AS17: Kapitel 2 - Grundlagen aus der Mathematik

Jürgen Hedderich

2020-04-13

Contents

zur Druckversion

Hinweis: Die thematische Gliederung zu den aufgeführten Befehlen und Beispielen orientiert sich an dem Inhaltsverzeichnis der 17. Auflage der 'Angewandten Statistik'. Nähere Hinweise zu den verwendeten Formeln sowie Erklärungen zu den Beispielen sind in dem Buch (Ebook) nachzulesen!

Aktuelle R-Umgebung zu den Beispielen

The R Project for Statistical Computing

```
sessionInfo()
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=German_Germany.1252  LC_CTYPE=German_Germany.1252
## [3] LC_MONETARY=German_Germany.1252 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_3.6.3  magrittr_1.5    tools_3.6.3     htmltools_0.4.0
##  [5] yaml_2.2.1      Rcpp_1.0.4      stringi_1.4.6   rmarkdown_2.1
##  [9] knitr_1.28      stringr_1.4.0   xfun_0.12       digest_0.6.25
## [13] rlang_0.4.5     evaluate_0.14
```

2.3 (Grund-) Rechenarten

```
12 + 32                                  # Addition
## [1] 44
43 - 15                                  # Subtraktion
## [1] 28

Zahlen <- c(2, 5, 7, 8, 9, 6)           # Werte in einem Vektor
sum(Zahlen)                              # Summe
## [1] 37

1:20
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
sum(1:20)                                # Summe zu Rangzahlen
## [1] 210

4 * 17                                   # Multiplikation
## [1] 68

56 / 8                                   # Division
## [1] 7

Zahlen <- c(2, 3, 4,5)
prod(Zahlen)                             # Produkt
## [1] 120

1:10
##  [1]  1  2  3  4  5  6  7  8  9 10
prod(1:10)                               # Fakultät
## [1] 3628800

                                         # größter gemeinsamer Teiler (ggT)
ggT <- function(m, n) ifelse (n==0, m, ggT(n, m %% n))
ggT(21, 35)
## [1] 7

                                         # kleinstes gemeinsames Vielfaches (kgV)
kgV <- function(m, n) abs(m*n) / ggT(m, n)
kgV(21, 35)
## [1] 105

ggT(3528, 3780); kgV(3528, 3780)
## [1] 252
## [1] 52920
```

2.3.2 Potenzen und Wurzeln

```
2^4                                     # Potenzieren
## [1] 16
12^4
## [1] 20736

sqrt(3)                                 # Radizieren in R
## [1] 1.732051
sqrt(234)
## [1] 15.29706
35^(5/3)
## [1] 374.4956

pi                                      # die Zahl Pi = 3.141593
## [1] 3.141593

exp(1)                                  # die Zahl e = 2.718282
## [1] 2.718282

log(12, base = exp(1))                  # Logarithmus zur Basis e
## [1] 2.484907

log10(16)                               # Logarithmus zur Basis 20
## [1] 1.20412

log2(20)                                # Logarithmus zur Basis 2
## [1] 4.321928
```

2.3.4 Rundungen

```
                                          # Rundungen
x <- c(1.347, 2.499, 3.501)
floor(x)
## [1] 1 2 3
ceiling(x)
## [1] 2 3 4

x <- c(1.347, 2.499, 3.501)
floor(x + 0.5)
## [1] 1 2 4
x <- c(-1.347, -2.499, -3.501)
ceiling(x - 0.5)
## [1] -1 -2 -4

                                          # Rundung für x>=0
rundung <- function(x, m) floor((x/m + 0.5))*m
rundung(x=1.3567, m=c(0.1, 0.01, 0.001, 0.25, 0.5))
## [1] 1.400 1.360 1.357 1.250 1.500

x <- seq(1.1, 1.4, 0.02)                  # Tabelle
y <- rundung(x, m=0.1)
z <- rundung(x, m=0.25)
cbind(x,y,z)[1:10,]
##          x   y    z
##  [1,] 1.10 1.1 1.00
##  [2,] 1.12 1.1 1.00
##  [3,] 1.14 1.1 1.25
##  [4,] 1.16 1.2 1.25
##  [5,] 1.18 1.2 1.25
##  [6,] 1.20 1.2 1.25
##  [7,] 1.22 1.2 1.25
##  [8,] 1.24 1.2 1.25
##  [9,] 1.26 1.3 1.25
## [10,] 1.28 1.3 1.25

round(x=1.3567, digits=c(1,2,3))
## [1] 1.400 1.360 1.357
```

