

Replikations-Skript zur Vorlesung 8: Formalia

Claudius Gräbner

KW 50 2020

Contents

1	Wachstumsraten	2
1.1	Darstellung durch Logarithmus	2
1.2	Gruppierung und Berechnung	2
2	Differenzialrechnung und Optimierung	3
2.1	Ableitungen	4
2.2	Optimierung	5
3	Lineare Algebra	8
3.1	Beispiel Keynes	8
3.2	Beispiel OLS	8
3.3	Herleitung des OLS Schätzers (optional)	10
4	Verteilungen	11
4.1	Theoretische und empirische Verteilungen	11
4.2	Grafische Darstellung	14

In diesem Dokument werden alle Abbildungen und Tabellen aus der siebten Vorlesung repliziert. Dabei gebe ich der Info wegen *allen* R Code. Entsprechend sind bestimmt auch einige Befehle dabei, die Sie jetzt noch nicht kennen.

Zudem nehme ich an, dass im Arbeitsverzeichnis der Ordner `data/T7/` existiert und in diesem folgende Datensätze enthalten sind (diese sind über die Repository zur Vorlesung verfügbar): `bip_growth.csv` und `AutoDaten.csv`.

Folgende Pakete werden zudem in diesem Skript verwendet:

```
library(tidyverse)
library(data.table)
library(ggpubr)
library(latex2exp)
library(icaeDesign)
library(here)
library(matlib)
library(fitdistrplus)
```

Beachten Sie, dass das Paket `icaeDesign` nicht über die zentrale Paketverwaltung verfügbar ist. Es muss folgendermaßen installiert werden:

```
devtools::install_github("graebner/icaeDesign")
```

1 Wachstumsraten

1.1 Darstellung durch Logarithmus

Wir starten mit $x = 2$ und lassen die Variable über 100 Schritte mit mit 4% pro Zeitschritt wachsen:

```
x <- c(2, rep(NA, 99))

for (i in 1:length(x)){
  x[i+1] <- x[i] * 1.04
}

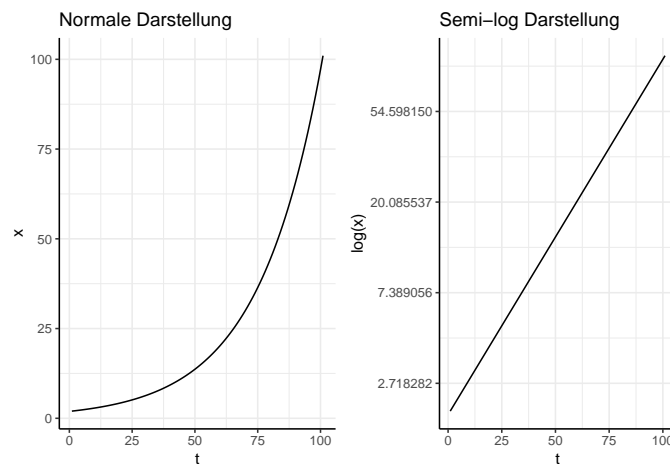
x_data <- data.frame(
  t = 1:101,
  x = x
)
```

Hier sehen wir, wie sich in der semi-log Darstellung eine gerade Linie ergibt:

```
p1 <- ggplot(
  data = x_data,
  mapping = aes(x=t, y=x)) +
  geom_line() +
  labs(title = "Normale Darstellung") +
  theme_bw() + theme(
    panel.border = element_blank(),
    axis.line = element_line()
  )

p2 <- p1 +
  labs(title = "Semi-log Darstellung", y="log(x)") +
  scale_y_continuous(trans = "log")

ggpubr::ggarrange(p1, p2, ncol = 2)
```



