Doris	62/1.57

Nun wollen wir diese Spalte aufteilen

```
student3 %>% separate(col = reatio, sep = "/",
  into = c("weight", " height"))
```

- **col** = Die Spalte in der die Observations sind.
- **sep** = Das Zeichen, welches die Observations trennt.
- **into** = Ein Array in welches die Observations nun geschrieben werden sollen.

name <chr>	weight <chr>	height <chr>
Adam	81	1.83
Bernd	71	1.75
Christian	55	1.69
Doris	62	1.57

4 Wichtige Befehle

4.1 Anfang

knitr::opts_chunk$set(echo = TRUE, error = T)

4.2 Librarys

```
library(tidyverse)
library(TeachingDemos)
```

4.3 Clean up code

- **STRG + i**

4.4 Datenerzeugung

- **sample** = Erzeugt ein sample aus den Werten in dem Array x mit der Länge size. Replace auf True wenn wir nur Unique werte aus x wollen.
- **runif** = Ein array mit länge n mit werten von min bis max

```
df <- tibble(
  id = 1:10,
  sex = sample(x = c("f", "m"), size = 10,
  replace = T),
  age = round(runif(n = 10, min = 10, max = 35)),
)
```

4.5 Tibble zeugs

4.5.1 Zeile Löschen

```
df %>% '['(-1,)
```

4.5.2 filtern

- **filter** = Kann ich nach Werten Filtern.
- **%in%** = falls ich mehrere sachen in einer variable filtern will

```
students %>% filter(sex == "m") %>% select(age)
flights %>%
  filter(carrier %in% c("AA", "DL"))
flights %>%
  filter(carrier == "AA" | carrier == "DL")
```

NA Werte filtern

- **!is.na(Zeile)** = Entfernt uns alle NA Werte

```
corona %>% filter(!is.na(new_cases))
```

4.5.3 select

- **select** = ich kann einzelne Spalten auswählen

```
students %>% select(age)
```

4.5.4 Zeile hinzufügen

- **add_row** = wir müssen jeden wert explizit angeben

```
students %>% add_row(id = 11, sex = "m",
age = 25, score1 = 4)
```

4.5.5 Werte in Kategorien

```
df <- df %>% mutate(
  grade = case_when(
    sum <= 37 ~ 5,
    sum > 37 & sum <= 45 ~ 4,
    sum > 35 & sum <= 55 ~ 3,
    sum > 55 & sum <= 65 ~ 2,
    sum > 65 ~ 1))
```

ifelse

```
ifelse(test = grade < 5, yes = 1,no = 0)
```

4.5.6 Werte sortieren

- **arrange** = Wir sortieren den Tibble nach der ausgewählten spalte. Wenn andersherum die Spalte mit desc umgeben

```
df %>% select(id, sex , grade) %>%
  arrange(sex) # oder arrange(desc(sex))
```

4.5.7 Summarise

- **summarise** = Erstellt eine Übersichtstabelle die nur die Spalten beschreibt die ich angebe, die spalte nach der ich groupe und die die ich anbgebe
- **quantile** - gibt mir die Quantile erst die Spalte dann welches quantil **Interquartile** = Q3 - Q1 v

```
df %>% group_by(sex) %>%
  summarise(mean_score = mean(sum),
            min_score = min(sum),
            max_score = max(sum),
            med_score = median(sum),
            Q1 = quantile(sum, 0.25),
            q1 = quantile(sum, 0.25),
            Q2 = quantile(sum, 0.5),
            Q3 = quantile(sum, 0.75))
```

trimmed mean

- **trim** - Wenn ich einen trimmes mean 40% habe, schneide ich jeweils 20% der kleinsten und grösten Werte weg

```
mean(observations, trim = 0.2)
```

4.5.8 Werte zählen

- **n** = um n() benutzen zu können, muss ich das in einem summarise benutzen
- **count** = count() kann ich immer benutzen, muss aber die spalte explizit angeben

```
er %>% group_by(group) %>%
  summarise(n = n())
er %>% count(group)
```

4.5.9 Rowwise

- **rowwise()** = Macht das selbe wie wenn ich ein group_by aber für jede zeile anwende

```
er %>% rowwise() %>%
  mutate(x = sum(ex1:ex5))
```

Ohne das rowwise würde ich die summer aller ex für alle Zeilen berechnen. Mit dem rowwise berechne ich jeweils die summe für jede zeile

4.5.10 Unique

- **unique** = Wenn ich alle Uniquen Zeilen Zeigen will. Gut kombinierbar mit select

4.5.11 Determine number and rates of Variables

4.5.12 Die Obersten n Werte

```
df %>% hat(10)
```

5 Diagramme uns so ein mist

5.1 plot - Standart ist scatterplot

Im Normalfall einfach x und y reinwerfen

- **type** - "l" gibt uns einen lineplot. in ?plot stehen die anderen optionen

```
plot(x = x_werte, y = y_werte, type = "l")
```

5.2 Boxplot

Ich habe hier ein Tibble mit 3 arrays mit Werten. Ich werfe diese in ein boxplot.

```
x <- tibble(on_player, beginner, tournament)
boxplot(x, ylab = 'beschreibung links')
boxplot(on_player, beginner, tournament,
names = c("a","b","c"))
```

5.2.1 symatrisch:

Wenn alles mittig ist

5.2.2 Skewness

Wenn der Rechte wisker länger ist: rigt skewed Wenn der Linke wisker länger ist: left skewed

5.2.3 Tilde - y ~ x

In y stehen die Werte Ich Ordne die Werte dem jeweilingen Typen in x zu

```
boxplot(werte ~ typen)
```

die linke variable ist in abhängigkeit der rechten variable

5.3 Kuchen chart

- **labels** - Die Beschreibung für jedes Kuchenstück
- **col** - Die Farben

```
pie(party$Results_2013,
    labels  = party$Party,
    col = party$colors)
```

5.4 Barplot

- **names.arg** - Name
- **col** - Farben :)

```
barplot(party$Results_2013,
        names.arg = results$Party,
        col = results$colors)
```

6 cumulative frequency distribution

selbe idee wie pnorm. Also h(700) sagt uns wieviele der werte unter 700 liegen

- **ecdf**

```
h <- ecdf(data)
plot(h)
```

6.0.1 less equal 800

```
h(800)
```

6.0.2 greater than 725

```
1 - h(725)
```

6.0.3 greater than 642 und less equal 777

```
h(777) - h(642)
```

6.0.4 equal 696

```
h(697) - h(695)
```

7 Linear Regression

7.1 Scatterplot

```
plot(x, y)
```

7.2 Covariance

Die Kovarianz misst den Grad, zu dem zwei Zufallsvariablen gemeinsam variieren

```
cov(x, y)
```

7.3 coefficient of correlation

Ist das selbe wie Covarianz aber genormt zeit wie stark zwei variablen zusammehängen bei 1 steigen sie perfekt zusammen bei -1 sinken sie perfekt zusammen 0 kein zusammenhang

```
cor(x, y)
```

7.4 coefficient of determination = Proportion of variation

Tells you how your model or line explains your data

- 1 -> your model explains 100% of your data
- 0.5 -> your model explains 50% of your data
- 0 -> your model is useless just like you

```
cor(x, y)^2
```

7.5 Regression line X Y

- Criterion is our Y
- predictor is our X
- a is our y achsenabschitt (intercept)
- b is our steigung (slope)

7.5.1 abline zeichnet unsere Linie

Wir müssen als erstes plotten **Im RMarkdown für abline den ganzen Codeblock ausführen**

```
plot(x, y)
model <- lm(y ~ x)
abline(model)
```

7.5.2 a und b bekommen

$y = a + b * x$

```
model <- lm(y ~ x)
a <- model$coefficients[1]
b <- model$coefficients[2]
```

7.5.3 Predict value for x = 8

```
model <- lm(y ~ x)
a <- model$coefficients[1]
b <- model$coefficients[2]
new_y <- a + b * 8
```

7.6 Regression line Y X

```
xy_model <- lm(x ~ y)
xy_a <- xy$coefficients[1]
xy_b <- xy$coefficients[2]
xya_new <- -(xy_a / xy_b)
xyb_new <- 1 / xy_b
abline(a = xya_new, b = xyb_new, col = "red")
```

8 Contingency table

40 | 10 | 50
20 | 10 | 30
10 | 10 | 20
70 | 30 | 100

8.1 Matrix bauen

Wir müssen nicht die gesamt felder betrachten

- **nrow** - wie viele Zeilen wir haben
- **ncol** - wie viele spalten wir haben Die daten werden automatisch eingeordnet

```
tab <- matrix(c(40, 10, 20, 10, 10, 10),
nrow = 3, ncol = 2, byrow = T)
```

8.2 Expected values

```
chisq.test(tab)$expected
```

8.3 X hoch 2

```
x2 <- chisq.test(tab)\$statistic
```

8.4 C

```
c <- sqrt((x2 / (x2 + sum(tab))))
```

8.5 C corr

0-0.3 keine assoziation 0.3-0.8 some kind of assoziation 0.8-0.1 strong assoziation

```
c_korr <- sqrt((min(length, height)/
    (min(length, height) -1)) * x2/(x2+sum(tab)))
```

9 R shinanigans

9.0.1 Ganzzahldivision

```
x %/% y
```

9.0.2 Modulo (x mod y)

```
x %% y
```

9.0.3 vim

imap jj <Esc>

10 Probability

10.1 Basic Rules

$P(A^c)$	$1 - P(A)$	Probability that A will not happen
$P(\emptyset)$	0	Probability of a null Event
$P(A \cap B)$	$P(A) * P(B)$ $P(A\|B) * P(B)$	Probability of A and B occurring
$P(A \setminus B)$	$P(A) - P(A \cap B)$	Probability of A without B
$P(A \cup B)$	$P(A) + P(B) - P(A \cap B)$	Probability of A or B occurring
$P(A\|B)$	$\frac{P(A \cap B)}{P(B)}$	Probability of A if B already happened

Table 1: A, B = Events; $P(x)$ = Probability of Event x

$A \subseteq B \implies P(A) \leq P(B)$

10.2 Crossproduct

$A \times B$

```r
omega <- expand.grid(x = 1:6, y = 1:6)
```

10.3 Union

$A \cup B$

```r
union(x = a, y = b)
```

10.4 Intersection

$A \cap B$

```r
intersect(x = a, y = b)
```

10.5 Difference

$A \setminus B$

```r
setdiff(x = a, y = b)
```

10.6 Examples

```r
# Cases where first die is 1
omega %>% filter(x = 1)
# Cases where sum of dice equals 7
omega %>% filter(x + y = 7)
# Probability of dice equals 12
count(omega %>% filter(x + y = 12)) / count(omega)
```

