The script first embeds the data: first four colours, twice, then the actual data, and finally the labels. Note that the application sums are read twice. This makes the bars twice as wide. The grant data on the other hand are only read once; the respective first values are set to zero. This is followed by two barplot() calls. Since we do not set horiz=TRUE this time, not bar, but column charts are plotted. The second call is done with add=TRUE, so that these columns are plotted on top of the first ones. To maintain correct proportions, grant data (y) have to be multiplied by 2. Alternatively, column width could have been adjusted accordingly within the barplot() calls. Finally follow four times three labels: halfway up, the values of the requested and granted sums, respectively (requested sums in the left columns), centrally over the columns the ratio of granted to requested sums in percent. The term format(round(x[1,i],1),nsmall=1),adj=0) ensures output of a decimal place even with integers. Further examples for bar charts will follow in Sect. 7.3.

6.2 Pie Charts and Radial Diagrams

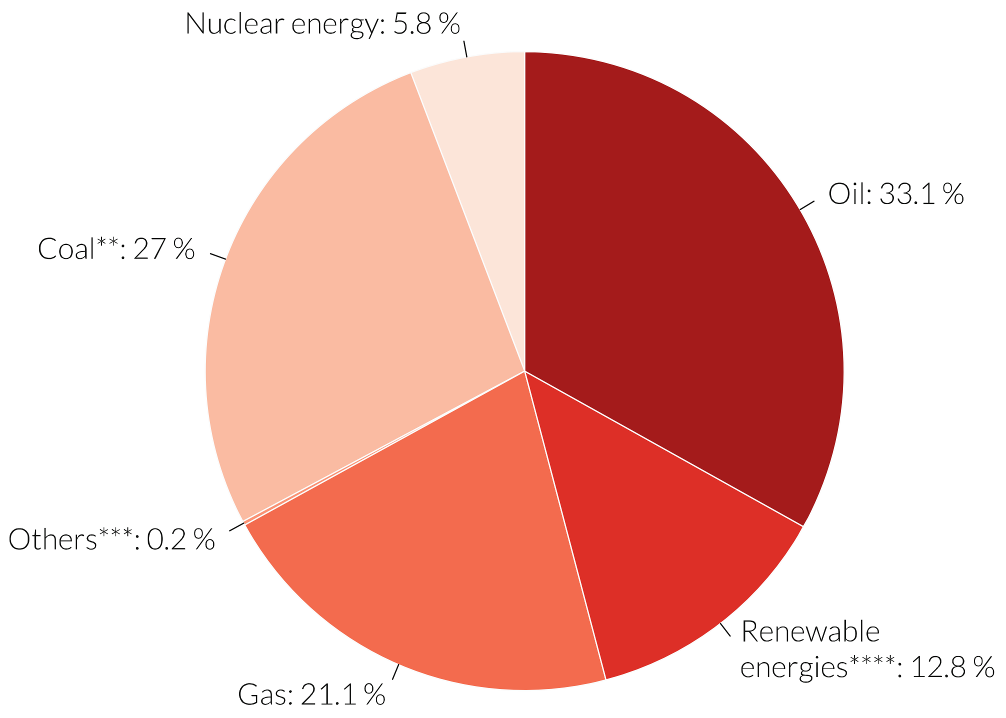
Pie charts are pretty to look at—nothing more. But no less for that, which is why I do not want to discourage their use. The frequently used argument that human perception finds it easier to compare, e.g., points on lines (dot charts) remains unchallenged. However, if the magnitude of individual data is so close together that they are hard to sort by their exact difference in magnitude, then the message of these data is that they are similar and small differences are therefore unimportant. In this section, we will not only explain regular pie charts, but also give examples of spie charts and radial polygons, and in Chap. 11 also radial column charts. The types differ in the variability of their radii and/or angles:

Type	Radius	Angle
Pie chart	Constant	Variable
Spie chart	Variable	Variable
Radial polygon	Variable	Constant
Radial column chart	Variable	Constant

6.2.1 Simple Pie Chart

Global energy mix (including sea and air transport)

Shares of energy sources in the primary energy supply in percent, 2008*



* Primary energy sources = primary energy production + imports - exports +/- stock changes
** Including peat
*** Bio matter, biodegradable waste (excluding industrial waste), water power, geothermal energy, solar, wind, and marine power.
**** Industrial waste and flammable waste that can serve as energy sources and are non-biodegradable

Source: German Federal Agency for Civic Education: keyword 'Enegiemix' [energy mix], www.bpb.de [website in German]

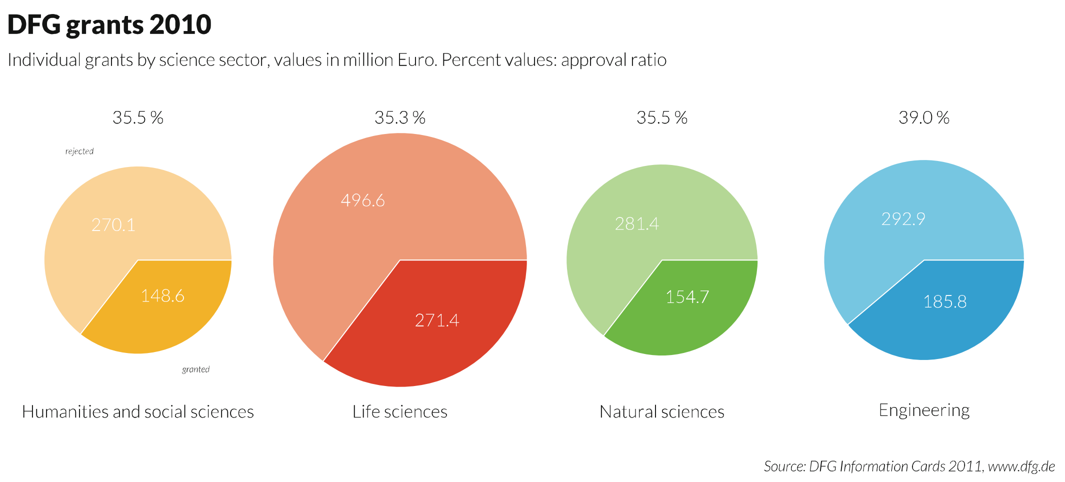
The figure shows data concerning the “Global energy mix”. Drawing a pie chart is more difficult than you think. With a pie chart, you can—and should—vary the order of the individual segments, as well as the rotation. Ideally, a segment should start at 0°, but that will not always be possible. Depending on the space you want to assign to the figure, and the attributes of the concrete data and the concrete labels, one should play around a little to find the optimal order of segments and rotation for the given case. In the present case, three or four iterations were necessary, i.e., changes of segment order and rotations, to achieve the presented result. If there are no substantive reasons for the use of multiple colours, one should use a colour palette with one colour for pie charts. The data are derived from a page of the Bundeszentrale für politische Bildung (German Federal Agency for Political Education).

```
pdf_file<-"pdf/piecharts_simple.pdf"
cairo_pdf(bg="grey98", pdf_file,width=11,height=11)

par(omi=c(2,0.5,1,0.25),mai=c(0,1.25,0.5,0.5),family="Lato Light↘
      ",las=1)
library(RColorBrewer)

# Create chart

pie.myData<-c(5.8,27.0,0.2,21.1,12.8,33.1)
energytypes<-c("Nuclear energy:","Coal**:","Others***:","Gas:","↘
      Renewable\nenergies****:","Oil:")
names(pie.myData)<-paste(energytypes,pie.myData,"%",sep=" ")
pie(pie.myData,col=brewer.pal(length(pie.myData),"Reds"),border↘
      =0,cex=1.75,radius=0.9,init.angle=90)

# Titling

mtext("Global energy mix (including sea and air transport)",3,↘
      line=2,adj=0,family="Lato Black",outer=T,cex=2.5)
mtext("Shares of energy sources in the primary energy supply* in↘
       percent, 2008",3,line=-0.75,adj=0,cex=1.65,font=3,outer=T↘
      )
mtext("* Primary energy sources = primary energy production + ↘
      imports - exports +/- stock changes",1,line=2,adj=0,cex↘
      =1.05,outer=T)
mtext("** Including peat",1,line=3.2,adj=0,cex=1.05,outer=T)
mtext("*** Bio matter, biodegradable waste (excluding industrial↘
       waste), water power, geothermal energy, solar, wind, and ↘
      marine power.",1,line=4.4,adj=0,cex=1.05,outer=T)
mtext("**** Industrial waste and flammable waste that can serve ↘
      as energy sources and are non-biodegradable",1,line=5.6,↘
      adj=0,cex=1.05,outer=T)
mtext("Source: German Federal Agency for Civic Education: ↘
      keyword 'Enegiemix' [energy mix], www.bpb.de [website in ↘
      German]",1,line=8,adj=1,cex=1.25,font=3,outer=T)
dev.off()
```

In the script, we start with the individual margin settings and then load the RColorBrewer package. Data are defined directly within the script, as are the labels for the segments. In the next line, these are complemented with their percent values and the percent symbol. The pie chart is plotted with pie(); we are using a Brewer colour palette for the segment colours, and the number of colour gradations results from the size of the vector containing the data. Suitable values for radius and initial angle (init.angle) can be found through trial and error. The labels follow at the end, as usual.

6.2.2 Pie Charts, Labels Inside (Panel)

About the figure: illustrations with multiple figures are obviously also an option with pie charts. Here, we once more refer to the data from the example in Sect. 6.1.12. As the granted sums are parts of the requested sums, pie charts are a suitable choice. The size of the circle symbolises the magnitude of the requested sum of the respective science sector. The approval rate, the ratio between granted and requested sums, becomes our heading, and the science sector the caption. The absolute numbers are written within the slices, which is explained in the subheading. The colours match the ones the DFG uses for the respective science sectors. Note that you have to use the square root for the radius, since doubling of the radius will quadruple the area.



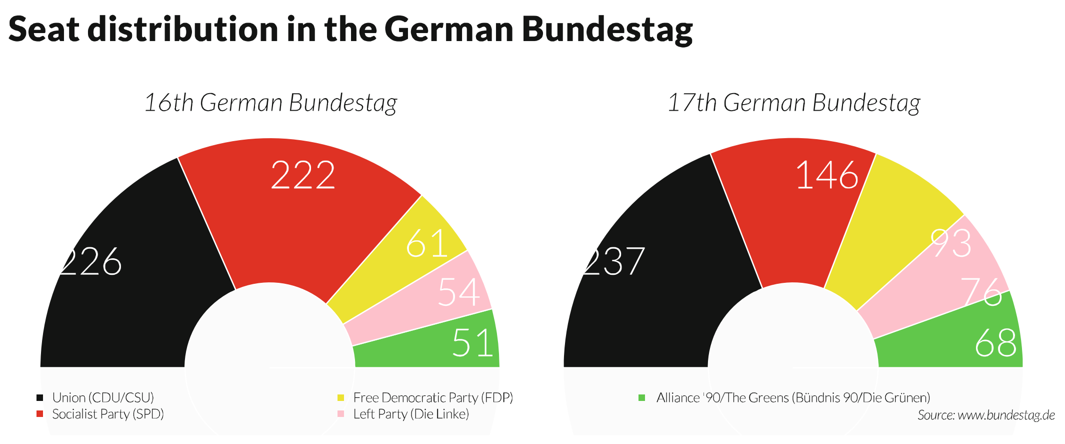
The data can be directly extracted from a PDF published on the DFG website “DFG Information Cards 2011” (card 9 “Research Grants”), and were manually entered into the R script.

```
pdf_file<-"pdf/piecharts_1x4.pdf"
cairo_pdf(bg="grey98", pdf_file,width=14,height=6)

library(plotrix)
par(omi=c(0.5,0.5,1,0.5),mai=c(0,0,0,0),xpd=T,mfcol=c(1,4),↘
      family="Lato Light",las=1)

# Import data

source("scripts/inc_data_dfg.r")

# Define charts and other elements

for (i in 1:4)
{
plot(1:5,type="n",axes=F,xlab="",ylab="")
values<-c(x[2,i]-y[2,i],y[2,i])
myCircle<-floating.pie(3,3,values,border="white",radius=2.1*sqrt↘
      (x[1,i]/max(x[1,])),col=c(myColours1[i],myColours2[i]))
pie.labels(3,3,myCircle,values,bg=NA,border=NA,radius=x[1,i]/max↘
      (x[1,]),cex=2,col="white")
if (i==1) pie.labels(3,3,myCircle,c("rejected","granted"),bg=NA,↘
      border=NA,radius=1.95,font=3)
text(3,4.7,cex=2,adj=0.5,paste(format(round(100*y[2,i]/x[1,i],1)↘
      ,nsmall=1),"%",sep=" "))
text(3,1.2,labelling[i],cex=2,adj=0.5)
}

# Titling

mtext("DFG grants 2010",3,line=4,adj=0,family="Lato Black",outer↘
      =T,cex=2)
mtext("Individual grants by science sector, values in million ↘
      Euro. Percent values: approval ratio",3,line=1,adj=0,cex↘
      =1.35,font=2,outer=T)
mtext("Source: DFG Information Cards 2011, www.dfg.de",1,line=2,↘
      adj=1.0,cex=1.1,font=3,outer=T)
dev.off()
```

This script uses the plotrix package by Jim Lemon to position the four pie charts at different locations within one figure using the floating.pie() function. We first import the data and the colour definitions as described in Sect. 6.1.12, and create four windows using mfcol(). In each of these, we first define an empty plot(), in whose middle (i.e. at x- and y-position 3) the pie chart is then plotted using floating.pie(). Since we then want to set the labels within the pie chart, floating.pie() does not only plot the result, but also saves the object into the variable myCircle. The radius of the pie chart is meant to symbolise the magnitude of the requested sums. Therefore, we calculate it using x[1,i]/max(x[1,]) as the ratio between the value of the requested sum of the i-th circle and the largest requested sum. All radii are then multiplied by the factor 2.1 to maximise the size of the circles within the figure. The next line plots the labels that are to go in the charts. We can achieve this by not multiplying the radius by the factor 2.1 here. The ratio, however, is set as before so the labels appear "centred" within the slices. Finally follows one condition: In the first pie chart, the slice contents should contain the additional labels "rejected" and "approved" in italics (font=3), effectively as a legend. Since all the pie charts look very similar, labelling one is enough. Now follow the headings and captions, comprising the approval rates and the science sectors, as in Sect. 6.1.12. Just like before, we use the text() function to do this. The title which was stored in the inc_title_dfg.r file, is plotted last.

6.2.3 *Seat Distribution (Panel)*

About the figure: this presentation type is often found in publications of election results. It involves semi-circles, i.e. pie charts in which the individual slices do not add up to an entire circle but only a "semi-circle"—or more precisely a "half donut". This chart type is very useful for seat distributions, as it is a stylised parliament: it adequately reflects the impression of the plenary chamber. The figure compares the seat distribution in the German Bundestag during the 16th and 17th election period. It particularly shows the severe loss of seats for the Social Democratic Party of Germany (SPD) from 222 to 146.



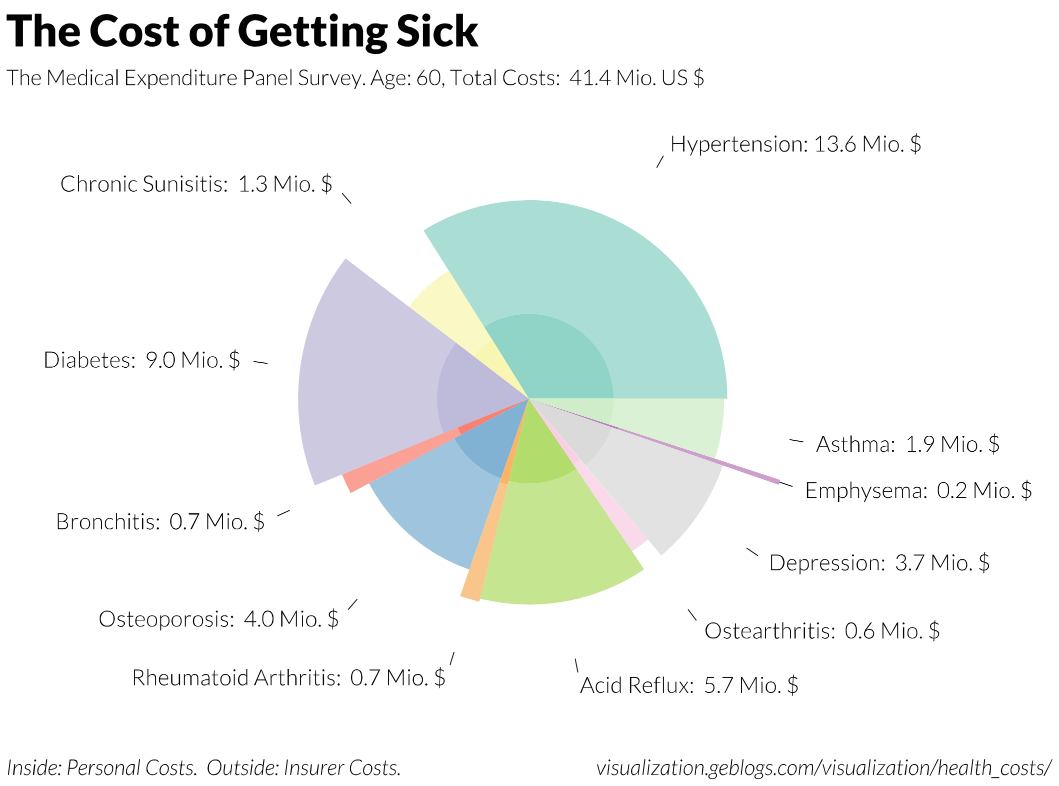
Data can be extracted from the Bundestag homepage (http://www.bundestag.de), and were manually entered into the script.

```
pdf_file<-"pdf/piecharts_allocation_of_seats_1x2.pdf"
cairo_pdf(bg="grey98", pdf_file,width=10,height=3.75)

par(omi=c(0.5,0.5,1,0.5),mai=c(0,0,0,0),xpd=T,mfcol=c(1,2),↘
      family="Lato Light")
library(plotrix)

# Define chart

plot(1:5,type="n",axes=F,xlab="",ylab="",xlim=c(1,5),ylim=c↘
      (1,10))
mySeats<-c(51,54,61,222,226)
myDes<-c(mySeats,""); mySlices<-50*mySeats /sum(mySeats)
myValues<-c(mySlices,50); myDisc<-100
MyColour<-c("white", "white", "black", "white", "white")

# Create chart

mySemiCircle<-floating.pie(3,1,myValues,border="white",radius↘
      =1.9,xpd=F,col=c("green","pink","yellow","red","black",par↘
      ("bg")))
pie.labels(3,1,mySemiCircle,myDes,bg=NA,border=NA,radius=1.5,cex↘
      =2,col=MyColour)
floating.pie(3,1,myDisc,border="white",col=par("bg"),radius=0.7,↘
      xpd=F)
mtext("16th German Bundestag",3,line=0,adj=0.5,font=3,cex=1.3)
```

```
par(xpd=T)
legend(1,0.5,c("Union (CDU/CSU)","Socialist Party (SPD)","Free ↘
      Democratic Party (FDP)","Left Party (Die Linke)"," ↘
      Alliance ’90/The Greens (Bündnis 90/Die Grünen)"),border=F↘
      ,pch=15,col=c("black","red","yellow","pink","green"),bty="↘
      n",cex=0.7,xpd=NA,ncol=3)
par(xpd=F)

# Define chart

plot(1:5,type="n",axes=F,xlab="",ylab="",xlim=c(1,5),ylim=c↘
      (1,10))
mySeats<-c(68,76,93,146,237)
myDes<-c(mySeats,""); mySlices <-50*mySeats/sum(mySeats)
myValues<-c(mySlices,50); myDisc<-100

# Create chart

semicirlce<-floating.pie(3,1,myValues,border="white",radius=1.9,↘
      xpd=F,col=c("green","pink","yellow","red","black",par("bg"↘
      )))
pie.labels(3,1,mySemiCircle,myDes,bg=NA,border=NA,radius=1.5,cex↘
      =2,col=MyColour)
floating.pie(3,1,myDisc,border="white",col=par("bg"),radius=0.7,↘
      xpd=F)
mtext("17th German Bundestag",3,line=0,adj=0.5,font=3,cex=1.3)

# Titling

mtext("Seat distribution in the German Bundestag",3,line=3,adj↘
      =0,family="Lato Black",outer=T,cex=1.8)
mtext("Source: www.bundestag.de",1,line=1,adj=1.0,cex=0.7,font↘
      =3,outer=T)
dev.off()
```

About the script: to our knowledge, R does not offer a direct option to draw semi-circle pie charts. However, we can manage with a trick. We first embed the plotrix package by Jim Lemon. As seen in the previous example, this package allows us to draw pie charts at a specific position within a plot. If we now use this method to place a pie chart with its centre exactly on the border of such a plot, and simultaneously ensure that the part outside said border is not displayed and that the non-displayed part is just a slice comprising half the actual values, then we are pretty close to our desired result. For the presented figure, we first define a window with one row and two columns using mfcol=c(1,2), i.e. two figures next to each other. Then an "empty" plot spanning 1–5 in x-plan and 1–10 in y-plan is defined using plot(). The number of seats for the individual parties is stored in the mySeats variable; myDes matches the mySeats variable, but we append an empty element at the end. Since we only require a semi-circle and not a full one, we also define a variable mySlices that contains the halved shares (50*mySeats/sum(mySeats)). The values for the pie chart are saved in myValues: these are the segment values from slices that add up to 50 (percent), complemented by the value 50 for the part that will then be invisible. For aesthetic reasons, we then superimpose a “disc”, a pie chart consisting of a single slice. The call for the pie chart is done via the floating.pie() function, which draws the pie chart at position 3, 1 into the “empty” plot, so that the centre point is exactly on the y-axis. The radius was chosen as 1.9, so that the width is adequately filled. With xpd=FALSE, the lower half of the pie chart is intentionally cut off. The segment colours are supposed to match the parties’ colours. Since we want to label the slices, we store the object created with floating.pie() into the mySemiCircle variable. Labelling follows in the next line using pie.labels(), but we choose a slightly smaller radius here, so the labels fit completely within the slices. Thirdly, the white pie chart disc that only consists of one value is drawn on top with radius 0.7; it has the effect that the first pie chart becomes a donut that resembles the appearance of a parliament. After the title comes the legend, preceded by xpd=TRUE, so it can be drawn into the bottom corner; then we reset the value to FALSE. Then comes a second call for the three already known pie chart elements for the second chart window, this time with the seat distribution during the 17th German Bundestag. As usual, the joint heading and captions in the outer margin follow last.

6.2.4 Spie Chart



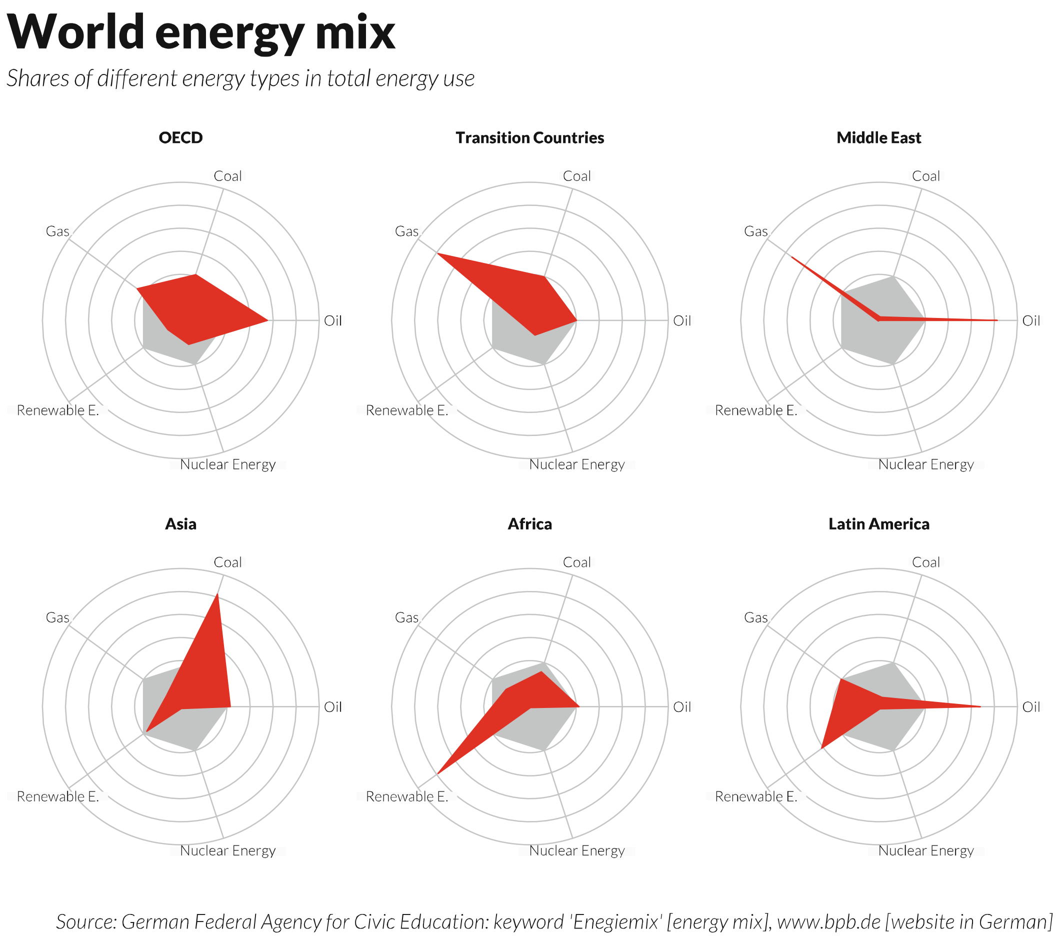
About the figure: Contrary to a common pie chart, a spie chart is an extension in which the statistical information is reflected not only by the magnitude of the angles but also by those of the radius. The present example, adapted from a conception by Ben Fry, shows insurance costs caused by sickness across the USA, according to the Medical Expenditure Panel Survey (MEPS) for the age group of 60-year-olds. Pictured are 11 different illnesses, with the radius reflecting the cost per insured patient and the angle the number of people affected by this illness. Therefore, slice area reflects the total cost for the illness. Personal costs are additionally shown as a second level, so that the area of the total cost that is not overlaid reflects the insurer's cost. Slice labelling should be in the form of a circle and therefore based on the longest slice. Data were taken from a Java plugin on the site http://visualization.geblogs.com/visualization/health_costs/ and data for 1 year were manually entered into an XLSX spreadsheet.

```
pdf_file<-"pdf/piecharts_spiechart.pdf"
cairo_pdf(bg="grey98", pdf_file,width=15,height=11)

par(omi=c(0.5,0.5,0.75,0.5),mai=c(0.1,0.1,0.1,0.1),family="Lato ↘
      Light",las=1)
library(RColorBrewer)

# Import data and prepare chart

x<-read.xls("myData/Healthcare_costs.xlsx",1,encoding="latin1")
attach(x)
n<-nrow(x)
myFactor<-max(sqrt(Acosts60))/0.8

# Define chart and other elements

plot.new()
myC0<-rep(NA,n)
myColours<-brewer.pal(n,"Set3")
for (i in 1:n)
{
  par(new=T)
  r<-col2rgb(myColours[i])[1]
  g<-col2rgb(myColours[i])[2]
  b<-col2rgb(myColours[i])[3]
  myC0[i]<-rgb(r,g,b,190,maxColorValue=255)
  myValue<-format(Total60/1000000,digits=1)
  myTotal<-paste(Disease,": ",myValue," Mio. $",sep="")
  if (Acosts60[i] == max(Acosts60)) {myDes<-myTotal} else {myDes↘
        <-NA}

  # Create slices

  pie(Patients60,border=NA,radius=sqrt(Acosts60[i])/myFactor,col↘
        =myC0,
      labels=myDes,cex=1.8)
  par(new=T)
  r<-col2rgb(myColours[i])[1]
  g<-col2rgb(myColours[i])[2]
  b<-col2rgb(myColours[i])[3]
  myC0[i]<-rgb(r,g,b,maxColorValue=255)
  pie(Patients60,border=NA,radius=sqrt(Pcosts60[i])/myFactor,col↘
       =myC0,labels=NA)
  myC0<-rep(NA,n)
}

# Titling

mtext("The Cost of Getting Sick",3,line=-1,adj=0,cex=3.5,family=↘
      "Lato Black",outer=T)
mtext("The Medical Expenditure Panel Survey. Age: 60, Total ↘
      Costs:  41.4 Mio. US $",3,line=-3.6,adj=0,cex=1.75,outer=T↘
      )
mtext("Inside: Personal Costs.  Outside: Insurer Costs.",1,line↘
      =0,adj=0,cex=1.75,outer=T,font=3)
mtext("visualization.geblogs.com/visualization/health_costs/",1,↘
      line=0,adj=1.0,cex=1.75,outer=T,font=3)
dev.off()
```

About the script: R offers functions for the creation of spie charts in different packages, such as spie() in the caroline package. However, we are only using basic functions included in the standard package, which do not limit our layout options.

After importing the data, we first construct a factor from the maximum of the square root of Acosts60. To adjust to the dimensions of the illustration, the value is divided by 0.8. After definition of the illustration using plot.new(), we define the vector myC0, which will later be filled with colour parameters. For this we use a Brewer colour palette as a basis. Then a loop is used to draw a one-slice pie chart for each data record, with individual radii depending on Acosts60. The trick lies in the fact that the colour myC0 is always set to NA at the end of the loop. In each run of the loop, only one slice is defined by colour: MyC0[i]. This means that two times i circles are drawn, of which only one slice each is visible. Every label is taken from the circle with the maximal radius, so that all labels are at the same distance from the centre. Since we are drawing two levels that should be colour-matched, we plot the lower levels as a transparent value of the chosen palette. To do this, we dissolve the selected colour from the Brewer palette into its RGB components. The lower circle slice is then plotted with intensity of 190, and the upper one with intensity of 255.

6.2.5 *Radial Polygons (Panel)*



About the figure: If there are multiple variables that have been measured in the same dimension, such as school grades or a person's characteristics on a scale from 1 to 10 or, such as here, shares of different energy types in the total energy mix in individual regions (see Sect. 6.2.1), then radial polygons are a good choice for presentation. Sometimes, the names spider plot, radar chart or net chart are also used. The values of the variables are plotted on radii at identical angles from a circle centre and then connected; the lengths of the radii correspond to the attributes.

Data: see annex A, worldenergymix.xlsx.

```
pdf_file<-"pdf/radial_polygons_2x3.pdf"
cairo_pdf(bg="grey98", pdf_file,width=12,height=12)

par(mfcol=c(2,3),omi=c(1,0.5,1,0.5),mai=c(0,0,0,0),cex.axis=0.9,↘
      cex.lab=1,xpd=T,col.axis="green",col.main="red",family="↘
      Lato Light",las=1)

library(plotrix)
library(gdata)

# Import data and prepare chart

myRegions<-read.xls("myData/worldenergymix.xlsx", encoding="↘
      latin1")
row.names(myRegions)<-myRegions$Region
myRegions$Region<-NULL
myLabelling<-c("Oil","Coal","Gas","Renewable E.","Nuclear Energy↘
      ")

myRegions<-myRegions[, c(1,2,3,4,5)]
myLabelling<-myLabelling[c(1,2,3,4,5)]

# Create charts

for (i in 2:nrow(myRegions))
{
radial.plot(rep(100/length(myRegions),length(myRegions)),labels=↘
      myLabelling,rp.type="p",main="",line.col="grey",show.grid=↘
      T,show.grid.labels=F,radial.lim=c(0,55),poly.col="grey")
radial.plot(myRegions[i,],labels="",rp.type="p",main="",line.col↘
      ="red",show.grid=F,radial.lim=c(0,55),poly.col="red",add=T↘
      )
mtext(row.names(myRegions)[i],line=2,family="Lato Black")
}

# Titling

mtext("World energy mix",line=2,cex=3,family="Lato Black",outer=↘
      T,adj=0)
mtext(line=-1,"Shares of different energy types in total energy ↘
      use",cex=1.5,font=3,outer=T,adj=0)
mtext(side=1, "Source: German Federal Agency for Civic Education↘
      : keyword 'Enegiemix' [energy mix], www.bpb.de [website in↘
       German]",line=2,cex=1.3,font=3,outer=T,adj=1)
dev.off()
```

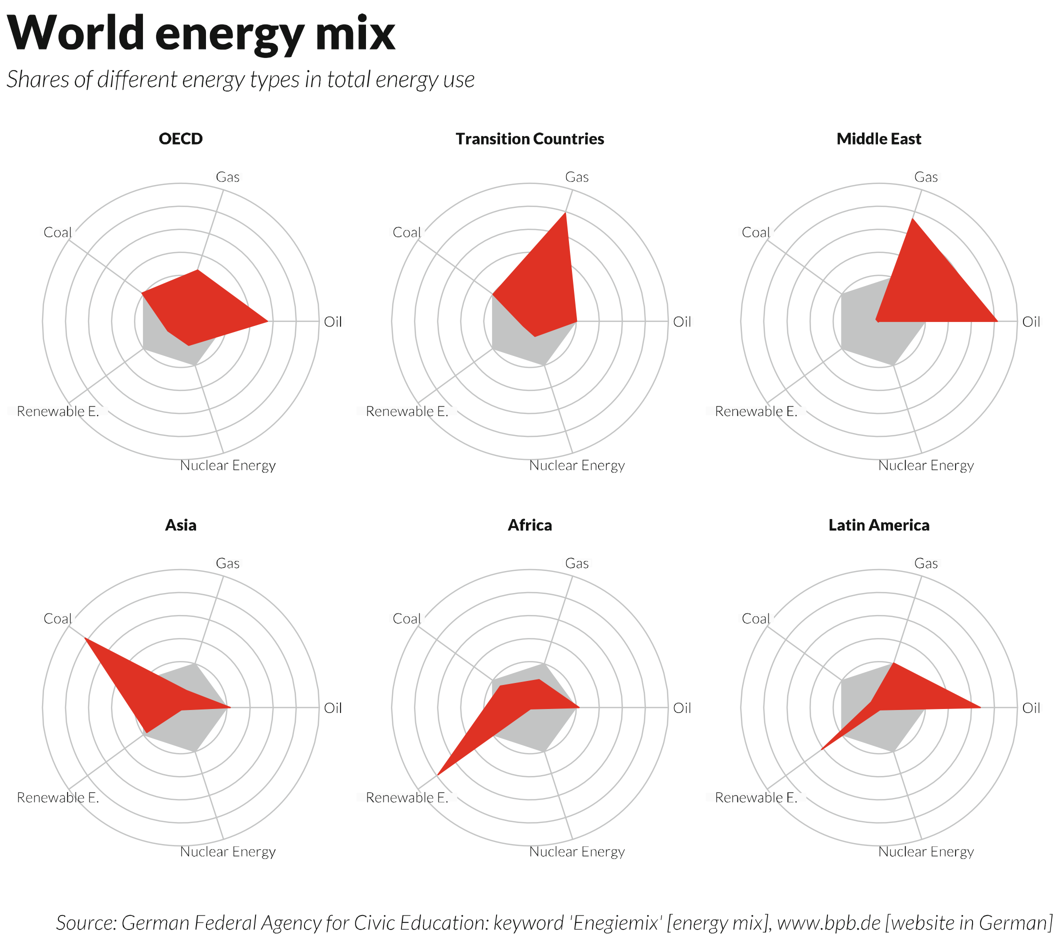
In the script, radial polygons are plotted using the radial.plot() function from the plotrix package by Jim Lemon. To do this, we first create row names with the regions from the data imported from an XLSX spreadsheet. We carry out the labelling directly in the script. The first line of data (the "world") is ignored, and we then use a loop to draw two radial polygon charts on top of each other for each region: the first, in grey colour, draws a polygon of equal edge lengths as a background. To do this, we transfer the parameter 100/length(regions) to the radial.plot() function; the number is the number of featured regions. The labels, too, are output here. The rp.type parameter specifies whether lines, polygons (areas) or symbols (points) are to be plotted. With show.grid=T, the drawing of guide circles is enabled. The radius is set with c(0,55). In the second call of radial.plot(), we use add=T to plot the actual data over this "reference polygon". The parameters correspond to the ones from the first call. At the end of the loop, the heading is set above the polygons.