1.2 Gruppierung und Berechnung

```
beispiel_daten <- fread(
  here("data/T7/bip_growth.csv")
) %>%
```

```

  arrange(-year)
beispiel_daten

##      country      BIP year
##  1: Austria 37941.04 2018
##  2: Germany 35866.00 2018
##  3: Austria 37140.79 2017
##  4: Germany 35477.89 2017
##  5: Austria 36469.39 2016
##  6: Germany 34858.79 2016
##  7: Austria 36129.03 2015
##  8: Germany 34370.64 2015
##  9: Austria 36123.43 2014
## 10: Germany 34076.90 2014

beispiel_daten <- beispiel_daten %>%
  arrange(country, year) %>%
  group_by(country) %>%
  mutate(
    BIP_Wachstum = (BIP-dplyr::lag(BIP))/abs(dplyr::lag(BIP))*100
  ) %>%
  ungroup()
beispiel_daten

## # A tibble: 10 x 4
##   country      BIP year BIP_Wachstum
##   <chr>      <dbl> <int>      <dbl>
## 1 Austria 36123.  2014         NA
## 2 Austria 36129.  2015        0.0155
## 3 Austria 36469.  2016         0.942
## 4 Austria 37141.  2017         1.84
## 5 Austria 37941.  2018         2.15
## 6 Germany 34077.  2014         NA
## 7 Germany 34371.  2015         0.862
## 8 Germany 34859.  2016         1.42
## 9 Germany 35478.  2017         1.78
## 10 Germany 35866.  2018         1.09

```

2 Differenzialrechnung und Optimierung

```

f_0 <- function(x) -x^2 + 4
f_0_d <- function(x) -2*x

data <- data.frame(x=seq(-2, 2, 0.05)) %>%
  rowwise() %>%
  mutate(y_2 = f_0(x),
         y_3 = f_0_d(x))

fun_max <- ggplot(data, aes(x=x, y=y_2)) +
  geom_line() +
  labs(
    title = TeX("$f(x)=-x^2 + 4$"),
    x="x", y="y"
  )

```

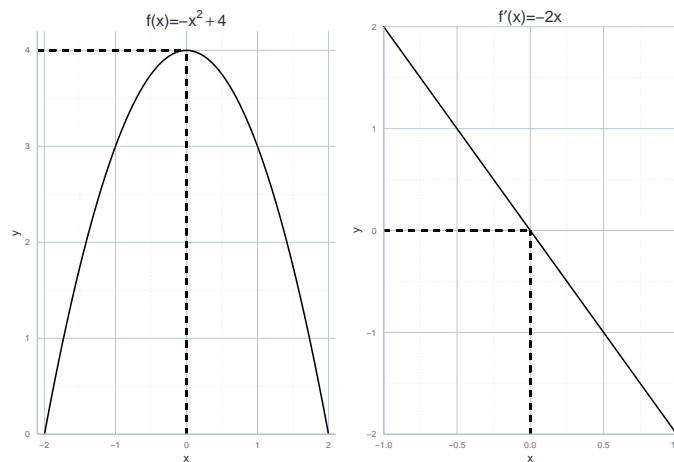
```

) +
geom_segment(x=0, xend=0, y=0, yend=4, linetype=2) +
geom_segment(x=-Inf, xend=0, y=4, yend=4, linetype=2) +
coord_cartesian(ylim = c(0, 4.2),
                xlim = c(-2.1, 2.1),
                expand = c(0)) +
theme_icae()

fun_deriv <- ggplot(data, aes(x=x, y=y_3)) +
  geom_line() +
  labs(
    title = TeX("$f'(x)=-2x$"),
    x="x", y="y"
  ) +
  geom_segment(x=0, xend=0, y=-Inf, yend=0, linetype=2) +
  geom_segment(x=-Inf, xend=0, y=0, yend=0, linetype=2) +
  coord_cartesian(ylim = c(-2, 2),
                xlim = c(-1, 1),
                expand = c(0)) +
  theme_icae()

comb_plot <- ggarrange(fun_max, fun_deriv, ncol = 2)
comb_plot

