

# Statistische Software (R)

**Paul Fink, M.Sc., Eva Endres, M.Sc.**

Institut für Statistik

Ludwig-Maximilians-Universität München

*Fortgeschrittene Grafiken*



# Das `type` Argument in `plot()`

```
> y <- rnorm(20)
> plot(y, type = "p")
> plot(y, type = "l")
> plot(y, type = "b")
> plot(y, type = "h")
```

`type` Argument interpretiert x- und y-Werte unterschiedlich:

`type="p"` x- und y- Werte repräsentieren die Koordinaten von Punkten

`type="l"` Punkte werden durch Linie verbunden

`type="o"`

`type="h"` Stabdiagramm mit x-Werte die Position der Stäbe und y-Werte als Höhe der Stäbe

# Weitere Elemente hinzufügen: *low-level*

Szenario für Grafik:

Kurven von mittleren Abweichungen bei 2 Maschinen:

```
> set.seed(122333)
> maschine1 <- rnorm(30)
> maschine2 <- rnorm(30)
```

Nun sollen beide in einer Grafik auftauchen:

```
> plot(maschine1, type = "l", main = "Verlauf von 2 Maschinen",
+      ylab = "mittlere Abweichung", xlab = "Tage", col = "red")
> # Versuch: Verlauf von y zu Plot hinzufuegen
> plot(maschine2, type = "l", main = "Verlauf von 2 Maschinen",
+      ylab = "mittlere Abweichung", xlab = "Tage", col = "blue")
> # Macht einen neuen Plot!!
```

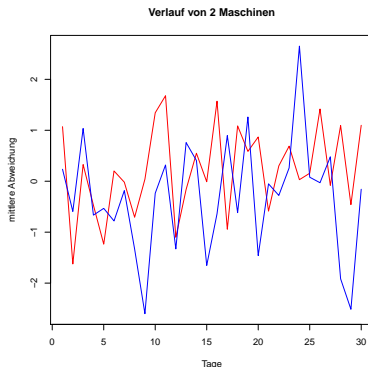
Nicht mit High-Level Funktionen zu machen!

⇒ Low-Level-Funktionen nutzen

# Weitere Elemente hinzufügen: *low-level*

- Zeichnen des Verlaufs der 1. Maschine mit `plot()` (high-level)
- Zeichnen des Verlaufs der 2. Maschine mit `lines()` (low-level)

```
> yrange <- range(maschine1, maschine2)
> plot(maschine1, type = "l",
+      main = "Verlauf von 2 Maschinen",
+      ylim = yrange, xlab = "Tage",
+      ylab = "mittlere Abweichung",
+      col = "red")
> lines(maschine2, col = "blue")
```



# Übersicht low-level Funktionen

Funktion	Beschreibung
<code>points()</code>	Punkte an Stellen $(x, y)$
<code>lines()</code>	Linien zwischen den Stellen $(x, y)$
<code>segments()</code>	Liniensegmente, ausgehend von $(x_0, y_0)$ zu allen Punkten in $(x, y)$
<code>arrows()</code>	ähnlich wie <code>segments()</code> , nur mit Pfeilspitzen
<code>xspline()</code>	Geglättete Kurve durch die Punkte $(x, y)$
<code>rect()</code>	Rechteck, mit linker unterer Ecke $(xl, yb)$ und rechter oberer Ecke $(xr, yt)$
<code>polygon()</code>	Polygonzug mit Knoten in $(x, y)$
<code>text()</code>	Text hinzufügen an Position $(x, y)$
<code>title()</code>	Beschriftung(en) zur Grafik hinzufügen
<code>axis()</code>	Achsen hinzufügen
<code>abline()</code>	Eine oder mehrere Geraden
<code>grid()</code>	Gitternetz

# Globale Konfiguration

Grafiklayout und globale Einstellungen ändern mit Funktion `par()`

`par(Parameter1 = Wert1, usw.)` vor ersten Grafikfunktion

Überblick über meist verwendete Parameter

Parameter	Beschreibung
<code>ask</code>	Wenn <code>TRUE</code> , Zeichnen eines neuen Plots durch Drücken der Eingabetaste
<code>cex</code>	Vergrößerung (siehe Tabelle in Folien 9)
<code>las</code>	Ausrichtung der Achsenbeschriftung
<code>mfrow, mfcoll</code>	Mehrere Grafiken in einem Display, Vektor <code>c(nr, nc)</code> , Einzeichnen zeilen- bzw. spaltenweise
<code>new</code>	Wenn <code>TRUE</code> , dann wird Grafikdisplay nicht geleert
<code>xaxis, xaxt</code>	Konfiguration der x-Achse
<code>yaxis, yaxt</code>	Konfiguration der y-Achse

