

---

## Grafiken mit **R**

## Inhaltsverzeichnis

<b>1</b>	<b>Beispielgrafiken</b>	<b>3</b>
1.1	Beispiel für einfache Grafiken . . . . .	3
1.2	Beispiel für komplexere Grafiken . . . . .	3
<b>2</b>	<b>Grafikausgabe</b>	<b>4</b>
<b>3</b>	<b>High-Level-Grafikfunktionen</b>	<b>6</b>
3.1	Die zentralen Konfigurationen von High-Level-Grafiken . . . .	8
3.2	Parameter-Sammlung . . . . .	9
<b>4</b>	<b>Bereiche eines Devices</b>	<b>12</b>
<b>5</b>	<b>Low-Level-Grafiken</b>	<b>15</b>
<b>6</b>	<b>Mathematische Beschriftung</b>	<b>16</b>
<b>7</b>	<b>Dynamische und Interaktive Grafiken</b>	<b>17</b>

## Grafiken mit R

### 1 Beispielgrafiken

#### 1.1 Beispiel für einfache Grafiken

```
> plot(1)
> plot(1:10)
> x <- 1:100
> y <- log(x)
> plot (x,y)
```

#### 1.2 Beispiel für komplexere Grafiken

Startseite des R Projektes: [www.r-project.org](http://www.r-project.org).

Webseite von Paul Murrell (Autor von *R Graphics*): [R Graphics](#)

## 2 Grafikausgabe

Wenn eine Grafik erzeugt wird, muss geklärt werden, auf welchem Gerät (*device*) die Grafik ausgegeben wird.

Tabelle 1: Devices unter Microsoft Windows

Funktion	Beschreibung
<code>jpeg()</code>	JPEG-Format
<code>pdf()</code>	PDF-Format, unterstützt auch mehrseitige Ausgaben
<code>pictex()</code>	Zum Einbinden in $\text{\LaTeX}$ -Dokumente
<code>png()</code>	PNG-Format
<code>postscript()</code>	Erstellt eine Postscriptdatei
<code>bmp()</code>	Bitmap-Format
<code>win.metafile()</code>	Erstellt ein Windows-Metafile
<code>win.print()</code>	Ausgabe an einen angeschlossenen Drucker

```
> jpeg("testgrafik.jpg")
> plot(1,2)
> dev.off()

null device
      1

> pdf("testgrafik")
> plot(1,2)

> pictex("testgrafik.tex")
> plot(1,2)

> bmp("testgrafik.bmp")
> plot(1,2)
> dev.off()

pdf
  2

> win.print("testgrafik.jpg")
> plot(1,2)
> win.print( printer = "")
```

```
null device  
1
```



Zeigt an, dass kein weitere Device geöffnet ist. Es ist aber möglich, dass mehrere Device gleichzeitig geöffnet sind. Im Beispiel:

```
pdf  
2
```

Es existieren zahlreiche Befehle zum Umgang mit einem Device. Die wichtigsten sind in der folgenden Tabelle aufgeführt. Wird eine Grafik erzeugt,

Tabelle 2: Befehle im Umgang mit mehreren Devices

Funktion	Beschreibung
<code>dev.list()</code>	Anzeige aller aktuellen Device
<code>graphics.off()</code>	Schließen aller Device
<code>dev.off()</code>	Schließen des aktuellen Device und anzeigen des neuen aktuellen

ohne dass ein Device definiert wurde, wird das in den Grundeinstellungen (`options("device")`) definierte Gerät verwendet. Die Standardeinstellung unter Windows ist `windows()`.<sup>1</sup>

Konkrete Eigenschaften und Parameter eines Devices können mit der Funktion `par()` (Abschnitt 3.2) gesetzt werden.

---

<sup>1</sup>Die Grafik wird in einem Fenster auf dem Bildschirm angezeigt.