```



2.1 Ableitungen

```

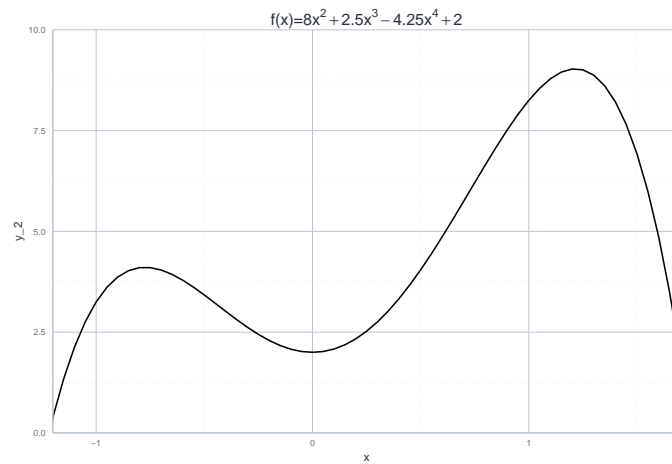
f_2 <- function(x) 8*x^2 + 2.5*x**3 - 4.25*x**4 + 2

data <- data.frame(x=seq(-2, 2, 0.05)) %>%
  rowwise() %>%
  mutate(y_2 = f_2(x))

mult_max <- ggplot(data, aes(x=x, y=y_2)) +
  geom_line() +
  ggtitle(TeX("$f(x)=8x^2 + 2.5x^3 - 4.25x^4 + 2$")) +
  coord_cartesian(ylim = c(0, 10),
                xlim = c(-1.2, 1.7), expand = c(0)) +
  theme_icae()

```

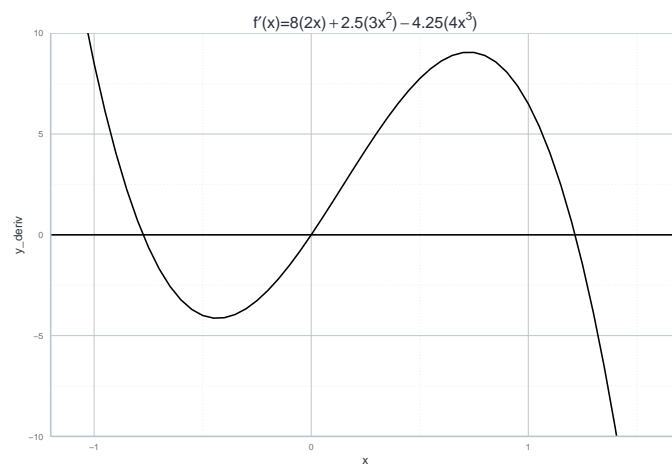
mult_max



```
f_expr <- expression(8*x^2 + 2.5*x^3 - 4.25*x^4 + 2)
f_deriv <- D(expr = f_expr, name = "x")
x <- seq(-2, 2, 0.05)
f_expr_vals <- eval(f_expr)

f_data <- data.frame(x=x,
                     y=f_expr_vals,
                     y_deriv = eval(f_deriv))

deriv_plot <- ggplot(f_data, aes(x=x, y=y_deriv)) +
  geom_line() +
  ggtitle(TeX("$f'(x)=8(2x) + 2.5(3x^2) - 4.25(4x^3)$")) +
  coord_cartesian(ylim = c(-10, 10),
                  xlim = c(-1.2, 1.7), expand = c(0)) +
  geom_hline(yintercept = 0) +
  theme_icae()
deriv_plot
```