# Mathematische Ausdrücke in Grafiken

Text mit z.B. griechischen Symbolen in `expression()` stecken:

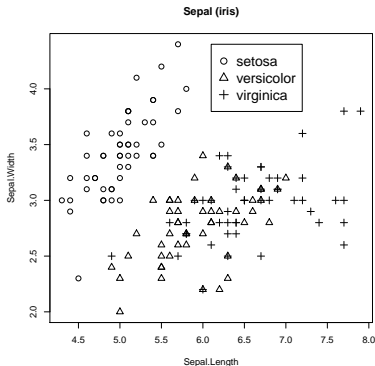
Beispiel aus `?plotmath`:

```
> x <- seq(-4, 4, len = 101)
> y <- cbind(sin(x), cos(x))
> matplot(x, y, type = "l", xaxt = "n",
+         main = expression(paste(plain(sin) * phi, " and ",
+                                 plain(cos) * phi)),
+         ylab = expression("sin" * phi, "cos" * phi), # only 1st is taken
+         xlab = expression(paste("Phase Angle ", phi)),
+         col.main = "blue")
> axis(1, at = c(-pi, -pi/2, 0, pi/2, pi),
+      labels = expression(-pi, -pi/2, 0, pi/2, pi))
```

# Legenden

Beispiel:

```
> with(iris,
+ plot(Sepal.Length, Sepal.Width,
+      main = "Sepal (iris)",
+      pch = as.numeric(Species),
+      cex = 1.2)
+ )
>
> legend(x = 6.1, y = 4.4,
+       legend = c("setosa",
+                  "versicolor",
+                  "virginica"),
+       cex = 1.5, pch = 1:3)
```



**Achtung:** Es nicht sichergestellt, dass die Legende zur Grafik passt!



# Umgang mit Farben

Übersicht über die built-in Farben: `colors()`

Farben sind sehr hilfreich in Grafiken, aber nicht jede Kombination!

Zum Umgang mit Farben: [Präsentation von Achim Zeileis \(2010\)](#)

Farbwahl abhängig von Verwendung: Beamer, Ausdruck in verschiedener Qualität, Bildschirm

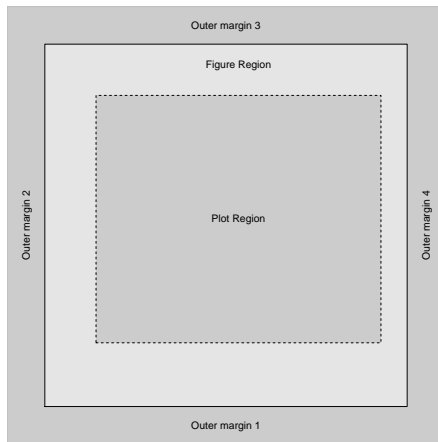
- Bei Abstufungen nicht zu ähnliche Farben wählen
- Idealerweise Farbblindheit berücksichtigen (Rot-Grün-Schwäche)
- Oft reichen auch Graustufen

Online Colorbrewer liefert einen guten Start zur Farbwahl:

<http://colorbrewer2.org/>

# Exkurs: Aufbau von traditionellen Grafiken<sup>1</sup>

Aufbau eines Plots:

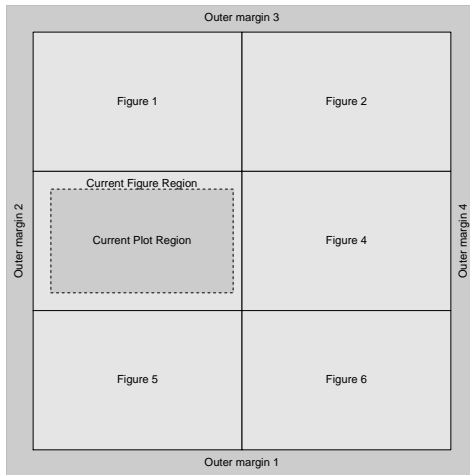


---

<sup>1</sup>Quelle: *Murrell (2011): R Graphics, 2nd Edition, Chapman & Hall*

# Exkurs: Aufbau von traditionellen Grafiken

## Aufbau mehrerer Plots:



# Exkurs: Aufbau von traditionellen Grafiken

Koordinatensystem:

