L01 – Base-Graphics – Aufgaben

12. April 2021

Contents

1 Sinus-Reihenentwicklung **1**
- 1.1 Farbe . . . 2
- 1.2 Weitere Basics . . . 3
- 1.3 Parameter für Linien . . . 3
- 1.4 Utilities . . . 3
- 1.5 Bilddatei exportieren . . . 3

2 Mandelbrot-Menge **3**

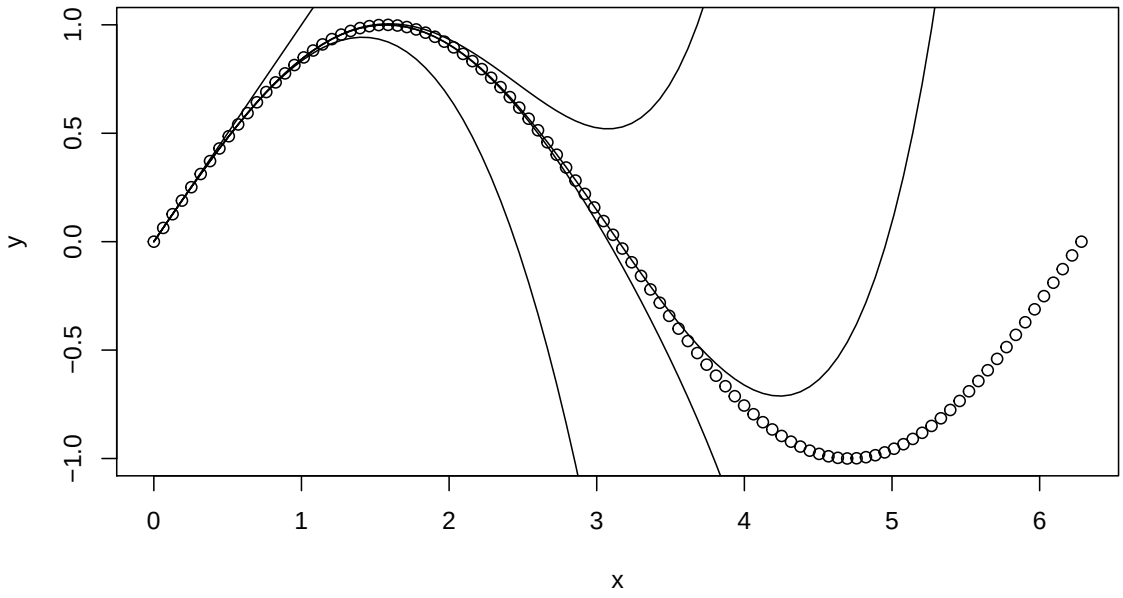
3 Vier Plots in einem Bild **5**

1 Sinus-Reihenentwicklung

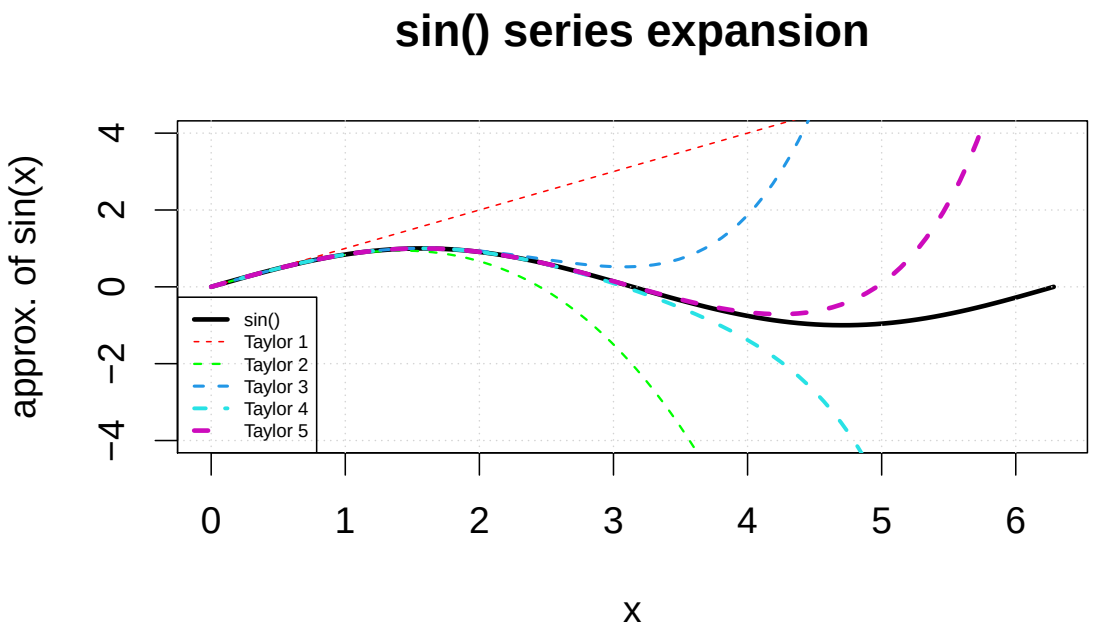
Wir visualisieren die Annäherung an die Sinus-Funktion mittels Polynomen (Taylor-Entwicklung).

```
sin1 <- function(x) x
sin2 <- function(x) sin1(x)-x^3/factorial(3)
sin3 <- function(x) sin2(x)+x^5/factorial(5)
sin4 <- function(x) sin3(x)-x^7/factorial(7)
sin5 <- function(x) sin4(x)+x^9/factorial(9)
x <- seq(0, 2*pi, length.out=100)
y <- sin(x)
y1 <- sin1(x)
y2 <- sin2(x)
y3 <- sin3(x)
y4 <- sin4(x)
y5 <- sin5(x)

plot(x, y)
lines(x, y1)
lines(x, y2)
lines(x, y3)
lines(x, y4)
lines(x, y5)
```



Das sieht etwas trist aus. Ziel dieser Aufgabe ist es, den Plot etwas aufzuhübschen, zB:



1.1 Farbe

Füge `plot()` und `lines()` ein Argument `col=...` hinzu. Anstelle von `...` steht die Spezifikation der Farbe. Dafür gibt es verschiedene Möglichkeiten:

- ein String der RGB-Codierung (**R**ed, **G**reen, **B**lue) hexadezimal der Form `"#RRGGBB"` wobei `RR`, `GG`, `BB` Werte zwischen `00` und `FF` sind mit `A=10`, ..., `F=15` und den Anteil der Grundfarben angeben, zB steht `"#FF00FF"` für einen lila Farbton.
- den Farbnamen als String. Alle Möglichkeiten werden mittels `colors()` ausgegeben. ZB `"yellow"`, `"khaki"`, oder `"navyblue"`
- eine natürliche Zahl, zB `col=2` (`col="2"` geht auch): Damit wird eine Farbe aus der voreingestellten Palette ausgewählt. `palette()` zeigt die Farben der voreingestellten Paletten an. Führe `barplot(rep(1,8), col=1:8)` aus um diese Farben angezeigt zu bekommen.

Teste die drei angegebenen Möglichkeiten eine Farbe anzugeben.

1.2 Weitere Basics

Mit den Argumenten `xlim=c(XMIN, XMAX)` und `ylim=c(YMIN, YMAX)` in `plot()` werden die Grenzen der x- bzw y-Achse gesetzt. Setze die Grenzen der y-Achse so, dass mehr von den Polynomfunktionen zu sehen ist.

Für Achsenbeschriftung setze die Argumente `xlab="XLABEL"` bzw `ylab="YLABEL"`. Wird `title("TITLE")` nach `plot()` ausgeführt, erhält der Plot die Überschrift `TITLE`. Füge dem Plot einen sinnvollen Titel sowie eine neue y-Achsenbeschriftung hinzu.

Lass nun auch die Sinus-Funktion als Linie statt als Punkte zeichnen. Nutze dazu das Argument `type` von `plot()`. Um herauszufinden auf welchen Wert `type` gesetzt werden muss, rufe die Hilfe `?plot` und suche im Abschnitt `Arguments` den Eintrag zu `type`.

1.3 Parameter für Linien

Lies in der Hilfe `?par` im Abschnitt `Graphical Parameters` die Einträge zu `lty` und `lwd`. Erhöhe die Linienbreite der geplotteten Funktionen und zeichne die Sinus-Kurve mit einem anderen Linientyp als die Polynome.

1.4 Utilities

Füge dem Plot mittels `grid()` Gitterlinien zur besseren Orientierung hinzu. Mit `lengend()` wird eine Legende in den Plot eingezeichnet. Passe folgendes Beispiel so an, dasse eine sinnvolle Legende entsteht.

```
legend("left", # "bottom", "topright", ... position of the legend
       c("Name1", "Name2", ...), # names of functions
       col=c(1, "red", "#00FF00", ...), # vector of colors
       lwd=c(1, 2, ...), # vector of line width
       lty=c("solid", "dashed", ...), # vector of line types
       cex=0.5) # text scaling (if needed)
```

1.5 Bilddatei exportieren

Nutze den Export-Button im Plots-Tab von RStudio, um den fertigen Plot als Bilddatei zu speichern. Poste diese Datei dann in dem heiCHAT-Raum deiner Übungsgruppe.

2 Mandelbrot-Menge

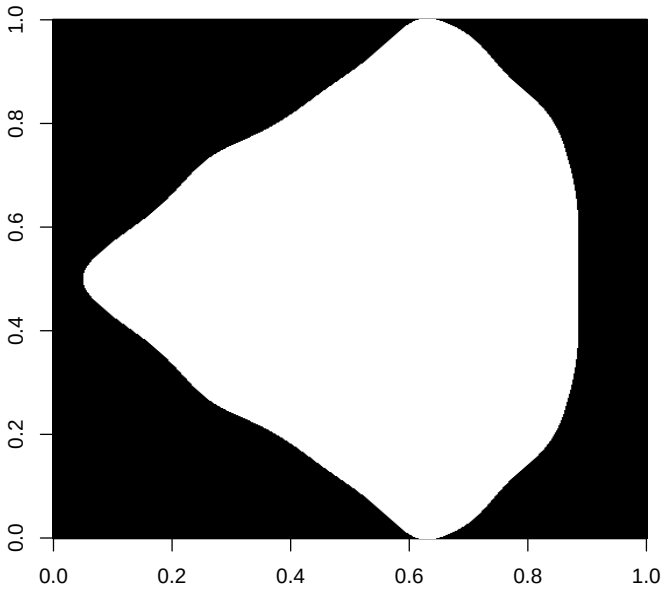
Ergänze den folgenden Code, um eine Visualisierung der Mandelbrot-Menge zu erhalten.

```
mandelbrot_iteration <- function(Z=0, dx=512, dy=512) {
  C  <- complex(real = rep(seq(-2.2, 1.0, length.out = dx), each = dy),
                imag = rep(seq(-1.2, 1.2, length.out = dy), dx))
  C <- matrix(C, dy, dx)
  Z <- Z^2 + C
```

```
  return(Z)
}

mandelbrot <- function(n, dx=512, dy=512) {
  X <- array(complex(1), dim=c(dx,dy,n))
  X[,,1] <- mandelbrot_iteration(dx=dx, dy=dy)
  for (i in 2:n) {
    X[,,i] <- mandelbrot_iteration(X[,,i-1], dx, dy)
  }
  res <- matrix(ncol=dx, nrow=dy)
  res[] <- as.integer(factor(rank(abs(t(X[,,n])))))-1
  return(res/max(res))
}

X <- mandelbrot(8)
dim(X) # X ist 512x512 Matrix
## [1] 512 512
range(X) # mit Werten zwischen 0 und 1
## [1] 0 1
mandelcolors <- c("white", "black") # TODO
image(X, col=mandelcolors, useRaster=TRUE)
```

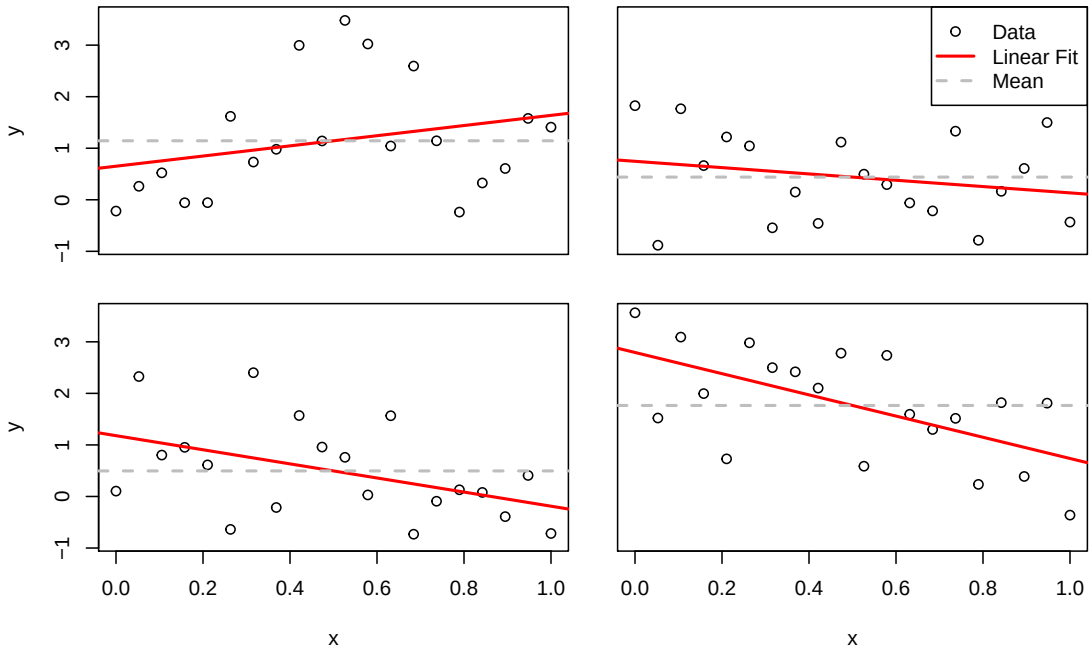


Die Funktion `image()` mit `useRaster=TRUE` übersetzt die Zahlenwerte der 512x512 Matrix `X` in Farbpixel.

- Ersetze `c("white", "black")` durch einen ansprechenderen Vektor von Farben. Nutze dazu eine Farbpalette, siehe `?rainbow`.
- Ändere die 8 in `X <- mandelbrot(8)` auf eine beliebige ganze Zahl zwischen 3 und 100.
- Setze `axes=FALSE` für die Funktion `image()`.
- Füge dem Bild mit `text()` (siehe `?text`) einen ansprechenden Titel hinzu.
- Erzeuge nun wieder eine Bild-Datei (512x512 Pixel) und poste sie in heiCHAT.

3 Vier Plots in einem Bild

Wir möchten 4 Plots in einem Bild anzeigen und dabei auf "unnötige" Achsen verzichten.



Ergänze den folgenden Code an den Stellen `# TODO`. Lies dazu die angegebenen Stellen der R-Dokumentation.

```r
# Generate Data ------------------------------------------------
set.seed(7)
n <- 20
x <- seq(0, 1, length.out=n)
get_random_data <- function(n, distri) {
  beta <- c(runif(1)*2-1, runif(1)*4-2)
  if (distri == "unif")
    beta[1] + beta[2] * x + runif(n)*sqrt(12)
  else
    beta[1] + beta[2] * x + rnorm(n)
}

data <- list()
data[[1]] <- get_random_data(n, "norm")
data[[2]] <- get_random_data(n, "norm")
data[[3]] <- get_random_data(n, "unif")
data[[4]] <- get_random_data(n, "unif")
total_ylim <- range(unlist(data))

# Plot ---------------------------------------------------------
# TODO 1 par(...): Setze mfrow, um mehrere Plots in einer Grafik darzusetellen.
# Siehe Hinweise unten.
for (i in seq_along(data)) {
  # TODO 2: ersetze {} jeweils durch einen passenden Aufruf von
  # par(mar=c(?, ?, ?, ?)) mit sinnvollen Werten zwischen 1 und 4 anstatt '?'
```

```
  if (i == 1) {}
  if (i == 2) {}
  if (i == 3) {}
  if (i == 4) {}
  plot.new() # erzeugt neuen leeren plot
  plot.window(c(0,1), total_ylim) # Limits setzen
  y <- data[[i]]
  plot.xy(xy.coords(x, y), type="p") # Datenpunkte einzeichnen
  # TODO 3: zeichne eine Box um den Plot: ?box
  if (i >= 3) { # nur in unteren plots
    # TODO 4: erzeuge x-Achse, siehe ?axis, ?title
  }
  if (i %% 2 == 1) { # nur in linken plots
    # TODO 5: erzeuge y-Achse, siehe ?axis, ?title
  }
  fit <- lm(y ~ x) # lineare Regression
  abline(fit, col="red", lwd=2) # Ergebnis einzeichnen
  abline(mean(y), 0, lty=2, col="gray", lwd=2) # Mittelwert einzeichnen
  if (i == 2) legend( # Legende
    "topright",
    c("Data", "Linear Fit", "Mean"),
    lwd = c(NA, 2, 2), # Linienbreite
    pch = c(1, NA, NA), # Symbol
    lty = c(NA, 1, 2), # Linientyp
    col = c("black", "red", "gray")
  )
}
```

Hinweise:

- Zu `#TODO 1`: Mit `par(mfrow=c(nrow, ncol))` wird angegeben, dass die folgenden `nrow*ncol` Plots in einer Graphik angezeigt werden. Führe zum Testen folgenden Code Zeile für Zeile aus:

```
par(mfrow=c(2, 3))
plot(sinpi)
plot(cospi)
plot(exp)
plot(sin)
plot(cos)
plot(sqrt)
```

- Bei einer Fehlermeldung `figure margins too large` führe zunächst `dev.off()` aus, um das *Graphics Device* zu resetten. Führe dann `windows()` aus, um ein externes *Graphics Device* zu öffnen. Ein Fenster sollte sich öffnen. Maximiere es und führe die Plot-Anweisungen erneut durch. Sie sollten nun in das neue Fenster geschrieben werden.
- Mit `par()` werden graphische Parameter persistent verändert. Um diese Änderungen Rückgängig zu machen, kann `dev.off()` ausgeführt werden. Dann sind die graphische Parameter wieder auf die Standardwerte gesetzt.
- Zu `#TODO 2`: Die Breite der Ränder (*margins*) eines Plots können durch `par(mar=c(bottom, left, top, right))` beeinflusst werden. ZB

```
par(bg="gray", mar=c(2, 3, 1, 0))
plot(sinpi)
```

Ändere mindestens zwei grafische Parameter (`col`, `lwd`, ...) in den Plots und passe die Legende entsprechend an. Poste das Ergebnis wieder in heiCHAT.