### 3 High-Level-Grafikfunktionen

High-Level-Grafikfunktionen sind Funktionen, die eine vollständige Grafik mit x- und y-Achsen, den Beschriftungen sowie den Daten erzeugen können. Beispiele dafür sind:

Tabelle 3: High-Level-Grafikfunktionen

Funktion	Beschreibung
<code>plot()</code>	wichtigste Funktion, generisch (für verschiedene Klassen verwendbar)
<code>barplot()</code>	Säulendiagramm
<code>boxplot()</code>	Boxplot
<code>curve()</code>	Funktionen zeichnen
<code>hist()</code>	Histogramm
<code>mosaicplots()</code>	Grafiken für kategoriale Daten
<code>pairs()</code>	Matrix von Scatterplots
<code>qqplot()</code>	QQ-Plots

Aufgrund der generischen Eigenschaften ist `plot()` eine zentrale und häufig verwendete Funktion.



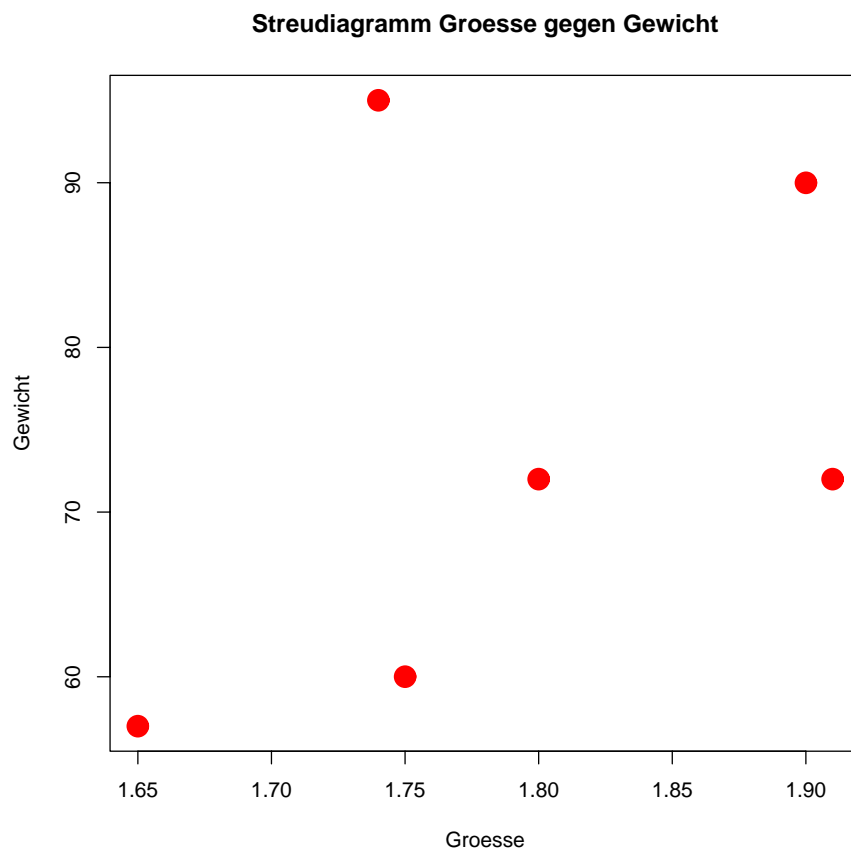
### 3 HIGH-LEVEL-GRAFIKFUNKTIONEN

---

Mit einfachen Modifikationen können weitere Informationen in die Grafik integriert werden:

```
> Gewicht <- c(60, 72, 57, 90, 95, 72)
> Groesse <- c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)

> plot(Groesse, Gewicht, type = "p", col = "red", lwd=10,
+ main = "Streudiagramm Groesse gegen Gewicht", xlab="Groesse",
+ ylab="Gewicht")
```



### 3.1 Die zentralen Konfigurationen von High-Level-Grafiken

Tabelle 4: Die wichtigsten Konfigurationen

Funktion	Beschreibung
<code>type = ...</code>	l = Linien, p = Punkte, b = Linien + Punkte, n = keine (nur Achsen), o = Überlagerung, h = vertikale Verbindungen zur x-Achse
<code>lwd = n</code>	n = Stärke der Linie bzw. des Punktes
<code>axes = FALSE</code>	Unterdrücken der Ausgabe der Achsen
<code>xlab = ...</code>	x-Achsenbeschriftung
<code>ylab</code>	y-Achsenbeschriftung
<code>xlim = c(Anfangswert, Endwert)</code>	Wertebereich der x-Achse
<code>ylim = c(Anfangswert, Endwert)</code>	Wertebereich der y-Achse
<code>main = ...</code>	Grafiküberschrift
<code>sub = ...</code>	Untertitel



## 3.2 Parameter-Sammlung

Die Eigenschaften eines Devices können mit einer zusätzlichen Spezifikation innerhalb der Funktion `par()` nachträglich verändert werden. Damit können beispielsweise an bereits erzeugten Grafiken zusätzliche Informationen angefügt werden. Eine Sammlung zentraler Argumente der Funktion `par()` sind in der folgenden Tabelle aufgeführt.

Tabelle 5: Argumente

Argument	Beschreibung
<code>mfrow = c(nrows, ncols)</code>	Aufteilung der Ausgabefläche in $n$ Zeilen und $n$ Spalten
<code>mfg = c(Zeile, Spalte)</code>	Position für den nächsten Plot, falls <code>mfrow</code> oder <code>mfc</code> gesetzt wurde, (z.B. zum Neuzeichnen einer Plotposition)
<code>pin = c(Breite, Hoehe)</code>	Größe des Plots in inches
<code>plt = c(x1, x2, y1, y2)</code>	zulässiger Plot-Bereich als Anteil des normalen Plot-Bereichs
<code>mar = c(unten, links,           oben, rechts)</code>	Margin: freier Raum zwischen Rand und Grafik, die Angabe erfolgt z.B. in inches
<code>bty = ...</code>	Form des Rahmens der Grafik, u.a. "n" für keinen Rahmen
<code>oma = c(unten, links,           oben, rechts)</code>	Breite des äußeren Rands in inches
<code>pch = n</code>	Plotsymbol für Punkte 1 bis 18
<code>lty = n</code>	Code: 1 = solid, 2 = dashed (gestrichelt), 3 = dotted (gepunktet), 4 = dotdash, 5 = longdash, 6 = twodash
<code>font = n</code>	Code: 1 = normal, 2=fett, 3=kursiv, 4 = fett + kursiv, 5 = Symbol
<code>font.axis = n</code>	Font-Code Achse
<code>font.main = n</code>	Font-Code Titel
<code>font.sub = n</code>	Font-Code Subtitel
<code>lfont.lab = n</code>	Font-Code für die x- und y-Achsen-Labels
<code>family = ...</code>	Name der Fontfamilie (z.B. serif, symbol) für Text
<code>col = ...</code>	Farbe für Beschriftung (Die Farbwahl erfolgt entweder wie in html: <i>RRGGBB</i> mit R/G/B-Anteil oder als Wort: z.B. green, lightred, darkblue, yellow oder mit der Angabe einer Zahl (1:n)
<code>col.axis = ...</code>	Farbe für die Achsenerklärung
<code>col.lab = ...</code>	Farbe für die x- und y-Achsen-Labels
<code>col.main = ...</code>	Farbe für den Titel
<code>col.sub = ...</code>	Farbe des Subtitels
<code>cex = n</code>	$n$ = Vergrößerungs- ( $>1$ ) bzw. Verkleinerungsfaktor ( $<1$ ) für Text beziehungsweise
<code>cex.axis = n</code>	Tickmark-Labels
<code>cex.lab = n</code>	Achsenbeschriftungen
<code>cex.main = n</code>	Titel
<code>cex.sub = n</code>	Font-Code Achse

Tabelle 6: Fortsetzung: Argumente

Argument	Beschreibung
<code>adj = n</code>	Textausrichtung: 0 = links, 0.5 = zentriert, 1 = rechts
<code>ann = T/F</code>	Ausgabe von Titel und Achsenbeschriftungen
<code>ask = T/F</code>	Benutzereingabe vor Ausgabe der Grafik ja/ nein
<code>bg = ...</code>	Hintergrundfarbe
<code>fg = ...</code>	Vordergrundfarbe
<code>crt = n</code>	Winkel für die Rotation einzelner Werte
<code>srt = n</code>	Winkel für Rotation einzelner Zeichensymbole
<code>lab = c(nx, ny, len)</code>	Anzahl der Tickmarks (Labels) auf den Achsen (nx, ny) bzw. Anzahl Zeichen für Beschriftung der Tickmarks
<code>las = n</code>	Ausrichtung der Tickmark (Label)-Beschriftung: 0 = parallel zur Achse, 1 = horizontal, 2 = senkrecht zur Achse, 3 = vertikal
<code>tck = n</code>	Länge der Tickmarks, je nach Vorzeichen weisen diese nach innen oder nach außen

## 4 Bereiche eines Devices

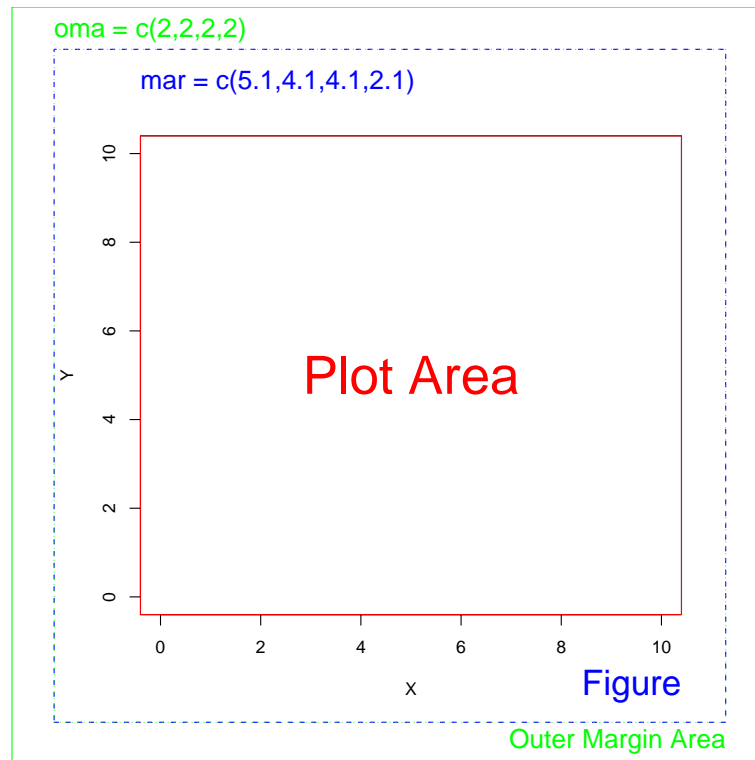
Jeder Device kann in verschiedene Bereiche eingeteilt werden.<sup>2</sup> Man kann in die folgenden drei Bereiche differenzieren:

- Plot Area
- Figure
- Outer Margin Area

Mit den oben aufgeführten Befehlen `mar()` und `oma()` können diese Bereiche verändert werden. Dabei ist auf die Reihenfolge (unten, links, oben rechts) bei der Angabe der Abstände zu achten.

