

R-Vorkurs

Teil 3

Martin Arnold & Jens Klenke

15.04.2021



Übersicht

1. Verteilungen und Zufallszahlen
2. Grafiken
3. Lineare Regression

Verteilungen und Zufallszahlen

Verteilungen

R kennt standardmäßig viele diskrete Verteilungen, z.B.

- Binomialverteilung (`binom`)
- Geometrische Verteilung (`geom`)
- Hypergeometrische Verteilung (`hyper`)
- Poisson-Verteilung (`pois`)
- ...

...sowie stetige Verteilungen:

- Normalverteilung (`norm`)
- t-Verteilung (`t`)
- Chi-Quadrat-Verteilung (`chisq`)
- F-Verteilung (`f`)
- Gleichverteilung (`unif`)
- ...

Verteilungen

Verteilungen - Funktionen

Für jede Verteilung stehen hierbei vier klassische Funktionen zur Verfügung:

- Berechnung von Punktwahrscheinlichkeiten/-dichten \Rightarrow Präfix **d** (density)
- Werte der Verteilungsfunktion \Rightarrow Präfix **p** (probability)
- Berechnen von Quantilen \Rightarrow Präfix **q** (quantile)
- Erzeugen von 'Zufallszahlen' \Rightarrow Präfix **r** (random)

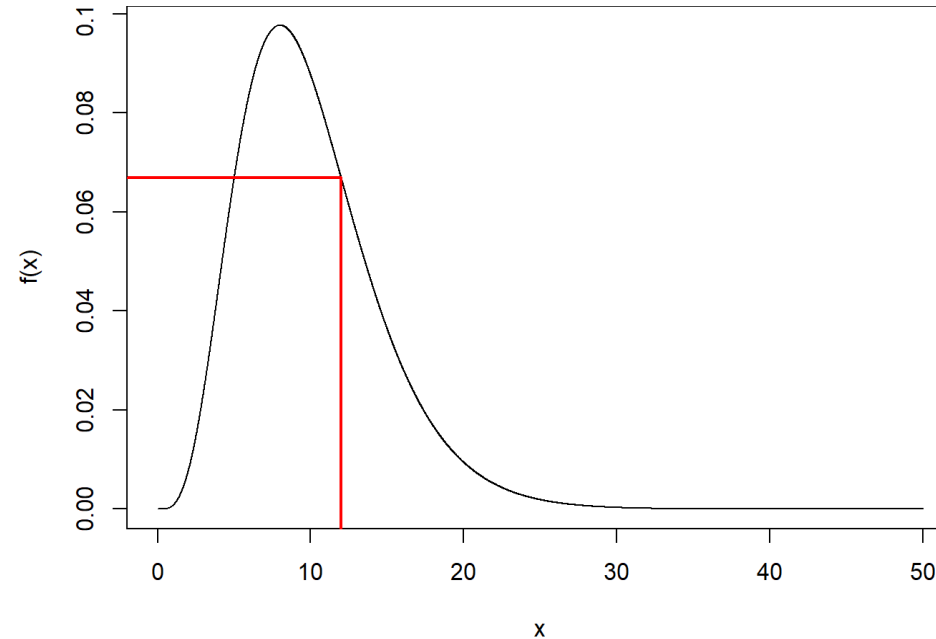
\Rightarrow Name der Funktion = Präfix + Verteilungskürzel (siehe vorherige Folie)

Beispiel:

- Quantil der Normalverteilung: **q** + norm = `qnorm()`
- Verteilungsfunktion der Binomialverteilung: **p** + binom = `pbinom()`
- etc.

Verteilungen

Beispiel - Chi-Quadrat-Verteilung



```
dchisq(12, df = 10) # Dichte an der Stelle x=12
```

```
## [1] 0.06692631
```

Verteilungen

Beispiel - Chi-Quadrat-Verteilung - Fortsetzung

Analog erhalten wir für die gleiche Verteilung einen bestimmten Wert der Verteilungsfunktion...

```
pchisq(8, df = 10)  # Wert der Verteilungsfunktion an der Stelle x=8
```

```
## [1] 0.3711631
```

...ein bestimmtes Quantil

```
qchisq(0.5, df = 10)  # 0.5-Quantil der Verteilung
```

```
## [1] 9.341818
```

...sowie Zufallszahlen

```
rchisq(5, df = 10)  # 5 chi^2-verteilte Zufallszahlen
```

```
## [1] 25.857183 12.706932 9.388072 6.988605 6.622895
```