2.4 Einführung in die Matrixalgebra

```
library(Matrix)
library(MASS)

A <- matrix( c(1, 2, 3, 6, 5, 4), nrow=2, ncol=3, byrow=TRUE)
A.trans <- t(A); A; A.trans             # Transponieren einer Matrix
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    6    5    4
##      [,1] [,2]
## [1,]    1    6
## [2,]    2    5
## [3,]    3    4

                                        # Addition zweier Matrizen
A <- matrix( c(1, 2, 3, 6, 5, 4), nrow=2, ncol=3, byrow=TRUE)
B <- matrix(c(4, 5, 6, 9, 8, 7), nrow=2, ncol=3, byrow=TRUE)
C <- A + B; A; B; C
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    6    5    4
##      [,1] [,2] [,3]
## [1,]    4    5    6
## [2,]    9    8    7
##      [,1] [,2] [,3]
## [1,]    5    7    9
## [2,]   15   13   11

                                        # Multiplikation mit einem Skalar
A <- matrix( c(1, 2, 3, 6, 5, 4), nrow=2, ncol=3, byrow=TRUE);
A; 2 * A
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    6    5    4
##      [,1] [,2] [,3]
## [1,]    2    4    6
## [2,]   12   10    8

                                        # Multiplikation zweier Matrizen
A <- matrix( c(1, 2, 3, 6, 5, 4), nrow=2, ncol=3, byrow=TRUE);
B <- matrix(c(4, 5, 6, 9, 8, 7), nrow=3, ncol=2, byrow=TRUE);
C <- A %*% B; A; B; C
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    6    5    4
##      [,1] [,2]
## [1,]    4    5
## [2,]    6    9
## [3,]    8    7
##      [,1] [,2]
## [1,]   40   44
## [2,]   86  103
```