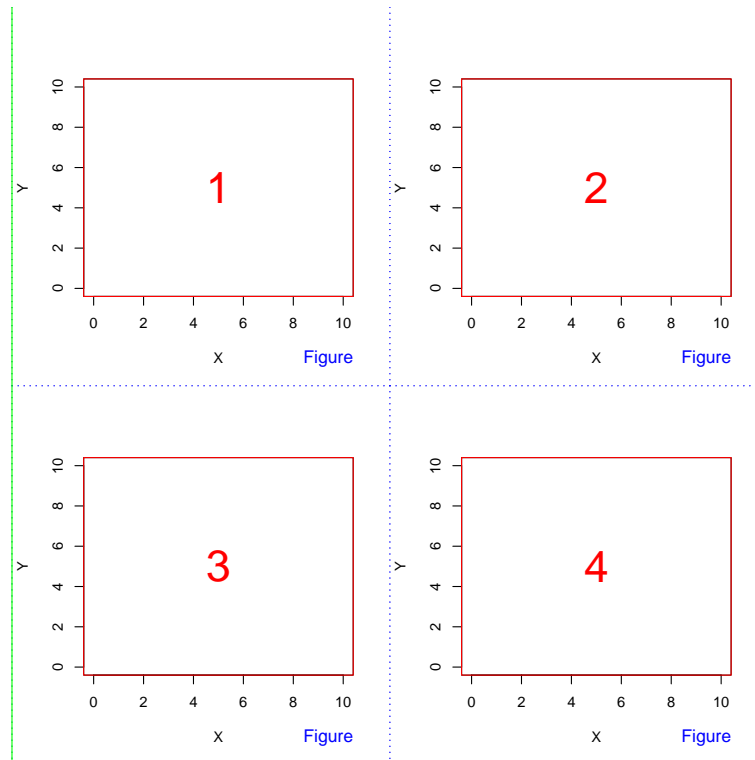
---

<sup>2</sup>Die folgenden Grafiken stammen von [Earl F. Glynn \(Stand Januar 2007\)](#)

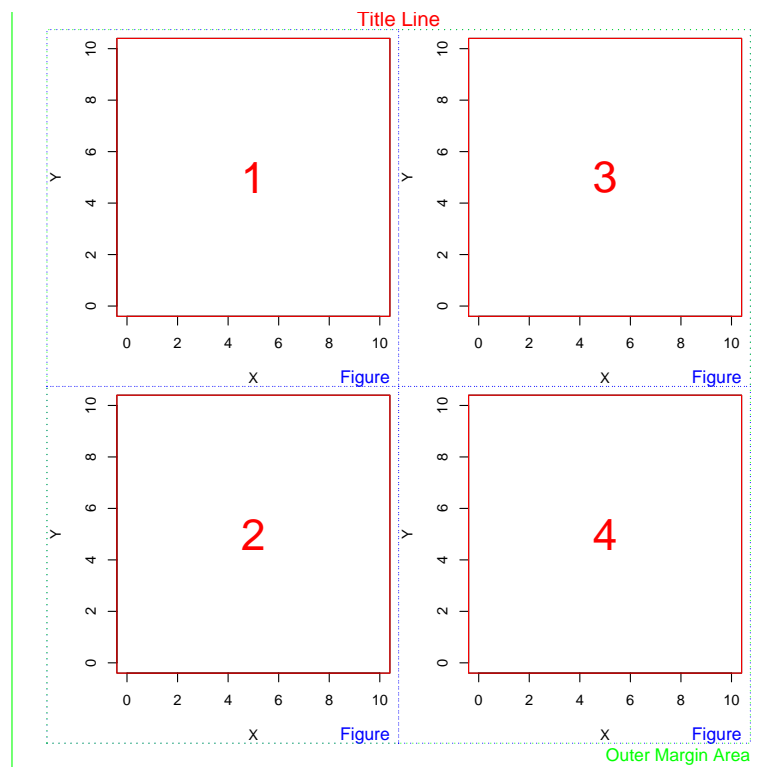


Ein Device kann auch für mehrere Grafiken spezifiziert werden.

```
> par(mfrow=c(2,2), mar=c(5.1,4.1,4.1,2.1), oma=c(0,0,0,0))
```



```
> par(mfcol=c(2,2), mar=c(4,4,0.5,0.5), oma=c(1.5,2,1,1))
```



## 5 Low-Level-Grafiken

Low-Level-Grafikfunktionen können zu bestehenden Grafiken zusätzliche Elemente hinzugefügt werden.

Die umfangreiche Funktionsvielfalt muss mit `?name` in Erfahrung gebracht werden.

