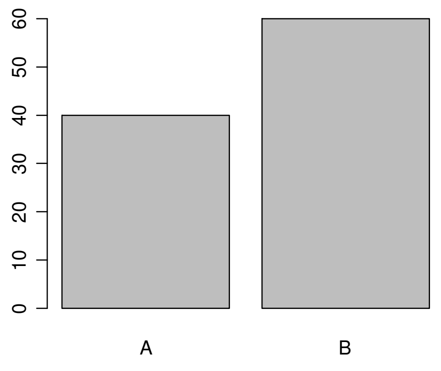
Part V
Customizing graphics

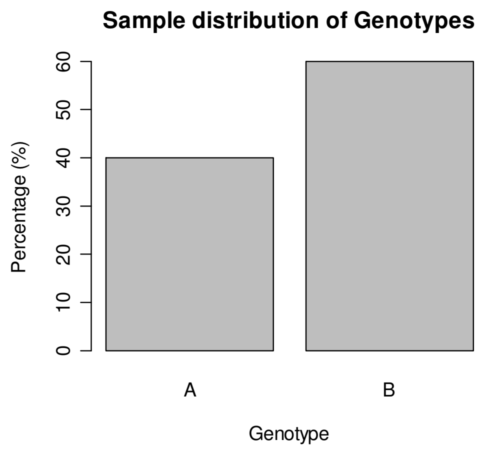
Labels
Without meaningful labels the produced graphics is useless. One can add parameters to the functions like plot, barplot, pie and hist to get the desired labels. To demonstrate how to add labels to graphics we make use the dataset *genodata*. Lets start with a simple barplot describing the distribution of genotypes:

```
barplot(prop.table(table(gene))*100)
```



We need to add labels.

```
barplot(prop.table(table(gene))*100, xlab="Genotype",
     ylab="Percentage (%)",
     main="Sample distribution of Genotypes")
```



The parameters available in almost all graphics-oriented functions are:

`main="Titel"`	– Title above the graph;
`xlab="x-axis"`	– annotation below the x-axis;
`ylab="y-axis"`	– annotation for y-axis;
`sub="Subtitle"`	– Annotation below the graph;

The same parameters tend to work across many different functions:

```
hist(fenotype, main="Fenotype", xlab="Intensity (in thousands)",
      ylab="Frequency")
pie(table(gene), main="Distribution of genotypes", sub="Figure 12.3")
plot(rnorm(100), ylab="Value", main="Random variables")
```

Remarks for a realy courageous reader: If needed, one can modify the size or font or color of the labels. See the help file for *par* (*?par*) and read about parameters *cex* (and about *cex.axis, cex.labels, cex.main*, ...); *font* (*font.axis, font.main*, ...); and *col.axis, col.main*, ...
One can use mathematical expressions and greek letters to annotate plots. See for example

```
example(plotmath)
demo(plotmath)
```

Colours and fills
Default colors are printer-friendly but may look somewhat dull for presentations. So we might want to add some vibrant colors to our plots. Changing the colors can be done using the *col*= parameter. There exist a few different possibilities to do this.

First, you may use the color names to get the desired look. Example:

```
barplot(table(gene), col=c("green","forestgreen"))
```

You can remember only a dosen colour names in english? Not a problem! All colour names available in R can be printed by using the command *colours()*. Examples of all colours and their names in R can be seen in http://www.ms.ut.ee/mart/R/varvid2.html.

As an alternative one might use some predefined palette of colors. Available palettes include for example *heat.colors*; *terrain.colors*; *topo.colors*; *rainbow* and others. Usage examples:

```
barplot(1:10, col=heat.colors(10))
pie(table(Plant), col=rainbow(15))
```

It is also possible to use color numbers. In the following example observations corresponding to genotype A is plotted using color number 2 (red) and all others are plotted with color 1 (black):

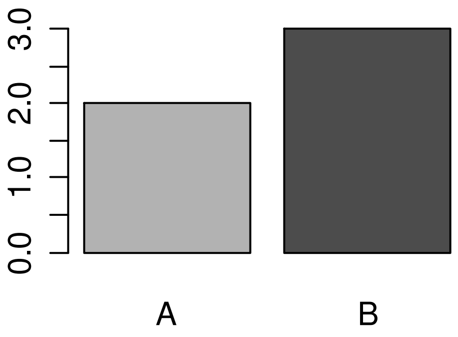
```
plot(fenotype, col=1+1*(gene=="A"), cex=3)
```

If one really wishes, one can also use rgb-values (give the exact amount of red, green and blue to be mixed to get the desired color. Each of the component should be within the interval of 0..1.):

```
barplot(table(gene), main="Distribution of genotypes",
      col=c("gold", "purple"), col.main=rgb(0.9,0.1,0.25))
```

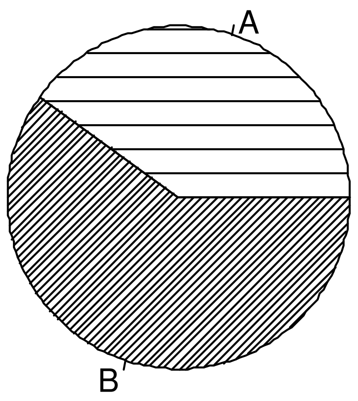
For printing purposes one might to play with the shades of gray instead:

```
barplot(table(gene), col=gray(c(0.7,0.3)))
```



If the aim is to produce printable graphics, one can also use the *density*= and *angle*= parameters to customize the look of the plots:

```
barplot(table(gene), density=c(5, 30), col=1)
barplot(table(gene), density=10, angle=c(10, 75), col=1)
pie(table(gene), density=c(8, 35), angle=c(0, 45))
```



Lower and upper units of the axes
Sometimes it is useful to have two or more graphs with the same x- or y-axes. One can specify the lower and upper units of the axes using the *xlim=* and *ylim=* parameters.

Examples
We continue to use the dataset *CO2*; If you have not done so beforehand you may issue command *data(CO2); attach(CO2)* before continuing.

Commands

```
hist(uptake[Treatment==“chilled“])
hist(uptake [Treatment==“nonchilled“])
```

will produce two different plots, one for each treatment. Unfortunately they may have slightly different units on the x-axis and hence the visual comparison of distribution is difficult. We can improve this:

```
hist(uptake[Treatment==“chilled“], xlim=c(0,50))
hist(uptake[Treatment==“nonchilled“], xlim=c(0,50))
```

One can similarily modify the limits of the plotted y-axis:
```
boxplot(uptake~Treatment, ylim=c(0,70))
```

Legend
Sometimes a legend would be needed. How to add a legend to a plot?

First make a plot, and then, (before closing the graphics window) we can add with another command a legend to the existing plot.

Example:
Make a plot, make some spare space for legend using parameter *ylim=*. Use the *pch=* parameter to change the shape of the dots in a plot:

```
plot(conc, uptake, col=1+(Treatment==“chilled“),
     pch=16, ylim=c(0,50))
```

Add the legend to the plot:

```
legend(700,10, c(“nonchilled“,“chilled“), col=1:2,
     pch=16)
```

Multiple plots

If you draw a new figure the previous one gets lost in R. However, if you want, you can change the default behaviour. If you run R under Windows, and the „R Graphics“ window is active, then one can select menu option *History-> Record*. All graphics created after this option has been selected are saved and can be recalled by pressing PgUp or PgDn keys while the graphics window is active.
Another option is to create a new graphics window for a new plot. This can be done in a program by issuing *windows()* command before drawing a new plot. For example run the following program all at once:

```
plot(conc, uptake)
windows()
hist(uptake)
```

to create two different graphs.

One can put many plots into the same window. You just have to say how many rows and columns of plots you want to have:

```
# 2 rows and 3 columns of plots in one window
par(mfrow=c(2,3))
# Now lets fill the window with 6 plots
hist(rnorm(100))
pie(table(gene))
barplot(table(gene))
hist(fenotype)
qqnorm(fenotype)
plot(gene,fenotype)
```

Keep in mind that usually one wants to use *xlim*= and/or *ylim*= parameters, if multiple plots are going to be drawn to the same window (to make the plots visually comparable):

```
par(mfrow=c(2,1))
hist(uptake[Treatment==“chilled“], xlim=c(0,50))
hist(uptake[Treatment==“nonchilled“], xlim=c(0,50))
```

How to get back to „normal“ from the divided window? If you close the graphics window, then the next plot drawn will use the default settings. One may of course issue another *par* command demanding one row and one column of plots in one window, as an alternative way to go back to „normal“ graphics window.

One can also collect the produced graphics by copying them to Word or to some other similar program, or you can save them in multiple formats for later use.

To see more graphics commands and options, or just to enjoy some nice graphics R can produce, type the following commands:

```
demo(graphics)
demo(image)
demo(persp)
```