



## Ziel

**Ziel** statistischer Graphiken ist die **Repräsentation** von Daten-gefügen, um Werte, Anzahlen, Verhältnisse, Zusammenhänge, Strukturen usw. unmittelbar sehen zu können, [1], [5], [8].

Beispiele (→ steht für *repräsentieren*):

- Stem and leaf Ziffern, y-Werte → Objekte, Größen
- Scatterplot Koordinaten → Messwerte
- Stabdiagramm Stablängen → Häufigkeiten
- Kreisdiagramm Sektoren → Anteile
- Boxplot Elemente → Extrema, Angeln, Ausreißer
- Chernoff face Gesichtselemente → Ausprägungen
- Dendrogramm Verbindungshöhen → Distanzen
- Mosaic-Plot Flächen → Häufigkeiten

Farbe, Form, Symbol, Größe → Infos, Gruppeneigenschaften

**Ziel für pic.plot():** Visualisierung von Kontingenztafeln

## Repräsentation von Objekten und Anzahlen

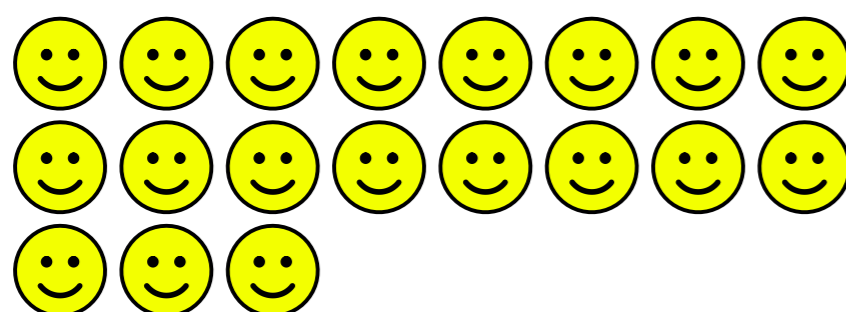
- Darstellung einzelner Objekte → Punkte, Symbole, Bilder:



- Darstellung von 19 Objekten → Reihe von Symbolen:



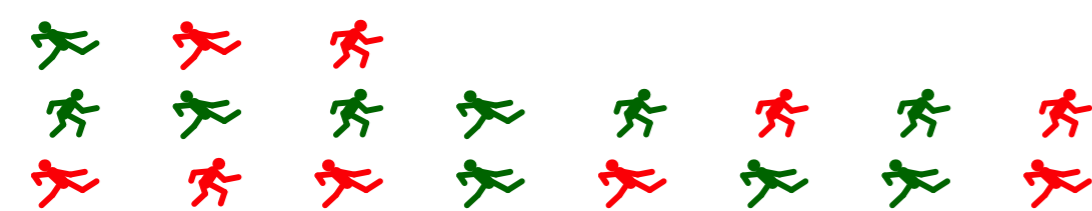
- Anzahlen → Aufteilung der Reihe auf mehrere **Stacks** für 19 Objekte → mit drei Stacks (mit max. Länge 8):



- Unterscheidung von Gruppen durch verschiedene Symbole → 12 bzw. 7 Strichmännchen zweier Gruppen:



- Unterscheidung von Gruppen durch Symbole *und* Farben:



## Das Projekt pic.plot()

- Urknall: R-Package `aplpack` – [6], Logo-Diskussionen sowie `pictogram()` von Ulrike Grömping, [2].
- Vision: Visualisierung mehrdimensionaler Häufigkeitstabellen mittels Repräsentation der zugrundeliegenden Elemente durch Pictogramme in schachbrettartigen Feldern (Panels).
- Anforderungen, die bisher erfüllt wurden:
  - schachbrettartige Zerlegung der Ausgabefläche in *Panels*
  - wiederholte Zerlegung der Fläche: horizontal wie vertikal
  - gruppenabhängige Farb- und Symbol-Wahl
  - einfache Bediensprache für Zerlegung, Farb-, Symbol-Wahl
  - Inputs aus den Klassen `table` und `data frame`
  - Transformation numerischer Größen
  - Verarbeitung gebrochener und negativer Einträge
  - Größen- und Randmodifikation für die Panels
  - diverse Layout-Anordnungen für Pictogramme in Panels
  - Symbole, Liniengraphiken und Bilder als Pictogramme
  - Erklärungstexte: Randbeschriftungen und Legenden
  - Verfeinerungen durch Low-Level-Routinen

<sup>ca</sup>Hans Peter Wolf, Fak. für Wirtschaftswissenschaften, Uni Bielefeld, e-Mail: [pwolf@wiwi.uni-bielefeld.de](mailto:pwolf@wiwi.uni-bielefeld.de)

## User Interface

Um die Argumentenvielfalt beherrschbar zu machen, erhielten Argumente je nach Kontext gleiche Namensanfänge (`grp`, `panel`, `pic`, `lab`). Die hinteren Namensteile weisen auf die Konfigurationszwecke hin (z. B. `cex`, `frame`, `space.factor`).

Die zulässigen Wertemengen wurden für die intendierten Fälle möglichst sinnvoll und sparsam entworfen. So lassen sich Zerlegungen der Ausgabefläche in x- und y-Richtung durch R-Formeln mit Variablennamen wie bei linearen Modellen ausdrücken. Variablen können aber auch einfach über Dimensionsnummern referenziert werden. Taucht auf der linken bzw. rechten Seite von `~` mehr als eine Variable auf – verbunden durch ein `+`-Zeichen –, führt das zu verschachtelten Zerlegungen des x- bzw. y-Bereichs.