11 probability distributions functions

1. “d” returns the height of the probability density function
2. “p” returns the cumulative density function
3. “q” returns the inverse cumulative density function (quantiles)
4. “r” returns randomly generated numbers

dgenom

11.1 Normalverteilung

```
pnorm(q = , mean = , sd = )
pnorm(q = 1.96, mean = 0, sd = 1)

```

q = Der Wert, bis zu dem wir $P(X \leq q)$ berechnen.
Die Ausgabe ist die Wahrscheinlichkeit, dass $X \leq q$ gilt.
Beispiel: berechnet $P(X \leq 1.96)$ für die Standardnormalverteilung.)

```
dnorm(x = , mean = , sd = )
dnorm(x = 0, mean = 0, sd = 1)
```

x = Der Wert, an dem die Dichte berechnet wird.
Beispiel: gibt die Dichte der Standardnormalverteilung bei $x = 0$ zurück.)

```
qnorm(p = , mean = , sd = )
qnorm(p = 0.975, mean = 0, sd = 1)
```

p = Das Quantil, also der Wahrscheinlichkeitswert (zwischen 0 und 1).
Die Ausgabe ist der x-Wert, sodass $P(X \leq x) = p$ gilt.
Beispiel: liefert das 97,5%-Quantil der Standardnormalverteilung.)

```
rnorm(n = , mean = , sd = )
rnorm(n = 10, mean = 0, sd = 1)
```

n = Anzahl der zu erzeugenden Zufallswerte.
Die Ausgabe sind n Zufallswerte aus der Normalverteilung.
Beispiel: erzeugt 10 Zufallswerte aus einer Standardnormalverteilung.)

11.2 Binomialverteilung

size = number of trials (zero or more)
prob = probability of success on each trial.

```
pbinom(q = , size = , prob = )
pbinom(q = 5, size = 10, prob = 0.3)
```

q = Die Anzahl der Erfolge, bis zu der $P(X \leq q)$ berechnet wird.
Die Ausgabe ist die Wahrscheinlichkeit, dass in $size$ Versuchen höchstens q Erfolge erzielt werden.
Beispiel: Berechnet $P(X \leq 5)$ für eine Binomialverteilung mit 10 Versuchen und einer Erfolgswahrscheinlichkeit von 0.3.)

```
dbinom(x = , size = , prob = )
dbinom(x = 3, size = 10, prob = 0.3)
```

x = Die Anzahl der Erfolge, für die die Wahrscheinlichkeit berechnet wird.
Beispiel: Gibt die Wahrscheinlichkeit zurück, genau 3 Erfolge in 10 Versuchen zu erzielen.)

```
qbinom(p = , size = , prob = )
qbinom(p = 0.975, size = 10, prob = 0.3)
```

p = Das Quantil, also der Wahrscheinlichkeitswert (zwischen 0 und 1).
Die Ausgabe ist die kleinste Anzahl von Erfolgen, sodass $P(X \leq x) \geq p$ gilt.
Beispiel: Liefert das 97,5%-Quantil der Binomialverteilung.)

```
rbinom(n = , size = , prob = )
rbinom(n = 10, size = 10, prob = 0.3)
```

n = Anzahl der zu erzeugenden Zufallszahlen.
Die Ausgabe sind n Zufallszahlen, die jeweils die Anzahl der Erfolge in $size$ Versuchen darstellen.
Beispiel: Erzeugt 10 Zufallszahlen aus einer Binomialverteilung mit 10 Versuchen und einer Erfolgswahrscheinlichkeit von 0.3.)

11.3 Hypergeometrische Verteilung

n = Nummer der Erfolge
M = Nummer der Misserfolge
k = Wie viele Versuche es gibt

```r
phyper(q = , m = , n = , k = )
phyper(q = 5, m = 20, n = 30, k = 10)
```

q = Die Anzahl der Erfolge, bis zu der $P(X \leq q)$ berechnet wird.
Die Ausgabe ist die Wahrscheinlichkeit, dass bei k Ziehungen aus einer Urne mit m Erfolgen und n Misserfolgen höchstens q Erfolge erzielt werden.
Beispiel: Berechnet $P(X \leq 5)$ für eine Hypergeometrische Verteilung mit $m = 20$, $n = 30$ und $k = 10$.

```r
dhyper(x = , m = , n = , k = )
dhyper(x = 3, m = 20, n = 30, k = 10)
```

x = Die Anzahl der Erfolge, für die die Wahrscheinlichkeit berechnet wird.
Beispiel: Gibt die Wahrscheinlichkeit zurück, genau 3 Erfolge bei 10 Ziehungen zu erzielen.

```r
qhyper(p = , m = , n = , k = )
qhyper(p = 0.975, m = 20, n = 30, k = 10)
```

p = Das Quantil, also der Wahrscheinlichkeitswert (zwischen 0 und 1).
Die Ausgabe ist die kleinste Anzahl von Erfolgen, sodass $P(X \leq x) \geq p$ gilt.
Beispiel: Liefert das 97,5%-Quantil der Hypergeometrischen Verteilung.

```r
rhyper(nn = , m = , n = , k = )
rhyper(nn = 10, m = 20, n = 30, k = 10)
```

nn = Anzahl der zu erzeugenden Zufallszahlen.
Die Ausgabe sind nn Zufallszahlen, die jeweils die Anzahl der Erfolge in k Ziehungen darstellen.
Beispiel: Erzeugt 10 Zufallszahlen aus einer Hypergeometrischen Verteilung mit $m = 20$, $n = 30$ und $k = 10$.