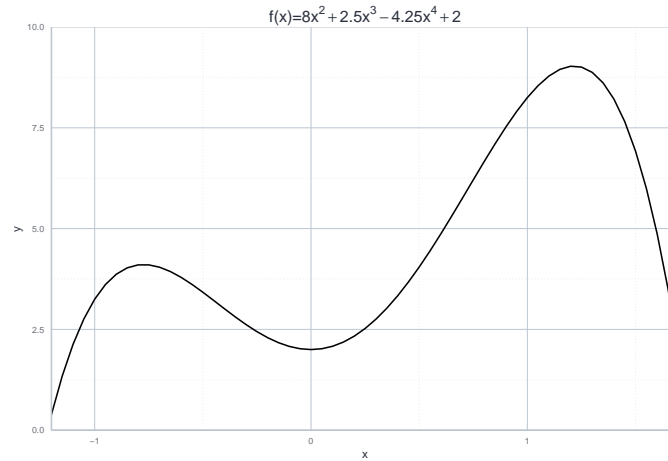


2.2 Optimierung

```
f_1 <- function(x) 8*x^2 + 2.5*x**3 - 4.25*x**4 + 2
```

```
data <- data.frame(x=seq(-2, 2, 0.05)) %>%
  rowwise() %>%
  mutate(y = f_1(x))

ggplot(data, aes(x=x, y=y)) +
  geom_line() +
  ggtitle(TeX("$f(x)=8x^2 + 2.5x^3 - 4.25x^4 + 2$")) +
  coord_cartesian(ylim = c(0, 10),
                  xlim = c(-1.2, 1.7), expand = c(0)) +
  theme_icae()
```



```
opt_obj <- optimize(f = f_1,
                    lower = -1.25, upper = 1.75,
                    maximum = FALSE)
```

```
opt_obj
```

```
## $minimum
## [1] -7.54766e-06
##
## $objective
## [1] 2
```

```
opt_obj <- optimize(f = f_1,
                    lower = -1.25, upper = 1.75,
                    maximum = TRUE)
```

```
opt_obj
```

```
## $maximum
## [1] 1.215492
##
## $objective
## [1] 9.032067
```

```
opt_obj <- optimize(f = f_1,
                    lower = -1.25, upper = 0,
                    maximum = TRUE)
```

```
opt_obj
```

```
## $maximum
## [1] -0.7743199
##
```

```
## $objective  
## [1] 4.108106
```

3 Lineare Algebra

3.1 Beispiel Keynes

Das Keynesianische Modell:

$$\begin{aligned}Y &= C + I + G \\ C &= a + bY\end{aligned}$$

Umgeformt:

$$\begin{aligned}Y - C &= I + G \\ -bY + C &= a\end{aligned}$$

$$\begin{pmatrix} 1 & -1 \\ -b & 1 \end{pmatrix} \times \begin{pmatrix} Y \\ C \end{pmatrix} = \begin{pmatrix} I + G \\ a \end{pmatrix}$$

$Ax = d$

Lösung in R für $G = 5$, $I = 4$, $a = 4$ und $b = 0.5$:

```
I_keynes <- 5
G_keynes <- 4
b_keynes <- 0.5
a_keynes <- 4

A_keynes <- matrix(c(1, -b_keynes, -1, 1), nrow = 2)
d_keynes <- matrix(c(I_keynes + G_keynes, a_keynes), ncol = 1)
Solve(A = A_keynes, b = d_keynes)

## x1    = 26
## x2    = 17
```

3.2 Beispiel OLS

Wir wissen nun, dass wir das lineare Regressionsmodell mit n Beobachtungen von p Variablen

$$\begin{aligned}y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \dots + \beta_p x_{1p} + \epsilon_1 \\ y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \dots + \beta_p x_{2p} + \epsilon_2 \\ &\vdots \\ y_n &= \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \dots + \beta_p x_{np} + \epsilon_n\end{aligned}$$

auch folgendermaßen schreiben können:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{1}$$

Nehmen wir jetzt folgenden Datensatz an:


```
ols_beispiel <- fread(here("data/T7/AutoDaten.csv"),
                      select = c("Auto"="character",
                                "Verbrauch"="double",
                                "PS"="double", "Zylinder"="double"))
```

ols_beispiel

```
##           Auto Verbrauch  PS Zylinder
## 1: Ford Pantera L      15.8 264        8
## 2:   Ferrari Dino      19.7 175        6
## 3: Maserati Bora       15.0 335        8
## 4:   Volvo 142E       21.4 109        4
```

Dies können wir schreiben als:

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \epsilon_1 \\ y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \epsilon_2 \\ y_3 &= \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + \epsilon_3 \\ y_4 &= \beta_0 + \beta_1 x_{41} + \beta_2 x_{42} + \epsilon_4 \end{aligned}$$

(2)

und mit Zahlen:

$$\begin{aligned} 15.8 &= \beta_0 + \beta_1 264 + \beta_2 8 + \epsilon_1 \\ 19.7 &= \beta_0 + \beta_1 175 + \beta_2 6 + \epsilon_2 \\ 15.0 &= \beta_0 + \beta_1 335 + \beta_2 8 + \epsilon_3 \\ 21.4 &= \beta_0 + \beta_1 109 + \beta_2 4 + \epsilon_4 \end{aligned}$$

(3)

Und als Matrix:

$$\begin{pmatrix} 1 & 264 & 8 \\ 1 & 175 & 6 \\ 1 & 335 & 8 \\ 1 & 109 & 4 \end{pmatrix} \times \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{pmatrix} = \begin{pmatrix} 15.8 \\ 19.7 \\ 15.0 \\ 21.4 \end{pmatrix}$$

