R-Vorkurs

Teil 3

Jens Klenke

31.03.2022

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

R-Funktionen

Für jede Verteilung stehen vier 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)

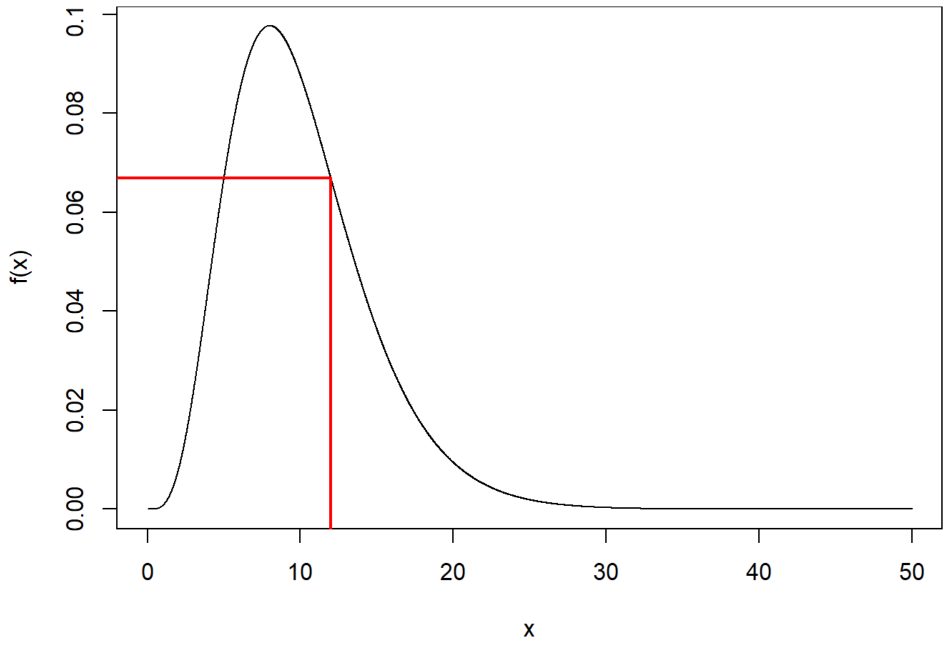
$\Longrightarrow$ Name der Funktion = Präfix + Verteilungskürzel (siehe vorherige Folie)

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

Analog erhalten wir für die gleiche Verteilung Werte 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]  7.693207  6.297943 14.408096  8.605373
## [5]  7.412510
```

Verteilungen

Seeds

Wiederholen wir den Code von Slide 7, erhalten wir andere zufällige Zahlen.

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

```
## [1]  9.605476  5.556216 10.051123 13.515931
## [5] 10.579882
```

Manchmal wollen wir Ergebnisse reproduzierbar machen. Dann muss ein sog. *seed* gesetzt werden. Damit können an jedem Computer die selben Zufallszahlen erzeugt werden.

```
set.seed(385)      # 385 ist ein Beispiel -- probiert andere seeds aus!
rchisq(5, df = 10)
```

```
## [1] 13.158928 14.475007  7.448275 18.419976
## [5]  8.822344
```

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

```
## [1] 13.158928 14.475007  7.448275 18.419976
## [5]  8.822344
```

Verteilungen

`sample()`

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

Beispiel Münzwurf:

```r
# Mögliche Ergebnisse definieren
K_Z <- c('Kopf', 'Zahl')
```

Fünfmaliges Werfen einer *fairen* Münze:

```r
# (*standardwert* prob = 1/length(K_Z) )
sample(K_Z, size = 5, replace = TRUE, prob = c(0.5, 0.5))
```

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

Fünfmaliges Werfen einer *unfairen* Münze:

```r
sample(K_Z, size = 5, replace = TRUE, prob = c(0.8, 0.2))
```

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

Verteilungen

Übungsaufgaben

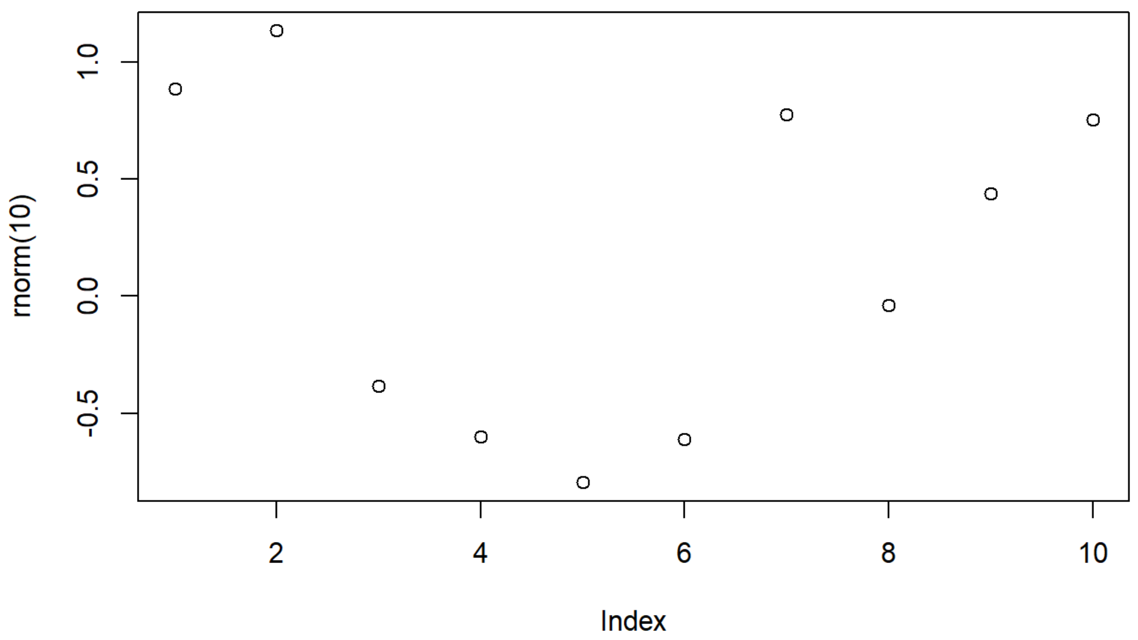
1. Sei $X \sim t(5)$. Berechne $P(X < 6)$, $P(3 < X \leq 7)$ und $P(X > 4)$.
2. Berechne das 0.95-Quantil einer $F(4, 5)$-verteilten Zufallsvariable.
3. Berechne die Wahrscheinlichkeit dafür, den Jackpot im Lotto zu gewinnen (d.h. 6 Richtige aus 49). Vernachlässige Zusatz- und Superzahlen (*Hinweis*: Benutze die hypergeometrische Verteilung).
4. Erzeuge 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 $i = 1, \ldots, n$).
5. Ziehe 10 mal $n = 10000$ standardnormalverteilte Zufallszahlen und berechne für jede Ziehung das arithmetische Mittel. Betrachte alle 10 Mittelwerte. Was fällt auf? Zusatz: Wiederhole die Simulation mit $X \sim t(1)$-verteilten Zufallszahlen (Cauchy-Verteilung) Was fällt nun auf? Erkläre! (*Hinweis*: Wikipedia-Eintrag zur Cauchy-Verteilung).