12 Expected Value und Varianz

12.1 Discrete Random Variablen

Erwartungswert(Mean) und Varianz einer diskreten Zufallsvariablen X mit Wahrscheinlichkeitsfunktion $p(x)$

$$E[X] = \sum_x x \cdot p \cdot (x)$$

$$\operatorname{Var}(X) = E\left[X^2\right] - (E[X])^2$$

```r
no_car <- 0.2
one_car <- 0.7
two_cars <- 0.1
expected_value <- (0*no_car) + (1*one_car) + (2*two_cars)
var <- ((0^2*no_car) + (1^2*one_car) + (2^2*two_cars)) - expected_value^2
```

Hier Berechnen wir erst Mean und dann die Var. Um zur SD zu gelangen müssen wir sqrt()
Um jetzt herauszufinden Wieviele Parkplätze wir bauen müssen um 99% der Autos parken zu können. Müssen wir die Anzahl der Häuser mal dem expected Value und Var rechnen

```r
n <- 1000
qnorm(.99, mean = n*expected_value, sd=sqrt(n*var))
```

12.2 X ist Binomialy distributed

$$E[X] = n \cdot p \quad , \quad \operatorname{Var}(X) = n \cdot p \cdot (1-p)$$

12.3 Brauche ich hier wohl uniformly und hyper??

13 Central Limit Theorem

13.1 Nach Maximum Sample size Umstellen n

⚠ Hier sollte alpha 0.5 sein, sonst Brute force ⚠
-
Quantilgleichung die bei der Normalapproximation der Binominalverteilung angewendet wird:

$$k + 0.5 = n \cdot p + \operatorname{qnorm}(\alpha) \cdot \sqrt{n \cdot p \cdot (1-p)}$$

Wir wissen, das wenn alpha = 0.5, ist qnorm(0.5) = 0.
Damit können wir $\sqrt{n \cdot p \cdot (1-p)}$ ignorieren!
- Jetzt haben wir also:

$$k + 0.5 = n \cdot p \quad \Longrightarrow \quad n = \frac{k + 0.5}{p}$$

Beispiel: Aus den Fakultäten B (25%) und C (30%) stammen insgesamt 55% aller Studierenden. Bei einer zufällig gezogenen Stichprobe der Größe n ist die Anzahl X der Studierenden aus B und C binomialverteilt, also X Bin(n, 0,55). Ein Raum bietet 80 Plätze, weshalb die Bedingung. Der Raum soll mit eine Chance von 50% ausreichen
P(X <= 80) >= 0,5 erfüllt sein muss. Bestimme das maximale n, für das diese Anforderung gilt.

$$k = 80, \quad p = 0.55, \quad \alpha = 0.5, \quad \operatorname{qnorm}(0.5) = 0$$

$$n = \frac{80.5}{0.55} \approx 146.36.$$

```r
n <- 80.5 / 0.55 #146.3636
```

Hier ist mein Bureforce Ansatz:

```r
new_p <- b + c
new_n <- 130:150
x <- pbinom(80, new_n, new_p)
x
new_n[max(which(x >= 0.5))] #146
```

14 Distributions

1.1)Binominale Distribution mit zurücklegen

⚠⚠ Mit zurücklegen ⚠⚠

$$P(X=k) \;=\; \binom{n}{k}\, p^k\,(1-p)^{\,n-k}$$

Beispiel: Wir haben 7 Weiße Bälle und 3 Rote Bälle: Wie Wie groß ist die Wahrscheinlichkeit, dass wir in n=5 Zügen k=2 rote Bälle ziehen? p ist 7/10

$$P(X=2) = \binom{5}{2} \left(\frac{3}{10}\right)^2 \left(\frac{7}{10}\right)^3.$$

```r
dbinom(x = 2, size = 5, prob = 3/10)
#0.1029193
```

x: Wie viele Rote Bälle wir bekommen wollen,

$size$: Wie Oft wir ziehen,

$prob$: Die prob einen Roten Ball zu ziehen

1.2)Hypergemoetric Distribution ohne zurücklegen

⚠⚠ Wie Binomial, aber ohne Zurücklegen ⚠⚠

$$P(X=k) = \frac{\binom{M}{k}\binom{N-M}{n-k}}{\binom{N}{n}}$$

N: Gesamtanzahl aller Elemente(z.B alle Kugeln),

M: Anzahl der Roten Kugeln gesamt,

n: Wie oft wir Ziehen,

k: Wie viele Roten wir Ziehen

```r
dhyper(x = 2, m = 3,n = 7,k = 5)
#0.4166667
```

x: wie viele von den gezogenen Bällen Rot sein sollen,

M: wie viele Rote Bälle,

n: wie viele nicht Rote,

k: wie viele Bälle wir ziehen

1.3)Multinomial Distribution mit zurücklegen

⚠⚠ Wie Binomial aber mit mehr als zwei Optionen.⚠⚠

Beispiel: Angenommen, wir haben $n=5$ Versuche. drei mögliche Ergebnisse (z.B. rot, blau, schwarz) mit $Rot = \frac{15}{20}$, $Grün = \frac{4}{20}$, $Blau = \frac{1}{20}$. Wir fragen: Wie groß ist die Wahrscheinlichkeit, dass genau Rot=2, $Grün$=2, $Blau$=1 auftritt?

