```
myRegions$Region<-NULL
myLabelling<-c("Oil","Coal","Gas","Renewable Energies","Nuclear\↘
      nenergy")

# Create chart

radial.plot(myRegions[2:3,],start=1,grid.left=T,labels=↘
      myLabelling,rp.type="p",main="",line.col=c(myC1,myC2),poly↘
      .col=c(myC1,myC2),show.grid=T,radial.lim=c(0,55),lwd=8)
legend("bottomleft",c("OECD","Asia"),pch=15,col=c(myC1,myC2),bty↘
      ="n",cex=1.5)

# Titling

mtext(line=3,"Energy mix: OECD and Asia by comparison",cex=2.5,↘
      adj=0,family="Lato Black")
mtext(line=1,"All values in percent",cex=1.5,adj=0,font=3)
mtext(side=1,line=2,"Source: German Federal Agency for Civic ↘
      Education: keyword 'Enegiemix' [energy mix], www.bpb.de [↘
      website in German]",cex=1.05,adj=1,font=3,outer=T)
dev.off()
```

The structure of the script does not essentially differ from the previous one. Instead of one, two columns of the data record are transferred as data. This is also the reason for the specification of two colours. The colours should be transparent.

6.3 Chart Tables

Here, "chart tables" refers to illustration types in which the arrangement of information has table character. Strictly speaking, this pragmatic definition also applies to bar charts, but there are a series of illustration forms that differ significantly from the form of bar charts and therefore justify their own category. We start with suggestions for two variants of so-called Gantt-charts, then follow with examples for a bump chart (Sect. 6.3.3), a heat map (Sect. 6.3.4), a mosaic plot (Sect. 6.3.5) and two examples for tree maps (Sect. 6.3.7 and 6.3.8).

Gantt charts are named after their inventor Henry L. Gantt, who developed this illustration form for visualisation of the individual operational steps within projects. The individual project steps are reflected by row as the span from the planned start to the planned end time of the project section. Additional optional elements include dependencies in the form of connecting lines between different spans as well as horizontal brackets for task groups. A common Gantt chart looks like Fig. 6.1 (built with LaTeX package gantt.sty by Martin Kumm).

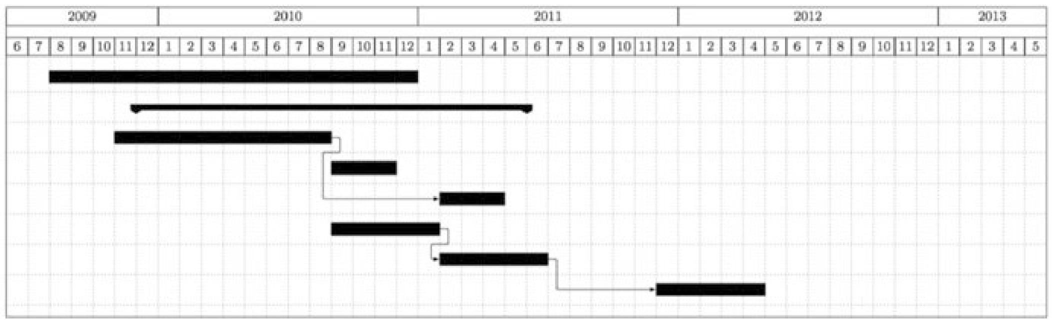
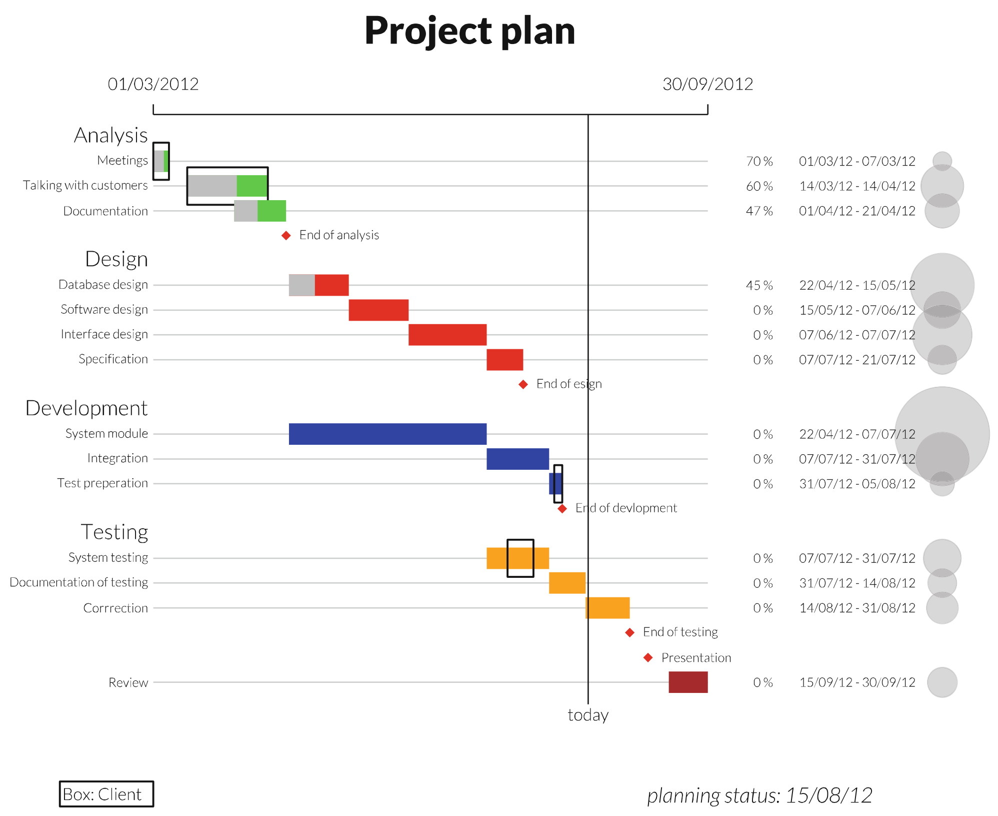
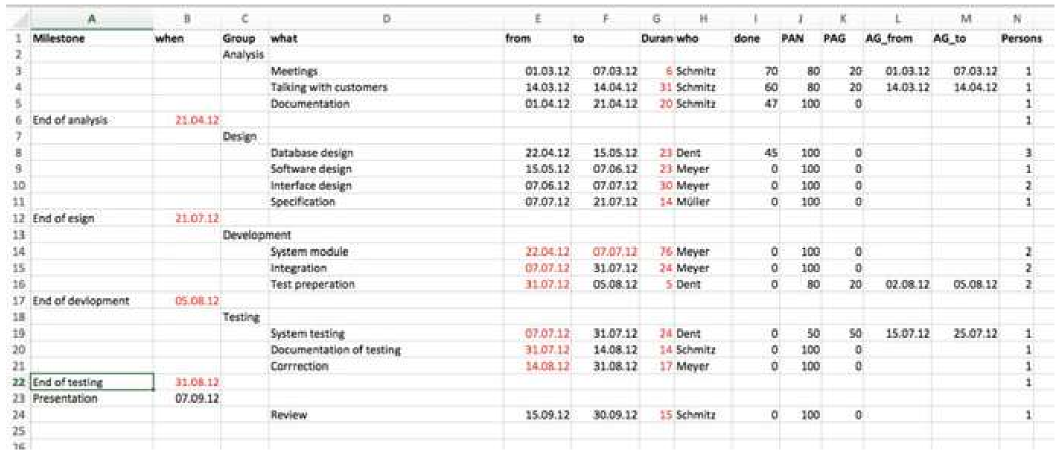


Fig. 6.1 Simplified Gantt chart

For the creation of Gantt charts, R provides e.g. the gantt.chart function within the plotrix package. Here we present a possibility of creating Gantt charts with the functions lines() and points().

6.3.1 *Simplified Gantt Chart*





	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Milestone	when	Group	what	from	to	Duran	who	done	PAN	PAG	AG_from	AG_to	Persons
2			Analysis											
3				Meetings	01.03.12	07.03.12	6	Schmitz	70	80	20	01.03.12	07.03.12	1
4				Talking with customers	14.03.12	14.04.12	31	Schmitz	60	80	20	14.03.12	14.04.12	1
5				Documentation	01.04.12	21.04.12	20	Schmitz	47	100	0			1
6	End of analysis	21.04.12												1
7			Design											
8				Database design	22.04.12	15.05.12	23	Dent	45	100	0			3
9				Software design	15.05.12	07.06.12	23	Meyer	0	100	0			1
10				Interface design	07.06.12	07.07.12	30	Meyer	0	100	0			2
11				Specification	07.07.12	21.07.12	14	Müller	0	100	0			1
12	End of esign	21.07.12												
13			Development											
14				System module	22.04.12	07.07.12	76	Meyer	0	100	0			2
15				Integration	07.07.12	31.07.12	24	Meyer	0	100	0			2
16				Test preperation	31.07.12	05.08.12	5	Dent	0	80	20	02.08.12	05.08.12	2
17	End of deviopment	05.08.12												
18			Testing											
19				System testing	07.07.12	31.07.12	24	Dent	0	50	50	15.07.12	25.07.12	1
20				Documentation of testing	31.07.12	14.08.12	14	Schmitz	0	100	0			1
21				Corrrection	14.08.12	31.08.12	17	Meyer	0	100	0			1
22	End of testing	31.08.12												1
23	Presentation	07.09.12												
24				Review	15.09.12	30.09.12	15	Schmitz	0	100	0			1
25														

Fig. 6.2 Data for project planning

The figure is based on the "classic" Gantt chart, but with a few variations. The basis is an XLSX spreadsheet in which data were collected (Fig. 6.2).

The figure groups the different tasks of the fictional project into "Analysis", "Design", "Development" and "Test" blocks. Each group is assigned its own colour. On top of that, there is a final point "Review", which does not belong to any group. Within each group, three to four tasks are defined and their duration is shown as a bar. The already completed part (the "completed" column in the XLS spreadsheet) is indicated by grey colour, while the part that is still due is shown in the colour of the respective group. Milestones are marked with red dots and are explained in a text. With projects, an important and unfortunately often neglected topic is the involvement of clients. Their involvement should appear in a project planning chart. One possibility is "superimposing" the task bars with a frame marking the duration of client participation. Usually, this will not span the entire duration of a task. Data are taken from the "CL_from" and "CL_until" columns of the XLS spreadsheet. Given that the graphic of the project plan serves as an overview, no detailed list of time periods is necessary; for these, references to the spreadsheet can be added if required. The figure is therefore limited to illustration of the start and end points, and to a line marking the current date "today". On the right side, the already completed parts of the tasks are identified in percent, and their start and end points are given. On the far side, the weight of the task is visualised. To do this, the product of the duration and number of persons per task is shown as a bubble. Due to the overlap for extensive tasks, these are best made transparent. Data are fictional and were entered into an XLS spreadsheet for illustration purposes.

```
pdf_file<-"pdf/tablecharts_gantt_simplified.pdf"
c0<-"black"; c1<-"green"; c2<-"red"; c3<-"blue"; c4<-"orange"; ↘
      c5<-"brown"
myColour_done<-"grey"
myColour<-c(c0,c1,c1,c1,c0,c0,c2,c2,c2,c2,c0,c0,c3,c3,c3,c0,c0,↘
      c4,c4,c4,c0,c0,c5)
source("scripts/inc_gantt_simplified.r")
```

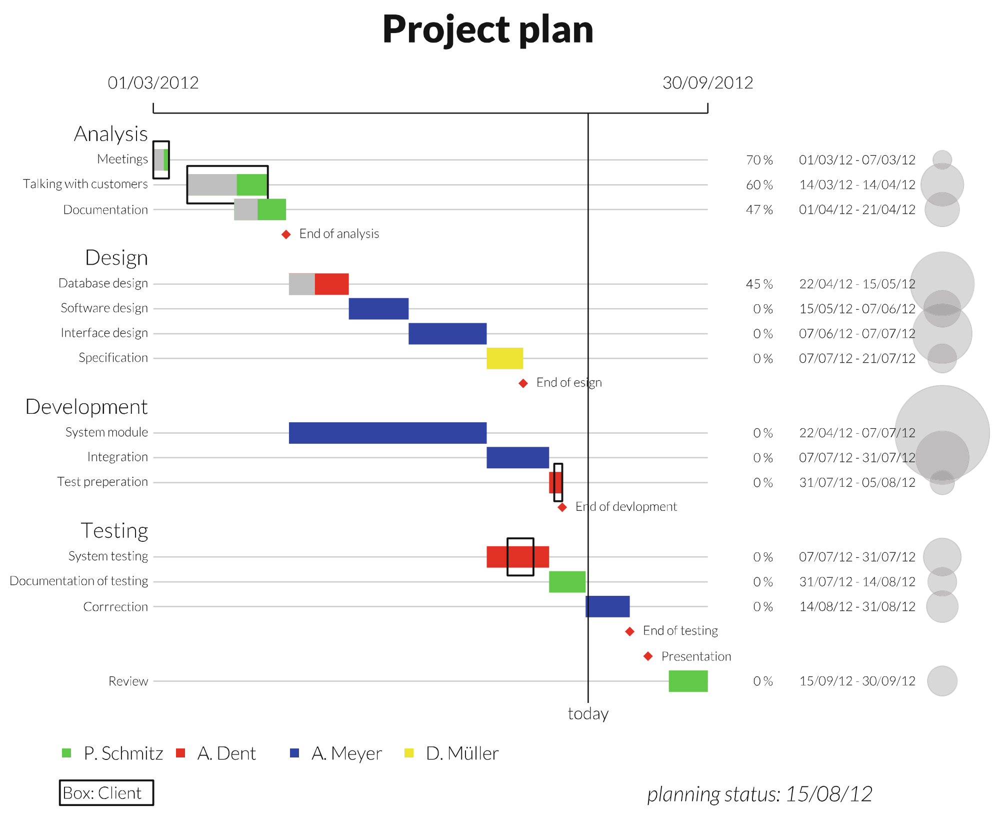
```
dev.off()
```

and

```
# inc_gantt_simplified.r

library(gdata)

cairo_pdf(bg="grey98", pdf_file,width=11.7,height=8.26)

par(lend=1,omi=c(0.25,1,1,0.25),mai=c(1,1.85,0.25,2.75),family="↘
      Lato Light",las=1)
mySchedule<-read.xls("mydata/projectplanning.xlsx", encoding="↘
      latin1")
n<-nrow(mySchedule)
myScheduleData<-subset(mySchedule,nchar(as.character(mySchedule$↘
      from))>0)
myBegin<-min(as.Date(as.matrix(myScheduleData[,c('from','to')]))↘
      )
myEnd<-max(as.Date(as.matrix(myScheduleData[,c('from','to')])))
attach(mySchedule)

plot(from,1:n,type="n",axes=F,xlim=c(myBegin,myEnd),ylim=c(n,1))
for (i in 1:n)
{
if (nchar(as.character(Group[i]))>0)
{
text(myBegin-2,i,Group[i],adj=1,xpd=T,cex=1.25)
}
else if (nchar(as.character(what[i]))>0)
{
x1<-as.Date(mySchedule[i,'from'])
x2<-as.Date(mySchedule[i,'to'])
x3<-x1+((x2-x1)*mySchedule[i,'done']/100)
x<-c(x1,x2)
x_done<-c(x1,x3)
y<-c(i,i)
segments(myBegin, i, myEnd, i, col="grey")
lines(x,y,lwd=20,col=myColour[i])
points(myEnd+90,i,cex=(mySchedule[i,'Persons']*mySchedule[i,'↘
      Durance'])**0.5,pch=19,col=rgb(110,110,110,50,↘
      maxColorValue=255),xpd=T)
if (x3-x1>1) lines(x_done,y,lwd=20,col=myColour_done)
if (mySchedule[i,'PAG'] > 0)
{
x4<-as.Date(mySchedule[i,'AG_from'])
x5<-as.Date(mySchedule[i,'AG_to'])
x_ag<-c(x4,x5)
rect(x4,i-0.75,x5,i+0.75,lwd=2)
}
text(myBegin-2,i,what[i],adj=1,xpd=T,cex=0.75)
text(myEnd+25,i,paste(done[i],"%",sep=" "),adj=1,xpd=T,cex=0.75)
text(myEnd+35,i,paste(format(x1,format="%d/%m/%y"),"-",format(x2↘
      ,format="%d/%m/%y"),sep=" "),adj=0,xpd=T,cex=0.75)
}
else # Milestone
{
x3<-as.Date(mySchedule[i,'when'])
myHalf<-(myEnd-myBegin)/2
if (x3-x1<myHalf)
{
points(as.Date(mySchedule[i,'when']),i,pch=18,cex=1.25,col="red"↘
      )
text(as.Date(mySchedule[i,'when'])+5,i,Milestone[i],adj=0,xpd=T,↘
      cex=0.75)
} else
{
points(as.Date(mySchedule[i,'when']),i,pch=18,cex=1.25,col="red"↘
      )
text(as.Date(mySchedule[i,'when'])-5,i,Milestone[i],adj=1,xpd=T,↘
      cex=0.75)
}
}
}
axis(3,at=c(myBegin,myEnd),labels=c(format(myBegin,format="%d/%m↘
      /%Y"),format(myEnd,format="%d/%m/%Y")))
myToday<-as.Date("15.08.2012", "%d.%m.%Y")
abline(v=myToday)

mtext("today",1,line=0,at=myToday)

# Titling

mtext("Project plan",3,line=2,adj=0,cex=2.25,family="Lato Black"↘
      ,outer=T)
mtext(paste("planning status: ",format(myToday,format="%d/%m/%y"↘
      ),sep=""),1,line=4,at=myEnd+20,cex=1.25,font=3)
rect(myBegin-36, n+5, myBegin, n+4, xpd=T,lwd=2)
text(myBegin-35, n+4.5, "Box: Client",xpd=T, adj=0)
```

The script first defines the colours for the individual groups (c0 to c5). Subsequently, the colouring of the individual bars are defined as elements of the colour vector. For ease of use, this was done manually. However, data could also be imported from a data table. First, rows containing date information are extracted from the XLSX spreadsheet, and the first and last occurring dates are identified. In the next step, the chart is scaled (not drawn) using plot() and the group name is written to the left of the start using text(). If the column “what” contains an entry, then the time periods and the completed part are determined, and time periods are plotted. On the right border, a point (a “bubble”) is plotted to indicate the size of the project part, i.e. the duration multiplied by the number of people involved. Since we are using the radius as a variable, we have to use the square root to ensure that area enlargement is proportionally correct. In the next step, the completed parts are plotted over the time periods. If the project client (PCL) has to be involved, a rectangle is drawn around the bars, with length and circumference taken from the data. Then follow the labelling of the rows at the beginning (with the “what”) and at the end with the percentage of the completed task part and the time period of the task. If there is no entry in the from and to columns, but in the when column, then it is a “milestone”. In those cases, they are plotted as points and labelled. Finally, an axis is plotted at the upper margin.

6.3.2 Simplified Gantt Chart: Colours by People

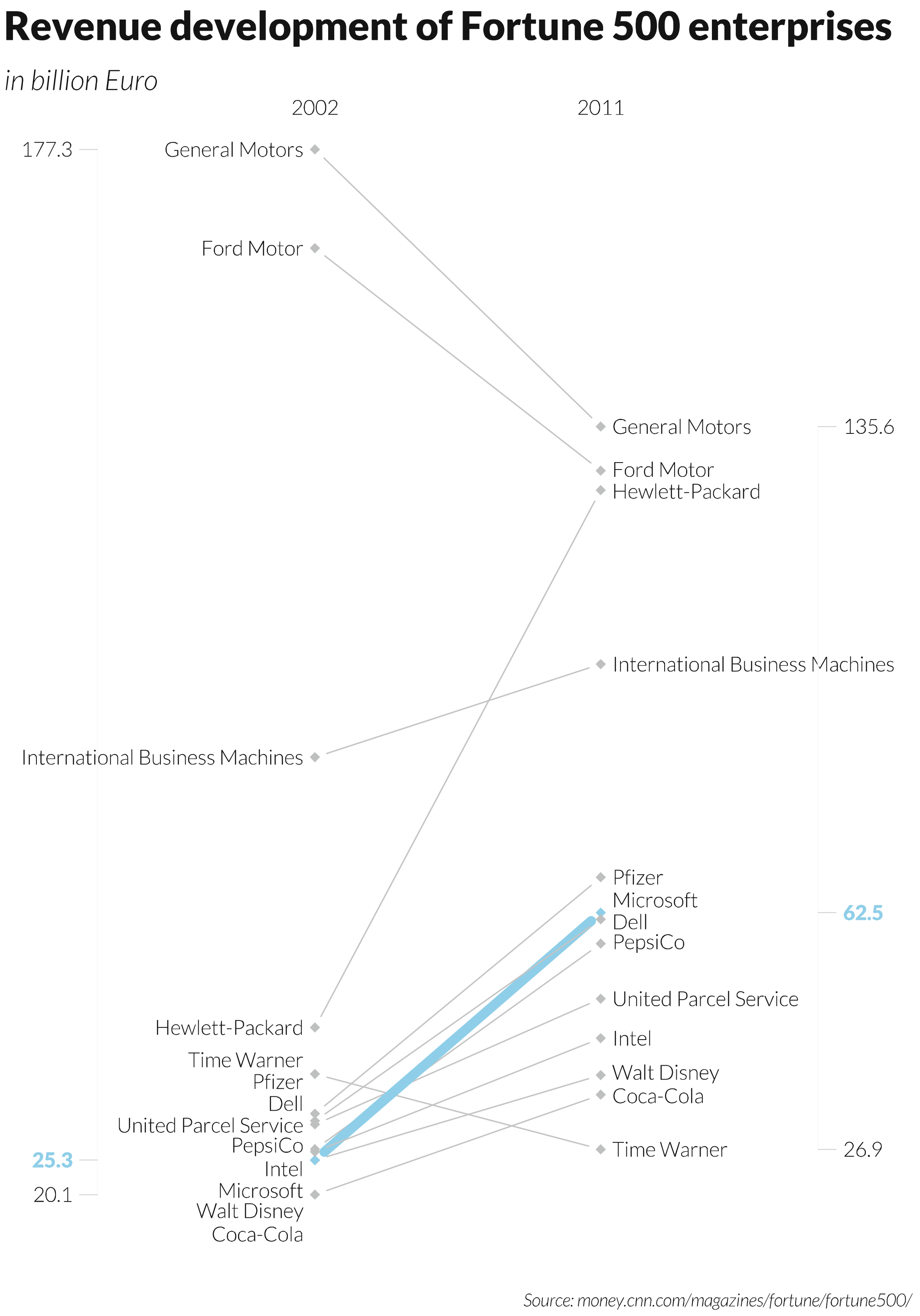


The figure essentially matches the previous one. Only the colouring is different here: it is based on the people involved in the project. Data are fictional and were entered into an XLS spreadsheet for illustration purposes.

```
pdf_file<-"pdf/tablecharts_gantt_simplified_who.pdf"
f0<-"black"; f1<-"green"; f2<-"red"; f3<-"blue"; f4<-"yellow"
farbe_erl<-"grey"
farbe<-c(f0,f1,f1,f1,f0,f0,f2,f3,f3,f4,f0,f0,f3,f3,f2,f0,f0,f2,↘
      f1,f3,f0,f0,f1)
source("scripts/inc_gantt_simplified.r")
legend(anfang-40,n+2,c("P. Schmitz","A. Dent","A. Meyer","D. Mü↘
      ller"),pch=15,col=c(f1,f2,f3,f4),bty="n",cex=1.1,horiz=T,↘
      xpd=T)
dev.off()
```

In the script, we only have to slightly redefine the colours for this case, and add a legend with the names of the people involved.

6.3.3 Bump Chart



The figure uses the bumpchart() function from the plotrix package by Jim Lemon. It would also be possible to directly use the matplot() function and add labels right-justified on the left side and left-justified on the right side using text(); the y-coordinates for the function would then simply be the revenue numbers. Depending on the value combination, however, labels could then overlap. The advantage of Jim Lemon's solution is that, in such cases, labels are automatically moved vertically by the appropriate amount.

About the figure: a bump chart usually compares two or more points in time for multiple numerical parameters; characteristic labels for these parameters are written at the ends of the connecting lines. There are two variants: one uses only the ranks and plots these on an ordinal scale; the other plots the actual values on an interval scale. In this example, revenue development of the Fortune 500 enterprises in the USA between 2002 and 2011 is compared. Here, the use of the actual values is a lot more informative than their ranks. It should be noted though, that the labels of individual points might overlap, depending on the data. In such cases, the labels should be attached with minimal, but identical line spacing, but in such a way that at least rank is maintained. On the left side, this is the case for all enterprises whose revenue is less than Hewlett-Packard's. As an example, Microsoft's revenue has been highlighted here. The data were first taken from a CNN website by year and copied into an XLS spreadsheet; a new sheet was used for each year. The data were then reorganised using the sqldf package and then saved as a binary R file fortune_revenue.RData.

```
library(sqldf)
f2011<-read.xls("data/fortune100.xlsx", sheet = 1)
f2010<-read.xls("data/fortune100.xlsx", sheet = 2)
f2009<-read.xls("data/fortune100.xlsx", sheet = 3)
f2008<-read.xls("data/fortune100.xlsx", sheet = 4)
f2007<-read.xls("data/fortune100.xlsx", sheet = 5)
f2006<-read.xls("data/fortune100.xlsx", sheet = 6)
f2005<-read.xls("data/fortune100.xlsx", sheet = 7)
f2004<-read.xls("data/fortune100.xlsx", sheet = 8)
f2003<-read.xls("data/fortune100.xlsx", sheet = 9)
f2002<-read.xls("data/fortune100.xlsx", sheet = 10)

total<-sqldf("select enterprise from f2011
union select enterprise from f2010
union select enterprise from f2009
union select enterprise from f2008
union select enterprise from f2007
union select enterprise from f2006
union select enterprise from f2005
union select enterprise from f2004
union select enterprise from f2003
union select enterprise from f2002")

x<-sqldf("select
total.enterprise,
f2002.revenue r2002
f2002.revenue r2003
f2002.revenue r2004
f2002.revenue r2005
f2002.revenue r2006
f2002.revenue r2007
f2002.revenue r2008
f2002.revenue r2009
f2002.revenue r2010
f2002.revenue r2011
from total
left join f2002 on total.enterprise=f2002.enterprise
left join f2003 on total.enterprise=f2003.enterprise
left join f2004 on total.enterprise=f2004.enterprise
left join f2005 on total.enterprise=f2005.enterprise
left join f2006 on total.enterprise=f2006.enterprise
left join f2007 on total.enterprise=f2007.enterprise
left join f2008 on total.enterprise=f2008.enterprise
left join f2009 on total.enterprise=f2009.enterprise
left join f2010 on total.enterprise=f2010.enterprise
left join f2011 on total.enterprise=f2011.enterprise
“)

row.names(x)<-x$enterprise
x$enterprise<-NULL
y<-t(x)
save(y, file=”fortune_revenue.RData.”)
```

The saved data can now be used in the script:

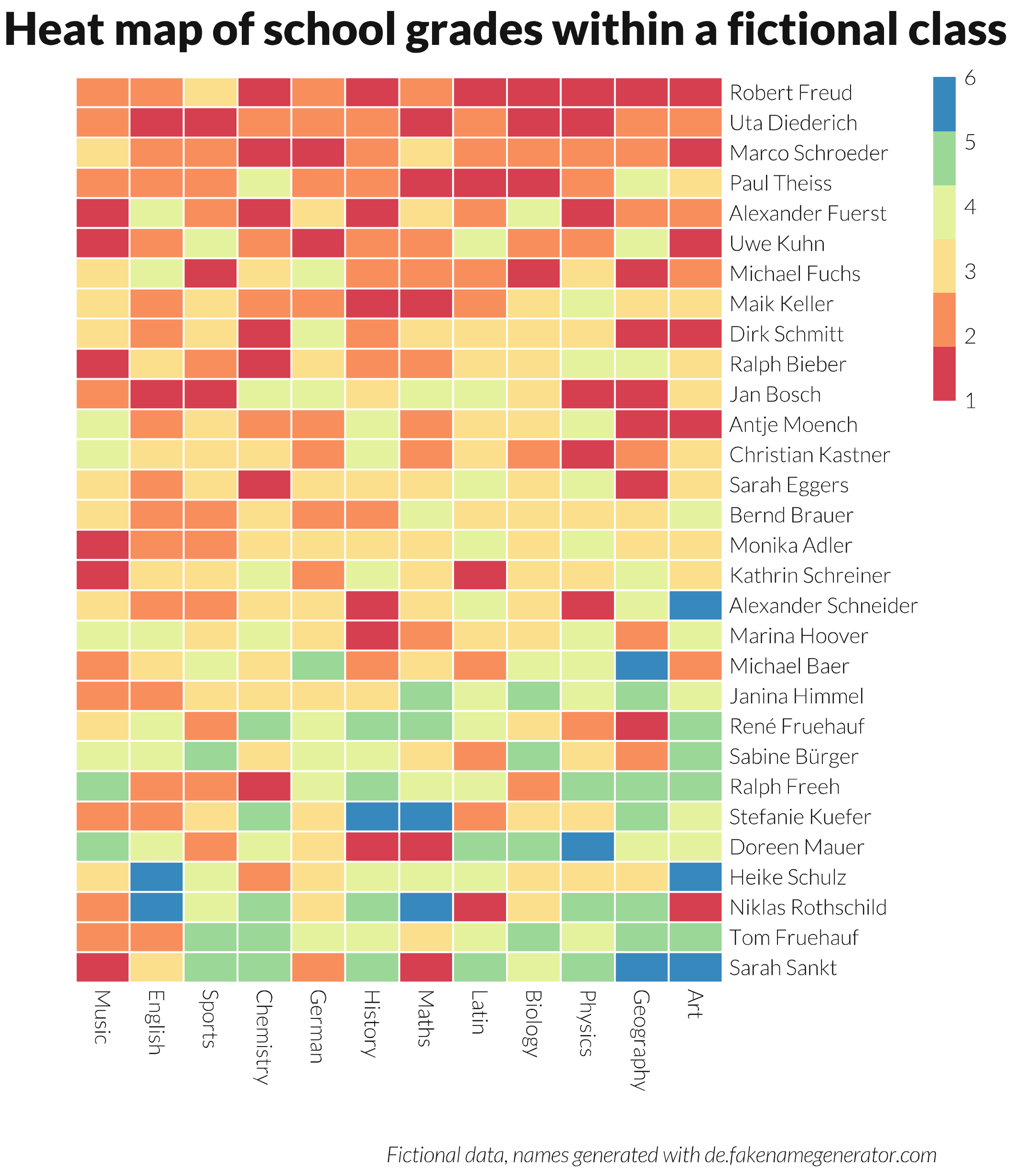
```
pdf_file<-"pdf/tablecharts_bumpchart.pdf"
cairo_pdf(bg="grey98", pdf_file,width=9,height=12)

par(omi=c(0.5,0.5,0.9,0.5),mai=c(0,0.75,0.25,0.75),xpd=T,family=↘
      "Lato Light",las=1)
library(plotrix)
library(gdata)


# Import data and prepare chart

z1<-read.xls("myData/bumpdata.xlsx", encoding="latin1")
rownames(z1)<-z1$name
z1$name<-NULL
myColours<-rep("grey",nrow(z1)); myLineWidth<-rep(1,nrow(z1))
myColours[5]<-"skyblue"; myLineWidth[5]<-8
par(cex=1.1)


# Create chart

bumpchart(z1,rank=F,pch=18,top.labels=c("2002","2011"),col=↘
      myColours,lwd=myLineWidth,mar=c(2,12,1,12),cex=1.1)

# Titling
```

```
mtext("Revenue development of Fortune 500 enterprises",3,line↘
      =1.5,adj=0,family="Lato Black",outer=T,cex=2.1)
mtext("Source: money.cnn.com/magazines/fortune/fortune500/",1,↘
      line=0,adj=1,cex=0.95,font=3,outer=T)

# Other elements

axis(2,col=par("bg"),col.ticks="grey81",lwd.ticks=0.5,tck=-↘
      0.025, at=c(min(z1$r2002), max(z1$r2002)),c(round(min(z1$↘
      r2002)/1000,digits=1), round(max(z1$r2002)/1000, digits=1)↘
      ))
axis(4,col=par("bg"),col.ticks="grey81",lwd.ticks=0.5,tck=-↘
      0.025, at=c(min(z1$r2011), max(z1$r2011)),c(round(min(z1$↘
      r2011)/1000,digits=1), round(max(z1$r2011)/1000, digits=1)↘
      ))

mtext("in billion Euro",3,font=3,adj=0,cex=1.5,line=-0.5,outer=T↘
      )

par(family="Lato Black")
axis(2,col=par("bg"),col.ticks="grey81",col.axis="skyblue",lwd.↘
      ticks=0.5,tck=-0.025,at=z1[5,1],round(z1[5,1]/1000, digits↘
      =1))
axis(4,col=par("bg"),col.ticks="grey81",col.axis="skyblue",lwd.↘
      ticks=0.5,tck=-0.025,at=z1[5,2],round(z1[5,2]/1000, digits↘
      =1))

dev.off()
```

In the script, we need the plotrix package for the bump chart. Data are read from a XLSX spreadsheet, and row names are created from the name column. Then, the name column is deleted so that the data frame only comprises the data to be plotted. A vector with identical values is defined for each colour and line width, and individually modified for the 5th data frame. Margins are set with the mar parameter within the bumpchart() function. At the end, we add two axis labels on both the left and right, stating the range of values. The first call sets the minima and maxima, the second the value for Microsoft.

6.3.4 Heat Map

About the figure: A heat map is a two-dimensional matrix, in which the cells are coloured depending on their value. It may be a table with individual data or aggregated values. There is no rule stating how the rows or columns should be arranged for illustration. If the order of both rows and columns is random or does not contain required information, then a cluster method can be used to group “similar” rows and/or columns. Additionally, dendrograms on the sides can show the grouping at different levels. Another variant is sorting the data. If comparable statistics can be done for the columns, then sorting by these is possible as well. An example for this is school grades. The figure shows a heat map of (fictional) school grades of a (fictional) class. Here, the best pupils are arranged at the top and the subjects with the best grades on the left. This gives a good impression of the grade distribution within a class. Of course, a comparison with one or several other classes would be an obvious option.

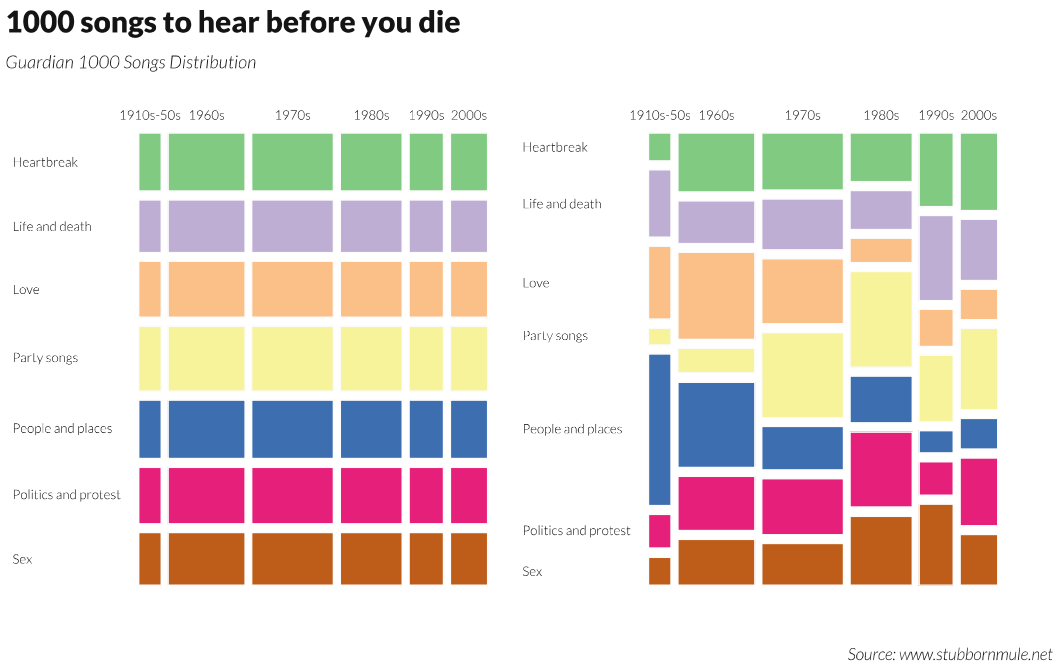


Data were generated with the help of the http://de.fakenamegenerator.com site, and entered into an XLS spreadsheet.

```
pdf_file<-"pdf/tablecharts_heatmap.pdf"
cairo_pdf(bg="grey98", pdf_file,width=7,height=8)

library(RColorBrewer)
library(pheatmap)
par(mai=c(0.25,0.25,0.25,1.75),omi=c(0.25,0.25,0.75,0.85),family↘
      ="Lato Light",las=1)

# Import data and prepare chart

myGrades<-read.xls("myData/grades.xlsx", encoding="latin1")
x<-as.matrix(myGrades[,2:13])
rownames(x)<-myGrades$names
x<-x[order(rowSums(x)), ]
x<-x[,order(colSums(x))]

# Create chart

plot.new()
pheatmap(x,col=brewer.pal(6,"Spectral"),cluster_rows=F,cluster_↘
      cols=F,cellwidth=25,cellheight=14,border_color="white",↘
      fontfamily="Lato Light")

# Titling

mtext("Heat map of school grades within a fictional class",3,↘
      line=1,adj=0.2,cex=1.75,family="Lato Black",outer=T)
mtext("Fictional data, names generated with de.fakenamegenerator↘
      .com",1,line=-1,adj=1,cex=0.85,font=3,outer=T)
dev.off()
```

About the script: The standard package stats provides a heatmap() function that can be used to create the corresponding charts. However, there is only limited potential to modify their appearance. The gplots package provides an extended heat map function called heatmap.2(); however, it uses the layout() function and can therefore not be used in a panel illustration defined with mfcol or mfrow. For this reason, we use the pheatmap() function from the package of the same name by Raivo Kolde for our purposes. Here, cell height and width can be defined individually. We forgo the use of dendrograms on the sides (cluster_rows=F, cluster_cols=F). Data are read from the grades.xlsx file, then row names are created from the "names" column and the column is deleted. Data are then sorted first by row and then by column, so that the best pupils are shown at the top and the best grades on the left. Then follows the call of the pheatmap() function. Before that, plot.new() has to be called though, since pheatmap() is not a high-level function from R's traditional base graphic environment, but is based on grid.

6.3.5 *Mosaic Plot (Panel)*

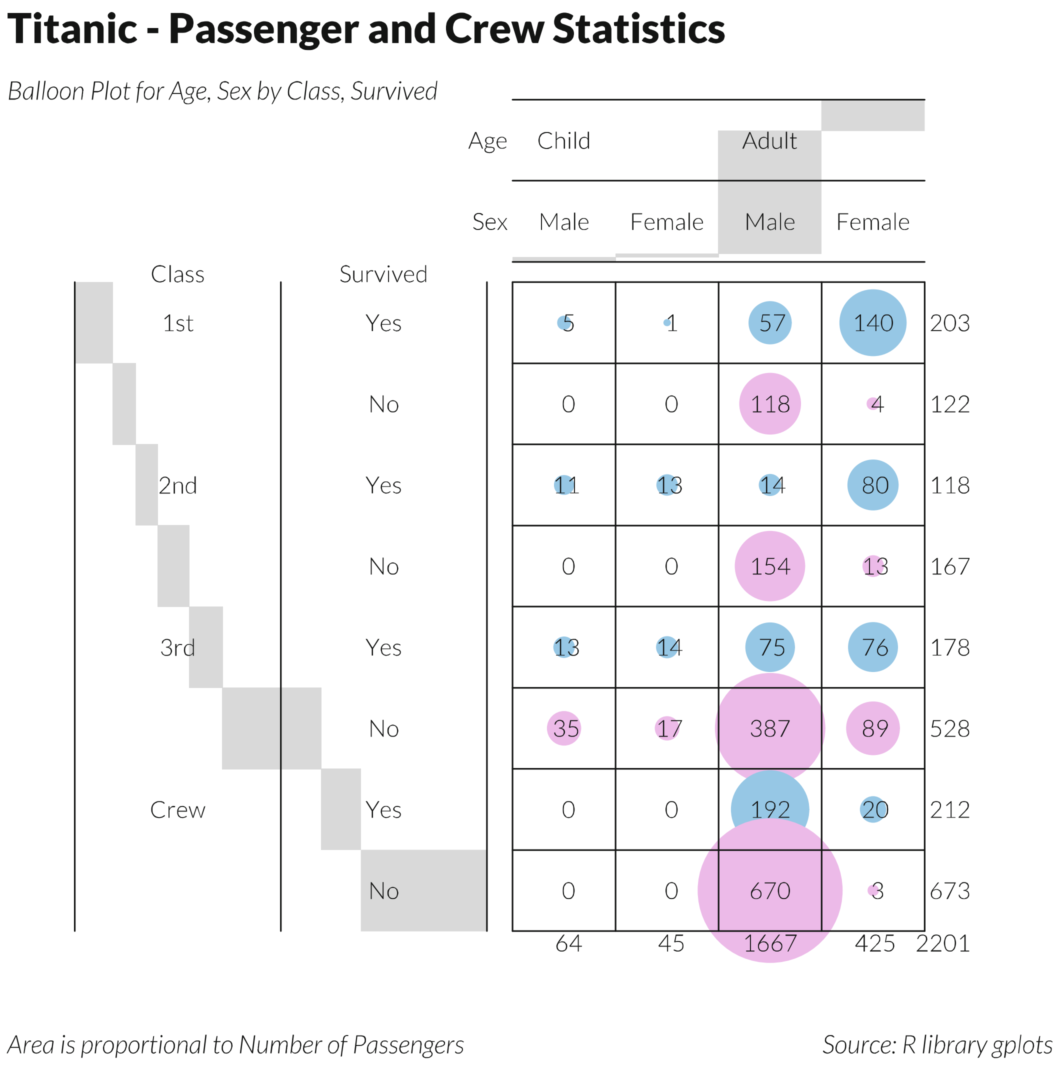


About the figure: in a mosaic plot, cells of a contingency table are shown in the form of rectangles, with the size of the rectangle corresponding to the frequency of the cell. This is generally also possible for multi-dimensional data and offers statistics specialists great help for gaining insights. Without prior knowledge, this illustration form requires some getting used to, but is useful in individual cases. It should be noted that the area varies in two dimensions and not independently, as is the case with a "bubble plot", a scatter plot with different size dots. Here, increasing one length will cause a shift of the subsequent elements. We are using an example from Sean Carmody that is also used in Wikipedia. The principle of constructing a two-dimensional mosaic plot is this: first, the rectangle of the entire tables is split in vertical slices, so that column width corresponds to the relative frequencies of the margin distribution of the column variable. In our case, this is the distribution of the number of songs within the individual epochs. In a second step, the area for each epoch is cut horizontally in such a way that the heights correspond to the relative frequencies of the row variable (in our case: the topics) in the respective epochs. In a two-dimensional case, the result is therefore a stacked 100% bar chart, in which column width corresponds to the relative frequencies of a second categorical variable. In my opinion, such a representation is especially useful if an independency table is set next to it for comparison, i.e. one that assumes identical frequencies for the row variable in those categories. The data are taken from a list that was compiled by The Guardian. A CSV file with the data is available on http://www.stubbornmule.net.

```
pdf_file<-"pdf/tablecharts_mosaicplot_1x2.pdf"
cairo_pdf(bg="grey98", pdf_file,width=10,height=6)

par(mai=c(0.25,0.0,0.0,0.25),omi=c(0.5,0.5,1.25,0.5),las=1,mfcol↘
      =c(1,2),family="Lato Light",las=1)
library(RColorBrewer)

# Import data and prepare chart

data<-read.csv("myData/1000.csv",as.is=c(F,T,F,T,T),sep=";")
data$DEKADE<-floor(data$YEAR/10) * 10
data$KDEKADE<-paste(data$DEKADE,"s",sep="")
data$KDEKADE[data$DEKADE < 1960]<-"1910s-50s"
tab<-table(data$KDEKADE,data$THEME)
utab<-chisq.test(tab)

# Create chart

mosaicplot(utab$expected,col=brewer.pal(7,"Accent"),main="",↘
      border=par("bg"))
mosaicplot(tab,col=brewer.pal(7,"Accent"),main="",border=par("bg↘
      "))

# Titling

mtext("1000 songs to hear before you die",3,line=3,adj=0,cex↘
      =1.5,family="Lato Black",outer=T)
mtext("Guardian 1000 Songs Distribution",3,line=1.5,adj=0,cex↘
      =0.9,font=3,outer=T)
mtext("Source: www.stubbornmule.net",1,line=1,adj=1.0,cex=0.85,↘
      font=3,outer=T)
dev.off()
```

In the script, we add an extra column DECADE to the period name in the data for illustration purposes, and an “s” is appended; then the dates before 1960 are summarised. For the mosaic plot, a table is generated from the data. We use the chis.test function from the stats package for the illustration of the “independency table”. The mosaicplot() function is part of the graphics package, which means that we do not have to load an additional package. The left mosaic plot shows the data under the assumption of independence, the right one the distribution of the actual data. In both cases, we use a qualitative Brewer palette and a margin-free background. No other settings have to be made. Extended features are provided by the vcd package, with which numerous variants of mosaic plots are possible. These graphics are based on grid though, not on the traditional graphic.

6.3.6 *Balloon Plot*



Class	Survived	Child Male	Child Female	Adult Male	Adult Female	
1st	Yes	5	1	57	140	203
	No	0	0	118	4	122
2nd	Yes	11	13	14	80	118
	No	0	0	154	13	167
3rd	Yes	13	14	75	76	178
	No	35	17	387	89	528
Crew	Yes	0	0	192	20	212
	No	0	0	670	3	673
		64	45	1667	425	2201

About the figure: The name "balloon plot" is slightly misleading, as it is frequently used to describe a scatter plot with variable dot size (a "bubble chart"). Here, it refers to a specific, graphically supplemented variant of contingency tables that has to date occasionally been used in biostatistics or mineralogy. The illustration form was implemented into R with the plots package by Gregory R. Warnes. The data are an example data frame frequently employed in R, the Titanic passengers, classified by sex, age (children and adults), their on-board status (1st, 2nd, 3rd class or crew) and their survival of the sinking (yes, no). The figure shows the data in the form of a bivariate frequency table, where the rows contain the status on the first level and the survival on the second level, the columns the age on the first level and the sex on the second level. In addition to the numbers showing cell and margin frequencies, cell frequencies are highlighted with a dot whose size is proportional to the number. The colour of the dots differentiates the survivors from the drowned.

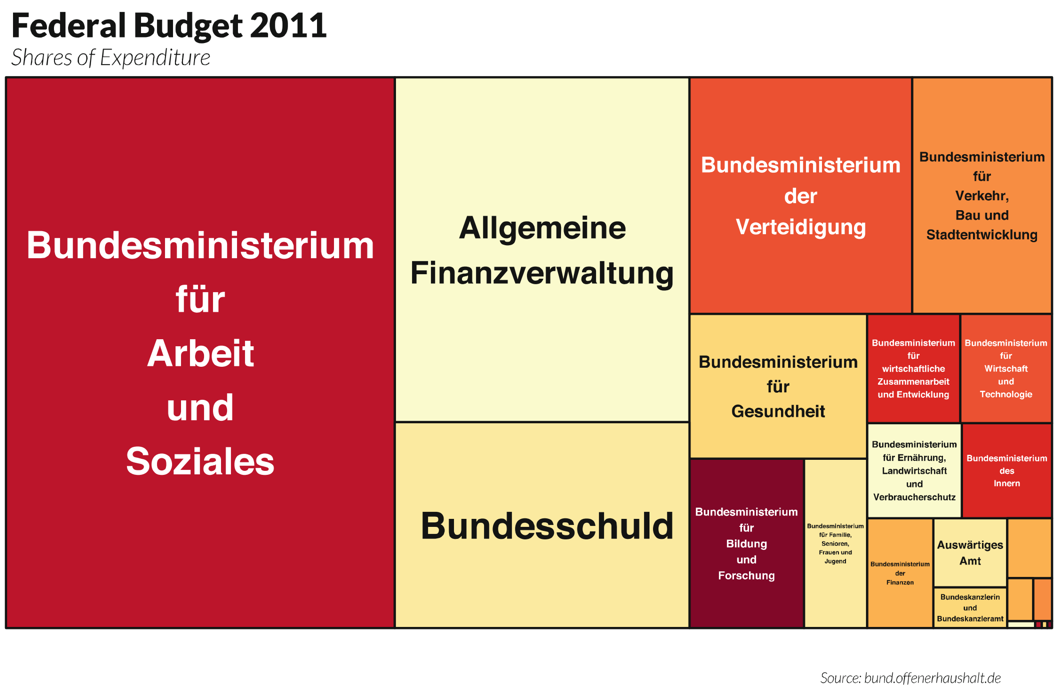
Margin frequencies are reflected in the header rows and columns as bar or column portions of 100% in the respective row or column. This type of illustration gives a better impression of the distribution than a “naked” contingency table would. Data are taken from the “Titanic” data frame supplied with R.

```
pdf_file<-"pdf/tablecharts_ballonplot.pdf"
cairo_pdf(bg="grey98",pdf_file,width=9,height=9)

par(omi=c(0.75,0.25,0.5,0.25),mai=c(0.25,0.55,0.25,0),family="↘
      Lato Light",cex=1.15)
library(gplots)

# Import data and prepare chart

data(Titanic)
myData<-as.data.frame(Titanic) # convert to 1 entry per row ↘
      format
attach(myData)
myColours<-Titanic
myColours[,,,"Yes"]<-"LightSkyBlue"
myColours[,,,"No"]<-"plum1"
myColours<-as.character(as.data.frame(myColours)$Freq)

# Create chart

balloonplot(x=list(Age,Sex),main="",
            y=list(Class=Class,
            Survived=gdata::reorder.factor(Survived,new.order=c↘
                  (2,1))),
            z=Freq,dotsize=18,
            zlab="Number of Passengers",
            sort=T,
            dotcol=myColours,
            show.zeros=T,
            show.margins=T)

# Titling

mtext("Titanic – Passenger and Crew Statistics",3,line=0,adj=0,↘
      cex=2,family="Lato Black",outer=T)
mtext("Balloon Plot for Age, Sex by Class, Survived",3,line=-2,↘
      adj=0,cex=1.25,font=3,outer=T)
mtext("Source: R library gplots",1,line=1,adj=1.0,cex=1.25,font↘
      =3,outer=T)
mtext("Area is proportional to Number of Passengers",1,line=1,↘
      adj=0,cex=1.25,font=3,outer=T)
dev.off()
```

The script is the example from the documentation of the ballonplot() function, in which only the colour selection and dot size have been adjusted. Data were loaded from the data frame “Titanic” that is supplied with R and converted into a data frame (the original data are an object of table type). The original table colours are used for the creation of the colours for the balloon plot. Data from the data frame are transferred to the function in the form of a list. We choose 18 as point size. Headings are created as before.

6.3.7 Tree Map

Tree maps are useful for the presentation of proportions. The New York Times used tree maps for the presentation of Obama's 2012 budget. A neat example with German data can be found on http://bund.offenerhaushalt.de. Here, a breakdown of the German federal budget can be found in tabular form and as a tree map, for both the entire budget and the individual categories. Data can be exported in JSON or RDF format.

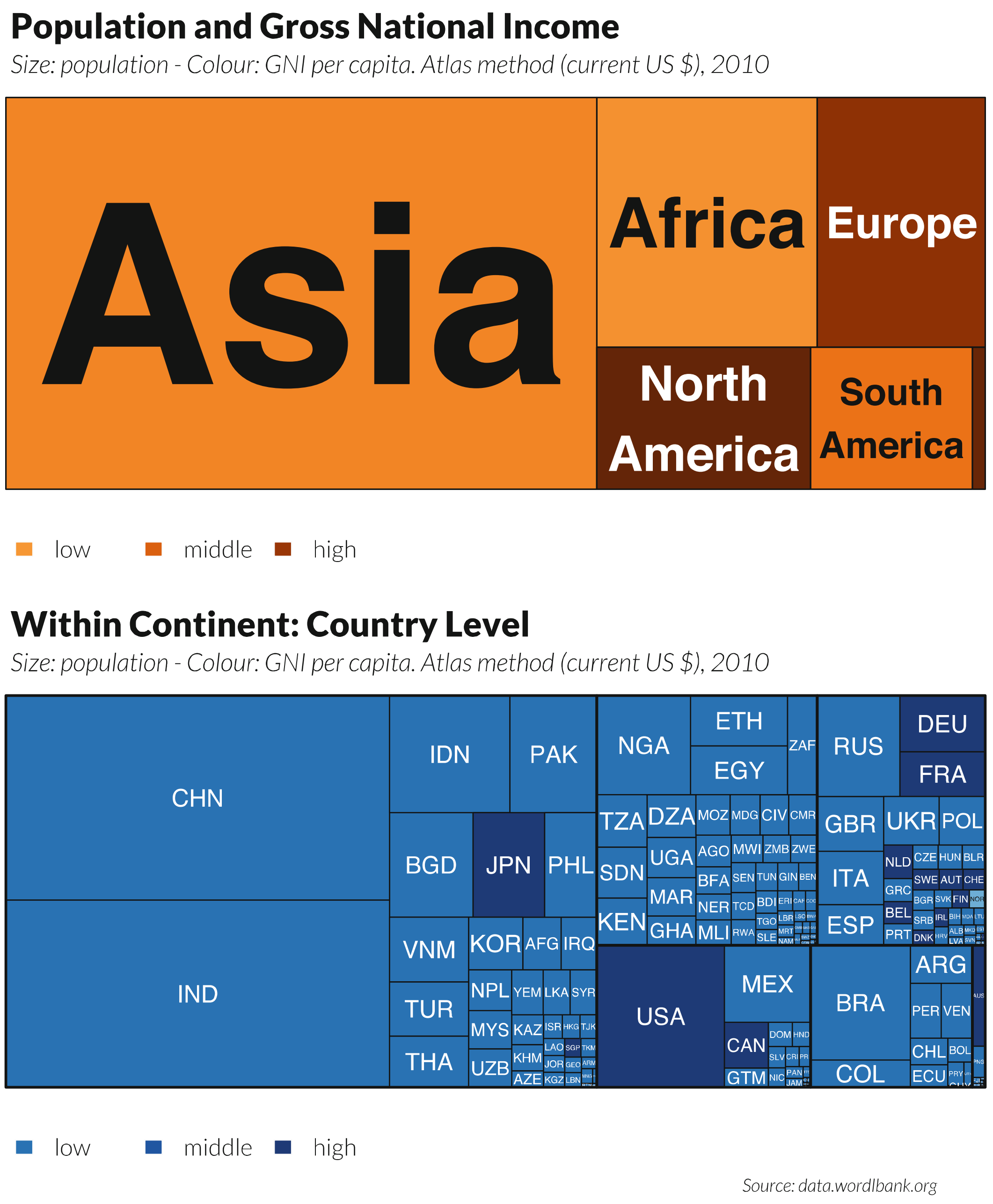


About the figure: A tree map shows the attribute of a cardinally scaled variable as nested rectangles. The size and order of the rectangles are calculated so that, with preset outer dimensions, the large rectangle is completely filled and the areas of the individual rectangles correspond to the size of the variables. There are different algorithms for calculation of the rectangles that each optimise different aspects of the subdivision. Mostly, a procedure is used that produces the maximum number of rectangles with aspect ratio approximating 1. Since the outer margins are always set, even hierarchies can be shown with tree maps: a created rectangle can again be considered the outer margin for a new subdivision of the attribute. The first example shows the shares of individual expenditures of the federal budget in 2011 as a tree map, which clearly depicts the unequal distribution of expenditures. The different colours are only used to separate the elements from each other. Labelling to the last element is not possible. Data are available on http://bund.offenerhaushalt.de, and were copied and pasted into an XLS spreadsheet.

```
pdf_file<-"pdf/tablecharts_treemap.pdf"
cairo_pdf(bg="grey98", pdf_file,width=11.69,height=7.5)

# par(omi=c(0.65,0.25,1.25,0.75),mai=c(0.3,2,0.35,0),family="↘
      Lato Light",las=1)
par(omi=c(0.55,0.25,1.15,0.75),family="Lato Light",las=1)
library(treemap)
library(gdata)


# Import data

federalbudget<-read.xls("mydata/federalbudget.xlsx",sheet=1, ↘
      encoding="latin1")

#  Create chart

plot.new()
treemap(federalbudget,title="",index="Title",type="index",vSize=↘
      "Expenditures",palette="YlOrRd",aspRatio=1.9,inflate.↘
      labels=T)

# Titling

mtext("Federal Budget 2011",3,line=3.8,adj=0,cex=2.2,family="↘
      Lato Black",outer=T)
mtext("Shares of Expenditure",3,line=2.3,adj=0,cex=1.5,outer=T,↘
      font=3)
mtext("Source: bund.offenerhaushalt.de",1,line=1,adj=1.0,cex↘
      =0.95,outer=T,font=3)
dev.off()
```

In the script, we use the treemap() function from the package of the same name by Martijn Tennekes, which essentially provides this exact function. Just like pheatmap(), the treemap() function is based on grid. If used within the frames of a traditional graphic approach, we first have to use plot() to define a figure, in which treemap() can then be called and finally a separate legend can be drawn. First, the variable for which a tree map is to be created has to be transferred to the function using index. The specification of multiple variables that are then hierarchically nested (see Sect. 6.3.8) is also possible. The colour of the tree map is set using type=index. Here, index means that the colours are based on the index variable. In our case, the colour does not carry information; the hues from the Brewer palette only serve to make the individual blocks as distinguishable as possible. The relative size of the rectangles is defined by the variable set using vSize. The command inflate.labels=T enlarges the labels of the rectangles so that they reach both sides of the rectangles. The aspRatio parameter is explicitly stated, but the dimensions of the figure also affect the aspect ratio.

6.3.8 *Tree Maps for Two Levels (Panel)*



About the figure: If two hierarchical levels are depicted, then the presentation of the tree map can become confusing. In such cases, it makes sense to initially plot the first level in a panel, and the second level in a second chart arranged like the first level. This is shown here using an example of data on population size and gross national income. Data for individual countries are available, but are first shown at the level of continents. The second chart shows the distribution of population size in the individual countries, but not in the order of countries, but first ordered by continent and within those nested by population size. As an additional variable, the gross national income (GNI) was colour-coded in three classes. Data are provided online by the World Bank. Data were filtered for the tree map examples used in this book, connected to the continent data and saved as a binary R file hnp.RData. The file is available for download on the book's website.

```
pdf_file<-"pdf/tablecharts_treemap_2a_inc.pdf"
cairo_pdf(bg="grey98", pdf_file,width=11.69,height=7.5)

par(omi=c(0.65,0.25,1.25,0.75),mai=c(0.3,2,0.35,0),family="Lato ↘
      Light",las=1)
library(treemap)
library(RColorBrewer)

# Read data and prepare chart

load("myData/hnp.RData")
myData<-subset(daten,daten$gni>0)
attach(myData)
popgni<-pop*gni
myData$popgni<-popgni
myContinents<-aggregate(cbind(pop,popgni) ~ kontinent,data=↘
      myData,sum)
kgni<-myContinents$popgni/myContinents$pop
myContinents$kgni<-kgni
kkgni<-cut(kgni,c(0,5000,10000,100000))
levels(kkgni)<-c("low","middle","high")
myContinents$kkgni<-kkgni
myContinents$nkkgni<-as.numeric(kkgni)

# Create chart and other elements

plot(1:1,type="n",axes=F)
treemap(myContinents,title="",index="kontinent", vSize="pop",↘
      vColor="kgni",type="value",palette="YlOrBr",aspRatio=2.5,↘
      position.legend="none",inflate.labels=T)
legend(0.35,0.6,levels(kkgni)[1:3],cex=1.65,ncol=6,border=F,bty=↘
      "n",fill= brewer.pal(5,"YlOrBr")[3:5],text.col="black",xpd↘
      =NA)

# Titling

mtext("Population and Gross National Income",3,line=2,adj=0,cex↘
      =2.4,outer=T,family="Lato Black")
mtext("Size: population – Colour: GNI per capita. Atlas method (↘
      current US $), 2010",3,line=0,adj=0,cex=1.75,outer=T,font↘
      =3)
mtext("",1,line=1,adj=1.0,cex=1.25,outer=T,font=3)
dev.off()
```

In the script, contrary to the previous example, two separate figures are created that are then connected in LaTeX. For the first figure, data from the hnp data frame are filtered so that only the countries with an entry for gni are kept. Then the product of pop and gni is calculated and added to the data frame. Then a second data frame continents is created by aggregation, in which the population and the product of population and the GNI are added up. This gives us the correct continent-related GNIs called cgni. From this, we create three classes “low”, “middle” and “high” called ccgni. Lastly, for the tree map, we require this as a numerical variable nccgni. As in the previous example, we have to call plot() to define the figure prior to plotting the tree map, since treemap() is based on grid. The tree map graphic is then plotted with the function of the same name. In contrast to the previous example, we set type=value, since the colour represents the attribute of an additional variable in this case. This variable is defined using vColor=cgni. We also specify that no legend be created, since we want to draw that separately at the end. The size of the rectangle is defined with vSize=pop. We then draw a legend using the legend() function. When doing this, we have to be careful to use the correct colours from the selected Brewer palette, as the treemap() function apparently assumes a fiver palette. The second tree map for the countries within the continents is created essentially like the first one. The only differences are that we define two variables using index=c(“continent”, “iso3”), because we want to keep the first level (continents) as order criteria. We also have to use fontsize.labels=c(0.1,20) to set specific font sizes for the two levels: 0.1 makes the labels of the first level invisible. The result is a tree map of all countries, but sorted within their continents.

```
pdf_file<-"pdf/tablecharts_treemap_2b_inc.pdf"
cairo_pdf(bg="grey98", pdf_file,width=11.69,height=7.5)

par(omi=c(0.65,0.25,1.25,0.75),mai=c(0.3,2,0.35,0),family="Lato ↘
      Light",las=1)
library(treemap)
library(RColorBrewer)

# Daten einlesen und Grafik vorbereiten

load("myData/hnp.RData")
myData<-subset(daten,daten$gni>0)
attach(myData)
kgni<-cut(gni,c(0,40000,80000))
levels(kgni)<-c("low","middle","high")
myData$kgni<-kgni
myData$nkgni<-as.numeric(kgni)

# Grafik definieren und weitere Elemente

plot(1:1,type="n",axes=F)
treemap(myData,title="",index=c("kontinent","iso3"), vSize="pop"↘
      ,vColor="nkgni",type="value",palette="Blues",aspRatio=2.5,↘
      fontsize.labels=c(0.1,20),position.legend="none")
```

```
legend(0.35,0.6,levels(kgni)[1:3],cex=1.65,ncol=3,border=F,bty="↘
      n",fill= brewer.pal(9,"Blues")[7:9],text.col="black",xpd=↘
      NA)

# Betitelung

mtext("Within Continent: Country Level",3,line=2,adj=0,cex=2.4,↘
      outer=T,family="Lato Black")
mtext("Size: population - Colour: GNI per capita. Atlas method (↘
      current US $), 2010",3,line=0,adj=0,cex=1.75,outer=T,font↘
      =3)
mtext("Source: data.wordlbank.org",1,line=1,adj=1.0,cex=1.25,↘
      outer=T,font=3)
dev.off()
```

The last step is to combine the two tree maps into one figure. To do that, both are sequentially embedded into LaTeX. Finally, the LaTeX embedding:

```
\documentclass{article}
\usepackage[paperheight=26.7cm, paperwidth=21cm, top=0cm, left=0↘
     cm, right=0cm, bottom=0cm]{geometry}
\usepackage{color}
\usepackage[abs]{overpic}
\definecolor{myBackground}{rgb}{0.9412,0.9412,0.9412}
\pagecolor{myBackground}
\begin{document}
\pagestyle{empty}
\begin{center}
\begin{overpic}[scale=0.70,unit=1mm]{../pdf/tablecharts_treemap_↘
     2a_inc.pdf}
\put(60,128){}
\end{overpic}
\begin{overpic}[scale=0.70,unit=1mm]{../pdf/tablecharts_treemap_↘
     2b_inc.pdf}
\put(60,28){}
\end{overpic}
\end{center}
\end{document}
```