6.2.6 Radial Polygons (Panel): Different Column Arrangement

The degree of dependence between the form of the illustration and the position of the columns can be seen if the second and third columns are swapped.

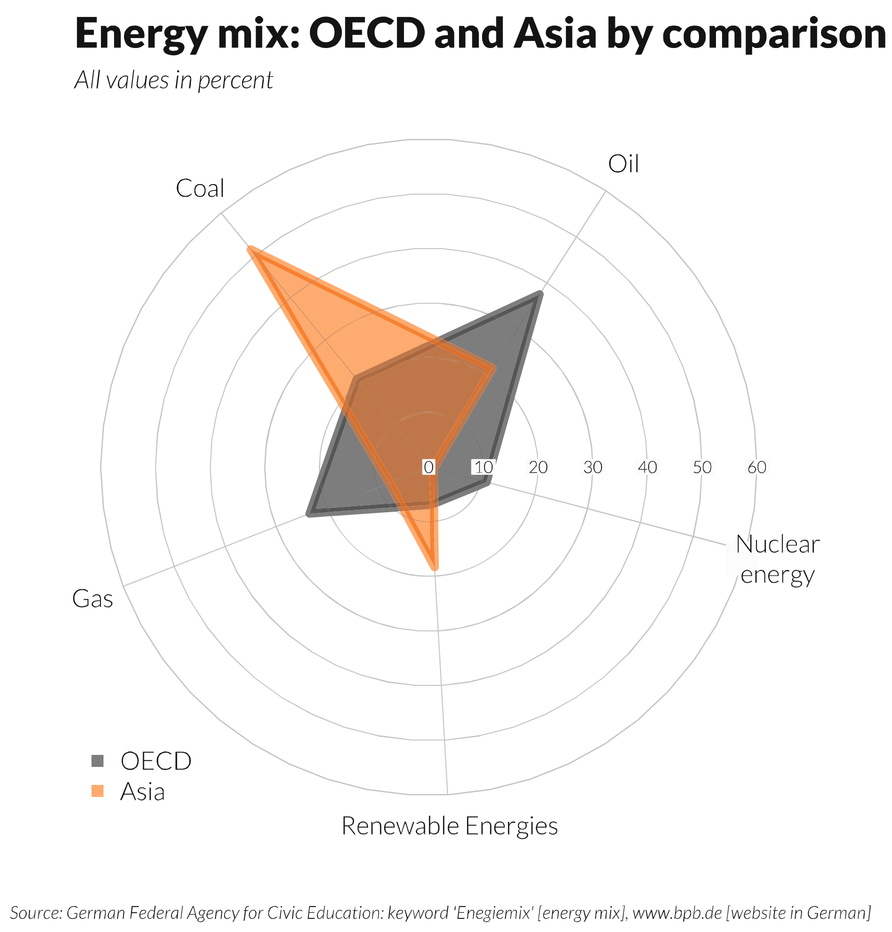
```
myRegions<-myRegions[, c(1,3,2,4,5)]
myLabelling<-myLabelling[c(1,3,2,4,5)]
```

The result, looking at e.g. the polygon for the Middle East, leaves a completely different impression, since the area of the polygon is now much bigger.



Therefore, this form of representation should be treated with caution. In my opinion, however, it is useful for the comparison of multiple persons, countries etc. with identically arranged categories.

6.2.7 Radial Polygons Overlay



About the figure: aside from a panel type, an illustration form in which graphics are stacked once more comes in handy. In this case, a coordinate system can also be added to the chart. If the areas occupied by the polygons are filled, then transparent colours should be chosen.

Data: see annex A, wordenergymix.xlsx.

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

par(omi=c(1,0.25,1,1),mai=c(0,2,0,0.5),cex.axis=1.5,cex.lab=1,↘
      xpd=T,family="Lato Light",las=1)
library(plotrix)

# Import data and prepare chart

myRegions<-read.xls("myData/worldenergymix.xlsx", encoding="↘
      latin1")
myC1<-rgb(80,80,80,155,maxColorValue=255)
myC2<-rgb(255,97,0,155,maxColorValue=255)
```

```
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).