Formel: $$\frac{n!}{x_1!\,x_2!\cdots x_k!}\, p_1^{x_1}\, p_2^{x_2} \cdots p_k^{x_k},$$

$$\frac{5!}{2!\,2!\,1!} \left(\frac{15}{20}\right)^2 \left(\frac{4}{20}\right)^2 \left(\frac{1}{20}\right)^1 = 0.3375.$$

```r
(factorial(5) / (factorial(2) * factorial(2) * factorial(1))) *
  ((15/20)^2 * (4/20)^2 * (1/20)^1) #0.03375
#Hier auch als Funktion
dmultinom(c(2, 2, 1),prob=c(15/20,4/20, 1/20))
```

1.4)Multivariate Hypergeometric Distribution

OHNE zurücklegen

⚠⚠ Wie Binomial aber mit mehr als zwei Optionen. ⚠⚠

Beispiel:

Angenommen, wir haben $n=5$ Versuche. drei mögliche Ergebnisse (z.B. rot, blau, schwarz) mit $Rot = \frac{15}{20}$, $Grün = \frac{4}{20}$, $Blau = \frac{1}{20}$. Wir fragen: Wie groß ist die Wahrscheinlichkeit, dass genau Rot=2, $Grün$=2, $Blau$=1 auftritt?

Formel: $$P\big(X_1 = k_1, X_2 = k_2, \ldots, X_r = k_r\big) = \frac{\binom{K_1}{k_1}\binom{K_2}{k_2}\ldots\binom{K_r}{k_r}}{\binom{N}{n}}$$

$$\frac{\binom{15}{2}\binom{4}{2}\binom{1}{1}}{\binom{20}{5}} \approx 0.04063467.$$

```r
(choose(15,2) * choose(4,2) * choose(1,1)) /choose(20,5) #0.04063467
```

1.4)Sequentielle Ziehung mit Zurücklegen

⚠⚠ mit Zurücklegen ⚠⚠

Wir haben insgesamt 20 Bälle, davon sind 15 Bälle nicht rot und 5 Bälle sind rot. Wir wollen die Wahrscheinlichkeit erst 4 nicht rote Bälle zu ziehen und dann ein roten Ball zu ziehen. - Geometrische Verteilung

P(Keinen roten Ball) = $\frac{15}{20}$
P(Einen roten Ball) = $\frac{5}{20}$

$$P(X = 5) = \left(1 - \frac{5}{20}\right)^4 \cdot \frac{5}{20}$$

```
(1 - (5 / 20))^4 * 5 / 20 #0.07910156
dgeom(x = 4, prob = 5/20) #Mit Funktion
```

Erst nehmen wir die chance (gegenwahrscheinlichkeit) keinen roten zu ziehen hoch 4 und dann mal die chance einen roten zu ziehen

1.6)Sequentielle Ziehung ohne Zurücklegen

⚠⚠ Ohne Zurücklegen ⚠⚠

Wir haben insgesamt 20 Bälle, davon sind 15 Bälle nicht rot und 5 Bälle sind rot. Wir wollen die Wahrscheinlichkeit erst 4 nicht rote Bälle zu ziehen und dann ein roten Ball zu ziehen. Negative hypergeometrische Verteilung

P(Keinen roten Ball) = $\frac{15}{20}$
P(Einen roten Ball) = $\frac{5}{20}$
P(einen roten Ball nach 4 Zügen) = $\frac{5}{16}$

$$P(X = 5) = \frac{\binom{15}{4}\binom{5}{0}}{\binom{20}{5}} \cdot \frac{5}{16}$$

Wir berechnen die Wahrscheinlichkeit 4 nicht rote Bälle zu ziehen Multipliziert mit der Wahrscheinlichkeit einen roten aus den verbleibenden Bällen zu Ziehen.

```
((choose(15,4)*choose(5,0))/choose(20,4)) * 5/16
#0.0880418
```

15 Type I and II Errors

	True state of H_0	
Statistical decision	H_0 True	H_0 False
Reject H_0	Type I Error	Correct
Do not reject H_0	Correct	Type II Error

Definitions:

- α: Probability of rejecting H_0 given that H_0 is true.
- β: Probability of not rejecting H_0 given that H_0 is false.