Verteilungen

Seeds

Wiederholen wir den Code von gerade, bekommen wir (wie erwartet) andere Zahlen.

```
rchisq(5, df = 10)
```

```
## [1]  9.361119 10.504950  5.448934  6.064690  2.002800
```

Manchmal erscheint es aber sinnvoll, Ergebnisse reproduzierbar zu machen. Dann bietet es sich an, einen sog. *seed* zu benutzen. Damit können an mehreren Rechnern die selben Zufallszahlen erzeugt werden.

```
set.seed(385)  
rchisq(5, df = 10)
```

```
## [1] 13.158928 14.475007  7.448275 18.419976  8.822344
```

```
set.seed(385)  
rchisq(5, df = 10)
```

```
## [1] 13.158928 14.475007  7.448275 18.419976  8.822344
```

Welchen seed man benutzt ist unerheblich, Hauptsache man benutzt denselben.

Verteilungen

sample()

Neben der Erzeugung von Zufallszahlen aus vordefinierten Verteilungen, kann man auch 'eigene' Zufallsexperimente durchführen

```
# Beispiel Münzwurf:  
  
# Mögliche Ergebnisse definieren  
K_Z <- c('Kopf', 'Zahl')  
  
# Fünfmaliger Wurf einer fairen Münze  
sample(K_Z, size = 5, replace = TRUE)
```

```
## [1] "Kopf" "Kopf" "Zahl" "Zahl" "Kopf"
```

```
# Fünfmaliger Wurf einer unfairen Münze  
sample(K_Z, size = 5, replace = TRUE, prob = c(0.8, 0.2))
```

```
## [1] "Kopf" "Kopf" "Kopf" "Zahl" "Kopf"
```


Verteilungen

Übungsaufgaben

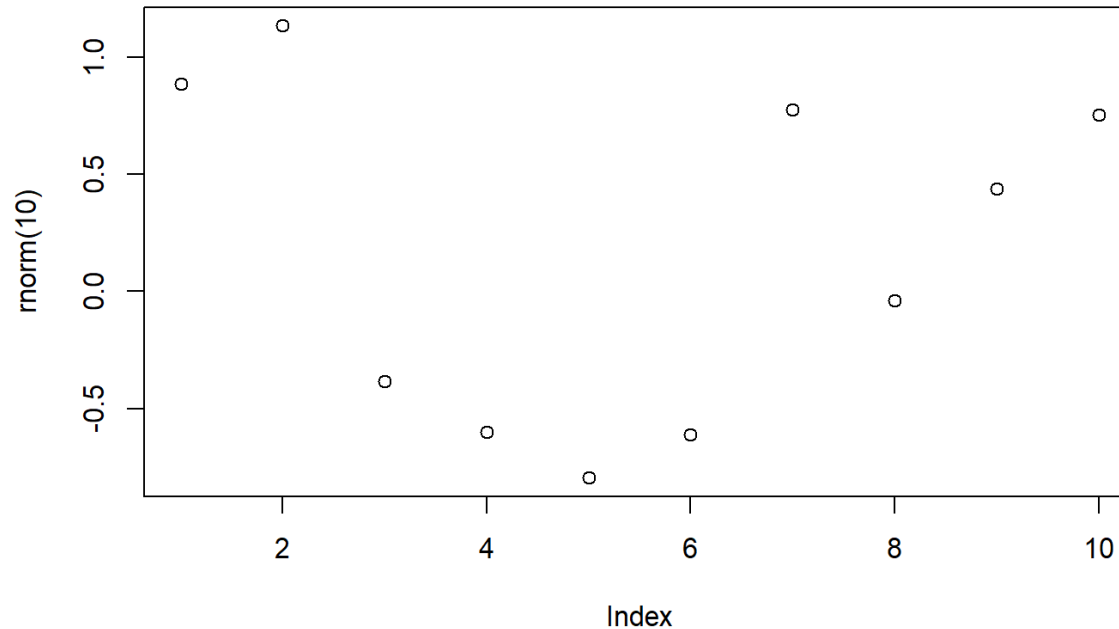
1. Sei $X \sim t(5)$. Berechnen Sie $P(X < 6)$, $P(3 < X \leq 7)$ und $P(X > 4)$.
2. Berechnen Sie das 0.95-Quantil einer $F(4, 5)$ -verteilten Zufallsvariable.
3. Berechnen Sie die Wahrscheinlichkeit dafür, den Jackpot im Lotto zu gewinnen (d.h. 6 Richtige aus 49). Vernachlässigen Sie bei Ihrer Berechnung Zusatz- oder Superzahlen (*Hinweis*: Benutzen Sie die hypergeometrische Verteilung).
4. Erzeugen Sie 20 $\chi^2(5)$ -verteilte Zufallszahlen ohne (!) dabei die `rchisq()`-Funktion zu benutzen (*Hinweis*: $\chi^2(n) = \sum_{i=1}^n Z_i^2$ mit $Z_i \sim \mathcal{N}(0, 1)$ für alle $i = 1, \dots, n$).
5. Ziehen Sie 10-mal standardnormalverteilte Zufallszahlen vom Umfang $n = 10000$ und berechnen Sie für jeden Durchlauf das arithmetische Mittel. Schauen Sie sich danach alle 10 Mittelwerte an. Was fällt Ihnen auf? Zusatz: Führen Sie dieselbe Aufgabe mit $X \sim t(1)$ -verteilten Zufallszahlen (sog. Cauchy-Verteilung) durch. Was fällt Ihnen nun auf? Können Sie sich das Ergebnis erklären? Schauen Sie sich ggf. den Wikipedia-Eintrag zur [Cauchy-Verteilung](#) an.
6. Zusatz: Verifizieren Sie, dass das Integral über die Dichtefunktion einer $\chi^2(15)$ -verteilten Zufallsvariable 1 ist. (Zum Integrieren in R können Sie `integrate()` benutzen.)

Grafiken

Basic Plot

Mit R ist es sehr leicht Grafiken zu erstellen, z.B. einen einfachen *Scatterplot* ...

```
plot(rnorm(10))
```

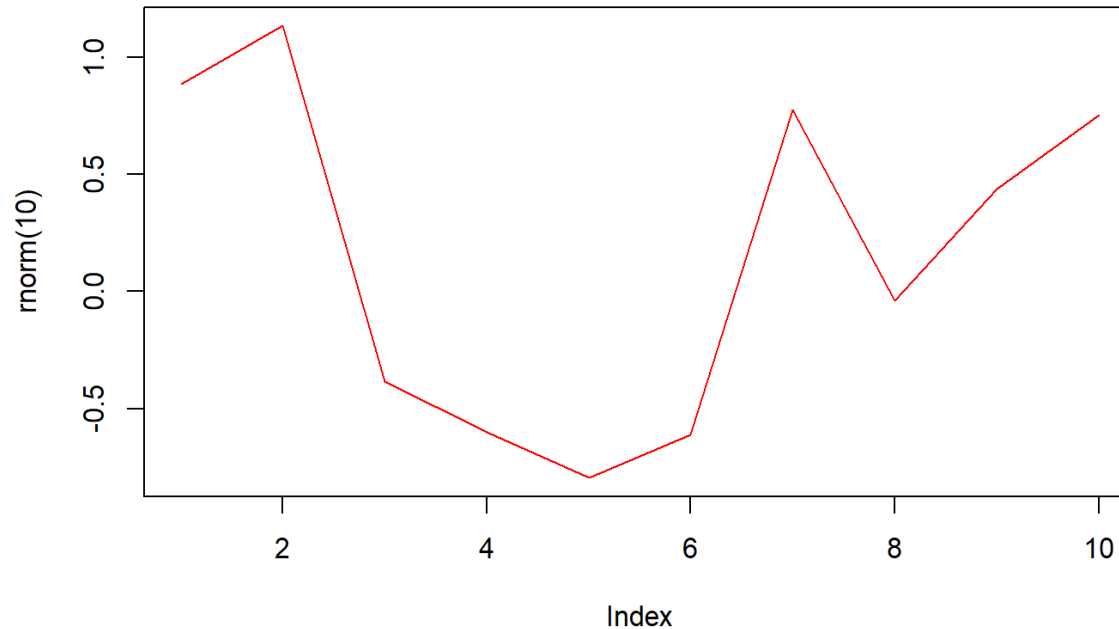


Grafiken

Basic Plot

...oder einen Linienplot in roter Farbe

```
plot(rnorm(10), type = 'l', col = 'red')
```



Grafiken

Cosmetics

Neben Parametern wie `col` können auch noch viele weitere kosmetische Dinge im Plot angepasst werden, z.B.:

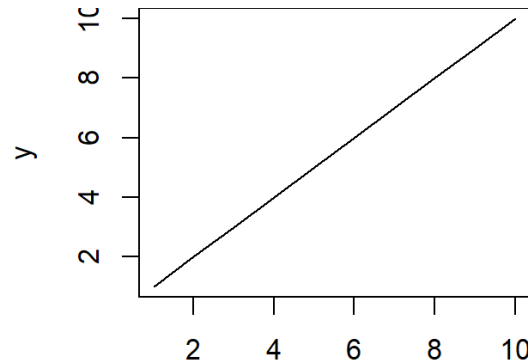
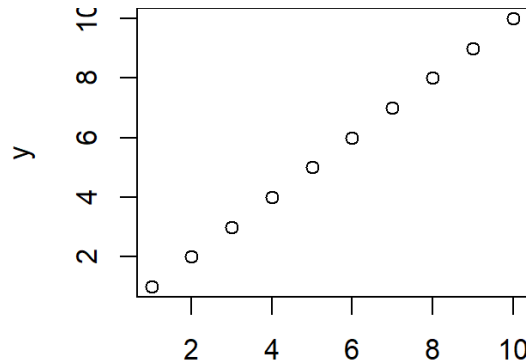
- `xlab/ylab`: Beschriftung x- bzw y-Achse
- `xlim/ylim`: Darzustellender Bereich von x- bzw. y-Achse
- `main/sub`: Titel/Untertitel der Grafik
- `pch`: Darstellung von Punkten in einer Grafik (Kreis, Quadrat, etc.)
- `lty/lwd`: Darstellung (durchgezogen, gestrichelt, etc.) und Breite von Linien in einer Grafik
- uvm. (siehe `?par`)

Grafiken

Mehrere Grafiken

Will man mehrere Grafiken in einem Fenster darstellen, kann man dies über den Parameter `mfrow` machen

```
par(mfrow = c(1, 2)) # Zwei Plots in einer Zeile (bzw. über zwei Spalten)
plot(1:10, xlab = 'x', ylab = 'y')
plot(1:10, xlab = 'x', ylab = 'y', type = 'l')
```



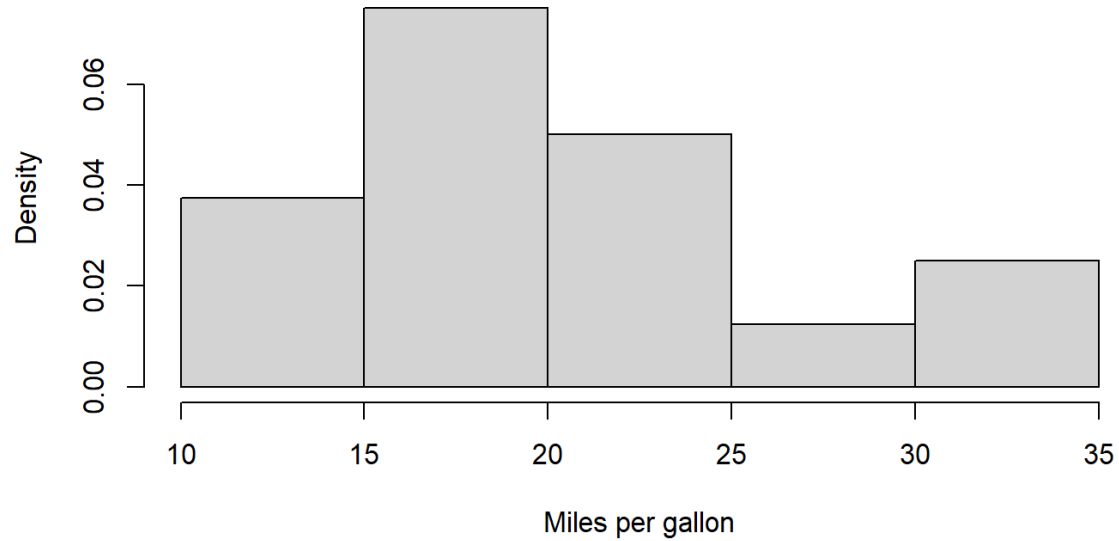
Man beachte, dass dies eine globale Einstellung ist. Will man also wieder zum ursprünglichen Set-Up zurück, sollte man es über `dev.off()` zurücksetzen.

Grafiken

Weitere Grafikenformen

`hist()`, `boxplot()`, `barplot()` und `pie()`.

Beispiel:



Grafiken

Funktionen die grafische Elemente hinzufügen

Es gibt auch Funktionen, die eine bestehende Grafik voraussetzen, bspw.