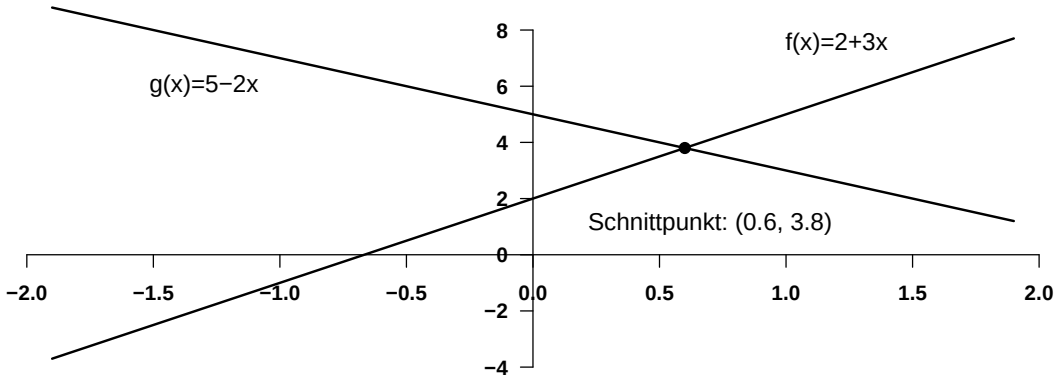
```r
                                        # Skalarprodukt zweier Vektoren
a <- 1:3
b <- 4:6
c <- t(a) %*% b; a; b; c
## [1] 1 2 3
## [1] 4 5 6
##      [,1]
## [1,]   32
                                        # Norm eines Vektors
a <- c(1, 2, 3, 4, 5, 6)
a.trans <- t(a)
a.norm <- sqrt(a.trans %*% a)
a; a.norm
## [1] 1 2 3 4 5 6
##          [,1]
## [1,] 9.539392
                                        # Bestimmung der Determinante
A     <- matrix(c(3, 1, 2, 4, 5, 6, 9, 7, 8), nrow=3, ncol=3, byrow=TRUE)
A.det <- det(A); A; A.det
##      [,1] [,2] [,3]
## [1,]    3    1    2
## [2,]    4    5    6
## [3,]    9    7    8
## [1] -18

                                        # Berechnung der inversen Matrix
A     <- matrix(c(3, 1, 2, 4, 5, 6, 9, 7, 8), nrow=3, ncol=3, byrow=TRUE)
A.inv <- solve(A)
A; round(A.inv, 2); round(A %*% A.inv, 2)
##      [,1] [,2] [,3]
## [1,]    3    1    2
## [2,]    4    5    6
## [3,]    9    7    8
##       [,1]  [,2]  [,3]
## [1,]  0.11 -0.33  0.22
## [2,] -1.22 -0.33  0.56
## [3,]  0.94  0.67 -0.61
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1

library(MASS)                           # auch mit ginv() aus library(MASS)
A.inv <- ginv(A)
A; round(A.inv, 2); round(A %*% A.inv, 2)
##      [,1] [,2] [,3]
## [1,]    3    1    2
## [2,]    4    5    6
## [3,]    9    7    8
##       [,1]  [,2]  [,3]
## [1,]  0.11 -0.33  0.22
## [2,] -1.22 -0.33  0.56
## [3,]  0.94  0.67 -0.61
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1

                                        # Loesung lineares Gleichungssystem
A     <- matrix(c(3, 1, 2, 4, 5, 6, 9, 7, 8), nrow=3, ncol=3, byrow=TRUE)
b     <- c(2,4,8); A; b
##      [,1] [,2] [,3]
## [1,]    3    1    2
## [2,]    4    5    6
## [3,]    9    7    8
## [1] 2 4 8
x     <- solve(A, b); round(x, 2)
## [1]  0.67  0.67 -0.33
A %*% x
##      [,1]
## [1,]    2
## [2,]    4
## [3,]    8

                                        # Eigenwerte und Eigenvektoren in A
A     <- matrix(c(3, 1, 2, 4), nrow = 2, ncol=2, byrow=TRUE);A
##      [,1] [,2]
## [1,]    3    1
## [2,]    2    4
l     <- eigen(A)$values; round(l, 2)
## [1] 5 2
x     <- eigen(A)$vectors; round(x, 2)
##       [,1]  [,2]
## [1,] -0.45 -0.71
## [2,] -0.89  0.71
```

2.5 Funktionen

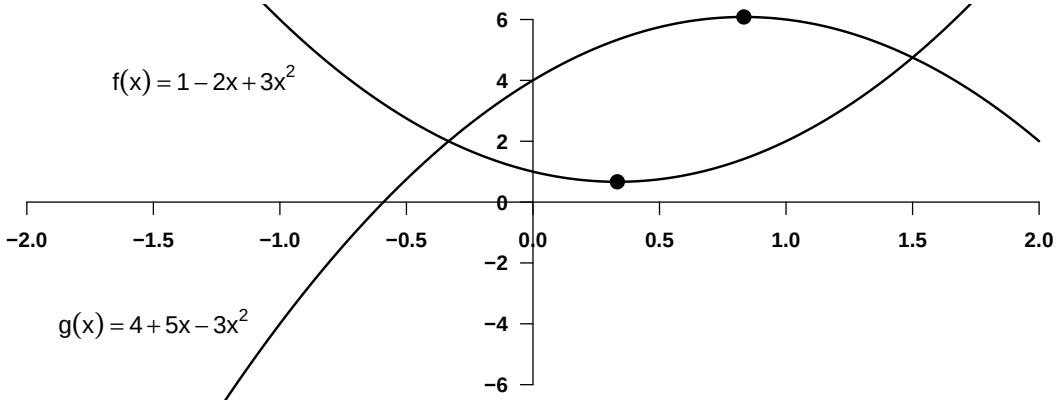
2.5.1 Lineare Funktionen

```
par(mfrow=c(1,1), font.axis=2, bty="l", ps=12, cex.axis=1.0)
x   <- seq(-1.9, +1.9, by=0.02)
y.1 <- 2 + 3 * x
y.2 <- 5 - 2 * x
plot(x, y.1, type="l", axes=F, xlab="", ylab="", las=1, lwd=1.9,
     xlim=c(-2, +2), ylim=c(-4,9))
axis(side = 1, pos = 0, xaxp=c(-2, 2, 8))
axis(side = 2, pos = 0, las=1)
lines(x, y.2, lwd=1.9)
points(0.6, 3.8, pch=16, cex=1.3)
text(1.2, 7.5, expression("f(x)=2+3x"), cex=1.2)
text(-1.3, 6, expression("g(x)=5-2x"), cex=1.2)
text(0.7, 1.1, "Schnittpunkt: (0.6, 3.8)", cex=1.2)
```