## Argumente von pic.plot()

Liste wichtiger Argumente mit Bedeutungshinweisen:

Bezug	Argument	Bedeutung
Input	<code>data</code>	Input: Tabelle oder data frame
Input	<code>vars.to.factors</code>	Transformation von Zahlen in factors
Gruppierung	<code>grp.xy</code>	Variablen für horizontale / vertikale Zerlegung
Gruppierung	<code>grp.color</code>	Variable für Farb-Wahl
Gruppierung	<code>grp.pic</code>	Variable für Pictogramm-Wahl
Gruppierung	<code>colors</code>	Menge der Farben
Gruppierung	<code>pics</code>	Menge der Pictogramme
Panels	<code>panel.prop.to.size</code>	Festlegung der Panelgrößen
Panels	<code>panel.margin</code>	Größen der Ränder
Panels	<code>panel.space.factor</code>	Zwischenraum zwischen Panels
Panels	<code>panel.frame</code>	Umrahmung der Panels
Pictogramme	<code>pic.horizontal</code>	horizontale oder vertikale Stacks
Pictogramme	<code>pic.stack.type</code>	Layout-Typ für Pictogramm-Platzierung
Pictogramme	<code>pic.cex</code>	Symbol-Größe
Pictogramme	<code>pic.aspect</code>	Symbol-Seitenverhältnis
Pictogramme	<code>pic.space.factor</code>	Zwischenraum zwischen Pictogrammen
Pictogramme	<code>pic.frame</code>	Umrahmung der Pictogramme
Beschriftung	<code>lab.side</code>	Seiten für Randbeschriftungen
Beschriftung	<code>lab.cex</code>	Beschriftungsgröße
Beschriftung	<code>lab.bboxes</code>	Hintergrund für Randbeschriftungen
Beschriftung	<code>main</code>	Titel

## Bereitstellung und weiteres Vorgehen

- Stand:
  - Konzept und Struktur von `pic.plot()` steht.
  - Alle wesentlichen Programm-Teile sind umgesetzt.
  - Demo mit über 75 Beispielen liegt als .R- und PDF-File vor.
  - Aktuelle Dateien zu `pic.plot()` sind zu finden unter [7]: Funktionsdefinition `pic.plot.R`, Beispiele: `pic.demo.pdf`
- Plan:
  - Check der Argumentenliste: Bezeichnungen, Wirkungen
  - Entwurf weiterer Generierungsfunktionen für Pictogramme
  - Durchführung zusätzlicher Tests
  - Aktualisierung der help page
  - Integration in das R-Paket `aplpack`
  - Bereitstellung der neuen Version von `aplpack` über CRAN

## Quellen

- [1] J. M. Chambers, W. S. Cleveland, B. Kleiner, P. A. Tukey (1988): Graphical Methods for Data Analysis.
- [2] U. Grömping (2015): `pictogram`, R package version 1.0-3, private communications.
- [3] M. Mazziotta, A. Pareto (2014): Non-compensatory Aggregation of Social Indicators: An Icon Representation. In: F. Crescenzi, S. Mignani (eds.), Statistical Methods and Applications from a Historical Perspective, Studies in Theoretical and Applied Statistics.
- [4] D. Meyer, A. Zeileis, K. Hornik (2006): The Structplot Framework: Visualizing Multi-way Contingency Tables with `vcd`. Journal of Statistical Software, 17(3), 1–48.
- [5] E. R. Tufté (2001): The visual Display of Quantitative Information.
- [6] H. P. Wolf (2015): `aplpack`, R package version 1.3.0, URL: <https://cran.r-project.org/web/packages/aplpack>
- [7] H. P. Wolf (2016): Web-Seite zu `aplpack`, URL: [http://www.wiwi.uni-bielefeld.de/lehrebereiche/statoekoinf/comet/wolf/wolf\\_aplpack](http://www.wiwi.uni-bielefeld.de/lehrebereiche/statoekoinf/comet/wolf/wolf_aplpack)
- [8] F. W. Young, P. M. Valero-Mora, M. Friendly (2006): Visual Statistics.

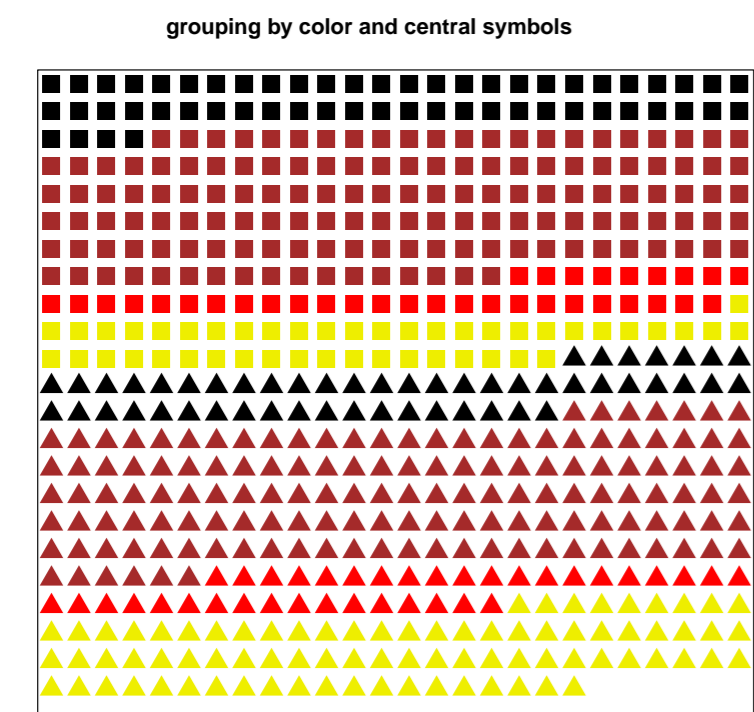


# Anhang: Beispiele

Folgende Beispiele stammen aus der oben erwähnten Demo, siehe [7].

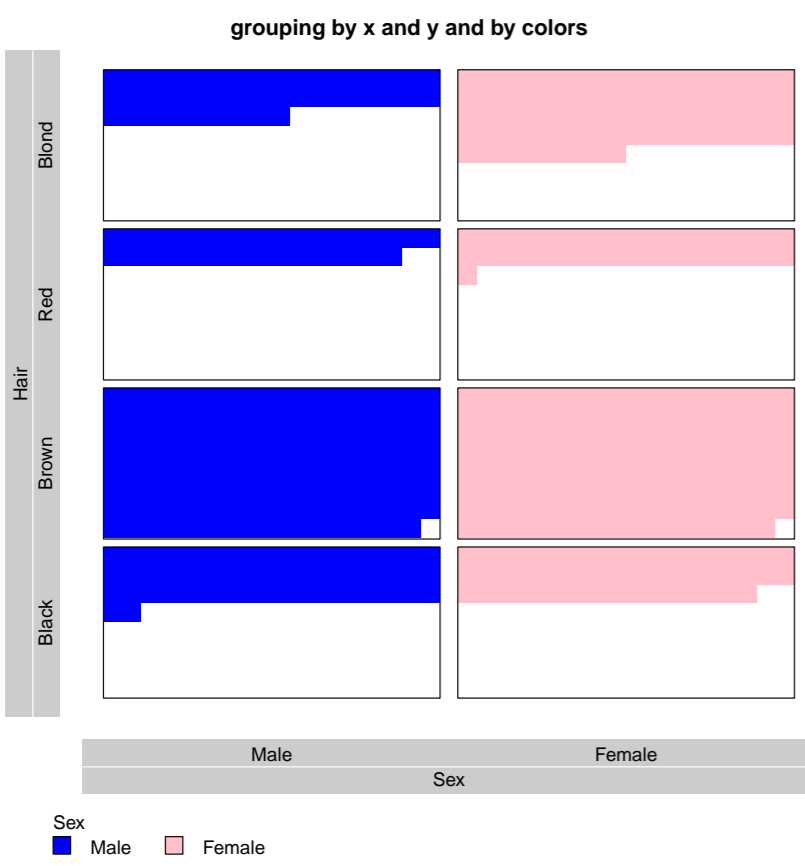
HairEyeColor: Geschlecht, Haarfarbe der Personen hervorgehoben

```
> pic.plot(aperm(HairEyeColor, c(2,1,3)), grp.xy = NULL,
+         grp.pic = Sex,
+         grp.color = Hair,
+         pics = c(15, 17),
+         colors = c("black", "brown", "red",
+                   "yellow2"),
+         pic.frame = FALSE,
+         pic.space.factor = 0,
+         lab.parallel = FALSE,
+         main = "grouping by color and central symbols")
```



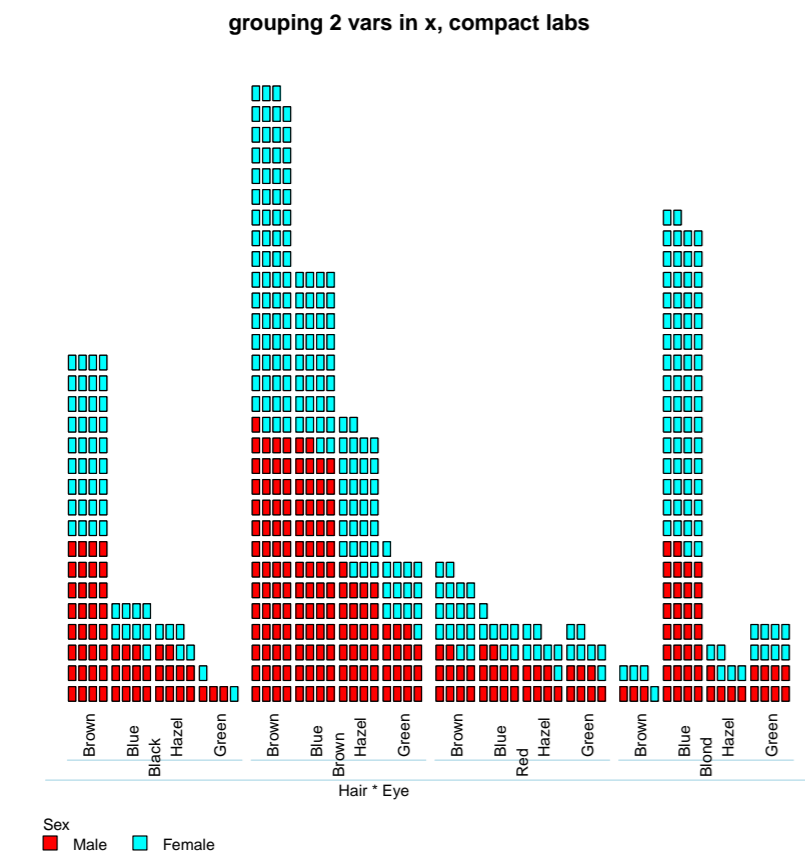
HairEyeColor: gruppiert nach Hair, Sex, ohne Space

```
> pic.plot(HairEyeColor,
+         grp.xy = Hair ~ Sex,
+         grp.color = Sex,
+         colors = c("blue", "pink"),
+         pic.frame = FALSE,
+         pic.space = 0,
+         main = "grouping by x and y and by colors")
```



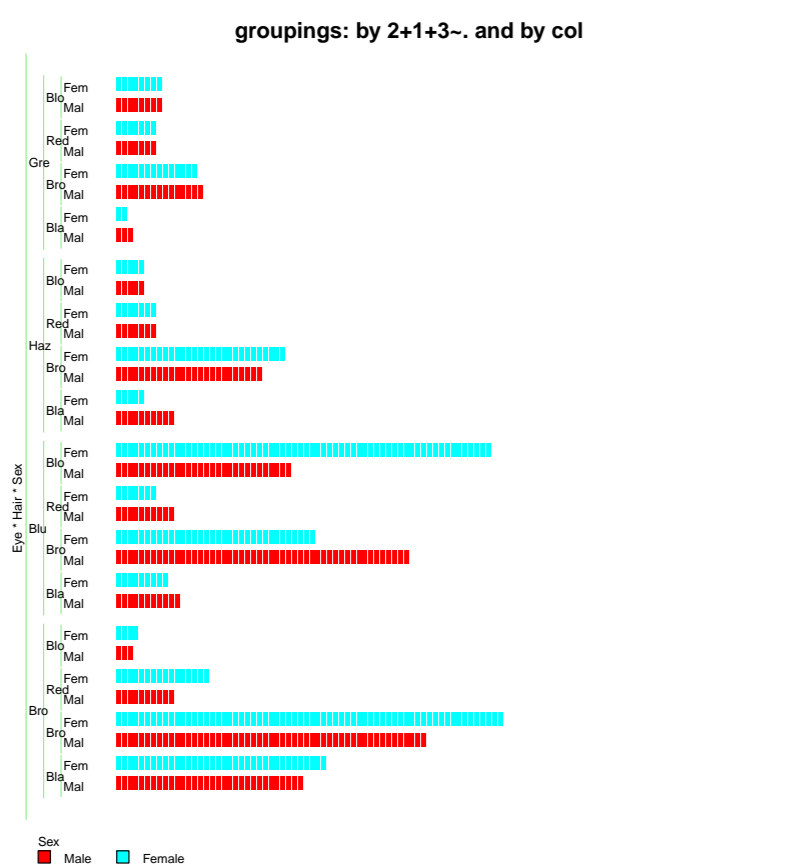
HairEyeColor: x-Bereich zerlegt nach Hair, Eye

```
> pic.plot(HairEyeColor,
+         grp.xy = 0 ~ 1 + 2,
+         grp.color = 3,
+         pic.stack.type = "bl",
+         lab.parallel = c(FALSE, NA, TRUE),
+         pic.aspect = .5,
+         panel.frame = FALSE,
+         lab.color = "lightblue",
+         lab.boxess = 0,
+         lab.cex = 0.8,
+         main = "grouping 2 vars in x, compact labs")
```



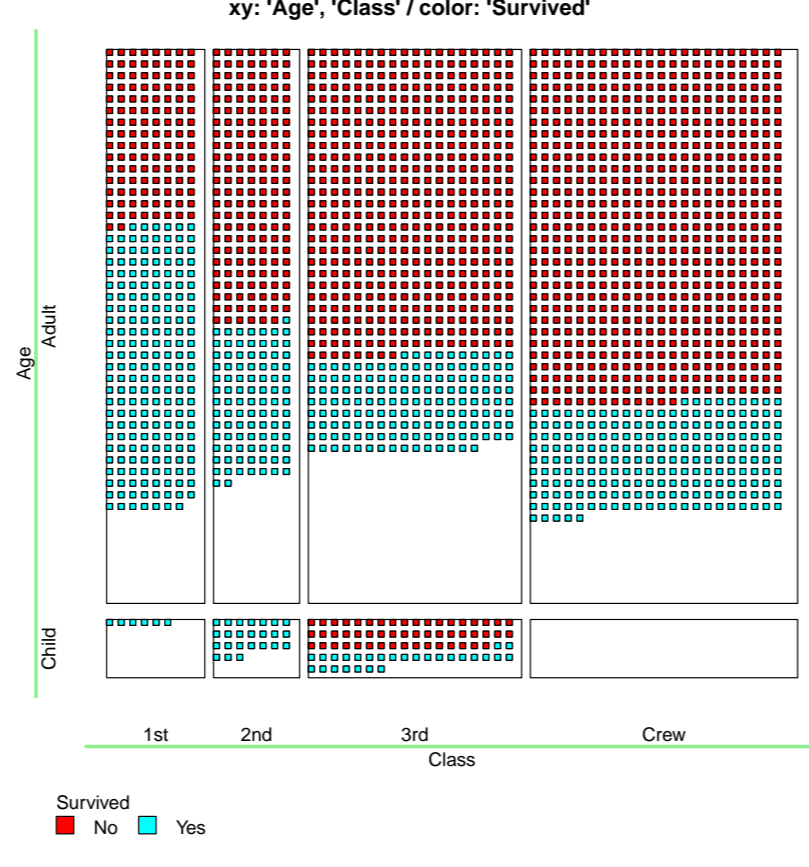
HairEyeColor: y-Bereich zerlegt nach Eye, Hair, Sex

```
> pic.plot(HairEyeColor,
+         grp.xy = 2 + 1 + 3 ~ 0,
+         grp.color = 3,
+         pic.aspect = .3,
+         pic.stack.type = "lt",
+         pic.frame = FALSE,
+         panel.frame = FALSE,
+         lab.parallel = c(FALSE, FALSE, TRUE),
+         lab.color = "lightgreen",
+         lab.boxess = FALSE,
+         lab.cex = 0.7,
+         lab.n.max = c(3,32),
+         main = "groupings: by 2+1+3-. and by col")
```



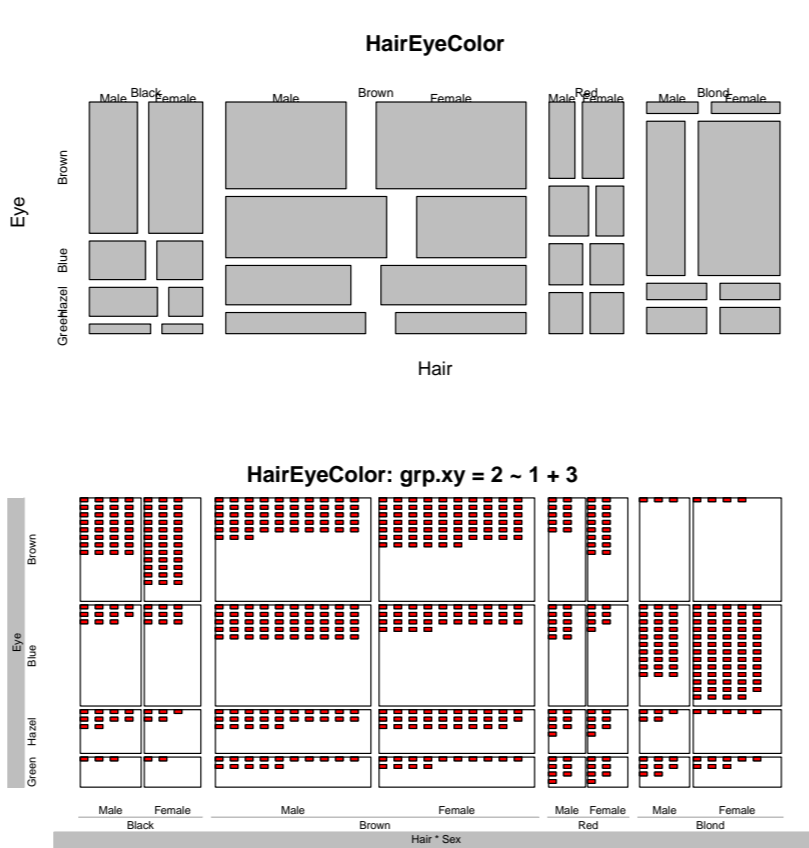
Titanic: Panelgrößen gemäß Randhäufigkeiten

```
> pic.plot(Titanic,
+         grp.xy = 3 ~ 1,
+         grp.color = 4,
+         pic.space.factor = 0.5,
+         panel.prop.to.size = c(TRUE, TRUE), # <-
+         lab.boxess = 0.3,
+         lab.color = "lightgreen",
+         main = "xy: 'Age', 'Class' / color: 'Survived'")
```



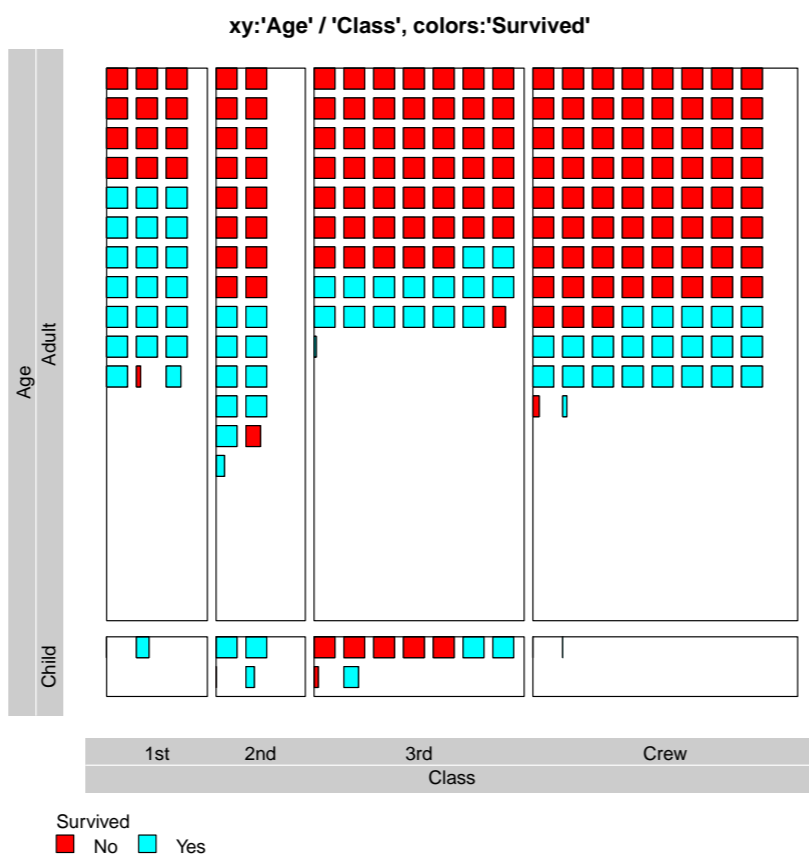
HairEyeColor: Mosaic-Plot versus Pictogramm-Plot, [4]

```
> par(mfrow = 2:1)
> mosaicplot(HairEyeColor)
> pic.plot(HairEyeColor,
+         grp.xy = 2 ~ 1 + 3,
+         lab.parallel = c(TRUE, TRUE, TRUE),
+         colors = "red",
+         pic.space.factor = 0.5,
+         pic.aspect = 2,
+         panel.reverse.y = TRUE,
+         panel.prop.to.size = TRUE,
+         lab.cex = .6,
+         lab.boxess = 1,
+         lab.color = "grey",
+         panel.margin = c(0.00, .035, 0.0, .050),
+         main = "HairEyeColor: grp.xy = 2 ~ 1 + 3'")
> par(mfrow=c(1,1))
```



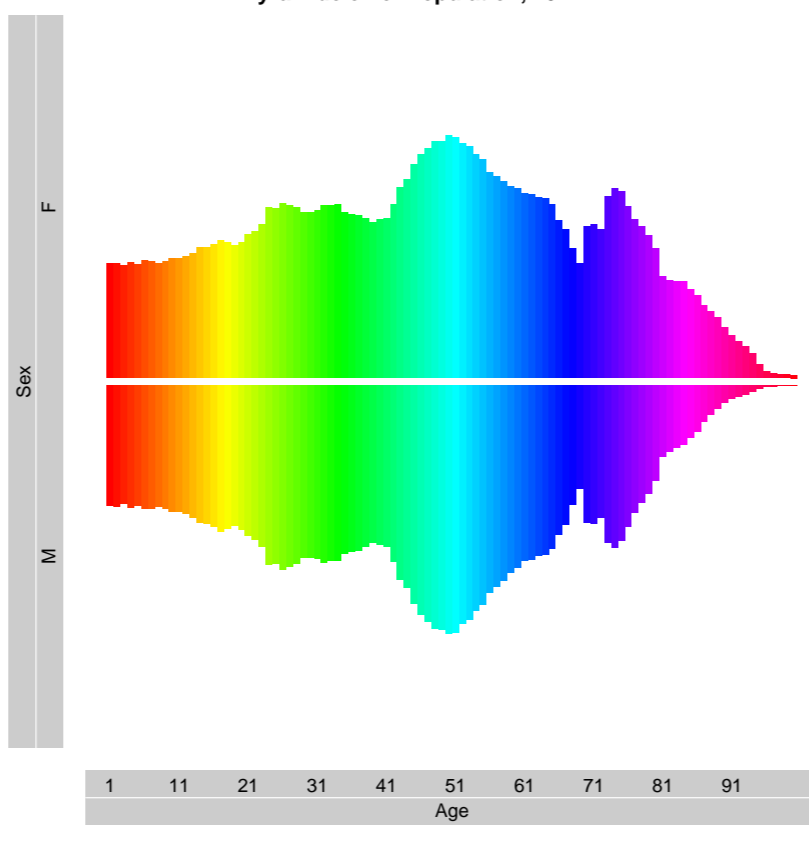
Titanic: gebrochene Pictogramme durch 10-Personen-Einheiten

```
> pic.plot(margin.table(Titanic / 10, c(1,3,4)),
+         grp.xy = 2 ~ 1,
+         grp.color = 3,
+         panel.prop.to.size = c(TRUE, TRUE),
+         main = "xy: 'Age' / 'Class', colors: 'Survived'")
```



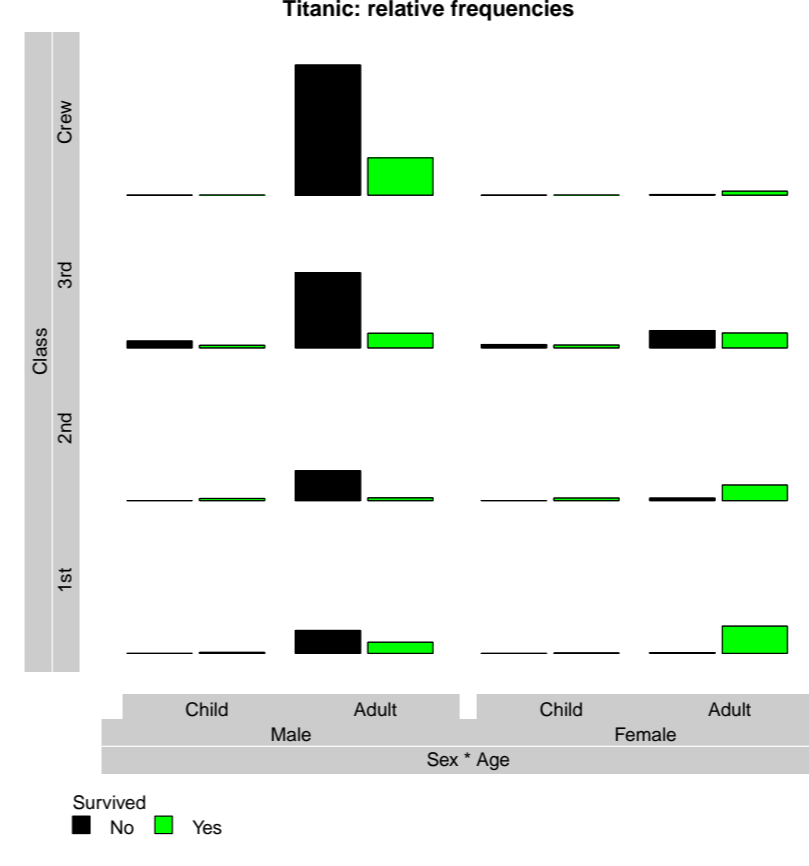
Bevölkerungspyramide, Einheit der Daten: in Tausend

```
> pic.plot(mw,
+         grp.xy = 2 ~ 1,
+         grp.color = 1,
+         pic.aspect = 2,
+         pic.stack.type = rep(c("t", "b"), nrow(mw)),
+         pic.horizontal = FALSE,
+         pic.space.factor = 0,
+         pic.frame = FALSE,
+         panel.frame = FALSE,
+         panel.space.factor = c(.0, .02),
+         lab.n.max = c(2, 10),
+         main = paste("Pyramide einer Population, 2014"))
```



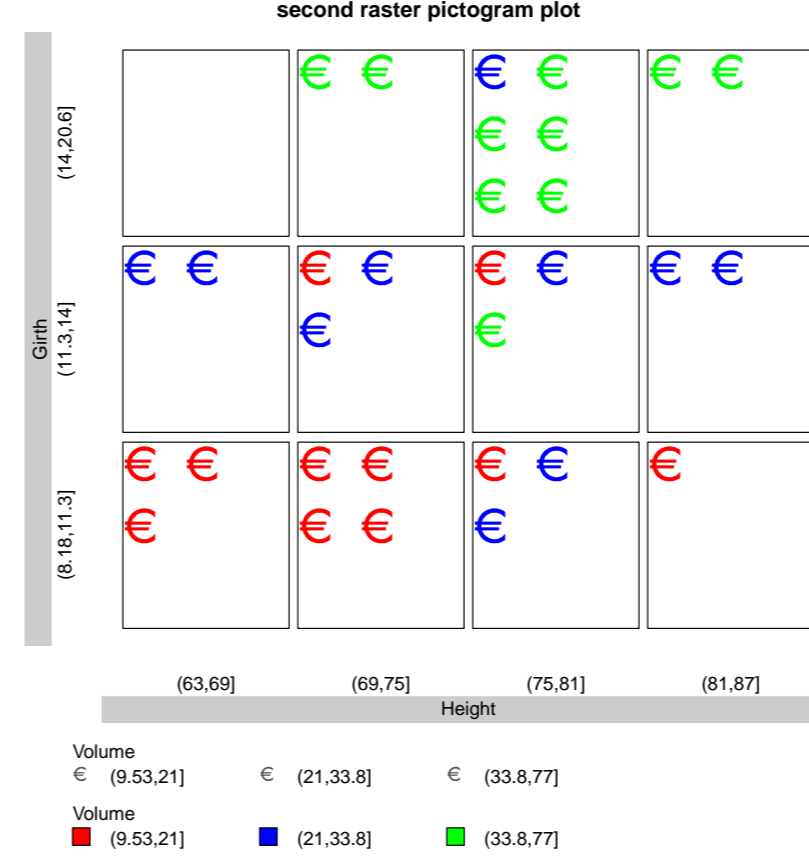
Titanic: Häufigkeiten relativ zur Maximalzahl

```
> pic.plot(Titanic / max(Titanic),
+         grp.xy = 1 ~ 2 + 3,
+         grp.color = 4,
+         colors = c("black", "green"),
+         pic.stack.type = "b",
+         pic.horizontal = FALSE,
+         panel.frame = FALSE,
+         panel.space.factor = 0.05,
+         pic.space.factor = 0.103,
+         pic.aspect = 0.5,
+         main = "Titanic: relative frequencies")
```



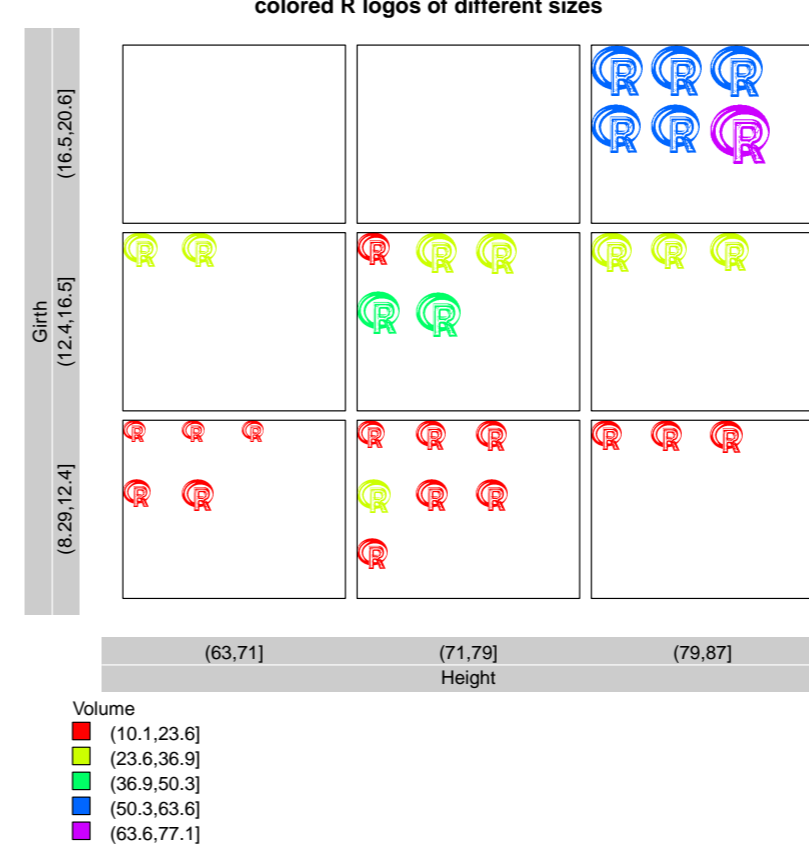
trees: konstruierte Rastergraphik zur Baumrepräsentation

```
> n <- 200; m <- n/2; x <- y <- seq(n <- 200); f <- floor
> image1 <- (outer(x-m)^2, (y-m)^2, FUN="+") > (.8*m)^2
> image2 <- (outer(x-m)^2, (y-m)^2, FUN="+") > (.6*m)^2
> image3 <- image1 / image2; image3[f(.75*n):n] <- 1
> image3[f(.47*n):f(.4*n)] <- f(.05*n):f(.65*n) <- 0
> image3[f(.53*n):f(.6*n)] <- f(.05*n):f(.60*n) <- 0
> image <- as.raster(image3)
> pic.plot(trees,
+         grp.xy = 1 ~ 2,
+         grp.color = 3,
+         grp.pic = 3,
+         colors = c("red", "blue", "green"),
+         pics = image,
+         pic.draft = FALSE,
+         pic.frame = FALSE,
+         vars.to.factors = c(.3, .4, .3),
+         lab.boxess = 1,
+         main = "second raster pictogram plot")
```

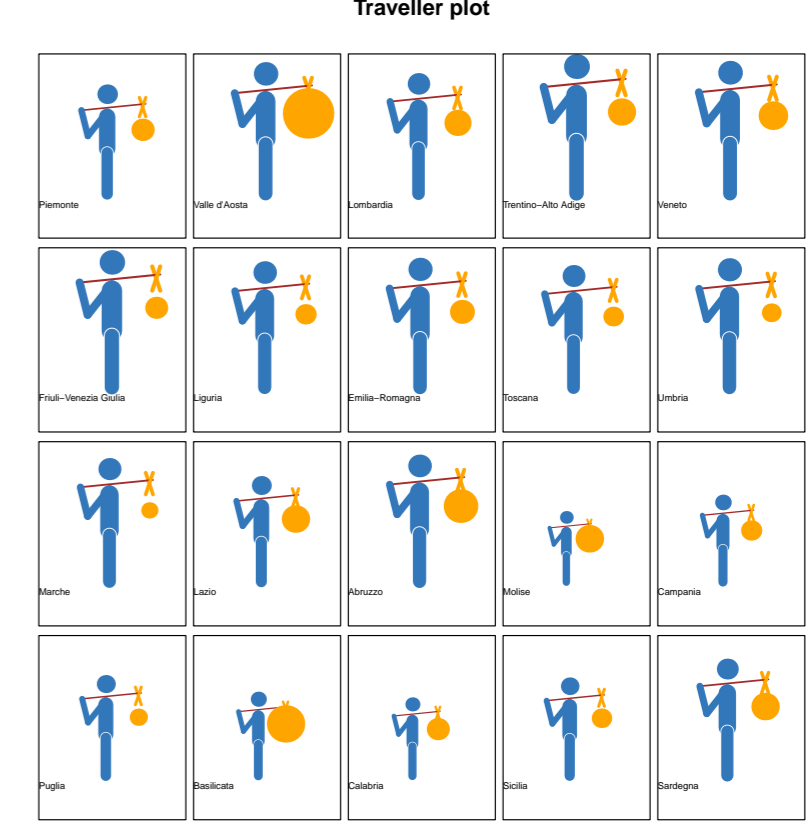


trees: skalierte und umgefärbte R-Icons zur Baumrepräsentation

```
> data <- cbind(trees, pic = 1,
+              fraction.1 = trees[,3] / max(trees[,3]))
> pic.plot(data,
+         grp.xy = 1 ~ 2,
+         grp.color = Volume,
+         grp.pic = pic,
+         pics = "R.pnm",
+         vars.to.factors = c(3, 3, 5),
+         pic.stack.type = "t1s",
+         pic.space.factor = 0.0,
+         pic.frame = FALSE,
+         lab.parallel = c(TRUE, TRUE, FALSE),
+         main = "colored R logos of different sizes")
```

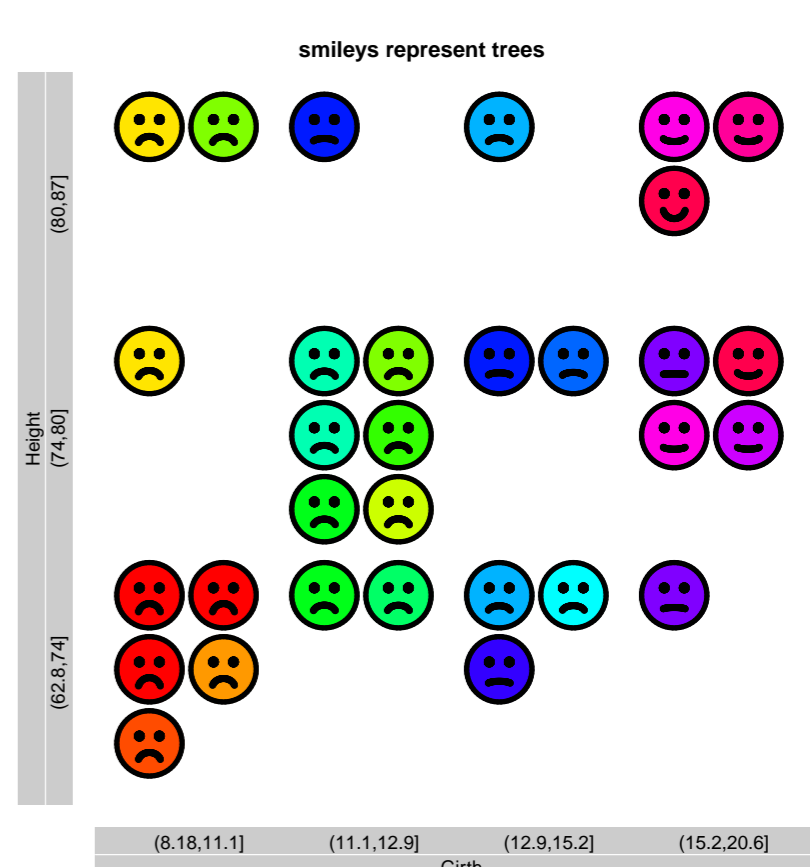


Traveller Icons zur Indikatorenrepräsentation, [3]



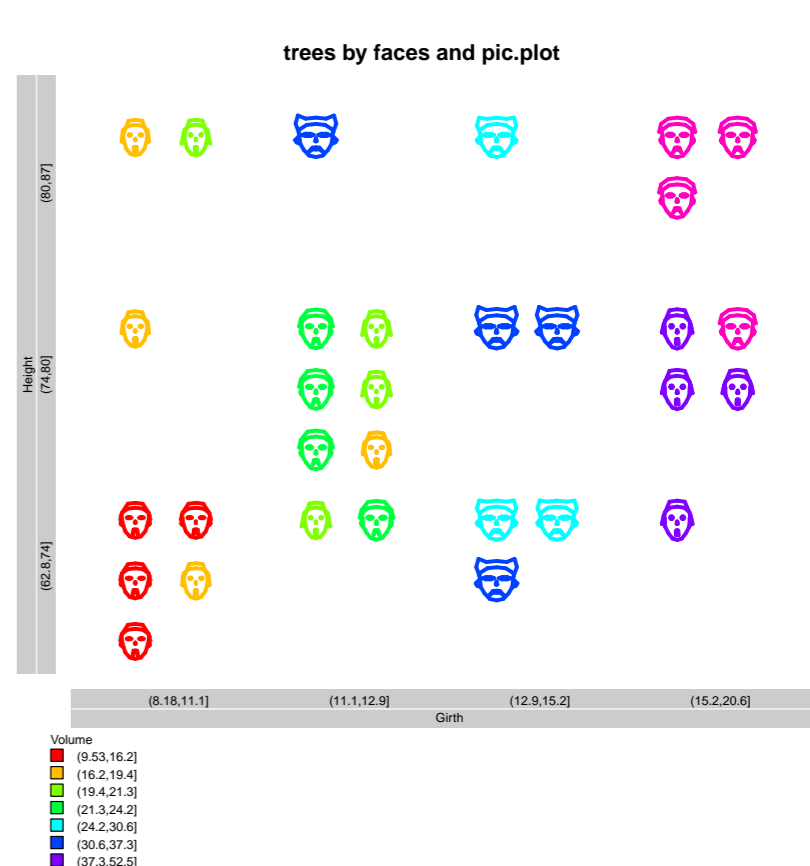
trees: Pictogramme erzeugt durch eine Generierungsfunktion

```
> smiley.gen <- function(data.row = NULL){
+   if(0 < length(data.row)){
+     idx <- data.row$idx
+     h <- min(trees[, "Volume"])
+     smile <- (trees[idx, "Volume"]-h)/
+       (max(trees[, "Volume"])-h)
+     res <- smiley(smiley = smile)
+   } else { res <- smiley() }
+   return(res)
+ }
> pic.plot(trees,
+         grp.xy = Height ~ Girth,
+         grp.pic = Volume,
+         grp.color = Volume,
+         vars.to.factors = c(.25, .3, .05),
+         pics = smiley.gen,
+         pic.space.factor = 0.1,
+         pic.frame = FALSE,
+         panel.frame = FALSE,
+         main = "smileys represent trees")
```



trees: Verwendung von faces() zur Volumenrepräsentation, [6]

```
> library(aplpack, lib.loc = "~/lib")
> faces.of.trees <- faces(trees, plot.faces = FALSE)
> f.list <- generate.fns(faces.of.trees, 1:31)
> pic.plot(trees,
+         grp.pic = 3,
+         grp.col = Volume,
+         vars.to.factors = c(.25, .3, .12),
+         pics = f.list,
+         pic.space.factor = 0.3,
+         pic.frame = FALSE,
+         panel.frame = FALSE,
+         lab.cex = 0.7,
+         lab.parallel = c(TRUE, TRUE, FALSE),
+         main = "trees by faces and pic.plot")
```

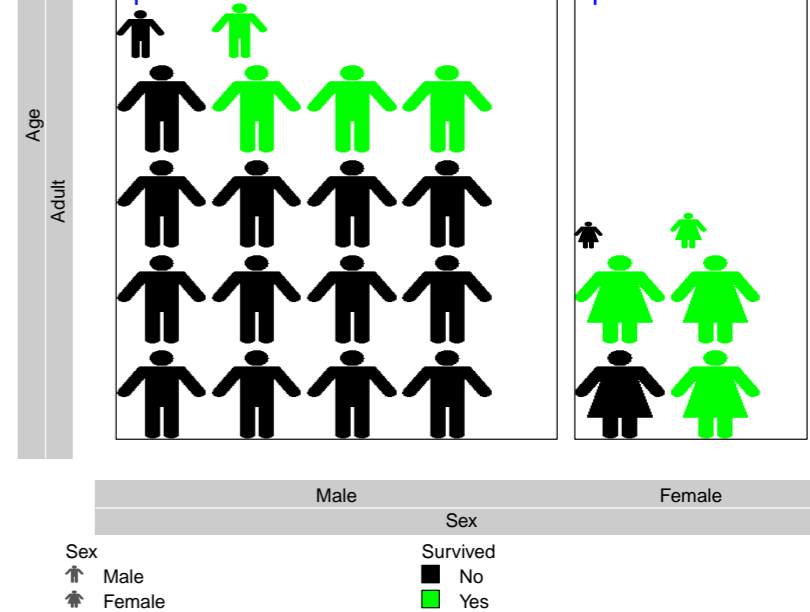


Titanic: Ergänzung von Texten

```
Sex
Age Male Female
Child 64 45
Adult 1667 425
```

```
Sex, Age, Survived of Titanic in 100
```

```
persons 64 2 persons 45 2
persons 1667 18 persons 425 6
```



Fazit

pic.plot() kann verschiedenste Repräsentationen mehrdimensionaler Datensätze erstellen. Die neuen Perspektiven führen vielleicht auch zu neuen Erkenntnissen. Im Gegensatz zu Tabellen können dekorative Pictogramm-Plots neugierig machen und Interesse wecken. Der Anwender muss deshalb einen geeigneten Kompromiss zwischen Information und Dekoration finden.