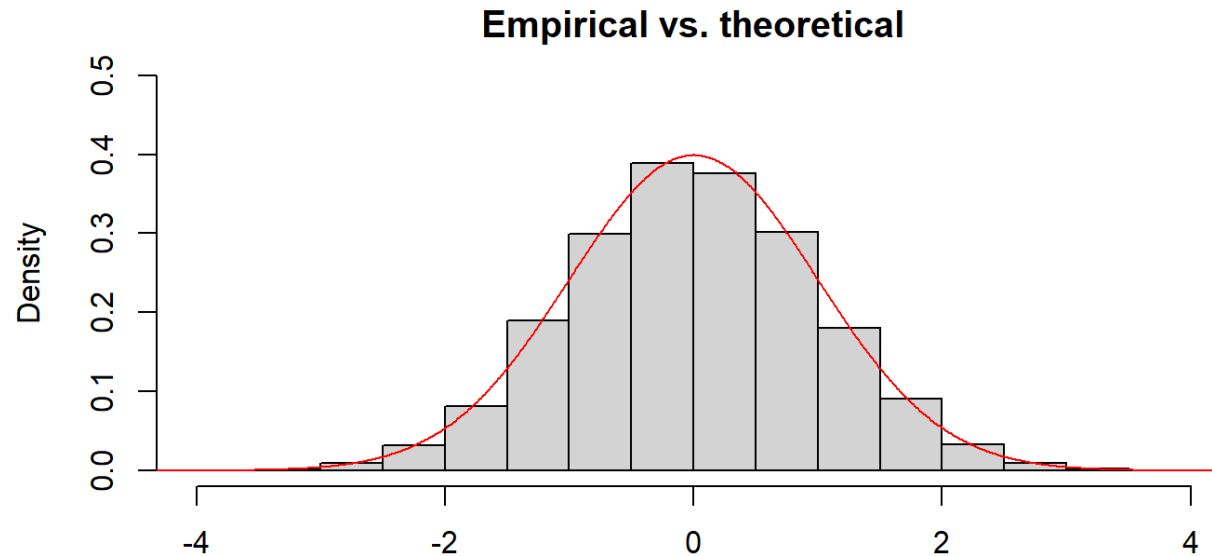
- `lines()`
- `points()`
- `abline()`
- `legend()`
- `text()`
- `arrows()`

Bis auf `abline()` erwarten alle Funktionen X- und Y-Koordinaten.

Grafiken

Funktionen die grafische Elemente hinzufügen - Beispiel

```
hist(rnorm(10000), freq = F, ylim = c(0, 0.5), xlab = '', main = 'Empirical vs. theoretical')  
x <- seq(-5, 5, 0.01)  
y <- dnorm(x)  
lines(x, y, col = 'red')
```



Grafiken

Übungsaufgaben

```
set.seed(385)
results <- rnorm(1000, mean = 100, sd = 15)
```

1. Kopieren Sie obigen Code und nehmen Sie an, dass dieser eine IQ-Testreihe mit 1000 Probanden simuliert. Zeichnen Sie ein Histogramm der Ergebnisse. Geben Sie Ihrem Plot anschließend eine passende Überschrift sowie passende Achsenbeschriftungen. Spezifizieren Sie darüber hinaus den Bereich von X- und Y-Achse auf $[40, 160]$ bzw. $[0, 0.03]$.
2. Hinterlegen Sie dem Plot die, dem IQ zugrundeliegende, theoretische Dichtefunktion, d.h. eine Normalverteilung mit $\mu = 100$ und $\sigma = 15$. Wählen Sie als Zeichenfarbe rot und machen Sie die einzuzeichnende Linie etwas breiter.
3. Zeichnen Sie einen Punkt in Form eines Dreiecks an das Maximum der theoretischen Dichte. Wählen Sie als Farbe blau.
4. Kennzeichnen Sie sowohl das 0.025- als auch das 0.975-Quantil der theoretischen Dichte, in dem Sie Vertikalen an diesen Punkten einzeichnen. Wählen Sie als Farbe grün.

Lineare Regression

Modell

Einfaches (univariates) lineares Regressionsmodell:

$$Y_t = \beta_0 + \beta_1 X_t + u_t, \quad t = 1, \dots, T$$

mit

- Y_t - Regressand (zu erklärende Variable),
- X_t - Regressor (erklärende Variable),
- u_t - Fehlerterm.

