Statistische Software (R)

Paul Fink, M.Sc.

Institut für Statistik
Ludwig-Maximilians-Universität München

Grafik

Allgemeines zu Grafiken

Leitmotiv: Sinnvolle leicht zu verstehende Grafiken

Leitfragen:

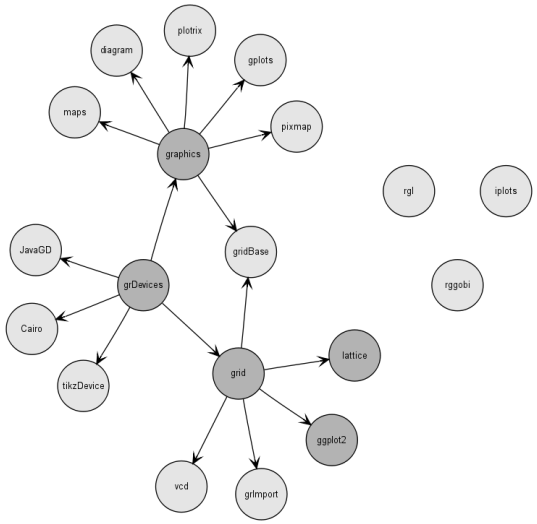
1. Welchen Grafiktyp verwenden?
2. Ist der Inhalt der Grafik klar dargestellt?
3. Einsatz von (einheitlichen) Farben?
4. Ist der Text lesbar?
5. Legende?

Literatur

R Graphics Second Edition
by Paul Murrel

Link zu den im Buch dargestellten Grafiken und dem zugehörigem R code:
http://www.stat.auckland.ac.nz/~paul/RG2e/

Übersicht von Grafikpaketen (nicht vollständig)



Grafikausgabe

- Die Grafikausgabe von `R` erfolgt in ein sogenanntes Gerät (Device)
- Ein Gerät ist zum Beispiel ein Fenster auf dem Bildschirm oder eine Datei
- Es stehen dabei zum Beispiel folgende Geräte zur Verfügung: `bitmap()`, `jpeg()`, `pdf()`, `pictex()`, `png()`, `postscript()`, `xfig()`, `x11()` bzw. `X11()`.
- Das Zusatzpaket `tikzDevice` stellt zusätzlich die Funktion `tikzdevice()` zur Verfügung, um Grafikcode direkt in TikZ-Code umzuwandeln zum Weiterverwenden mit LaTeX. Ist auf CRAN nicht mehr verfügbar, jedoch weiterhin unter R-Forge.

Grafikausgabe

- Unter Windows(TM) gibt es auch noch `bmp()` und `win.metafile()`.
- Beispiel:

```
x11()
```

  öffnet ein Grafikfenster auf dem Bildschirm und

```
dev.off()
```

  schließt das Fenster wieder.
- Beispiel: Zeichnen der Dichte der Standard-Normalverteilung auf dem Bildschirm mit der Grafikfunktion `curve()`.

```
x11()
curve(from = -3, to = 3, dnorm(x), ylab = "Dichte der N(0,1)-Verteilung")
```


Grafikausgabe

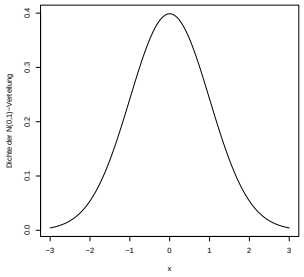
- Statt auf dem Bildschirm, lassen wir uns die Grafik direkt in ein gewünschtes Dateiformat ausgeben, zum Beispiel PDF oder Postscript.

```
pdf(file="dichteN01.pdf")
curve(from=-3, to=3, dnorm(x), ylab="Dichte der N(0,1)-Verteilung")
dev.off()
```

  Die Datei `"dichteN01.pdf"` wird dabei im aktuellen Arbeitsverzeichnis angelegt. Sie können das Ergebnis z.B. mit dem Acrobat Reader betrachten. Die Postscript-Version geht ganz analog:

```
postscript(file="dichteN01.ps")
curve(from = -3, to = 3, dnorm(x), ylab = "Dichte der N(0,1)-Verteilung")
dev.off()
```


Grafikausgabe



- Bei der Ausgabe in eine Datei muss dass Gerät immer geschlossen werden mit `dev.off()`! Erst beim Schließen wird die Grafik in die Datei gespeichert.