16 Relevante Übersetzungen

1. Dispersion: Streuung (vermutlich SD gemeint)
2. Scatter: Streuung (vermutlich SD gemeint)

17 P-Value

Hypothese	Test-Typ	p-Wert Berechnung
$H_0 : \mu \geq \mu_0$	Einseitig (links)	$p = \texttt{pnorm}(z)$
$H_0 : \mu \leq \mu_0$	Einseitig (rechts)	$p = 1 - \texttt{pnorm}(z)$
$H_0 : \mu = \mu_0$	Zweiseitig	$p = 2 \cdot \texttt{pnorm}(-\lvert z \rvert)$

18 Prob Type II error β

18.1 Normal Distribution - two sided

Was brauchen wir:

Symbol	Bedeutung
ub	Upper bound
lb	lower bound
mu1	Test Mean - gegeben in der aufgabestellung

```
beta <- pnorm(ub, mean = mu1, sd = sigma/sqrt(n)) - pnorm(lb, mean = mu1, sd = sigma/sqrt(n))
```

18.2 Normal Distribution - Rechtsseitiger Test

```
beta <- pnorm(ub, mean = mu1, sd = sigma/sqrt(n))
```

18.3 Normal Distribution - Linksseitiger Test

```
beta <- 1 - pnorm(lb, mean = mu1, sd = sigma/sqrt(n))
```

19 Library

library(TeachingDemos)

19.1 Binomial Distribution - two sided

Was brauchen wir:

Symbol	Bedeutung
p0	population proportion
p1	Test Proportion - gegeben in der aufgabestellung

```
beta <- pbinom(qbinom(1 - alpha, size = n, prob = p0), size = n, prob = p1)
```

- Der Index 0 z.b. μ_0 bedeutet, dass es sich um einen gegebenen Wert, und nicht um einen geschätzten Wert handelt.

I) Gauß Test:

Hauptziel: Hier wird die Hypothese über den Mittelwert (μ) getestet

Mean μ ist unbekannt, wir kennen SD σ

Gegeben muss sein:

$$H_0 : \mu = \mu_0, \quad H_0 : \mu \leq \mu_0, \quad H_0 : \mu \geq \mu_0$$

Symbol	Bedeutung
n	Stichprobengröße
σ_0	Standardabweichung der gesamtheit
$\overline{X}_{(n)}$	Sample Mean

Decision Rule R**:**

$$T = \frac{\overline{X} - \mu_0}{\frac{\sigma_0}{\sqrt{n}}} \in R \implies \text{reject } H_0$$

Rejection Region R**:**

H_0	rejection region R
$\mu = \mu_0$	$(-\infty, -u_{1-\frac{\alpha}{2}}) \cup (u_{1-\frac{\alpha}{2}}, \infty)$
$\mu \leq \mu_0$	$(u_{1-\alpha}, \infty)$
$\mu \geq \mu_0$	$(-\infty, -u_{1-\alpha})$

$u_{1-\frac{\alpha}{2}}$ = qnorm(1 - (alpha / 2))

Beispiel:

```
n <- 100
sd <- 0.3
sample_mean <- 10.1
alpha <- 0.1
#H0: mu = 10, H1: mu != 10
mu0 <- 10
#Rejection region
ru <- qnorm(1 - (alpha / 2))
rl <- -qnorm(1 - (alpha / 2))
#[-inf , -1.644854] or [1.644854, inf]
#teststatistic
t <- (sample_mean - mu0) /(sd / sqrt(n))
#3.333333
t > ru
#True
#we reject h0 because we are in the rejection
 ↪ region
p_value <- 2* pnorm(-abs(t))
#0.0008581207
p_value < alpha
#True we reject H0
```

hier vielleichtg noch z test einfpgen

II) t-Test:

Hauptziel: Hier wird die Hypothese über den Mittelwert (μ) getestet.

Mean μ und SD σ_0 sind unbekannt

⚠ Mean μ_0 wird durch H_0 gegeben ⚠

Gegeben muss sein:

$$H_0 : \mu = \mu_0, \quad H_0 : \mu \leq \mu_0, \quad H_0 : \mu \geq \mu_0$$

Symbol	Bedeutung
n	Stichprobengröße
$S_{(n)}$	Sample SD
$\overline{X}_{(n)}$	Sample Mean

Decision Rule:

$$T = \frac{\overline{X} - \mu_0}{\frac{S_{(n)}}{\sqrt{n}}} \in R \implies \text{reject } H_0$$

Rejection Region R**:**

H_0	Rejection Region R
$\mu = \mu_0$	$(-\infty, -t_{n-1,1-\frac{\alpha}{2}}) \cup (t_{n-1,1-\frac{\alpha}{2}}, \infty)$
$\mu \leq \mu_0$	$(t_{n-1,1-\alpha}, \infty)$
$\mu \geq \mu_0$	$(-\infty, -t_{n-1,1-\alpha})$

$t_{n-1,1-\frac{\alpha}{2}}$ = qt(1-alpha/2, n-1)

exact ⚠ (nur möglich wenn wir Sample habe) ⚠:

```
t.test(x = sample, mu = mu0, alternative =
 ↪ "two.sided", conf.level = 1-alpha)
```

Approx Beispiel:

```
#H0: mu >= 250, h1: < 250
n <- 82
sample_mu <- 248
sample_sd <- 5
alpha <- 0.05
mu0 <- 250
R <- -qt(1-alpha, n-1)
#[ , -1.663884]
t <- (sample_mu - mu0) / ((sample_sd) / sqrt(n))
#-3.622154
t < R
#We reject the H0
p_value <- pt(t,n - 1)
#0.0002540167
p_value < alpha
#True We reject the H0
```

P-value Berechnen:

```
pt(t, n-1) #H0: mu >= mu0
1-pt(t, n-1) #H0: mu <= mu0
2*pt(-abs(t), n-1) #H0: mu = mu0
```

III) Test für Varianz σ_0^2:

Hauptziel: Hier wird die Hypothese über die Varianz (σ_0^2) getestet.