über z.B. OLS können wir dann die geschätzten Koeffizienten $\widehat{\beta}_0$ und $\widehat{\beta}_1$ erhalten.

Beispiel:

Im Datensatz **mtcars** sind etliche Informationen zu verschiedenen Automodellen hinterlegt. Wir vermuten nun, dass die Reichweite eines Autos (*mpg*) davon abhängt, wie schwer es ist (*wt*). D.h. wir haben folgendes geschätztes Modell:

$$\widehat{mpg}_t = \widehat{\beta}_0 + \widehat{\beta}_1 wt_t$$

(Welche Richtung des Zusammenhangs vermuten Sie?)

Lineare Regression

Parameterschätzung

Einfache lineare Regression über `lm()` (= linear model)

```
model <- lm(mpg ~ wt, data = mtcars)
```

Die Ergebnisse werden in einer Liste gespeichert

```
names(model)
```

```
## [1] "coefficients" "residuals"    "effects"      "rank"         "fitted.values"  
## [6] "assign"       "qr"           "df.residual"  "xlevels"      "call"  
## [11] "terms"       "model"
```

Lineare Regression

Regressionsoutput

Einen kompakten Überblick über die wichtigsten Informationen erhält man auch mit `summary()`

```
summary(model)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   37.2851     1.8776  19.858 < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528,    Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

Lineare Regression

Punktewolke und Regressionslinie

Grafisch sieht das Ganze dann so aus:

```
plot(mtcars$wt, mtcars$mpg, xlab = 'Weight', ylab = 'Miles per gallon')  
abline(model, col = 'red')
```

Lineare Regression

Vorhersage

Haben wir das Regressionsmodell geschätzt, können wir Vorhersagen treffen. Nehmen wir an, dass ein neues Auto mit einem Gewicht von $X_{new} = 3$ (in 1000 lbs) auf den Markt kommt. Was für eine Reichweite wird dieses dann in etwa haben?

```
new_weight <- data.frame(wt = 3)
predict(model, newdata = new_weight)
```

```
##           1
## 21.25171
```

Lineare Regression

Dummy-Variablen/Regression ohne Achsenabschnitt

Dummyvariablen (hier Getriebeart *am* mit Automatik = 0, Manuell = 1) können über die Funktion `factor()` als zusätzlicher Regressor mitaufgenommen werden.

```
model_wd <- lm(mpg ~ wt + factor(am), data = mtcars)
model_wd$coefficients
```

```
## (Intercept)          wt factor(am)1
## 37.32155131 -5.35281145 -0.02361522
```

(Wie sind die Koeffizienten zu interpretieren?)

Will man eine Regression ohne Achsenabschnitt (β_0) durchführen, bedarf es einer `-1` (oder alternativ `+0`) am Ende der Regressionsformel.

```
model_wd_no_int <- lm(mpg ~ wt + factor(am) - 1, data = mtcars)
model_wd_no_int$coefficients
```

```
##          wt factor(am)0 factor(am)1
## -5.352811  37.321551   37.297936
```

(Was hat sich an der Interpretation der Koeffizienten geändert?)

Lineare Regression

Übungsaufgaben

1. Betrachten Sie im Folgenden den Datensatz **faithful**, der Daten zum Old Faithful Geysir im Yellowstone Nationalpark enthält. Sowohl die Dauer einer Eruption in Min. (`eruptions`) als auch die Wartezeit bis zur nächsten Eruption in Min. (`waiting`) sind als Variable im Datensatz verfügbar. Unterstellen Sie folgendes Regressionsmodell

$$\text{waiting}_t = \beta_0 + \beta_1 \text{eruptions}_t + u_t$$

und schätzen Sie die entsprechenden Parameter $\widehat{\beta}_0, \widehat{\beta}_1$. Zeichnen Sie anschließend eine geeignete Grafik und interpretieren Sie diese.

2. Verschaffen Sie sich mit `summary()` einen Überblick über ihr in 1.) erhaltenes Ergebnis. Interpretieren Sie die geschätzten Koeffizienten und speichern Sie anschließend das R^2 (*Multiple R-squared*) in der Variablen `R2` ab. (Hinweis: Schauen Sie sich die, beim Ausführen von `summary()`, ausgegebene Datenstruktur genauer an.)
3. Sie beobachten einen zusätzlichen Datenpunkt für die Dauer einer Eruption von $X_{new} = 4$. Sagen Sie die entsprechende Wartezeit bis zur nächsten Eruption vorher.