Traditionelle Grafiken (`graphics` Paket)

Zunächst lassen sich die traditionellen Grafik Funktionen in zwei Klassen unterteilen

- *high-level* Grafik-Funktionen erstellen vollständige Grafiken, z.B. Streudiagramme, Boxplots, Histogramme, etc.

- *low-level* Grafik-Funktionen bieten zum einen eine feinere Kontrolle von Details und zum anderen die Möglichkeit zusätzliche Elemente, z.B. beschreibende Etiketten, zusätzliche Linien, etc., zu einer bestehenden Grafik hinzuzufügen.
  Grafiken selber zusammen stellen nach dem Baukasten-Prinzip.

Die `plot()` Funktion

Die `plot()` Funktion ist die wichtigste *high-level* Funktion im traditionellen Grafik System. In vielen Fällen ist es die einfachste Variante um vollständige Grafiken zu erstellen.

Die folgenden drei Befehle erstellen identische Streudiagramme.

```
> plot(pressure)
> plot(pressure$temperature, pressure$pressure)
> plot(pressure ~ temperature, data = pressure)
```

In der ersten Variante werden alle darzustellenden Datenpunkte in einem Objekt der Klasse `data.frame` mit zwei Variablen übergeben. In der zweiten Variante werden die x- und y-Werte einzeln an die Argumente übergeben. Die dritte Variante verwendet eine Formel der Form `y` $\sim$ `x` und zusätzlich die Daten in denen die Variablen enthalten sind.

Die `plot()` Funktion ist generisch

Die `plot()` Funktion kann mit denselben Daten auf unterschiedliche Weise aufgerufen werden und liefert ein identisches Ergebnis.

Abhängig vom Typ der Daten werden von `plot()` unterschiedliche Grafiken erzeugt, wird z.B. für ein Objekt der Klasse `factor` an das Argument `x` übergeben, so wird direkt ein Balkendiagramm erstellt (siehe `?plot.factor`).

Diagramme für eine Variable

Funktion	Daten	Beschreibung
`plot()`	numeric	Scatterplot
`plot()`	factor	Barplot
`plot()`	1-D table	Barplot
`barplot()`	numeric (Höhe der Balken)	Barplot
`pie()`	numeric	Pie chart
`dotchart()`	numeric	Dotplot
`boxplot()`	numeric	Boxplot
`hist()`	numeric	Histogramm
`stripchart()`	numeric	1-D scatterplot
`stem()`	numeric	Stem-and-leaf plot

Diagramme für zwei Variablen

Funktion	Daten	Beschreibung
`plot()`	numeric, numeric	Scatterplot
`plot()`	numeric, factor	Stripcharts
`plot()`	factor, numeric	Boxplots
`plot()`	factor, factor	Spineplot
`plot()`	2-D table	Mosaic plot
`sunflowerplot()`	numeric, numeric	Sunflower Scatterplot
`smoothScatter()`	numeric, numeric	Smoothed scatterplot
`boxplot()`	list of numeric	Boxplots
`barplot()`	matrix	Stacked/side-by-side Barplot
`dotchart()`	matrix	Dotplot
`stripchart()`	list of numeric	Stripcharts
`spineplot()`	numeric, factor	Spinogram
`cdplot()`	numeric, factor	Conditional density plot
`fourfoldplot()`	2x2 table	Fourfold display
`assocplot()`	2-D table	Association plot
`mosaicplot()`	2-D table	Mosaic plot

Diagramme für viele Variablen

Funktion	Daten	Beschreibung
`plot()`	data frame	Scatterplot Matrix
`pairs()`	matrix	Scatterplot Matrix
`matplot()`	matrix	Scatterplot
`stars()`	matrix	Star plots
`image()`	numeric, numeric, matrix	Image plot
`contour()`	numeric, numeric, matrix	Contour plot
`filled.contour()`	numeric, numeric, matrix	Filled contour
`persp()`	numeric, numeric, matrix	3-D surface
`symbols()`	numeric, numeric, numeric	Symbol scatterplot
`coplot()`	formula	Conditioning plot
`mosaicplot()`	N-D table	Mosaic plot

Spezielle Plots

Das traditionelle Grafik System und Pakete, die darauf aufbauen, beinhalten eine Vielzahl an Funktionen um spezielle Grafiken zu erzeugen.