Das können wir wiederum in R lösen:

```
ols_X <- matrix(c(1, 264, 8, 1, 175, 6, 1, 335, 8, 1, 109, 4),
                ncol = 3, byrow = T)
ols_y <- matrix(c(15.8, 19.7, 15.0, 21.4), ncol = 1)

solve(t(ols_X) %*% ols_X) %*% t(ols_X) %*% ols_y
```

```
##           [,1]
## [1,] 26.37086491
## [2,] -0.01783627
## [3,] -0.68592421
```

Oder direkt mit `lm()`:

```
lm(Verbrauch~PS+Zylinder, data = ols_beispiel)

##
## Call:
## lm(formula = Verbrauch ~ PS + Zylinder, data = ols_beispiel)
##
## Coefficients:
## (Intercept)          PS          Zylinder
##    26.37086    -0.01784    -0.68592
```

3.3 Herleitung des OLS Schätzers (optional)

Wir wissen bereits, dass die Residuen einer Schätzung gegeben sind durch:

$$e = y - X\hat{\beta}$$

Wir können die Summe der Residuen (RSS) in Matrixschreibweise schreiben als:

$$e'e = \begin{pmatrix} e_1 & e_2 & \dots & e_n \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} = \begin{pmatrix} e_1 \times e_1 & e_2 \times e_2 & \dots & e_n \times e_n \end{pmatrix}$$

Wir können dann schreiben:¹

$$\begin{aligned} e'e &= (y - X\hat{\beta})' (y - X\hat{\beta}) \\ &= y'y - \hat{\beta}' X'y - y' X\hat{\beta} + \hat{\beta}' X' X\hat{\beta} \\ &= y'y - 2\hat{\beta}' X'y + \hat{\beta}' X' X\hat{\beta} \end{aligned}$$

Wir wollen diesen Ausdruck nun minimieren. Dazu leiten wir nach dem Vektor der zu schätzenden Koeffizienten $\hat{\beta}$ ab:

$$\frac{\partial e'e}{\partial \hat{\beta}} = -2X'y + 2X'X\hat{\beta} = 0$$

Diese Gleichung können wir nun umformen zu:

$$\begin{aligned} 2X'X\hat{\beta} &= 2X'y \\ X'X\hat{\beta} &= X'y \end{aligned}$$

Da gilt, dass $(X'X)^{-1}(X'X) = I$ multiplizieren wir beide Seiten mit $(X'X)^{-1}$:

$$\begin{aligned} (X'X)^{-1} X'X\hat{\beta} &= (X'X)^{-1} X'y \\ \hat{\beta} &= (X'X)^{-1} (X'y) \end{aligned} \tag{4}$$

Damit haben wir den Schätzer für $\hat{\beta}$ hergeleitet.

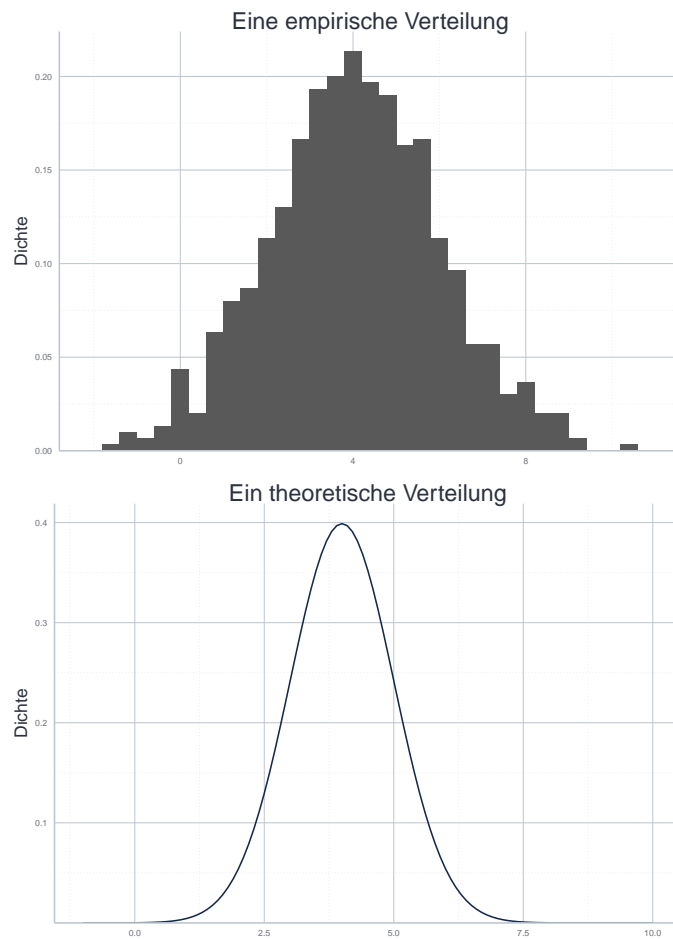
¹Beachte dabei, dass $y'X\hat{\beta} = (y'X\hat{\beta})' = \hat{\beta}'X'y$.