6. Zusatz: Prüfe, dass das Integral über die Dichtefunktion einer $\chi^2(15)$-verteilten Zufallsvariable 1 ist. (*Hinweis*: Nutze `integrate()` zum Integrieren einer R-Funktion)

Grafiken

Basic Plot

Mit R ist es einfach Grafiken zu erstellen, z. B. einen *Scatterplot*...

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

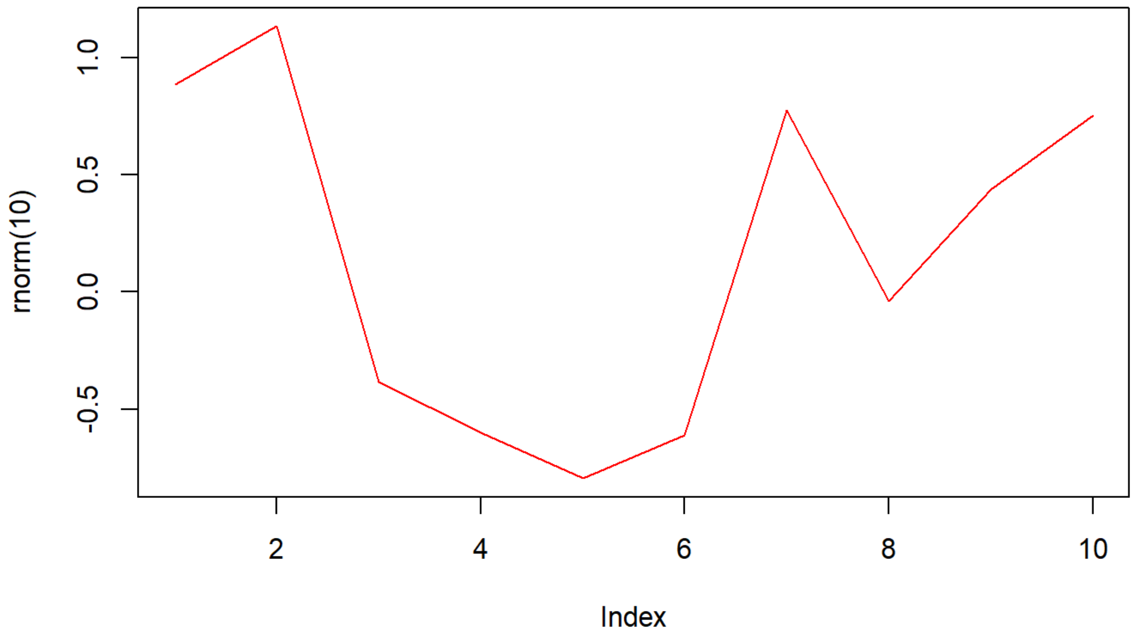


Grafiken

Basic Plot

...oder einen Linienplot in roter Farbe

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



Grafiken

Cosmetics

Neben Argumenten wie `col` können weitere Eigenschaften des Plots angepasst werden, z.B.:

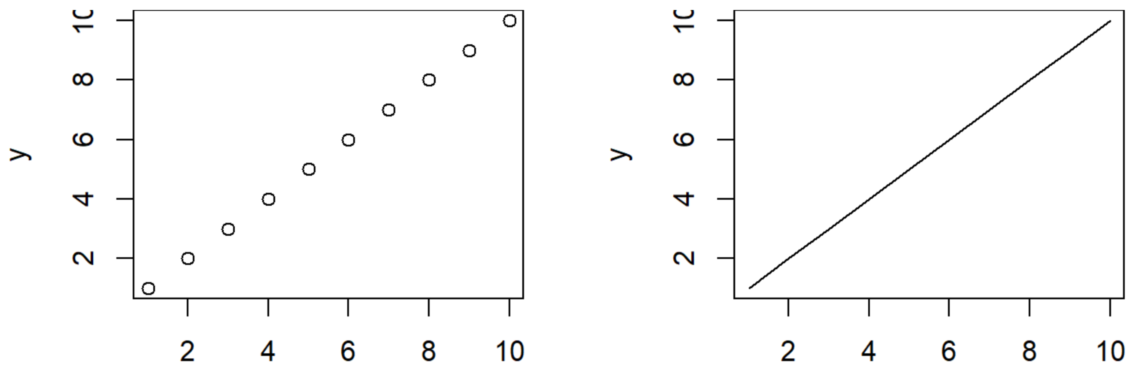
- `xlab`/`ylab`: Beschriftung von x-/y-Achse
- `xlim`/`ylim`: Darzustellender Bereich von x-/y-Achse
- `main`/`sub`: Titel/Untertitel der Grafik
- `pch`: Symbol für Datenpunkte in einer Grafik (Kreis, Quadrat, Dreieck etc.)
- `lty`/`lwd`: Darstellung (durchgezogen, gestrichelt, etc.) und Breite von Linien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

Grafiken

Mehrere Grafiken

Für mehrere Grafiken in einem plot muss der Parameter `mfrow` genutzt werden:

```
par(mfrow = c(1, 2))  # Zwei Plots in einer Zeile (bzw. in 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, muss man mit `dev.off()` zurücksetzen.

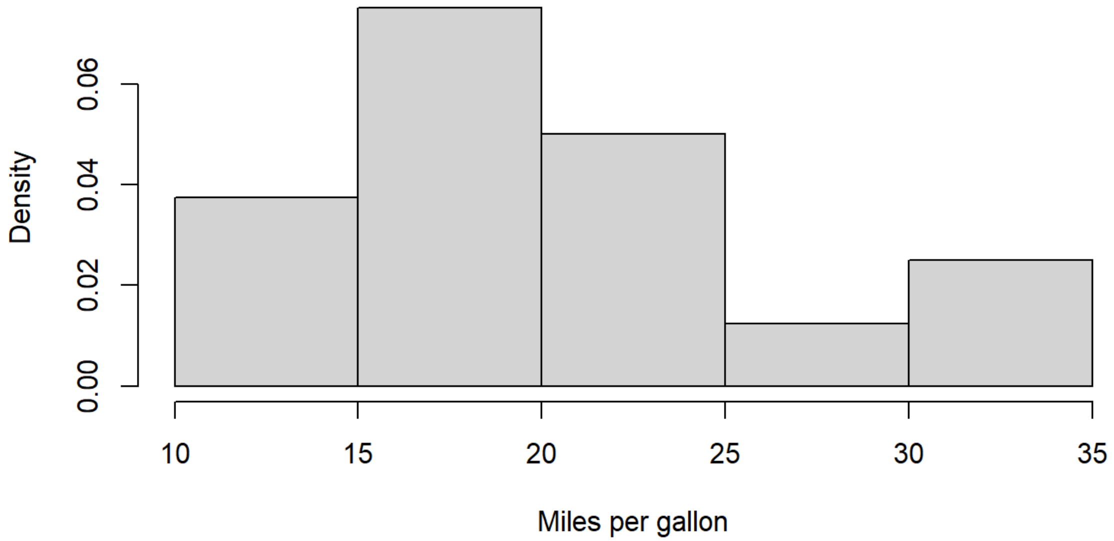
Grafiken

Weitere Grafikenformen

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

Beispiel:

```
hist(mtcars$mpg, breaks = 5, freq = F, main = '', xlab = 'Miles per gallon')
```



Grafiken

Funktionen, die grafische Elemente hinzufügen

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

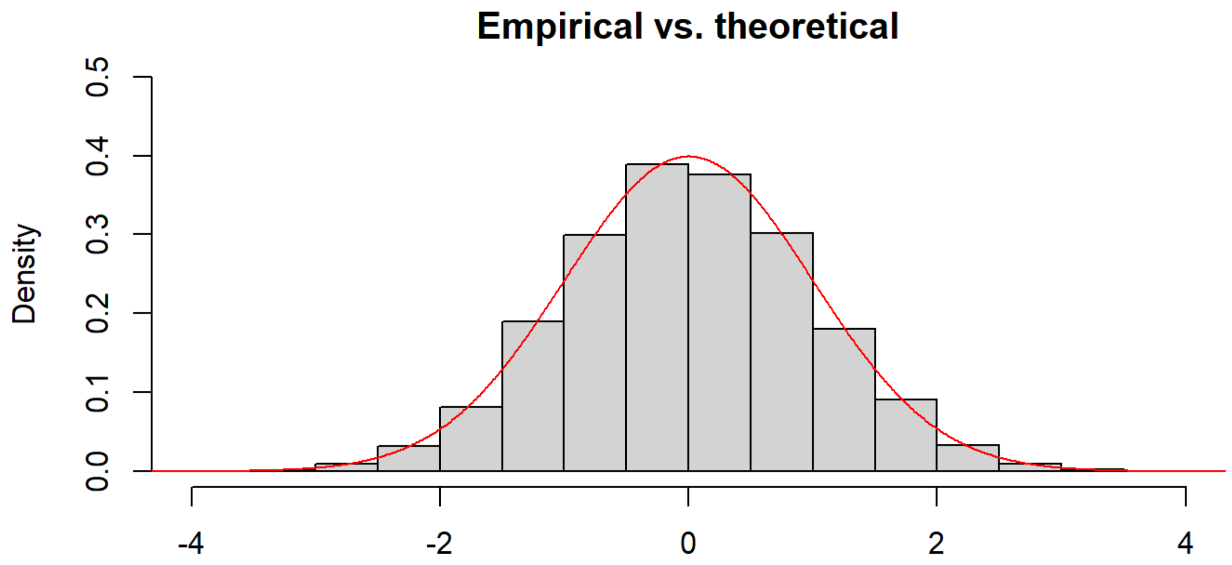
- `lines()`
- `points()`
- `abline()`
- `legend()`
- `text()`
- `arrows()`

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

Grafiken

Funktionen, die grafische Elemente hinzufügen – Beispiel

```r
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

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

1. Kopiere obigen Code und nimm an, dass dieser eine IQ-Testreihe mit 1000 Probanden simuliert. Zeichne ein Histogram der Ergebnisse. Beschrifte den plot und setze die Bereiche von X- und Y-Achse auf $[40, 160]$ und $[0, 0.03]$.
2. Zeichne die zugrundeliegende theoretische Dichtefunktion, d.h. eine Normalverteilung mit $\mu = 100$ und $\sigma = 15$, in den Plot.
3. Zeichne ein blaues Dreieck an das Maximum der theoretischen Dichte.
4. Markiere sowohl das 0.025- als auch das 0.975-Quantil der theoretischen Dichte mit vertikalen grünen Linien.

Lineare Regression

Modell

Einfaches (univariates) lineares Regressionsmodell:

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

wobei

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

Mit KQ-Methode können wir die Koeffizientenschätzer $\widehat{\beta_0}$ und $\widehat{\beta_1}$ erhalten.

Beispiel:

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

$$mpg_t = \beta_0 + \beta_1 wt_t + u_t$$

(Welche Richtung hat dieser Zusammenhangs vermutlich?)

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"
##  [3] "effects"       "rank"
##  [5] "fitted.values" "assign"
##  [7] "qr"            "df.residual"
##  [9] "xlevels"       "call"
## [11] "terms"         "model"
```

Lineare Regression

Regressionsoutput

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

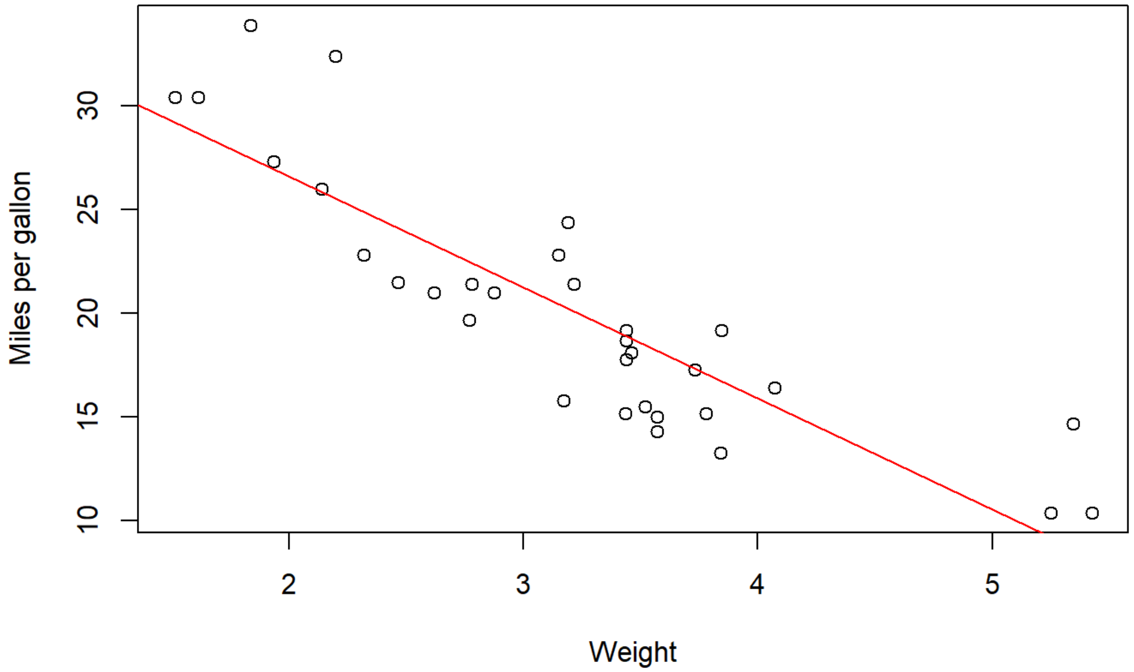
```
summary(model)
```

```
##
## 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
## (Intercept)  37.2851     1.8776  19.858
## wt           -5.3445     0.5591  -9.559
##             Pr(>|t|)
## (Intercept)  < 2e-16 ***
## wt          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 Regressionsgerade

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



Lineare Regression

Vorhersage

Mit dem geschätzten Modell können wir Vorhersagen machen. Angenommen ein neues Auto mit einem Gewicht von $X_{new} = 3$ (in 1000 lbs) kommt auf den Markt. Was für eine Reichweite schätzen wir für dieses Auto?

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

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

Lineare Regression

Dummy-Variablen/Regression ohne Achsenabschnitt

Kategorische Variablen (hier Getriebeart *am* mit Automatik = 0, manuell = 1) können über die Funktion `factor()` als 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
```

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
```

Lineare Regression

Übungsaufgaben

1. Betrachte 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 Variablen im Datensatz verfügbar. Unterstelle das Regressionsmodell

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

   und schätze die Parameter β_0, β_1. Zeichne anschließend eine geeignete Grafik und interpretiere.
2. Erhalte mit `summary()` einen Überblick über die Regression aus 1. Interpretiere die geschätzten Koeffizienten und speichere anschließend das R^2 (*Multiple R-squared*) in `R2`.
3. Angenommen wir beobachten einen weiteren Datenpunkt für die Dauer einer Eruption $X_{new} = 4$. Triff eine Vorhersage für die Wartezeit bis zur nächsten Eruption!