Mean μ und SD σ sind unbekannt

⚠ Kein σ_0 da σ gegeben durch H_0 ⚠

⚠ Also kein Schätzwert ⚠

Gegeben muss sein:

$$H_0 : \sigma^2 = \sigma_0^2, \quad H_0 : \sigma^2 \leq \sigma_0^2, \quad H_0 : \sigma^2 \geq \sigma_0^2$$

Symbol	Bedeutung
$S_{(n)}$	Sample SD
$\bar{X}_{(n)}$	Sample Mean

Decision Rule:

$$T = \frac{(n-1)\, S_{(n)}}{\sigma_0^2} \in R \quad \Longrightarrow \quad \text{reject } H_0.$$

Rejection Region R:

H_0	rejection region R
$\sigma^2 = \sigma_0^2$	$(0,\, \chi^2_{n-1,\frac{\alpha}{2}}) \cup (\chi^2_{n-1,1-\frac{\alpha}{2}}, \infty)$
$\sigma^2 \leq \sigma_0^2$	$(\chi^2_{n-1,1-\alpha}, \infty)$
$\sigma^2 \geq \sigma_0^2$	$(0,\, \chi^2_{n-1,\alpha})$

Beispiel:

```r
#h0: sd >= 7, h1: sd <7
n <- 82
sample_mu <- 248
sample_sd <- 5
alpha <- 0.05
sd0 <- 7
#Rejection region
R <- qchisq(alpha, n-1)
#[ , 61.26148
#Teststatistics
t <- ((n - 1) * sample_sd)/sd0
#57.85714
t < r
#We reject H0, in R area
p_value <- pchisq(t, n-1)
#0.02419782
p_value < alpha
#we reject H0
```

⚠ Hier noch var.test

IIII) Bernoulli Test für Probability p_0:

Hauptziel: Zu prüfen, ob die beobachtete Erfolgsrate $\hat{p}$ signifikant von der vorgegebenen Wahrscheinlichkeit p_0 abweicht

Probability p_0 ist unbekannt

$$\text{Number of successes: } X = \sum_{i=1}^{n} X_i \sim B(n,p), \quad \text{d.h. } \mathbb{E}(X) = np$$

$$\mathrm{Var}(X) = np(1-p).$$

Gegeben muss sein:

$$H_0 : p = p_0, \quad H_0 : p \leq p_0, \quad H_0 : p \geq p_0$$

Symbol	Bedeutung
n	Stichprobengröße
X	Number of successes
$\hat{p}$	$\frac{X}{n}$ Example Probability

Teststatistic

$$T = \frac{\hat{p} - p_0}{\sqrt{\frac{p_0(1-p_0)}{n}}}, \quad \text{mit } \hat{p} = \frac{X}{n}.$$

Rejection Region R

H_0	Rejection Area R
$p = p_0$	$(-\infty, -u_{1-\frac{\alpha}{2}}) \cup (u_{1-\frac{\alpha}{2}}, \infty)$
$p \leq p_0$	$(u_{1-\alpha}, \infty)$
$p \geq p_0$	$(-\infty, -u_{1-\alpha})$

Normal Approximation:

```r
#a) 80% immunity rate
#b) H0: p <= 80, H1: p > 80
p0 <- 0.8; n <- 200; x <- 172
alpha <- 0.05
phut <- x / n
#Rejection region
R <- qnorm(1 - alpha)
#r <- [1.644854, ]
#teststatistic
t <- (phut-p0)/sqrt((p0 * (1 - p0)) / n)
#2.12132
t > R
#We reject H0
p_value <- 1 - pnorm(t)
#0.01694743
p_value < alpha
#We reject H0
```

Exact test:

```r
#exact
binom.test(172, p = 0.8, n = n, alternative =
  'greater', conf.level = 1-alpha)
#0.01793
```

0) Alle Infos 2-Sample Tests:

I) 2-Sample Gauss Test:

Hauptziel: Hier wird die Hypothese über die Mittelwerte (μ_1, μ_2) getestet

Means sind unbekannt, wir kennen σ_1, σ_2

Gegeben muss sein:

$$H_0 : \mu_1 = \mu_2, \quad H_0 : \mu_1 \leq \mu_2, \quad H_0 : \mu_1 \geq \mu_2$$

Symbol	**Bedeutung**
n_1, n_2	Stichprobengrößen
σ_1, σ_2	SD der gesamtheiten
$\overline{X}_{(n_1)}, \overline{Y}_{(n_2)}$	Sample Means

Teststatistik:

$$T = \frac{\overline{X}_{(n_1)} - \overline{Y}_{(n_2)}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \sim N(0,1)$$

Decision Rule R:

$$T \in R \implies \text{reject } H_0$$

Rejection Region R:

H_0	Rejection Region R
$\mu_1 = \mu_2$	$(-\infty, -u_{1-\frac{\alpha}{2}}) \cup (u_{1-\frac{\alpha}{2}}, \infty)$
$\mu_1 \leq \mu_2$	$(u_{1-\alpha}, \infty)$
$\mu_1 \geq \mu_2$	$(-\infty, u_\alpha)$