Die meisten dieser Funktionen sind Variationen des einfachen Streudiagramms mit Punkten/Linien in kartesischen Koordinaten. Z.B. `qqplot` und `qqnorm` werden verwendet um beobachtete Werte gegen Werte einer theoretischen Verteilung abzutragen.

Argumente für Grafikfunktionen

Oft möchte man nur kleine Anpassungen an den Grafiken vornehmen, wie zum Beispiel den Titel oder die Achsenbeschriftung ändern oder die Plot-Region.

Argument	Beschreibung
`main`	Haupttitel der Grafik
`xlab, ylab`	Titel der X-Achse bzw Y-Achse
`xlim, ylim`	Vektor mit Minimum/Maximum für Werte in der Plot-Region in X bzw. Y Richtung
`cex`	Generelle Vergrößerung
`cex.main, cex.axis, cex.lab`	Vergrößerung von Titel, Achsenbeschriftung und -titeln relativ zu `cex`
`col`	Farbe der Objekte in der Plot-Region Kann auch wie `cex` auf Titel und Achsen angewendet werden
`axes`	Bei `FALSE` werden keine Achsen eingezeichnet
`xaxt="n", yaxt="n"`	Kein Einzeichnen von X bzw. Y-Achse
`lty, lwd`	Linientyp, Linienbreite
`type`	`"p"`: Punkte, `"l"`: Linien, `"b"`: Punkte und Linien, `"h"`: Stäbe, `"n"`: Nichts einzeichen

Das `type` Argument

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

In Abhängigkeit vom `type` Argument werden die x- und y-Werte unterschiedlich interpretiert, für

- `type = "p"` wird ein Streudiagramm dargestellt, x- und y-Werte repräsentieren die Koordinaten der Punkte,
- `type = "l"` oder `type = "o"` wird ein Liniendiagramm dargestellt, x- und y-Werte werden durch eine Linie verbunden,
- `type = "h"` wird ein Stabdiagramm dargestellt, wobei die x-Werte die Position der Stäbe und die y-Werte die Höhe der Stäbe liefern.

Weitere Elemente hinzufügen: low-level

Funktion	Beschreibung
`points()`	Punkte an Stellen (x, y)
`lines()`	Linien zwischen den Stellen (x, y)
`segments()`	Liniensegmente, ausgehend von $(x0, y0)$ zu allen Punkten in (x, y)
`arrows()`	ähnlich wie `segements`, nur mit Pfeilspitzen an dem/n Ende/n
`xspline()`	Geglättete Kurve durch die Punkte (x, y)
`rect()`	Rechteck, mit linker unterer Ecke (xl, yb) und recher oberer Ecke (xr, yt)
`polygon()`	Polygonzug mit Knoten in (x, y)
`text()`	Text hinzufügen an Position (x, y)
`title()`	Beschriftung zur Grafik hinzufügen (Titel und/oder Achsen)
`axis()`	Achsen hinzufügen; X-Achse: `side="1"`, Y-Achse: `side="2"`
`abline()`	Eine oder mehrere Geraden
`grid()`	Gitternetz

Globale Konfiguration

Das Grafiklayout kann man durch setzten von Grafikparametern in der Funktion `par` ändern. Dazu muss `par(Grafikparameter1 = Wert1, usw.)` vor der ersten Grafikfunktion aufgerufen werden

Hier nur ein Überblick über die wichtigsten Parameter

Parameter	Beschreibung
`ask`	Wenn `TRUE`, Zeichnen eines neuen Plots durch die Drücken der Eingabetaste
`cex`	Vergrößerung (siehe Tabelle davor)
`las`	Ausrichtung der Achsenbeschriftung
`mfrow, mfcol`	Mehrere Grafiken in einem Display, Vektor der Form `c(nr, nc)`, Einzeichen zeilen- bzw. spaltenweise
`new`	Wenn `TRUE`, dann wird Grafikdisplay nicht geleert
`xaxp, xaxs, xaxt, xlog`	Konfiguration der x-Achse
`yaxp, yaxs, yaxt, ylog`	Konfiguration der y-Achse

Mathematische Ausdrücke in Grafiken

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,
+           pch = as.numeric(Species), cex = 1.2))
> legend(6.1, 4.4, c("setosa", "versicolor", "virginica"),
+        cex = 1.5, pch = 1:3)
```

Achtung: Es nicht sichergestellt, dass die Legende zur Grafik passt. Stimmen die Symbole? Stimmen die Bezeichnungen? Dies liegt in der Verantwortung des Anwenders.

Die Zusatzpakete `lattice` und `ggplot2` bieten mehr Unterstützung.

Online Dokumentation:
http://had.co.nz/ggplot2/
by Hadley Wickham

Umgang mit Farben

Farben sind sehr hilfreich in Grafiken, jedoch sollte man aufpassen, welche man verwendet.

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

Zum Umgang mit Farben:
Präsentation von Achim Zeileis (2010)
Präsentation von Paul Murrel (2002)

Online Colorbrewer:
http://colorbrewer2.org/

Die Wahl hängt aber auch immer von dem Gerät ab für das die Grafik verwendet wird, i.e. Beamer, Ausdruck in verschiedenen Qualitäten

Interaktive Grafiken

In `R` werden hauptsächlich statische Grafiken erstellt.

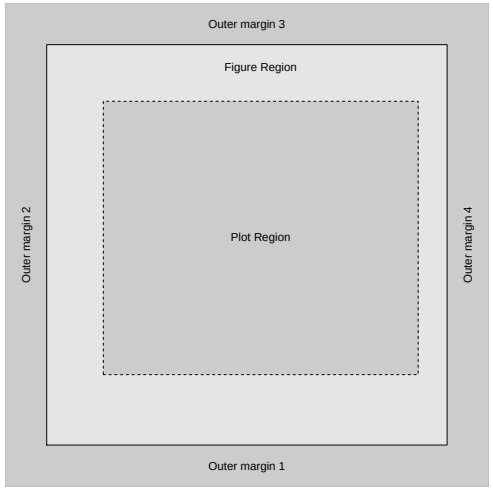
Interaktivität kann erreicht werden, wenn man vor Erstellung mehrerer Plots `par(ask=TRUE)` setzt: Eine neue Grafik wird dann nur nach Betätigung der Eingabetaste (Return) erstellt.

Interaktive Bestimmung von Punkten innerhalb einer bereits erstellten Grafik geht über die Funktionen `identify` und `locator`. Achtung: Beide Funktionen sind fehlerbehaftet und sollten vermieden werden!

Das Paket `iplots` bietet interaktive Grafiken:
http://www.rosuda.org/iplots/

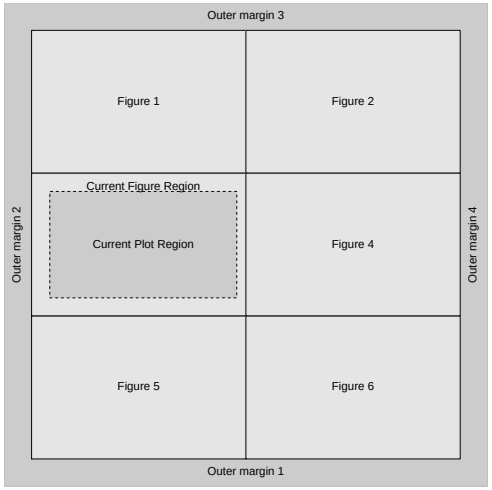
Exkurs: Aufbau von Grafiken

Eine traditionelle R-Grafik hat den folgenden Aufbau:



Exkurs: Aufbau von Grafiken

Für mehrere Grafiken sieht es dann so aus:



Exkurs: Aufbau von Grafiken

Die Darstellung innerhalb einer Figure hat den Aufbau: