Stochastische Ausarbeitung

Bela Schinke

16.04.2023

Contents

Einleitung

Viel Spaß mit meiner Ausarbeitung :)
Quellen werden am Anfang jedes Kapitels aufgeführt.

Allgemeine Formulierungshilfe von [ChatGPT].
Text Completion für Freitext und (hauptsächlich) R Code von [Copilot].
Das Skript von [Engel] wurde bei fast allen Aufgaben verwendet.

Aufagabe 1

χ^2-Anpassungstest

Theorieteil

[Wikipedia, a]

In der Multinomialverteilung haben wir 4 Kategorien, welche jeweils Binomial verteilt sind. Für große n ist die Binomialverteilung normalverteilt mit $\mu = n \cdot p$ und $\sigma = \sqrt{n \cdot p \cdot (1-p)}$. Sei a_1, a_2, a_3, a_4 die Anzahl der Beobachtungen in den Kategorien. Damit ist $\frac{a_j - n \cdot p_j}{\sqrt{n \cdot p_j}} \sim N(0,1)$.
Das wegfallen des $(1-p)$ im Nenner kommt daher, dass es sich um eine Multinomialverteilung handelt.
Also ist $\frac{(a_j - n \cdot p_j)^2}{n \cdot p_j} \sim (N(0,1))^2$
Damit ist die Summe $\sum_{j=1}^{4} \frac{(a_j - n \cdot p_j)^2}{n \cdot p_j \cdot (1-p_j)} \sim \chi^2_3$.

Da die p-Werte der χ^2-Verteilung bekannt sind, kann so ein einfacher Hypothesentest durchgeführt werden:

Anwendung

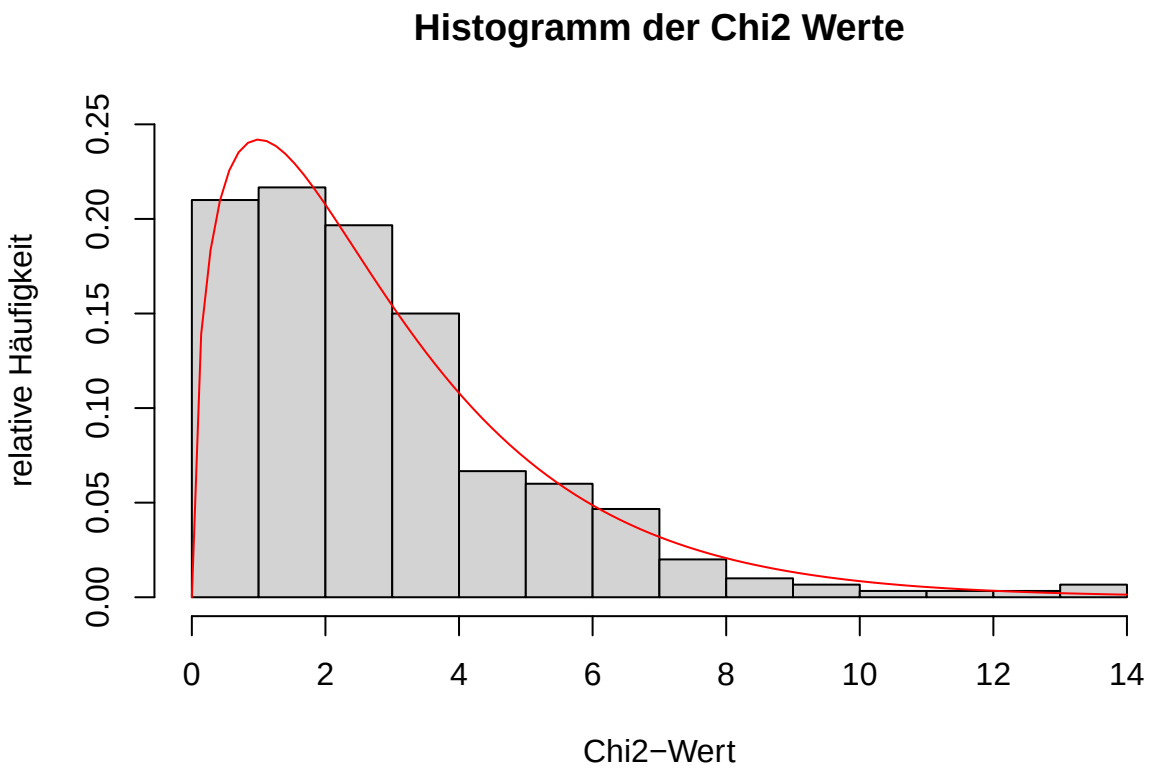
$$H_0 : p_1 = \frac{1}{8}, p_2 = \frac{1}{4}, p_3 = \frac{1}{2}, p_4 = \frac{1}{8}$$
$$H_1 : p_1 \neq \frac{1}{8} \vee p_2 \neq \frac{1}{4} \vee p_3 \neq \frac{1}{2} \vee p_4 \neq \frac{1}{8}$$

Simulieren wir nun den Versuch:

```
set.seed(123)
#Diese Funktion simuliert die Multinomialverteilung und berechnet die Chi2-Teststatistik
simulateAnpassungstest <- function(){
  n <- 1000
  p <- c(1/8, 1/4, 1/2, 1/8)
  a <- rmultinom(1, n, p)
  sum((a - n*p)^2/(n*p))
}
```

```
results <- c()
#Wir berechnen 300-mal die Teststatistik und speichern sie in results
for (i in 1:300){
  results =c(results ,simulateAnpassungstest())
}
```

```
#Wir plotten die Verteilung der Teststatistik
#und vergleichen sie mit der Chi2-Verteilung mit 3 Freiheitsgraden
hist(results, freq=FALSE, main = "Histogramm der Chi2 Werte", xlab = "Chi2-Wert",
     ylab = "relative Häufigkeit", ylim = c(0, 0.25))
curve(dchisq(x, 3), add = TRUE, col = "red")
```



$t-$Test mit unbekannter Varianz

Theorieteil

[Wikipedia, b]
Beim zweiseitigen $t-$Test mit unbekannter Varianz wird die Nullhypothese $H_0 : \mu = \mu_0$ gegen die Alternative $H_1 : \mu \neq \mu_0$ getestet. Als Teststatistik wird $\frac{\sqrt{n}(\bar{X}_n - \mu_0)}{S_n}$ verwendet.
Herleitung:
Seien $X_1, X_2, \ldots, X_n \sim N(\mu, \sigma^2)$ unabhängig und identisch verteilt.
Das arithmetische Mittel $\bar{X}_n$ ist normalverteilt mit $\mu = \mu$ und $\sigma = \frac{\sigma}{\sqrt{n}}$.

Bei bekannter Varianz wäre $\sqrt{n}\frac{\bar{X}_n - \mu_0}{\sigma} \sim N(0,1)$. Wir müssen allerdings die Varianz mit der empirischen Varianz ersetzen, also ist $\sqrt{n}\frac{\bar{X}_n - \mu_0}{S_n} \sim t_{n-1}$.(T-Verteilung folgt aus Störung durch die Varianzschätzung, Skript 4.55)

Das heißt für den Test müssen wir nur die Teststatistik $T = \frac{\sqrt{n}(\bar{X}_n - \mu_0)}{S_n}$ berechnen und anschließend deren p-Wert bestimmen. Wenn dieser kleiner als $1 - \alpha$ ist, wird die Nullhypothese verworfen.

Anwendung

Test:

```r
set.seed(123)
mu0 <- 2
sigma <- 4
data <- rnorm(1000, mu0, sigma)
alpha <- 0.05

#Berechne mean und varianz der Daten
dataVar <- var(data)
dataMean <- mean(data)

#Berechne Teststatistik
T <- sqrt(length(data))*(dataMean-mu0)/sqrt(dataVar)

#Teststatistik
print(paste("T = ", T))
```

```
## [1] "T =  0.514279000759497"
```

```r
#p-Wert von T
print(paste("p-Wert von T = ", pt(T, length(data)-1, lower.tail = FALSE)))
```

```
## [1] "p-Wert von T =  0.303585342303021"
```

```r
#Testresultat
if (pt(T, length(data)-1, lower.tail = FALSE) > 1-alpha){
  print("Nullhypothese wird verworfen")
} else {
  print("Nullhypothese wird nicht verworfen")
}
```

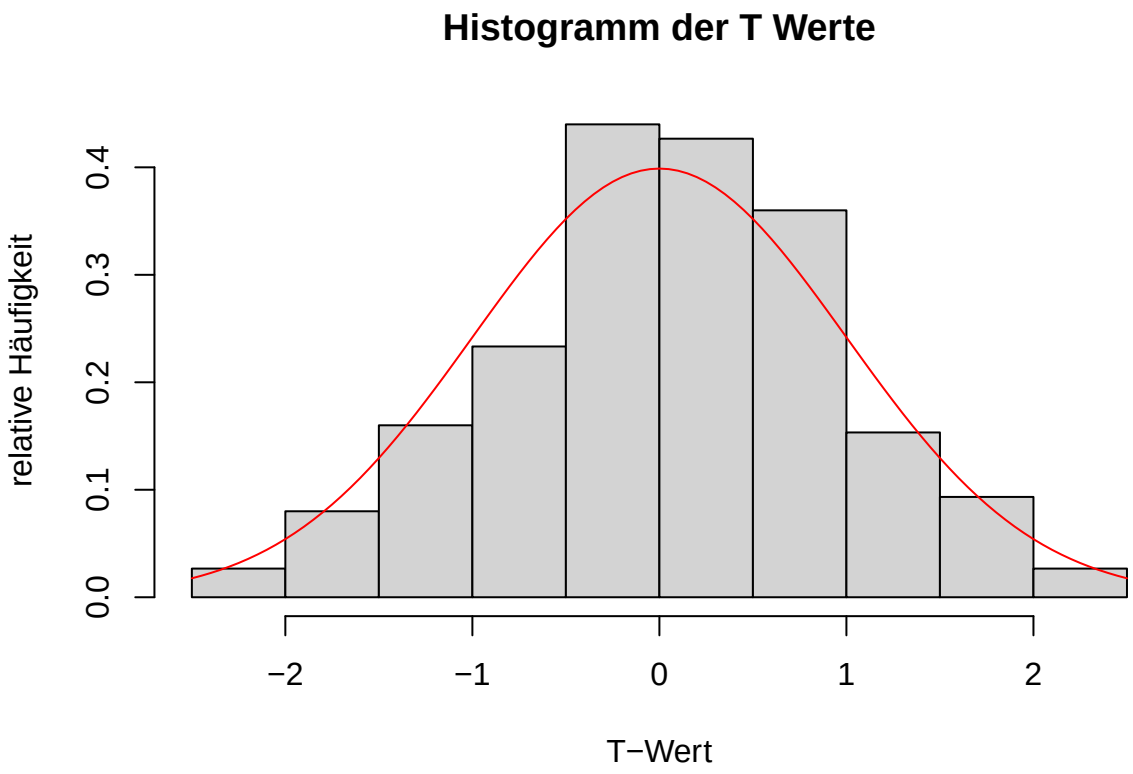
```
## [1] "Nullhypothese wird nicht verworfen"
```

Verteilung der Teststatistik:

```r
set.seed(123)
Tdata = c()

#Berechne Teststatistik 300-mal
for (i in 1:300){
  data <- rnorm(1000, mu0, sigma)
  mu0 <- 2
  alpha <- 0.05

  T <- sqrt(length(data))*(mean(data)-mu0)/sqrt(var(data))
  Tdata = c(Tdata, T)
}
#Histogramm der Teststatistik, vergleiche mit t-Verteilung
hist(Tdata, freq=FALSE, main = "Histogramm der T Werte",
     xlab = "T-Wert", ylab = "relative Häufigkeit")
curve(dt(x, length(data)-1), add = TRUE, col = "red")
```



Der Kolmogorov-Smirnov-Test prüft, ob eine Stichprobe aus einer spezifischen Verteilung stammt. Er vergleicht die empirische kumulative Verteilungsfunktion der Stichprobe mit der kumulativen Verteilungsfunktion der Population und berechnet den maximalen Abstand zwischen den beiden Funktionen.

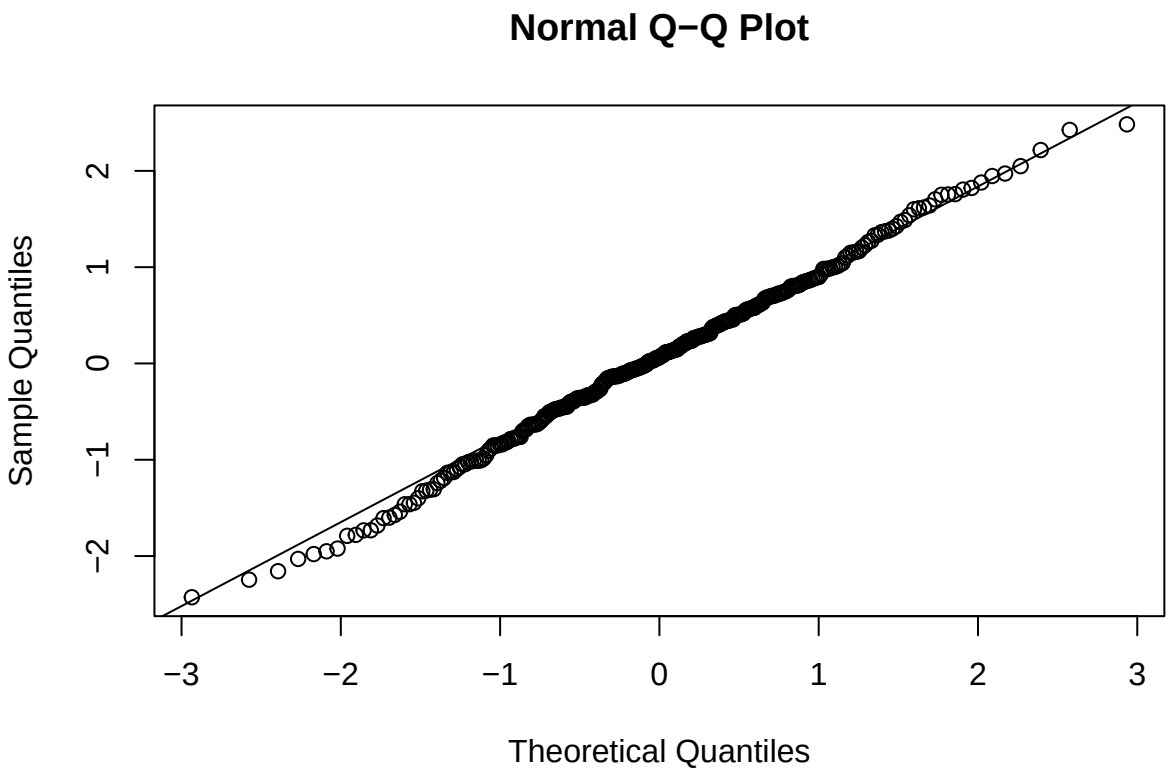
```
#Kolmogorov-Smirnoff-Test
print(ks.test(Tdata, "pt", length(data)-1))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Tdata
## D = 0.068075, p-value = 0.124
## alternative hypothesis: two-sided
```

Da der p-Wert des Kolmogorov-Smirnoff-Tests kleiner als $1 - \alpha = 0.95$ ist, kann die Nullhypothese nicht verworfen werden, also ist die Verteilung der Teststatistik t-verteilt.

QQ-Plot:

```
qqnorm(Tdata)
qqline(Tdata)
```



Ergebnis: Alles deutet darauf hin, dass die Teststatistik t-verteilt ist.