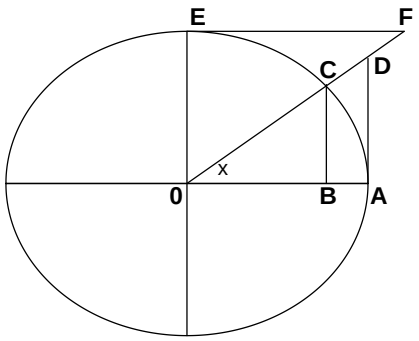
2.5.2 Nichtlineare Funktionen

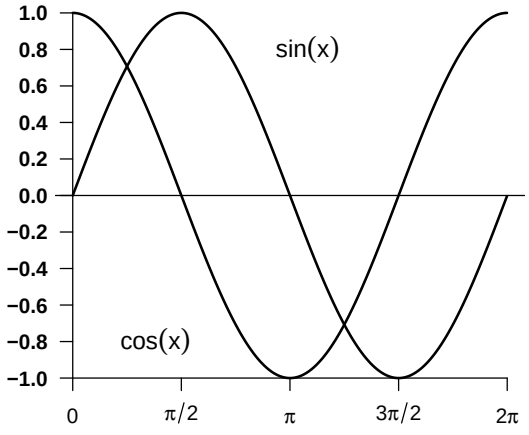
```
par(mfrow=c(1,1), font.axis=2, bty="l", ps=12, cex.axis=1.0)
x <- seq(-2, +2, by=0.02)
y.1 <-  1 - 2*x + 3*x^2
y.2 <-  4 + 5*x - 3*x^2
plot(x, y.1, type="l", axes=F, xlab="", ylab="", las=1, lwd=1.9,
     xlim=c(-2, +2), ylim=c(-6, +6))
axis(side = 1, pos = 0, xaxp=c(-2, 2, 8))
axis(side = 2, pos = 0, las=1)
lines(x, y.2, lwd=1.9)
text(-1.3, 4.0, expression(f(x) == 1 - 2*x + 3*x^2), cex=1.2)
text(-1.5, -4.0, expression(g(x)==4 + 5*x - 3*x^2), cex=1.2)
points(-(-2/(2*3)),    1-((-2)^2 /   (4*3)), pch=19, cex=1.5)
points(-(5 /(2*(-3))), 4-( 5^2   /(4*(-3))), pch=19, cex=1.5)
```



2.5.3 Periodische Funktionen

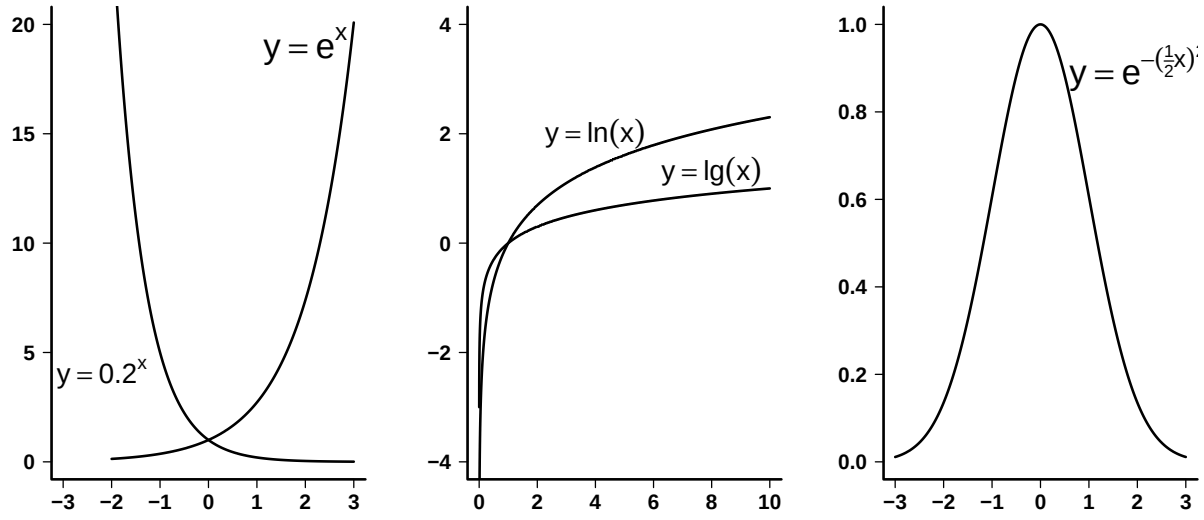
```
par(mfrow=c(1,2), font.axis=2, bty="l", ps=12, cex.axis=1.0)
x <- seq(-1.0, + 1.0, by=0.0002)
y <- sqrt(1-x^2)
plot(c(x, x, 1), c(y, -y, 0), type="l", axes=FALSE, xlab="", ylab="",
     las=1, xlim=c(-1.2, +1.2), ylim=c(-1.2, +1.2))
text(0.2,0.091, expression(x), cex=1.0)
lines(c(0.0, 0.0), c(-1,+1)); lines(c(-1,+1),    c(0,0))
lines(c(0.0, 1.2), c(0.0, 1.0));   lines(c(0.0, 1.2), c(1.0, 1.0))
lines(c(1.0, 1.0), c(0.0, 0.822)); lines(c(0.77, 0.77), c(0.0, 0.65))

text(-0.06, -0.08, "0", cex=1.2, font=2)
text(1.06, -0.08, "A", cex=1.2, font=2)
text(0.78, -0.08, "B", cex=1.2, font=2)
text(0.78, +0.75, "C", cex=1.2, font=2)
text(1.08, +0.78, "D", cex=1.2, font=2)
text(0.06, +1.1, "E", cex=1.2, font=2)
text(1.21, +1.1, "F", cex=1.2, font=2)

x <- seq(0, 2*pi, by=0.01)
y.1 <- cos(x);  y.2 <- sin(x)
plot(x, y.1, type="l", axes=F, xlab="", ylab="", lwd=2.0,
     xlim=c(0, 2*pi), xaxp=c(0, 2*pi, 5), ylim=c(-1, +1))
axis(side=2, pos = 0, yaxp=c(-1, 1, 10), las=1, cex=1.0)
axis(side=1, pos = -1.0, at=c(0, pi/2, pi, 3*pi/2, 2*pi),
     labels=expression(0, pi/2, pi, 3*pi/2, 2*pi), cex=2.0)
lines(x, y.2, lwd=2.0)
text(3.4, 0.8, expression(sin(x)), cex=1.2)
text(1.2, -0.8, expression(cos(x)), cex=1.2)
abline(h=0)
```



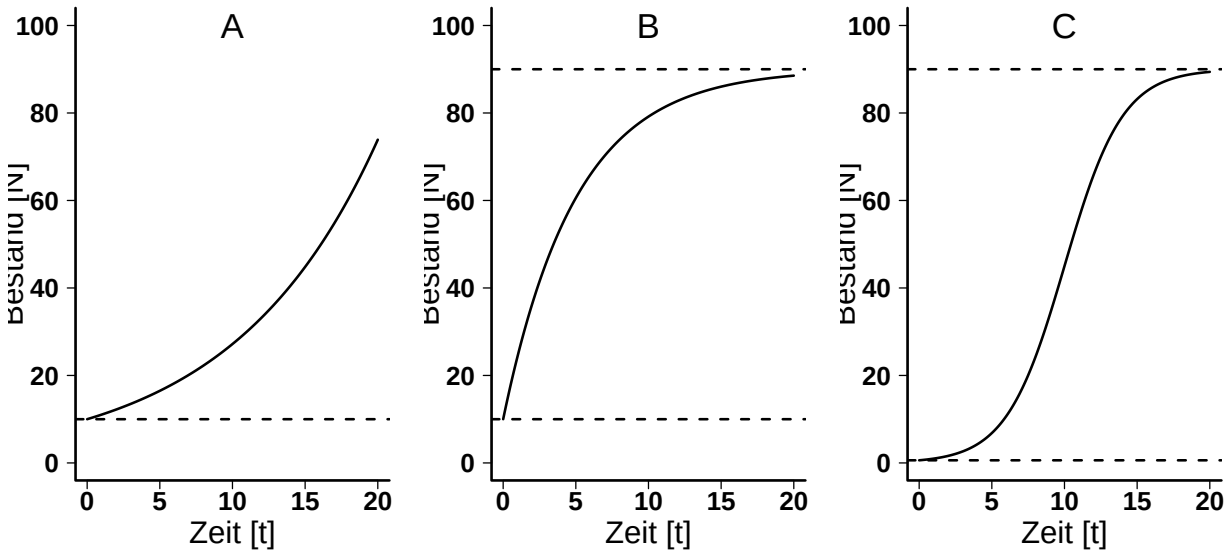


2.5.4 Expontialfunktion / Wachstumsfunktionen

```
                                    # Exponentialfunktionen
par(mfrow=c(1,3), lwd=2.0, font.axis=2, bty="l", ps=12, cex.axis=1.5)
x <- seq(-2, 3, by=0.001)
y.1 <- exp(x)
y.2 <- (1/5)^x
plot(x, y.1, type="l", axes=TRUE, xlab="", ylab="", las=1,
     xlim=c(-3, 3), ylim=c(0, 20))
lines(x, y.2)
text(2, 19, expression(y==e^x), cex=2.5)
text(-2.2, 4, expression(y==0.2^x) , cex=2.0)

                                    # Logarithmusfunktion
x <- seq(0, 10, by=0.001)
y.1 <- log(x)
y.2 <- log10(x)
plot(x, y.1, type="l", axes=TRUE, xlab="", ylab="", las=1,
     xlim=c(0, 10), ylim=c(-4,4))
lines(x, y.2)
text(4, 2, expression(y==ln(x)), cex=2.0)
text(8, 1.3, expression(y==lg(x)), cex=2.0)

                                    # Glockenkurve
x <- seq(-3, 3, by=0.001)
y <- exp(-0.5*x^2)
plot(x, y, type="l", axes=TRUE, xlab="", ylab="", las=1,
     xlim=c(-3, 3), ylim=c(0, 1))
text(2.0, 0.9, expression(y == e^-(frac(1,2)*x)^2), cex=2.5)
```

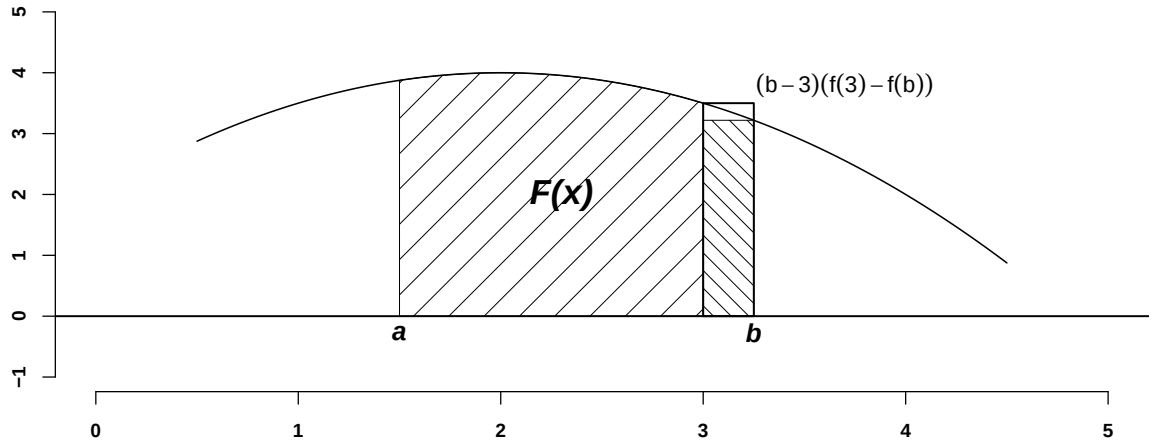


```r
                                    # Wachstumsfunktionen
par(mfrow=c(1,3), lwd=2, font.axis=2, bty="l", ps=15, cex.axis=1.5)

                                    # exponentielles Wachstum
e <- exp(1); t <- seq(0, 20, by=0.1); lambda <- 0.1;  N.0 <- 10
N <- N.0 * e^(lambda*t)
plot(t, N, ylim=c(0,100), type="l", las=1, cex.lab=1.8,
     xlab="Zeit [t]", ylab="Bestand [N]")
abline(h=10, lty=2); text(10, 100, "A", cex=2)

                                    # exponentielles wachstum mit Sättigung
N.max  <- 90; lambda <- -0.2; N.0    <- 10
N <- N.max - (N.max-N.0)*e^(lambda*t)
plot(t, N, ylim=c(0,100), type="l", las=1, cex.lab=1.8,
     xlab="Zeit [t]", ylab="Bestand [N]")
abline(h=10, lty=2); abline(h=90, lty=2); text(10, 100, "B", cex=2)

                                   # logististische Wachstumsfunktion
N.max <- 90; a <- 5; b <- 0.5
N <- N.max / (1+e^(a - b*t))
plot(t, N, ylim=c(0,100), type="l", las=1, cex.lab=1.8,
     xlab="Zeit [t]", ylab="Bestand [N]")
N.0 <- N.max / (1+e^a)
abline(h=N.max, lty=2); abline(h=N.0, lty=2); text(10, 100, "C", cex=2)
```



2.5.5 Fläche unter einer Funktion

```
par(lwd=1.5, font.axis=2, bty="n", ps=10, cex.axis=1.07, mfrow=c(1,1))
x <- seq(0.5, 4.5, by=0.001)
f <- function(x) - 0.5*x^2 + 2*x + 2
y <- f(x)
plot(x,y, type="l", axes=TRUE, lwd=1.2, xlab=" ", ylab=" ",
                                          xlim=c(0,5), ylim=c(-1,5))
x1 <- seq(1.5, 3, by=0.1)
y1 <- f(x1)
polygon(c(1.5, x1, 3), c(0, y1, 0), density=5, lwd=0.7)
polygon(c(3, 3, 3.25, 3.25), c(0, f(3), f(3), 0))
polygon(c(3, 3, 3.25, 3.25), c(0, f(3.25), f(3.25), 0),angle=135,
        lwd=0.7, density=10)
abline(h=0)
text(2.3, 2, "F(x)", cex=2, font=4)
text(1.5, -0.28, "a", cex=1.5, font=4)
text(3.25, -0.28, "b", cex=1.5, font=4)
text(3.7, 3.8, expression((b-3)*(f(3)-f(b))), cex=1.3, font=4)
```



2.6 Kombinatorik

Funktion permn() in library(combinat)

```r
library(combinat)                                   # Permutationen
x <- c("a","b","c")
permn(x)
## [[1]]
## [1] "a" "b" "c"
##
## [[2]]
## [1] "a" "c" "b"
##
## [[3]]
## [1] "c" "a" "b"
##
## [[4]]
## [1] "c" "b" "a"
##
## [[5]]
## [1] "b" "c" "a"
##
## [[6]]
## [1] "b" "a" "c"

n <- 8                                              # Produkt 1:n
prod(1:n)
## [1] 40320

n <- 20
prod(1:(2*n -2)) / (2^(n-1)*prod(1:(n-1)))
## [1] 8.200795e+21

n <- 9; k <- 7                                      # Binomialkoeffizient
choose(n, k)
## [1] 36

combn(letters[1:5], 3)                              # Kombinationen
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,] "a"  "a"  "a"  "a"  "a"  "a"  "b"  "b"  "b"  "c"
## [2,] "b"  "b"  "b"  "c"  "c"  "d"  "c"  "c"  "d"  "d"
## [3,] "c"  "d"  "e"  "d"  "e"  "e"  "d"  "e"  "e"  "e"

choose(10, 3) * prod(1:3)                           # Verteilung von Preisen
## [1] 720

choose(5+12-1, 12)                                  # Bonbonsorten
## [1] 1820

26^3 + 26^2 + 26                                    # Worte
## [1] 18278

sum(choose(10, 1:10))                               # Ausstattungsvarianten
## [1] 1023
```

```
                                          # Multinomialkoeffizient
karten <- 52                              # Anzahl der Karten

spieler <- 4                              # Anzahl der Spieler

k.spiel <- karten/spieler                 # Anzahl Karten pro Spieler

prod(1:karten)/(prod(1:k.spiel)^spieler)
## [1] 5.364474e+28
```