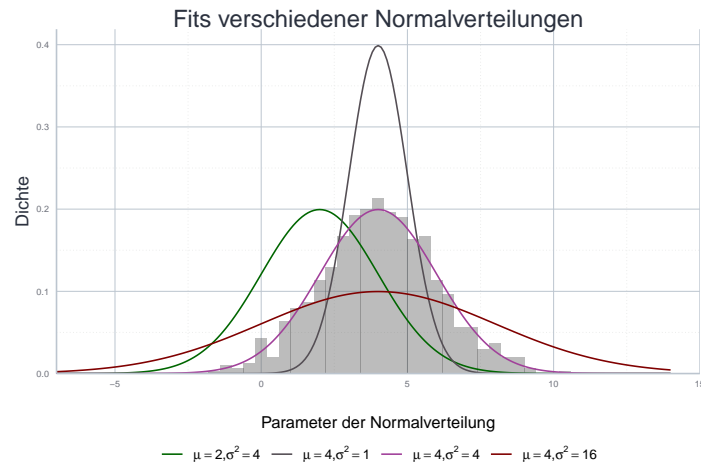
4 Verteilungen

4.1 Theoretische und empirische Verteilungen

Erstellen der Daten:

```
ggplot(data = sample_data) +  
  geom_histogram(mapping = aes(x=r, stat(density)), binwidth = 0.4) +  
  scale_x_continuous(expand = c(0, 1)) +  
  scale_y_continuous(name = "Dichte",  
    expand = expansion(c(0, 0.05), c(0, 0))) +  
  ggtitle("Eine empirische Verteilung") +  
  theme_icae() +  
  theme(axis.title.x = element_blank(),  
    plot.title = element_text(size = 16),  
    axis.title.y = element_text(size = 12))
```





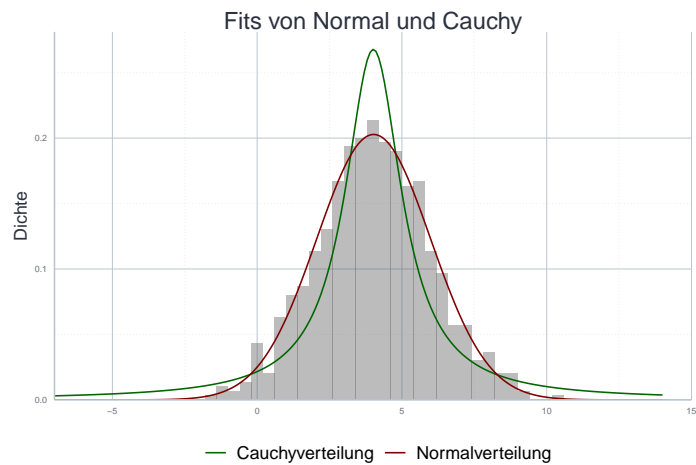
Wir fitten die Normalverteilung mit dem Paket `fitdistrplus`:

```
fit_dist <- fitdistr(data = sample_data$r,
                    distr = "norm")
fit_dist[["estimate"]]
```

```
##      mean      sd
## 4.023254 1.967130
```

```
fit_dist2 <- fitdistr(data = sample_data$r,
                     distr = "cauchy")
fit_dist2[["estimate"]]
```

```
## location    scale
## 4.011809 1.188676
```