Beispiel:

```r
m1 <-  c(5.46, 5.34, ..., 5.82)
m2 <- c(5.45, 5.31, 4.11, ..., 4.09)
sd1 <- 0.5
sd2 <- 0.6
n1 <- length(m1)
n2 <- length(m2)
#test the H0: mu1 >= mu2
alpha <- 0.05
#rejection Region
r <- qnorm(alpha)
#[ , -1.644854]
#teststistic
t <- (mean(m1) - mean(m2)) /
  sqrt((sd1^2 / n1) + (sd2^2 / n2))
#1.027782
p_value <- pnorm(t)
#0.8479739
#we fail to reject H0 since we are outside of the
↪  rejection area
```

II) 2-Sample t-Test (Varianzen gleich und unbekannt):

Hauptziel: Hier wird die Hypothese über die Mittelwerte (μ_1, μ_2) getestet

Means μ_1, μ_2 sind unbekannt und $\sigma_1 = \sigma_2$

Gegeben muss sein:

$$H_0 : \mu_1 = \mu_2, \quad H_0 : \mu_1 \leq \mu_2, \quad H_0 : \mu_1 \geq \mu_2$$

⚠ Es muss für x und y ein Sample gegeben sein ⚠

Beispiel:

```
x <- c(7.06, 11.84, ..., 8.54)
y <- c(8.68, 6, 7.82, 4.7, ..., 12.36)
#H0: X >= Y, H1 x < y
####################################
#case: equal variances:
t.test(x, y, alternative = 'less', paired = F, var.equal = T, conf.level = 0.95)
#p-value = 0.0181
#we can reject the h0 since we are under 0.05
```

III) Welch test (Varianzen ungleich, aber unbekannt):

Hauptziel: Hier wird die Hypothese über die Mittelwerte (μ_1, μ_2) getestet

Means μ_1, μ_2 sind unbekannt und $\sigma_1 \neq \sigma_2$

Gegeben muss sein:

$$H_0 : \mu_1 = \mu_2, \quad H_0 : \mu_1 \leq \mu_2, \quad H_0 : \mu_1 \geq \mu_2$$

⚠ Es muss für x und y ein Sample gegeben sein ⚠

Beispiel:

```
x <- c(7.06, 11.84, ..., 8.54)
y <- c(8.68, 6, 7.82, 4.7, ..., 12.36)
#H0: X >= Y, H1 x < y
####################################
#case: unequal variances:
t.test(x, y, alternative = 'less', paired = F, var.equal = F, conf.level = 0.95)
#p-value = 0.01596
#we can reject the h0 since we are under 0.05
```

⚠ Das einzige was sich ändert ist: var.equal = F ⚠

IV) Two Paired Sample t-Test

⚠ Wenn Z.B einzelne Partienten vorher nacher ⚠

Hauptziel: Wir berechnen als erstes den Unterschied aller Werte der beiden Samples, und dann schauen ob der Mean signifikant Unterschiedlich von 0 ist.

σ ist unbekannt

Gegeben muss sein:

$$H_0 : \mu_1 = 0, \quad H_0 : \mu_1 \leq 0, \quad H_0 : \mu_1 \geq 0$$

⚠ Es muss für x und y ein Sample gegeben sein ⚠

Beispiel:

-

```
#H0: mu = 0, H1: mu != 0
x <- c(16, 15, 11, 20, ..., 15, 14, 16)
y <- c(13, 13, 10,.., 10, 15, 11, 16)
t.test(x = x, y = y, alternative = 'two.sided', paired = T, var.equal = T, conf.level = 0.95, mu = 0)
#0.0007205
```

⚠ Das einzige was sich ändert ist: paired = T ⚠

V) Testing two Variances - F Test

Hauptziel: Wir vergleichen die beiden sample Varianzen.

σ ist unbekannt

Gegeben muss sein:

$$H_0 : \sigma_1 = \sigma_2, \quad H_0 : \sigma_1 \leq \sigma_2, \quad H_0 : \sigma_1 \geq \sigma_2$$

⚠ Es muss für x und y ein Sample gegeben sein ⚠

Beispiel:

```
x <- c(102.4, 101.3, ..., 100.1)
y <- c(98.4, 101.7, ..., 101.0)
#H0: sd_x <= sd_y, H1: sd_x > sd_y
alpha <- 0.05
var.test(x = x, y = y, alternative = 'greater', conf.level = 1-alpha)
#p-value = 0.03404
```

Beispiel: Erst H0 dass vars gleich sind. Wenn nicht reject, dann müssten wir mein Mean test, var.equal auf True

```
#H0: sigma_x = sigma_y, H1: sigma_x != sigma_y
var.test(x = x, y = y, alternative = 'two.sided', conf.level = 1-alpha)
#p-value = 0.8814
#we fail to reject H0, also Varianz gleich oder fast gleich
###############################################
#H0: mu_x <= mu_y, H1: mu_x > mu_y
alpha <- 0.025
t.test(x = x, y = y, alternative = 'greater', paired = F, var.equal = TRUE, conf.level = 1-alpha)
#p-value = 0.02374
#we reject H0
```