$t-$Test mit bekannter Varianz

Theorieteil

[Wikipedia, b] Genau wie oben, nur dass wir die Varianz nicht schätzen müssen. Deshalb ist die Teststatistik $\sqrt{n}\frac{\bar{X}_n - \mu_0}{\sigma} \sim N(0,1)$. Der Rest bleibt gleich:

Anwendung

```
set.seed(123)
mu0 <- 2
sigma <- 4
data <- rnorm(1000, mu0, sigma)
alpha <- 0.05

dataMean <- mean(data)

T <- sqrt(length(data))*(dataMean-mu0)/sigma

#Teststatistik
print(paste("T = ", T))
```

```
## [1] "T =  0.51000790152087"
```

```
#p-Wert von T unter Normalverteilung
print(paste("p-Wert von T = ", pnorm(T, lower.tail = FALSE)))
```

```
## [1] "p-Wert von T =  0.305022963064507"
```

```
#Testresultat
if (pnorm(T, lower.tail = FALSE) > 1-alpha){
  print("Nullhypothese wird verworfen")
} else {
  print("Nullhypothese wird nicht verworfen")
}
```

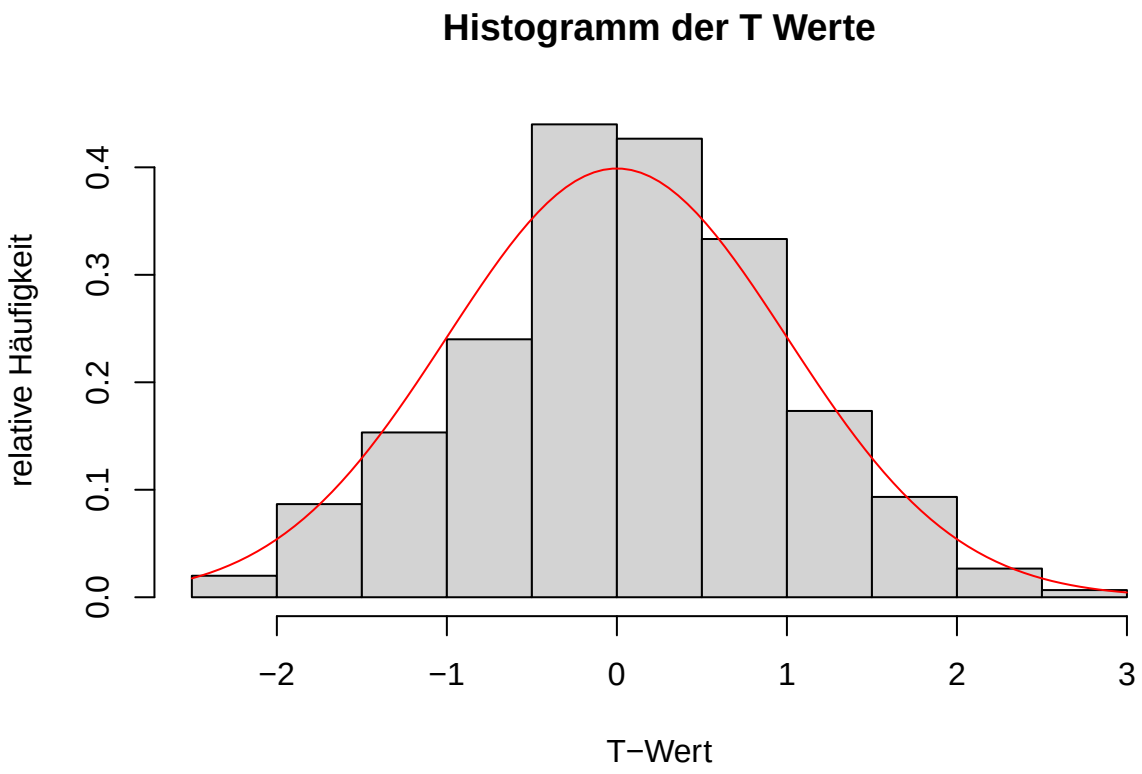
```
## [1] "Nullhypothese wird nicht verworfen"
```

Verteilung der Teststatistik:

```
set.seed(123)
Tdata = c()

for (i in 1:300){
  data <- rnorm(1000, mu0, sigma)
  alpha <- 0.05

  T <- sqrt(length(data))*(mean(data)-mu0)/sigma
  Tdata = c(Tdata, T)
}
hist(Tdata, freq=FALSE, main = "Histogramm der T Werte", xlab = "T-Wert", ylab = "relative Häufigkeit")
curve(dnorm(x), add = TRUE, col = "red")
```



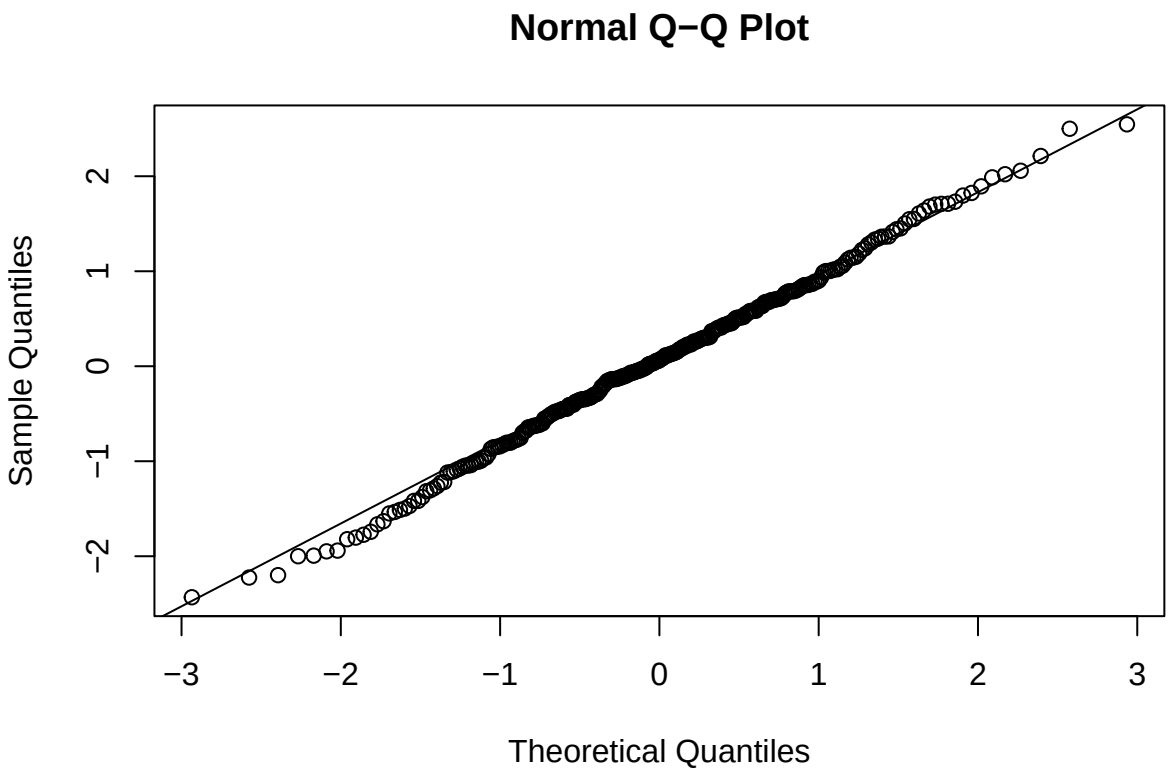

```
#Kolmogorov-Smirnoff-Test
print(ks.test(Tdata, "pnorm", lower.tail = FALSE))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Tdata
## D = 0.99456, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Da der p-Wert des Kolmogorov-Smirnoff-Tests kleiner als $1 - \alpha = 0.95$ ist, kann die Nullhypothese nicht verworfen werden, also ist die Verteilung der Teststatistik t-verteilt.

QQ-Plot:

```
qqnorm(Tdata)
qqline(Tdata)
```



Ergebnis: Alles deutet darauf hin, dass die Teststatistik normalverteilt ist.

Aufgabe 2

Lineare Regression mit R

Theorieteil

Ein lineares Modell geht davon aus, dass eine Abhängige Variable (hier der Preis) von einer oder mehreren unabhängigen Variablen (hier die Temperatur, der Niederschlag bei der Ernte und der Niederschlag im Winter) + einem zufälligen Fehler erzeugt wird. Die lm-Funktion findet die Parameter, die das Modell (nach der squared error Methode) am besten beschreiben.
Die Formel für das lineare Modell lautet:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

Die Parameter $\beta_0, \beta_1, \beta_2, \beta_3$ werden durch die lm-Funktion bestimmt. Die Fehlerterm ϵ ist normalverteilt mit Erwartungswert 0 und Varianz σ^2.

Anwendung

Daten einlesen

```r
data <- read.table("Datensaetze/wine.txt", header = TRUE)
```

Überblick über die Daten

```r
head(data)
```

```
##   year price temp h.rain w.rain
## 1 1952    37 17.1    160    600
## 2 1953    63 16.7     80    690
## 3 1955    45 17.1    130    502
## 4 1957    22 16.1    110    420
## 5 1958    18 16.4    187    582
## 6 1959    66 17.5    187    485
```

```r
summary(data)
```

```
##       year          price             temp           h.rain
##  Min.   :1952   Min.   : 10.00   Min.   :15.00   Min.   : 38.0
##  1st Qu.:1960   1st Qu.: 14.00   1st Qu.:16.15   1st Qu.: 88.0
##  Median :1967   Median : 22.00   Median :16.40   Median :123.0
##  Mean   :1967   Mean   : 28.81   Mean   :16.47   Mean   :144.8
##  3rd Qu.:1974   3rd Qu.: 35.00   3rd Qu.:17.00   3rd Qu.:185.5
##  Max.   :1980   Max.   :100.00   Max.   :17.60   Max.   :292.0
##      w.rain
##  Min.   :376.0
##  1st Qu.:543.5
##  Median :600.0
##  Mean   :608.4
##  3rd Qu.:705.5
##  Max.   :830.0
```

Lineares Modell erstellen

```r
model <- lm(price ~ temp + h.rain + w.rain, data = data)
summary(model)
```

```
##
## Call:
## lm(formula = price ~ temp + h.rain + w.rain, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.580  -8.601  -4.057   6.813  29.064
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -365.45179   77.63849  -4.707 9.66e-05 ***
```

```
## temp          22.50086    4.28502   5.251 2.51e-05 ***
## h.rain        -0.09296    0.03746  -2.481   0.0208 *
## w.rain         0.06103    0.02247   2.717   0.0123 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.33 on 23 degrees of freedom
## Multiple R-squared:  0.6421, Adjusted R-squared:  0.5954
## F-statistic: 13.75 on 3 and 23 DF,  p-value: 2.389e-05
```

In dem linearen Modell hat der Temperaturkoeffzient ein positives Vorzeichen, was bedeutet, dass der Preis mit steigender Temperatur steigt.
Der Koeffizient für Niederschlag bei der Ernte hat ein negatives Vorzeichen, was bedeutet, dass der Preis mit steigendem Niederschlag bei der Ernte sinkt.
Der Koeffizient für Niederschlag im Winter hat ein positives Vorzeichen, was bedeutet, dass der Preis mit steigendem Niederschlag im Winter steigt.

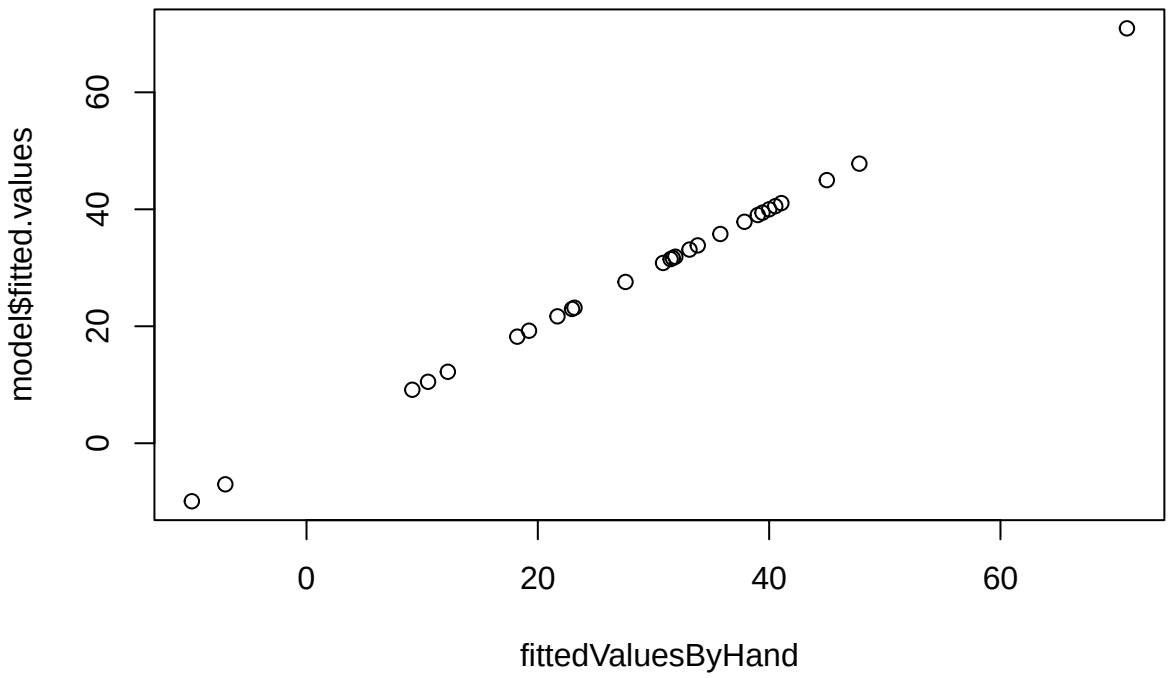
Angepasste Werte berechnen

Die angepassten Werte sind die Werte, die das lineare Modell für die Abhängige Variable berechnet. Sie können wir mit Hilfe der Hut-Matrix berechnen (siehe Skript 9.11):

```
X = matrix(c(rep(1,length(data$temp)),data$temp,data$h.rain,data$w.rain),ncol=4)
HutMatrix = X%*%solve(t(X)%*%X)%*%t(X)
fittedValuesByHand = HutMatrix%*%data$price
```

Vergleichen mit den aus dem linearen Modell berechneten Werten:

```
plot(fittedValuesByHand, model$fitted.values)
```




```
if (all((fittedValuesByHand - model$fitted.values)<0.0001)){
  print("Die beiden Werte sind annähernd gleich")
} else {
  print("Die beiden Werte sind nicht annähernd gleich")
}
```

```
## [1] "Die beiden Werte sind annähernd gleich"
```

Die beiden Werte sind annähernd gleich (Unterschiede durch Rundungsfehler), was bedeutet, dass die Berechnung der angepassten Werte mit Hilfe der Hut-Matrix korrekt ist.

Bestimmtheitsmaß

Das Bestimmtheitsmaß ist ein Maß für die Güte des linearen Modells und liegt zwischen 0 und 1. Es gibt den Anteil der Varianz durch die Regressionsvariablen an der Gesamtvarianz wieder. Grundsätzlich gilt je größer der Wert, desto besser ist das Modell.
Berechne das Bestimmtheitsmaß R^2 (Skript 9.16):

```
R2 = 1 - sum((data$price - fittedValuesByHand)^2)/sum((data$price - mean(data$price))^2)
print(paste("R2 = ", R2))
```

```
## [1] "R2 =  0.642088980919609"
```

zufälliger Fehler

Der zufällige Fehler ist normalverteilt mit Erwartungswert 0 und Varianz σ^2, in unserer Gleichung oben ϵ. Berechne die Varianz des geschätzten zufälligen Fehler $\hat{\sigma}^2$ (Skript 9.2.2):

```
sigma.hat.sq = sum((data$price - fittedValuesByHand)^2)/(length(data$price)-4)
print(paste("sigma.hat= ", sqrt(sigma.hat.sq)))
```

```
## [1] "sigma.hat=  13.3296898554664"
```

Die berechneten Werte stimmen mit den aus dem linearen Modell berechneten Werten überein. (siehe oben)

Erweiterung des linearen Modells

Wir fügen das Jahr der Ernte als weitere unabhängige Variable hinzu:

```
model2 <- lm(price ~ temp + h.rain + w.rain + year, data = data)
summary(model2)
```

```
##
## Call:
## lm(formula = price ~ temp + h.rain + w.rain + year, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.077  -9.040  -1.018   3.172  26.991
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1305.52761  597.31137   2.186  0.03977 *
## temp          19.25337    3.92945   4.900 6.72e-05 ***
## h.rain        -0.10121    0.03297  -3.070  0.00561 **
## w.rain         0.05704    0.01975   2.889  0.00853 **
## year          -0.82055    0.29140  -2.816  0.01007 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.69 on 22 degrees of freedom
## Multiple R-squared:  0.7369, Adjusted R-squared:  0.6891
## F-statistic: 15.41 on 4 and 22 DF,  p-value: 3.806e-06
```

Der Koeffizient für das Jahr hat ein negatives Vorzeichen, was bedeutet, dass der Preis mit steigendem Jahr sinkt. Das kann daran liegen, dass alte Weine seltener sind und deshalb teurer sind, oder dass der verlängerte Reifeprozess die Qualität der Weine steigert und deshalb der Preis steigt.

Vergleich der Güte der Modelle

```
print(paste("R2 Vergleich: ","old model: ", summary(model)$r.squared,
                             " new model: ", summary(model2)$r.squared))
```

```
## [1] "R2 Vergleich:  old model:  0.642088980919612  new model:  0.736908910750188"
```

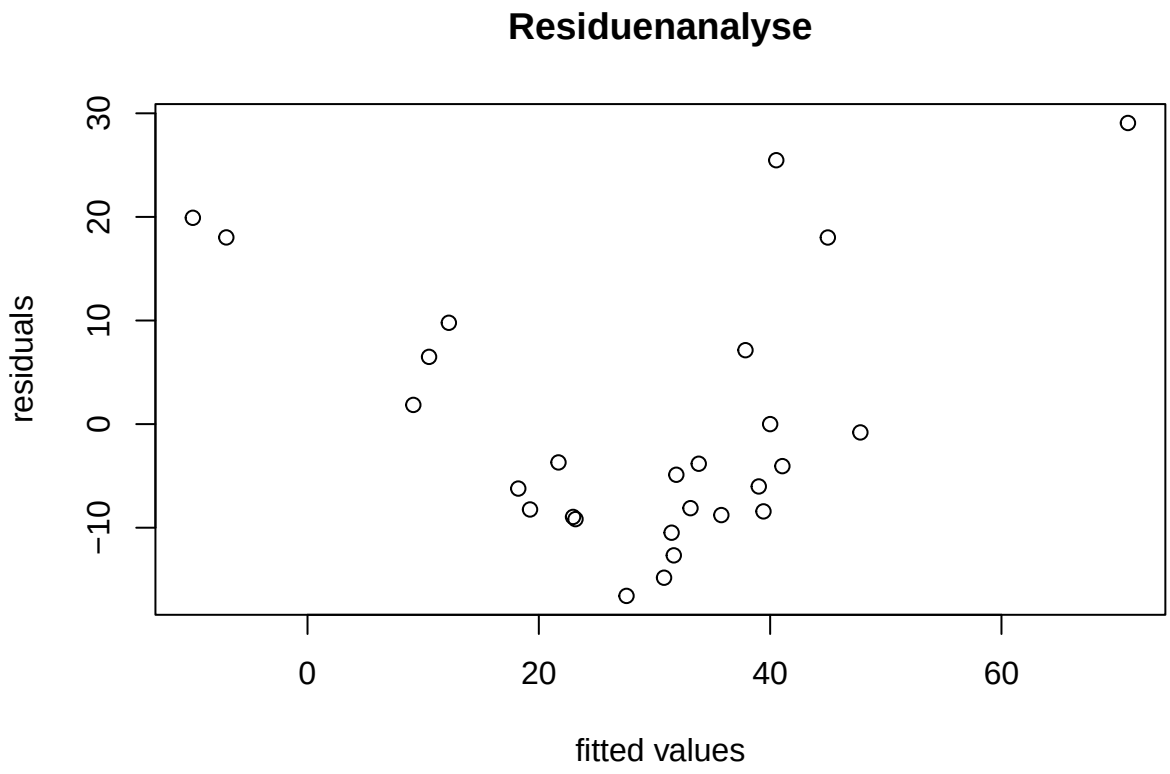
```
print(paste("Sigma Vergleich: ","old model: ", summary(model)$sigma,
                                " new model: ", summary(model2)$sigma))
```

```
## [1] "Sigma Vergleich:  old model:  13.3296898554663  new model:  11.6852540044809"
```

Das neue Modell hat ein besseres Bestimmtheitsmaß und einen kleineren geschätzten zufälligen Fehler. Das neue Modell wirkt also besser als das alte Modell. Man sollte aber auch noch den adjustierten R^2 betrachten.

Residuenanalyse

```
plot(model$fitted.values, model$residuals, xlab = "fitted values", ylab = "residuals", main = "Residuena
```



Die Residuen sind nicht unbedingt zufällig verteilt, man erkennt eine u-förmige Struktur. Das kann daran liegen, dass der zufällige Fehler nicht unabhängig normalverteilt ist.

Lineares Log Modell

```
model3 <- lm(log(price) ~ temp + h.rain + w.rain + year, data = data)
summary(model3)
```

```
##
## Call:
## lm(formula = log(price) ~ temp + h.rain + w.rain + year, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.4450 -0.2362  0.0181  0.1888  0.5100
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 40.6777920 14.3374580   2.837  0.00959 **
## temp         0.6187109  0.0943199   6.560 1.35e-06 ***
## h.rain      -0.0037482  0.0007915  -4.736  0.00010 ***
## w.rain       0.0011972  0.0004740   2.526  0.01924 *
## year        -0.0243519  0.0069947  -3.481  0.00212 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2805 on 22 degrees of freedom
## Multiple R-squared:  0.8308, Adjusted R-squared:  0.8001
## F-statistic: 27.01 on 4 and 22 DF,  p-value: 3.294e-08
```

In der Summary steht nun auch der zufällige Fehler für die logarithmierten Daten. Um einen Vergleichbaren Wert zu erhalten, müssen wir den geschätzten zufälligen Fehler für die normalen Daten berechnen:

```
true.fitted.values = exp(model3$fitted.values)
model3.sigma.hat.sq = sum((data$price - true.fitted.values)^2)/(length(data$price)-4)
print(paste("sigma.hat= ", sqrt(model3.sigma.hat.sq)))
```

```
## [1] "sigma.hat=  8.41672958083313"
```

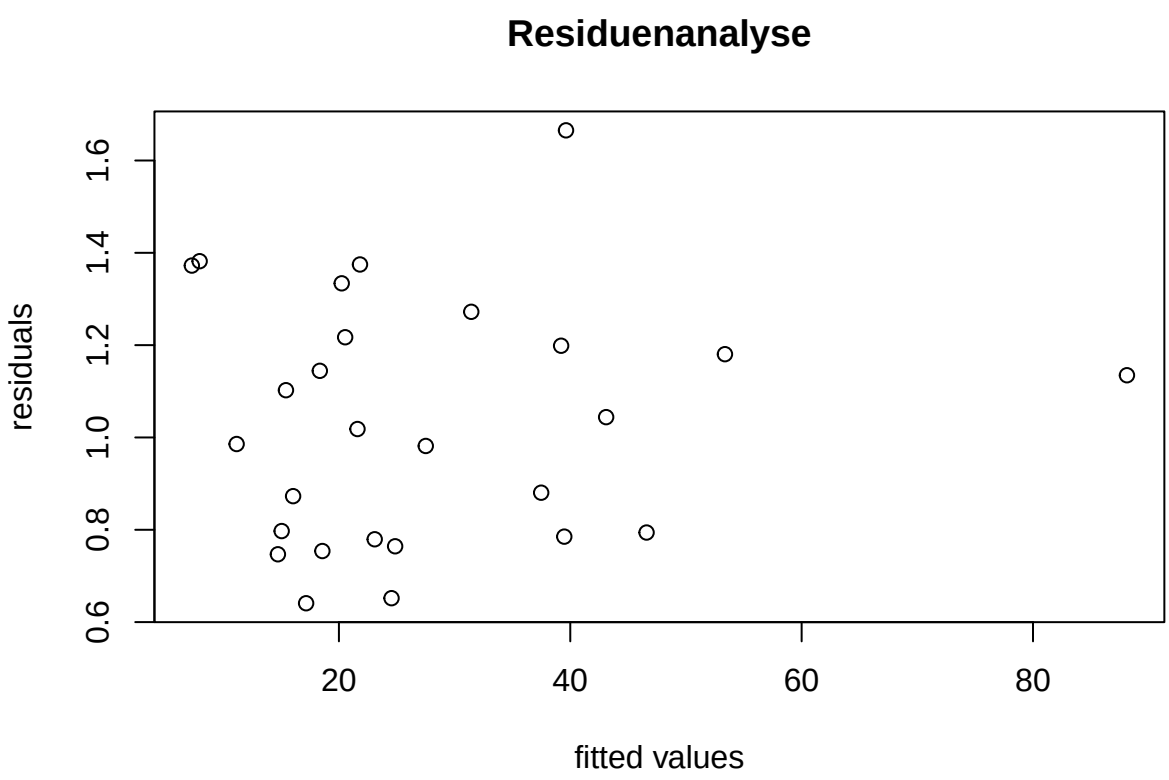
```
model3.R2=1-sum((data$price-true.fitted.values)^2)/sum((data$price-mean(data$price))^2)
print(paste("R2= ", model3.R2))
```

```
## [1] "R2=  0.857300737700796"
```

Die angepassten Werte sind immernoch besser als die der vorherigen Modelle.

Residuenanalyse

```
plot(exp(model3$fitted.values), exp(model3$residuals), xlab = "fitted values", ylab = "residuals", main
```



Sieht relativ unabhängig normalverteilt aus.
Vergleich mit Modell 2: Modell 3 ist deutlich besser als Modell 2, da die Residuen von Modell 3 unabhängig normalverteilt wirken, und die von Modell 2 nicht.

Vergleich der 3 Modelle

```
Vergleichstabelle <- data.frame(
  Model = c("Model 1", "Model 2", "Model 3"),
  R2 = c(summary(model)$r.squared, summary(model2)$r.squared, model3.R2),
  Adjusted.R2 = c(summary(model)$adj.r.squared, summary(model2)$adj.r.squared,
                  1-(1-model3.R2)*(length(data$price)-1)/(length(data$price)-4)),
  Sigma = c(summary(model)$sigma, summary(model2)$sigma, sqrt(model3.sigma.hat.sq))
)
knitr::kable(Vergleichstabelle)
```

Model	R2	Adjusted.R2	Sigma
Model 1	0.6420890	0.5954049	13.32969
Model 2	0.7369089	0.6890742	11.68525
Model 3	0.8573007	0.8386878	8.41673

Das dritte Modell ist signifikant besser als die anderen beiden Modelle, das sieht man an der höheren Adjusted R^2 und dem kleineren geschätzten zufälligen Fehler.

References

OpenAI ChatGPT. https://chat.openai.com/.

Github Copilot. https://copilot.github.com/.

Dirk Engel. Kursunterlagen für das praktikum zur stochastik.

Wikipedia. Chi-squared test. https://de.wikipedia.org/wiki/Chi-Quadrat-Test, a.

Wikipedia. Einstichproben-t-test. https://de.wikipedia.org/wiki/Einstichproben-t-Test, b.