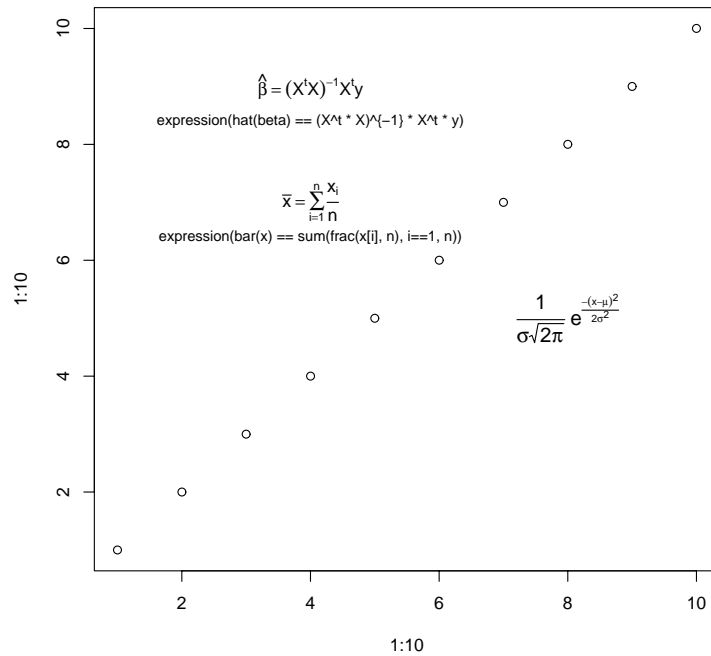
Tabelle 7: Low-Level-Grafikfunktionen

Funktion	Beschreibung
<code>abline()</code>	Linien (z.B. Regressionsgeraden aus x,y)
<code>arrows()</code>	Pfeile
<code>grid()</code>	Gitternetz
<code>legend()</code>	Legende
<code>lines()</code>	Linien (Angabe von x- und y-Koordinaten)
<code>mtext()</code>	Text in den Rändern zwischen den Bereichen
<code>points()</code>	Punkte
<code>text()</code>	Text in der Grafik
<code>locator()</code>	Ermittelt durch Klick in eine Grafik die Koordinaten
<code>identify()</code>	Bei Klicken in die Nähe des definierten Punktes, wird eine Ausgabe erzeugt

## 6 Mathematische Beschriftung

Oft ist es von Nutzen, Grafiken mit mathematischen Symbolen und/ oder griechischen Buchstaben zu ergänzen. Dafür kann man die Funktion `expression()` verwenden.

```
> plot(1:10, 1:10)
> text(4, 9, expression(hat(beta) == (X^t * X)^{-1} * X^t * y))
> text(4, 8.4, "expression(hat(beta) == (X^t * X)^{-1} * X^t * y)", cex = .8)
> text(4, 7, expression(bar(x) == sum(frac(x[i], n), i==1, n)))
> text(4, 6.4, "expression(bar(x) == sum(frac(x[i], n), i==1, n))", cex = .8)
> text(8, 5, expression(paste(frac(1, sigma*sqrt(2*pi)), " ",
+   plain(e)^{frac(-(x-mu)^2,2*sigma^2)})), cex = 1.2)
```



Eine Liste aller Optionen findet man unter

`> ?plotmath`



## Praxis

1. Erstellen Sie wiederum eine Grafik (Streudiagramm) mit den beiden Variablen Einkommen und Alter. Beschriften Sie die Achsen.
2. Zeichnen Sie die Regressionsgerade ein.
3. Zeichnen Sie außerdem einen Punkt in die Grafik ein, der deutlich zu erkennen ist.
4. Beschriften Sie diesen Punkt mit *Der springende Punkt*.
5. Lesen Sie den Datensatz *allison\_2.sav* ein.
6. Erzeugen Sie für Männer und Frauen getrennte Boxplots zum Einkommen in einem Device. Beschriften Sie die Abbildungen mit Legende und Überschrift.
7. Erstellen Sie 4 Bildungsgruppen und erzeugen Sie pro Gruppe ein Streudiagramm für Einkommen und Alter.
8. Zeichnen Sie die Regressionsgerade ein. Beschriften Sie die Gerade mit der entsprechenden Funktion ( $y = a + bx$ ).
9. Heben Sie die Grafiken farblich hervor.

## 7 Dynamische und Interaktive Grafiken

Einen Einblick in die Möglichkeiten erhält man mit den Beispielen im Paket `rgl`.

```
> library(rgl)
> example(surface3d)
> example(plot3d)
```

Weitere illustrative Beispiel findet man im Paket `iplots`.

```
> library(MASS)
> data(Cars93)
> attach(Cars93)
> ibar(Cylinders)

> iplot(EngineSize, Horsepower)
> subs <- iset.selected()
> iabline(lm(Horsepower ~ EngineSize,
> subset=subs))
```