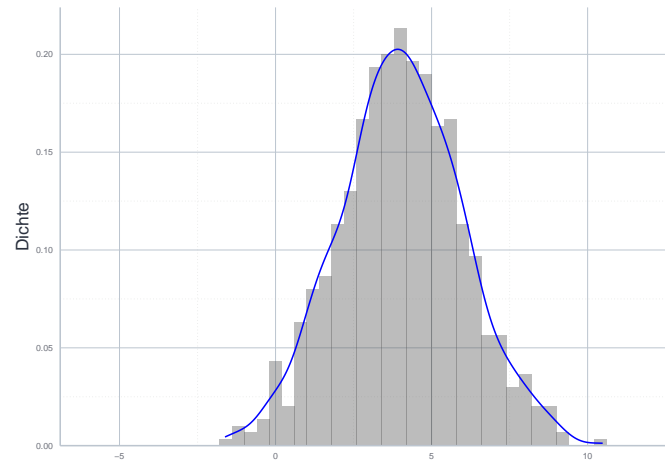
Der nicht-parametrische Fit:

```
ggplot(data = sample_data) +
  geom_histogram(
    mapping = aes(x=r, stat(density)),
    binwidth = 0.4, alpha=0.4) +
  coord_cartesian(xlim = c(-6, 12)) +
  stat_density(mapping = aes(x=r,
                             color="blue",
                             geom="line")) +
  scale_y_continuous(name = "Dichte",
```

```

        expand = expansion(c(0, 0.05), c(0, 0))
      ) +
    theme_icae() +
    theme(legend.title = element_blank(),
          legend.text = element_text(size = 12),
          axis.title.x = element_blank(),
          plot.title = element_text(size = 16),
          axis.title.y = element_text(size = 12))

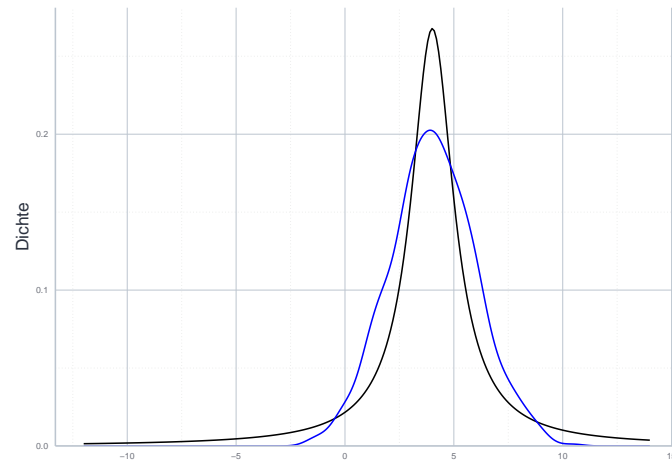
```



```

ggplot(data = sample_data) +
  geom_line(data = filter(
    y_vals_pdf,
    Verteilung=="Cauchyverteilung"),
    mapping = aes(x=x, y=y)) +
  stat_density(mapping = aes(x=r),
    color="blue",
    geom="line") +
  scale_y_continuous(name = "Dichte",
    expand = expansion(c(0, 0.05), c(0, 0))
  ) +
  theme_icae() +
  theme(legend.title = element_blank(),
        legend.text = element_text(size = 12),
        axis.title.x = element_blank(),
        plot.title = element_text(size = 16),
        axis.title.y = element_text(size = 12))

```



4.2 Grafische Darstellung

```
data("anscombe")
head(anscombe)
```

```
##   x1 x2 x3 x4   y1   y2   y3   y4
## 1 10 10 10  8 8.04 9.14  7.46 6.58
## 2  8  8  8  8 6.95 8.14  6.77 5.76
## 3 13 13 13  8 7.58 8.74 12.74 7.71
## 4  9  9  9  8 8.81 8.77  7.11 8.84
## 5 11 11 11  8 8.33 9.26  7.81 8.47
## 6 14 14 14  8 9.96 8.10  8.84 7.04
```

Die folgende Tabelle gibt die Werte der quantitativen Kennzahlen an:

Kennzahl	Wert
Mittelwert von x	9
Mittelwert von y	7.5
Varianz von x	11
Varianz von y	4.13
Korrelation zw. x und y	0.82

Nur die grafische Inspektion zeigt, wie unterschiedlich die Verteilungen tatsächlich sind:

