R Graphics

An Introduction to Graphical Representation of Data using R

Dr. Christoph Scherber, Agroecology, University of Goettingen,
cscherb1@gwdg.de
09.10.2007

Introduction

One of the most tedious things for beginners using R is to find out how to **configure graphs** properly. This manual is intended as an introduction to the vast possibilities R offers for the professional display of data.

The material is structured in **three chapters**:

(1) Standard graphs, which is what everybody is using all the time
(2) Trellis displays, with which an in-depth "dissection" of datasets is possible; data that are conventionally treated as "multivariate" can be visualized very conveniently.
(3) Some additional graphics features from interesting R libraries; examples are shown for barplots, 3D surface plots and for the display of meteorological data.

Christoph Scherber,
October 2007.

1. Standard plots

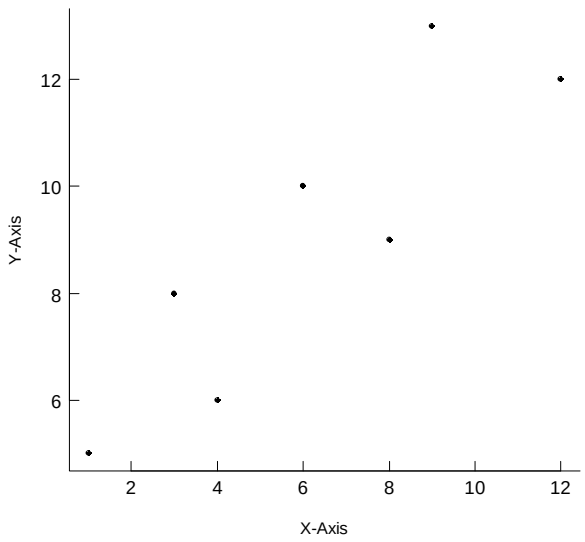
```
x<-c(1,3,4,6,8,9,12)
y<-c(5,8,6,10,9,13,12)
```

Quick plotting starts like this:
`plot(x,y)`

More advanced plotting, however, involves a step-wise approach:

- Set up an empty plotting region (type="n")
- Draw the points
- Draw the axes
- Write the title(s)
- Add regression lines etc.

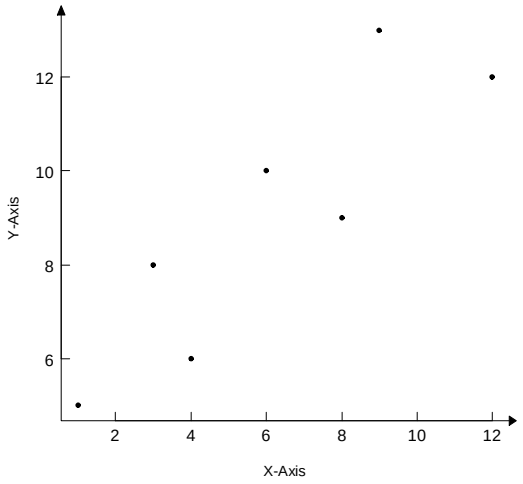
```
par(tck=0.02,las=1,cex.axis=1.3,cex.lab=1.2,mgp=c(3,0.5,0),bty="L")

x<-c(1,3,4,6,8,9,12)
y<-c(5,8,6,10,9,13,12)
plot(x,y,type="n",axes=F,xlab="",ylab="")
points(x,y,pch=16)
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-Axis")
box()
```



Suppose now that we want to **add arrows to the axes**; there are, in principle, two possibilities for this. One is to use the

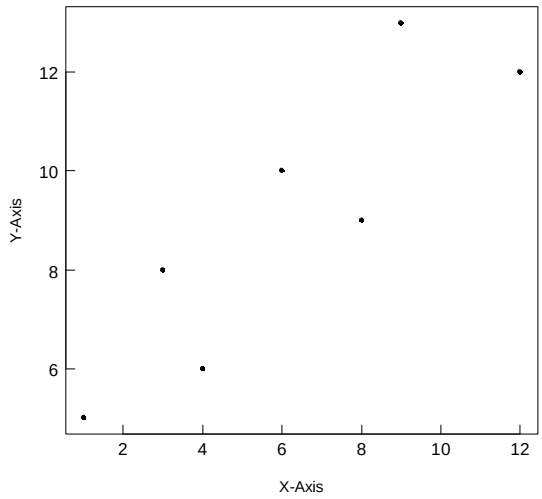
arrows() command (which doesnt´t look too good) - the second (a bit more complicated one) is to use polygon():

```
u <- par("usr")
u

# first possibility:
arrows(u[1], u[3], u[2], u[3], code = 2, xpd =
TRUE,angle=20,length=0.2)

arrows(u[1], u[3], u[1], u[4], code = 2, xpd =
TRUE,angle=20,length=0.2)

#second possibility:
polygon(c(u[2],u[2],u[2]+0.2),c(u[3]-
0.1,u[3]+0.1,u[3]),xpd=TRUE,col="black",lty=1)

polygon(c(u[1]-
0.1,u[1]+0.1,u[1]),c(u[4],u[4],u[4]+0.2),xpd=TRUE,col="black",
lty=1)
```



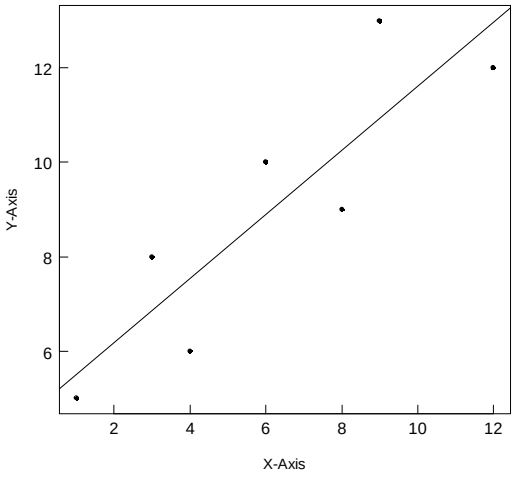
Usually, you wouldn´t really need those arrowheads, so you just proceed using par(bty="o"):

```
par(tck=0.02,las=1,cex.axis=1.3,cex.lab=1.2,mgp=c(3,0.5,0),bty
="o")

x<-c(1,3,4,6,8,9,12)
y<-c(5,8,6,10,9,13,12)
plot(x,y,type="n",axes=F,xlab="",ylab="")
points(x,y,pch=16)
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-Axis")
box()
```



You could then add a regression line to the graph by typing

abline(lm(y~x))



There is a lot more to play around; here are some of the most useful graphics parameters from the par() command:

Specification of graphics parameters using par():

- **adj**

The value of adj determines the way in which text strings are justified. A value of 0 produces **left-justified** text, 0.5 **centered** text and 1 **right-justified** text. (Any value in [0, 1] is allowed, and on most devices values outside that interval will also work.) Note that the adj argument of text also allows adj = c(x, y) for different adjustment in x- and y- direction.

- **ann**

If set to FALSE, high-level plotting functions calling plot.default do not annotate the plots they produce with axis titles and overall titles. The default is to do annotation.

- **ask**

logical. If TRUE (and the R session is interactive) the user is asked for input, before a new figure is drawn. As this applies to the device, it also affects output by packages grid and lattice. It can be set even on non-screen devices.

- **bg**

The color to be used for the background of plots. When called from par() it also sets new=FALSE. A description of how colors are specified is given below. Note that some graphics functions such as plot.default and points have an argument of this name with a different meaning.

- **bty**

A character string which determined the type of box which is drawn about plots. If bty is one of "o", "l", "7", "c", "u", or "]" the resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box.

- **cex**

A numerical value giving the amount by which plotting text and symbols should be scaled relative to the default. Note that some graphics functions such as plot.default have an argument of this name which multiplies this graphical parameter.

- **cex.axis**

The magnification to be used for axis annotation relative to the current setting of cex. (Some functions such as points accept a vector of values which are recycled. Other uses will take just the first value if a vector of length greater than one is supplied.)

- **cex.lab**

The magnification to be used for x and y labels relative to the current setting of cex.

- **cex.main**

The magnification to be used for main titles relative to the current setting of cex.

- **cex.sub**

The magnification to be used for sub-titles relative to the current setting of cex.

- **col**

A specification for the default plotting color. A description of how colors are specified is given below. (Some functions such as lines accept a vector of values which are recycled. Other uses will take just the first value if a vector of length greater than one is supplied.)

- **col.axis**

The color to be used for axis annotation.

- **col.lab**

The color to be used for x and y labels.

- **col.main**

The color to be used for plot main titles.

- **col.sub**

The color to be used for plot sub-titles.

- **crt**

A numerical value specifying (in degrees) how single characters should be rotated. It is unwise to expect values other than multiples of 90 to work. Compare with srt which does string rotation.

- **family**

The name of a font family for drawing text. The maximum allowed length is 200 bytes. This name gets mapped by each graphics device to device-specific font descriptions. The default value is "" which means that the default device fonts will be used. Standard values are "serif", "sans", "mono", and "symbol" and the Hershey font families are also available. (Different devices may define others, and some devices will ignore this setting completely.) This can be specified inline for text.

- **fg**

The color to be used for the foreground of plots. This is the default color used for things like axes and boxes around plots. When called from par() this also sets parameter col to the same value.

A description of how colors are specified is given below.

- **fig**

A numerical vector of the form c(x1, x2, y1, y2) which gives the (NDC) coordinates of the figure region in the display region of the device. If you set this, unlike S, you start a new plot, so to add to an existing plot use new=TRUE as well.

- **fin**

The figure region dimensions, (width,height), in inches. If you set this, unlike S, you start a new plot.

- **font**

An integer which specifies which font to use for text. If possible, device drivers arrange so that 1 corresponds to plain text, 2 to bold face, 3 to italic and 4 to bold italic. Also, font 5 is expected to be the symbol font, in Adobe symbol encoding. On some devices font families can be selected by family to choose different sets of 5 fonts.

- **font.axis**

The font to be used for axis annotation.

- **font.lab**

The font to be used for x and y labels.

- **font.main**

The font to be used for plot main titles.

- **font.sub**

The font to be used for plot sub-titles.

- **gamma**

the gamma correction, see argument gamma to hsv. This is only accepted if the current device has support for changing the gamma correction: currently only windows and quartz do. (X11 has support for setting the gamma correction when opening the device, but not for changing it.)

- **lab**

A numerical vector of the form c(x, y, len) which modifies the default way that axes are annotated. The values of x and y give the (approximate) number of tickmarks on the x and y axes and len specifies the label length. The default is c(5, 5, 7). Note that this only affects the way the parameters xaxp and yaxp are set when the user coordinate system is set up, and is not consulted when axes are drawn. len is unimplemented in R.

- **las**

numeric in {0,1,2,3}; the style of axis labels.
0: always parallel to the axis [default],
1: always horizontal,
2: always perpendicular to the axis,
3: always vertical.
Note that other string/character rotation (via argument srt to par) does not affect the axis labels.

- **lend**

The line end style. This can be specified as an integer or string:
0 and "round" mean rounded line caps [default];
1 and "butt" mean butt line caps;
2 and "square" mean square line caps.

- **lheight**

The line height multiplier. The height of a line of text (used to vertically space multi-line text) is found by multiplying the current font size both by the current character expansion and by the line height multiplier. Default value is 1.

- **ljoin**

The line join style. This can be specified as an integer or string:
0 and "round" mean rounded line joins [default];
1 and "mitre" mean mitred line joins;
2 and "bevel" mean bevelled line joins.

- **lmitre**

The line mitre limit. This controls when mitred line joins are automatically converted into bevelled line joins. The value must be larger than 1 and the default is 10. Not all devices will honour this setting.

- **lty**

The line type. Line types can either be specified as an integer (0=blank, 1=solid, 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash) or as one of the character strings "blank", "solid", "dashed", "dotted", "dotdash", "longdash", or "twodash", where "blank" uses ‘invisible lines’ (i.e., does not draw them).

Alternatively, a string of up to 8 characters (from c(1:9, "A":"F")) may be given, giving the length of line segments which are alternatively drawn and skipped. See section ‘Line Type Specification’ below.

Some functions such as lines accept a vector of values which are recycled. Other uses will take just the first value if a vector of length greater than one is supplied.

- **lwd**

The line width, a positive number, defaulting to 1. The interpretation is device-specific, and some devices do not implement line widths less than one. (See the help on the device for details of the interpretation.)

Some functions such as lines accept a vector of values which are recycled. Other uses will take just the first value if a vector of length greater than one is supplied.

- **mai**

A numerical vector of the form c(bottom, left, top, right) which gives the margin size specified in inches.

- **mar**

A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(5, 4, 4, 2) + 0.1.

- **mex**

mex is a character size expansion factor which is used to describe coordinates in the margins of plots. Note that this does not change the font size, rather specifies the size of font used to convert between mar and mai, and between oma and omi.

- **mfcol, mfrow**

A vector of the form c(nr, nc). Subsequent figures will be drawn in an nr-by-nc array on the device by columns (mfcol), or rows (mfrow), respectively.

In a layout with exactly two rows and columns the base value of "cex" is reduced by a factor of 0.83: if there are three or more of either rows or columns, the reduction factor is 0.66.

Consider the alternatives, layout and split.screen.

- **mfg**

A numerical vector of the form c(i, j) where i and j indicate which figure in an array of figures is to be drawn next (if setting) or is being drawn (if enquiring). The array must already have been set by mfcol or mfrow.

For compatibility with S, the form c(i, j, nr, nc) is also accepted, when nr and nc should be the current number of rows and number of columns. Mismatches will be ignored, with a warning.

- **mgp**

The margin line (in mex units) for the axis title, axis labels and axis line. The default is c(3, 1, 0).

- **mkh**

The height in inches of symbols to be drawn when the value of pch is an integer. Completely ignored currently.

- **new**

logical, defaulting to FALSE. If set to TRUE, the next high-level plotting command (actually plot.new) should not clean the frame before drawing “as if it was on a new device”.

- **oma**

A vector of the form c(bottom, left, top, right) giving the size of the outer margins in lines of text.

- **omd**

A vector of the form c(x1, x2, y1, y2) giving the outer margin region in NDC (= normalized device coordinates), i.e., as fraction (in [0,1]) of the device region.

- **omi**

A vector of the form c(bottom, left, top, right) giving the size of the outer margins in inches.

- **pch**

Either an integer specifying a symbol or a single character to be used as the default in plotting points. See points for possible values and their interpretation.

- **pin**

The current plot dimensions, (width,height), in inches.

- **plt**

A vector of the form c(x1, x2, y1, y2) giving the coordinates of the plot region as fractions of the current figure region.

- **ps**

integer; the pointsize of text (but not symbols). Unlike the pointsize argument of most devices, this does not change the relationship between mar and mai (nor oma and omi), but it does scale cin and csi.

What is meant by ‘pointsize’ is device-specific, but most devices mean a multiple of 1bp, that is 1/72 of an inch.

- **pty**

A character specifying the type of plot region to be used; "s" generates a square plotting region and "m" generates the maximal plotting region.

- **srt**

The string rotation in degrees. See the comment about crt.

- **tck**

The length of tick marks as a fraction of the smaller of the width or height of the plotting region. If tck >= 0.5 it is interpreted as a fraction of the relevant side, so if tck = 1 grid lines are drawn. The default setting (tck = NA) is to use tcl = -0.5 (see below).

- **tcl**

The length of tick marks as a fraction of the height of a line of text. The default value is -0.5; setting tcl = NA sets tck = -0.01 which is S' default.

- **usr**

A vector of the form c(x1, x2, y1, y2) giving the extremes of the user coordinates of the plotting region. When a logarithmic scale is in use (i.e., par("xlog") is true, see below), then the x-limits will be 10 ^ par("usr")[1:2]. Similarly for the y-axis.

- **xaxp**

A vector of the form c(x1, x2, n) giving the coordinates of the extreme tick marks and the number of intervals between tick-marks when par("xlog") is false. Otherwise, when log coordinates are active, the three values have a different meaning: For a small range, n is negative, and the ticks are as in the linear case, otherwise, n is in 1:3, specifying a case number, and x1 and x2 are the lowest and highest power of 10 inside the user coordinates, 10 ^ par("usr")[1:2]. (The "usr" coordinates are log10-transformed here!)

- n=1
- will produce tick marks at 10^j for integer j,
- n=2
- gives marks k 10^j with k in {1, 5},
- n=3

- gives marks k 10^j with k in {1, 2, 5}.
- See axTicks() for a pure R implementation of this.

This parameter is reset when a user coordinate system is set up, for example by starting a new page or by calling plot.window or setting par("usr"): n is taken from par("lab"). It affects the default behaviour of subsequent calls to axis for sides 1 or 3.

- **xaxs**

The style of axis interval calculation to be used for the x-axis. Possible values are "r", "i", "e", "s", "d". The styles are generally controlled by the range of data or xlim, if given. Style "r" (regular) first extends the data range by 4 percent and then finds an axis with pretty labels that fits within the range. Style "i" (internal) just finds an axis with pretty labels that fits within the original data range. Style "s" (standard) finds an axis with pretty labels within which the original data range fits. Style "e" (extended) is like style "s", except that it is also ensures that there is room for plotting symbols within the bounding box. Style "d" (direct) specifies that the current axis should be used on subsequent plots. (Only "r" and "i" styles are currently implemented)

- **xaxt**

A character which specifies the x axis type. Specifying "n" suppresses plotting of the axis. The standard value is "s": for compatibility with S values "l" and "t" are accepted but are equivalent to "s": any value other than "n" implies plotting.

- **xlog**

A logical value (see log in plot.default). If TRUE, a logarithmic scale is in use (e.g., after plot(*, log = "x")). For a new device, it defaults to FALSE, i.e., linear scale.

- **xpd**

A logical value or NA. If FALSE, all plotting is clipped to the plot region, if TRUE, all plotting is clipped to the figure region, and if NA, all plotting is clipped to the device region.

- **yaxp**

A vector of the form c(y1, y2, n) giving the coordinates of the extreme tick marks and the number of intervals between tick-marks unless for log coordinates, see xaxp above.

- **yaxs**

The style of axis interval calculation to be used for the y-axis. See xaxs above.

- **yaxt**

A character which specifies the y axis type. Specifying "n" suppresses plotting.

- **ylog**

A logical value; see xlog above.

- **Color Specification**

Colors can be specified in several different ways. The simplest way is with a character string giving the color name (e.g., "red"). A list of the possible colors can be obtained with the function colors. Alternatively, colors can be specified directly in terms of their RGB components with a string of the form "#RRGGBB" where each of the pairs RR, GG, BB consist of two hexadecimal digits giving a value in the range 00 to FF. Colors can also be specified by giving an index into a small table of colors, the palette. This provides compatibility with S. Index 0 corresponds to the background color.

Additionally, "transparent" or (integer) NA is transparent, useful for filled areas (such as the background!), and just invisible for things like lines or text. Semi-transparent colors are available for use on devices that support them.

The functions **rgb**, **hsv**, **hcl**, **gray** and **rainbow** provide additional ways of generating colors.

- **Line Type Specification**

Line types can either be specified by giving an index into a small built-in table of line types (1 = solid, 2 = dashed, etc, see lty above) or directly as the lengths of on/off stretches of line. This is done with a string of an even number (up to eight) of characters, namely non-zero (hexadecimal) digits which give the lengths in consecutive positions in the string. For example, the string "33" specifies three units on followed by three off and "3313" specifies three units on followed by three off followed by one on and finally three off. The 'units' here are (on most devices) proportional to lwd, and with lwd = 1 are in pixels or points or 1/96 inch.

The five standard dash-dot line types (lty = 2:6) correspond to c("44", "13", "1343", "73", "2262").

Note that NA is not a valid value for lty.

Prior to R 2.4.0, zero hex digits were accepted and handled in a device-dependent way, e.g. X11() mapped them to one, pdf() regarded zero as terminating the string and windows() gave a zero-length stretch.

Colors

Here is a list of colors that can be used with R; the list can also be reproduced using **colors()**:

```
  [1] "white"              "aliceblue"          "antiquewhite"
  [4] "antiquewhite1"      "antiquewhite2"      "antiquewhite3"
  [7] "antiquewhite4"      "aquamarine"         "aquamarine1"
 [10] "aquamarine2"        "aquamarine3"        "aquamarine4"
 [13] "azure"              "azure1"             "azure2"
 [16] "azure3"             "azure4"             "beige"
 [19] "bisque"             "bisque1"            "bisque2"
 [22] "bisque3"            "bisque4"            "black"
 [25] "blanchedalmond"     "blue"               "blue1"
 [28] "blue2"              "blue3"              "blue4"
 [31] "blueviolet"         "brown"              "brown1"
 [34] "brown2"             "brown3"             "brown4"
 [37] "burlywood"          "burlywood1"         "burlywood2"
 [40] "burlywood3"         "burlywood4"         "cadetblue"
 [43] "cadetblue1"         "cadetblue2"         "cadetblue3"
 [46] "cadetblue4"         "chartreuse"         "chartreuse1"
 [49] "chartreuse2"        "chartreuse3"        "chartreuse4"
 [52] "chocolate"          "chocolate1"         "chocolate2"
 [55] "chocolate3"         "chocolate4"         "coral"
 [58] "coral1"             "coral2"             "coral3"
 [61] "coral4"             "cornflowerblue"     "cornsilk"
 [64] "cornsilk1"          "cornsilk2"          "cornsilk3"
 [67] "cornsilk4"          "cyan"               "cyan1"
 [70] "cyan2"              "cyan3"              "cyan4"
 [73] "darkblue"           "darkcyan"           "darkgoldenrod"
 [76] "darkgoldenrod1"     "darkgoldenrod2"     "darkgoldenrod3"
 [79] "darkgoldenrod4"     "darkgray"           "darkgreen"
 [82] "darkgrey"           "darkkhaki"          "darkmagenta"
 [85] "darkolivegreen"     "darkolivegreen1"    "darkolivegreen2"
 [88] "darkolivegreen3"    "darkolivegreen4"    "darkorange"
 [91] "darkorange1"        "darkorange2"        "darkorange3"
 [94] "darkorange4"        "darkorchid"         "darkorchid1"
 [97] "darkorchid2"        "darkorchid3"        "darkorchid4"
[100] "darkred"            "darksalmon"         "darkseagreen"
[103] "darkseagreen1"      "darkseagreen2"      "darkseagreen3"
[106] "darkseagreen4"      "darkslateblue"      "darkslategray"
[109] "darkslategray1"     "darkslategray2"     "darkslategray3"
[112] "darkslategray4"     "darkslategrey"      "darkturquoise"
[115] "darkviolet"         "deeppink"           "deeppink1"
[118] "deeppink2"          "deeppink3"          "deeppink4"
[121] "deepskyblue"        "deepskyblue1"       "deepskyblue2"
[124] "deepskyblue3"       "deepskyblue4"       "dimgray"
[127] "dimgrey"            "dodgerblue"         "dodgerblue1"
[130] "dodgerblue2"        "dodgerblue3"        "dodgerblue4"
[133] "firebrick"          "firebrick1"         "firebrick2"
[136] "firebrick3"         "firebrick4"         "floralwhite"
[139] "forestgreen"        "gainsboro"          "ghostwhite"
[142] "gold"               "gold1"              "gold2"
[145] "gold3"              "gold4"              "goldenrod"
[148] "goldenrod1"         "goldenrod2"         "goldenrod3"
[151] "goldenrod4"         "gray"               "gray0"
[154] "gray1"              "gray2"              "gray3"
[157] "gray4"              "gray5"              "gray6"
[160] "gray7"              "gray8"              "gray9"
[163] "gray10"             "gray11"             "gray12"
[166] "gray13"             "gray14"             "gray15"
[169] "gray16"             "gray17"             "gray18"
[172] "gray19"             "gray20"             "gray21"
[175] "gray22"             "gray23"             "gray24"
[178] "gray25"             "gray26"             "gray27"
[181] "gray28"             "gray29"             "gray30"
[184] "gray31"             "gray32"             "gray33"
[187] "gray34"             "gray35"             "gray36"
```

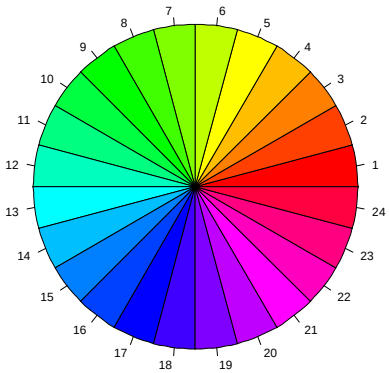
```
[190] "gray37"          "gray38"          "gray39"
[193] "gray40"          "gray41"          "gray42"
[196] "gray43"          "gray44"          "gray45"
[199] "gray46"          "gray47"          "gray48"
[202] "gray49"          "gray50"          "gray51"
[205] "gray52"          "gray53"          "gray54"
[208] "gray55"          "gray56"          "gray57"
[211] "gray58"          "gray59"          "gray60"
[214] "gray61"          "gray62"          "gray63"
[217] "gray64"          "gray65"          "gray66"
[220] "gray67"          "gray68"          "gray69"
[223] "gray70"          "gray71"          "gray72"
[226] "gray73"          "gray74"          "gray75"
[229] "gray76"          "gray77"          "gray78"
[232] "gray79"          "gray80"          "gray81"
[235] "gray82"          "gray83"          "gray84"
[238] "gray85"          "gray86"          "gray87"
[241] "gray88"          "gray89"          "gray90"
[244] "gray91"          "gray92"          "gray93"
[247] "gray94"          "gray95"          "gray96"
[250] "gray97"          "gray98"          "gray99"
[253] "gray100"         "green"           "green1"
[256] "green2"          "green3"          "green4"
[259] "greenyellow"     "grey"            "grey0"
[262] "grey1"           "grey2"           "grey3"
[265] "grey4"           "grey5"           "grey6"
[268] "grey7"           "grey8"           "grey9"
[271] "grey10"          "grey11"          "grey12"
[274] "grey13"          "grey14"          "grey15"
[277] "grey16"          "grey17"          "grey18"
[280] "grey19"          "grey20"          "grey21"
[283] "grey22"          "grey23"          "grey24"
[286] "grey25"          "grey26"          "grey27"
[289] "grey28"          "grey29"          "grey30"
[292] "grey31"          "grey32"          "grey33"
[295] "grey34"          "grey35"          "grey36"
[298] "grey37"          "grey38"          "grey39"
[301] "grey40"          "grey41"          "grey42"
[304] "grey43"          "grey44"          "grey45"
[307] "grey46"          "grey47"          "grey48"
[310] "grey49"          "grey50"          "grey51"
[313] "grey52"          "grey53"          "grey54"
[316] "grey55"          "grey56"          "grey57"
[319] "grey58"          "grey59"          "grey60"
[322] "grey61"          "grey62"          "grey63"
[325] "grey64"          "grey65"          "grey66"
[328] "grey67"          "grey68"          "grey69"
[331] "grey70"          "grey71"          "grey72"
[334] "grey73"          "grey74"          "grey75"
[337] "grey76"          "grey77"          "grey78"
[340] "grey79"          "grey80"          "grey81"
[343] "grey82"          "grey83"          "grey84"
[346] "grey85"          "grey86"          "grey87"
[349] "grey88"          "grey89"          "grey90"
[352] "grey91"          "grey92"          "grey93"
[355] "grey94"          "grey95"          "grey96"
[358] "grey97"          "grey98"          "grey99"
[361] "grey100"         "honeydew"        "honeydew1"
[364] "honeydew2"       "honeydew3"       "honeydew4"
[367] "hotpink"         "hotpink1"        "hotpink2"
[370] "hotpink3"        "hotpink4"        "indianred"
[373] "indianred1"      "indianred2"      "indianred3"
[376] "indianred4"      "ivory"           "ivory1"
```

```
[379] "ivory2"               "ivory3"               "ivory4"
[382] "khaki"                "khaki1"               "khaki2"
[385] "khaki3"               "khaki4"               "lavender"
[388] "lavenderblush"        "lavenderblush1"       "lavenderblush2"
[391] "lavenderblush3"       "lavenderblush4"       "lawngreen"
[394] "lemonchiffon"         "lemonchiffon1"        "lemonchiffon2"
[397] "lemonchiffon3"        "lemonchiffon4"        "lightblue"
[400] "lightblue1"           "lightblue2"           "lightblue3"
[403] "lightblue4"           "lightcoral"           "lightcyan"
[406] "lightcyan1"           "lightcyan2"           "lightcyan3"
[409] "lightcyan4"           "lightgoldenrod"       "lightgoldenrod1"
[412] "lightgoldenrod2"      "lightgoldenrod3"      "lightgoldenrod4"
[415] "lightgoldenrodyellow" "lightgray"            "lightgreen"
[418] "lightgrey"            "lightpink"            "lightpink1"
[421] "lightpink2"           "lightpink3"           "lightpink4"
[424] "lightsalmon"          "lightsalmon1"         "lightsalmon2"
[427] "lightsalmon3"         "lightsalmon4"         "lightseagreen"
[430] "lightskyblue"         "lightskyblue1"        "lightskyblue2"
[433] "lightskyblue3"        "lightskyblue4"        "lightslateblue"
[436] "lightslategray"       "lightslategrey"       "lightsteelblue"
[439] "lightsteelblue1"      "lightsteelblue2"      "lightsteelblue3"
[442] "lightsteelblue4"      "lightyellow"          "lightyellow1"
[445] "lightyellow2"         "lightyellow3"         "lightyellow4"
[448] "limegreen"            "linen"                "magenta"
[451] "magenta1"             "magenta2"             "magenta3"
[454] "magenta4"             "maroon"               "maroon1"
[457] "maroon2"              "maroon3"              "maroon4"
[460] "mediumaquamarine"     "mediumblue"           "mediumorchid"
[463] "mediumorchid1"        "mediumorchid2"        "mediumorchid3"
[466] "mediumorchid4"        "mediumpurple"         "mediumpurple1"
[469] "mediumpurple2"        "mediumpurple3"        "mediumpurple4"
[472] "mediumseagreen"       "mediumslateblue"      "mediumspringgreen"
[475] "mediumturquoise"      "mediumvioletred"      "midnightblue"
[478] "mintcream"            "mistyrose"            "mistyrose1"
[481] "mistyrose2"           "mistyrose3"           "mistyrose4"
[484] "moccasin"             "navajowhite"          "navajowhite1"
[487] "navajowhite2"         "navajowhite3"         "navajowhite4"
[490] "navy"                 "navyblue"             "oldlace"
[493] "olivedrab"            "olivedrab1"           "olivedrab2"
[496] "olivedrab3"           "olivedrab4"           "orange"
[499] "orange1"              "orange2"              "orange3"
[502] "orange4"              "orangered"            "orangered1"
[505] "orangered2"           "orangered3"           "orangered4"
[508] "orchid"               "orchid1"              "orchid2"
[511] "orchid3"              "orchid4"              "palegoldenrod"
[514] "palegreen"            "palegreen1"           "palegreen2"
[517] "palegreen3"           "palegreen4"           "paleturquoise"
[520] "paleturquoise1"       "paleturquoise2"       "paleturquoise3"
[523] "paleturquoise4"       "palevioletred"        "palevioletred1"
[526] "palevioletred2"       "palevioletred3"       "palevioletred4"
[529] "papayawhip"           "peachpuff"            "peachpuff1"
[532] "peachpuff2"           "peachpuff3"           "peachpuff4"
[535] "peru"                 "pink"                 "pink1"
[538] "pink2"                "pink3"                "pink4"
[541] "plum"                 "plum1"                "plum2"
[544] "plum3"                "plum4"                "powderblue"
[547] "purple"               "purple1"              "purple2"
[550] "purple3"              "purple4"              "red"
[553] "red1"                 "red2"                 "red3"
[556] "red4"                 "rosybrown"            "rosybrown1"
[559] "rosybrown2"           "rosybrown3"           "rosybrown4"
[562] "royalblue"            "royalblue1"           "royalblue2"
[565] "royalblue3"           "royalblue4"           "saddlebrown"
```

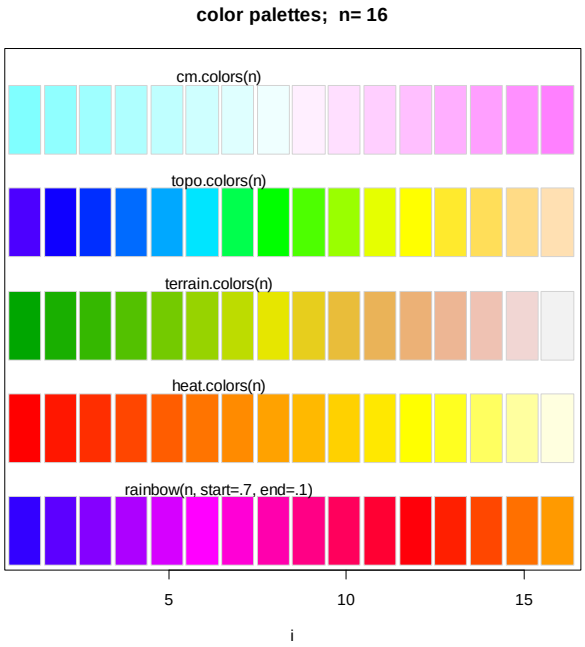
```
[568] "salmon"          "salmon1"         "salmon2"
[571] "salmon3"         "salmon4"         "sandybrown"
[574] "seagreen"        "seagreen1"       "seagreen2"
[577] "seagreen3"       "seagreen4"       "seashell"
[580] "seashell1"       "seashell2"       "seashell3"
[583] "seashell4"       "sienna"          "sienna1"
[586] "sienna2"         "sienna3"         "sienna4"
[589] "skyblue"         "skyblue1"        "skyblue2"
[592] "skyblue3"        "skyblue4"        "slateblue"
[595] "slateblue1"      "slateblue2"      "slateblue3"
[598] "slateblue4"      "slategray"       "slategray1"
[601] "slategray2"      "slategray3"      "slategray4"
[604] "slategrey"       "snow"            "snow1"
[607] "snow2"           "snow3"           "snow4"
[610] "springgreen"     "springgreen1"    "springgreen2"
[613] "springgreen3"    "springgreen4"    "steelblue"
[616] "steelblue1"      "steelblue2"      "steelblue3"
[619] "steelblue4"      "tan"             "tan1"
[622] "tan2"            "tan3"            "tan4"
[625] "thistle"         "thistle1"        "thistle2"
[628] "thistle3"        "thistle4"        "tomato"
[631] "tomato1"         "tomato2"         "tomato3"
[634] "tomato4"         "turquoise"       "turquoise1"
[637] "turquoise2"      "turquoise3"      "turquoise4"
[640] "violet"          "violetred"       "violetred1"
[643] "violetred2"      "violetred3"      "violetred4"
[646] "wheat"           "wheat1"          "wheat2"
[649] "wheat3"          "wheat4"          "whitesmoke"
[652] "yellow"          "yellow1"         "yellow2"
[655] "yellow3"         "yellow4"         "yellowgreen"
```

A nice example of color usage in R is the rainbow() color palette used for plotting a pie chart:

```
pie(rep(1, 24), col = rainbow(24), radius = 0.9)
```



In addition to the rainbow colors, there are several other built-in color palettes in R:

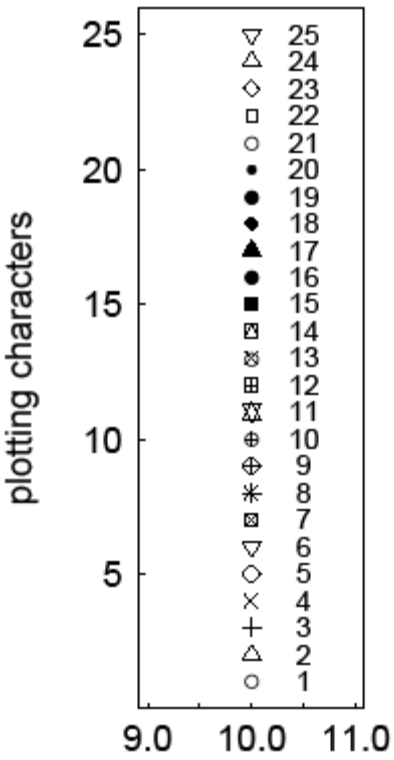


Just for background information, these palettes have been called using

```
demo.pal <-
  function(n, border = if (n<32) "light gray" else NA,
           main = paste("color palettes;  n=",n),
           ch.col = c("rainbow(n, start=.7, end=.1)",
"heat.colors(n)","terrain.colors(n)", "topo.colors(n)",
"cm.colors(n)"))
{
    nt <- length(ch.col)
    i <- 1:n; j <- n / nt; d <- j/6; dy <- 2*d
    plot(i,i+d, type="n", yaxt="n", ylab="", main=main)
    for (k in 1:nt) {
        rect(i-.5, (k-1)*j+ dy, i+.4, k*j,
             col = eval(parse(text=ch.col[k])), border =
border)
        text(2*j,  k * j +dy/4, ch.col[k])
    }
}
n <- if(.Device == "postscript") 64 else 16

demo.pal(n)
```

Plotting characters

Here is an overview of R´s default plotting characters.

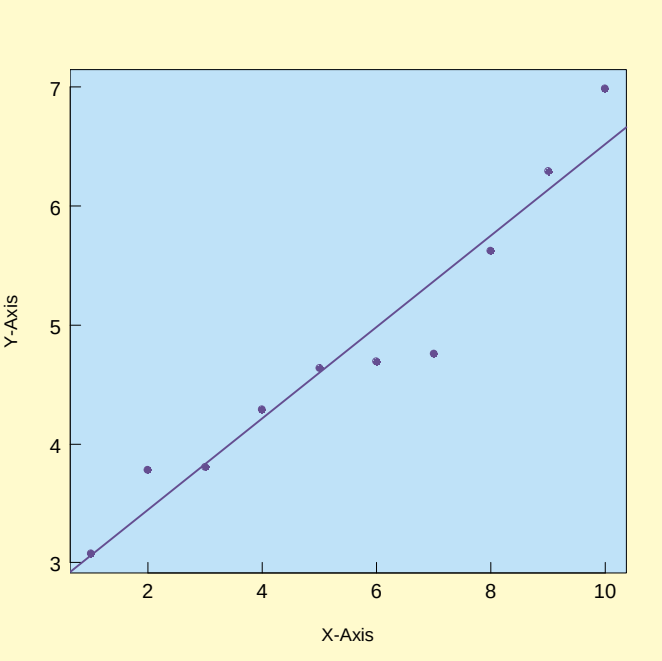


This figure has been created using the pdf() device like this:

```
pdf(file="C:\\characters.pdf",pointsize=14,width=3,height=7)

par(mgp=c(3,0.5,0),tck=0.02,cex.axis=1.3,cex.lab=1.3,lwd=1,las
=1)

plot(rep(10,25),1:25,pch=1:25,xlim=c(9,11),xlab="",ylab="plott
ing characters")
text(rep(10.5,25),1:25,1:25)
dev.off()
```

Let´s use some of the above-mentioned commands to make our original plot look fancier:

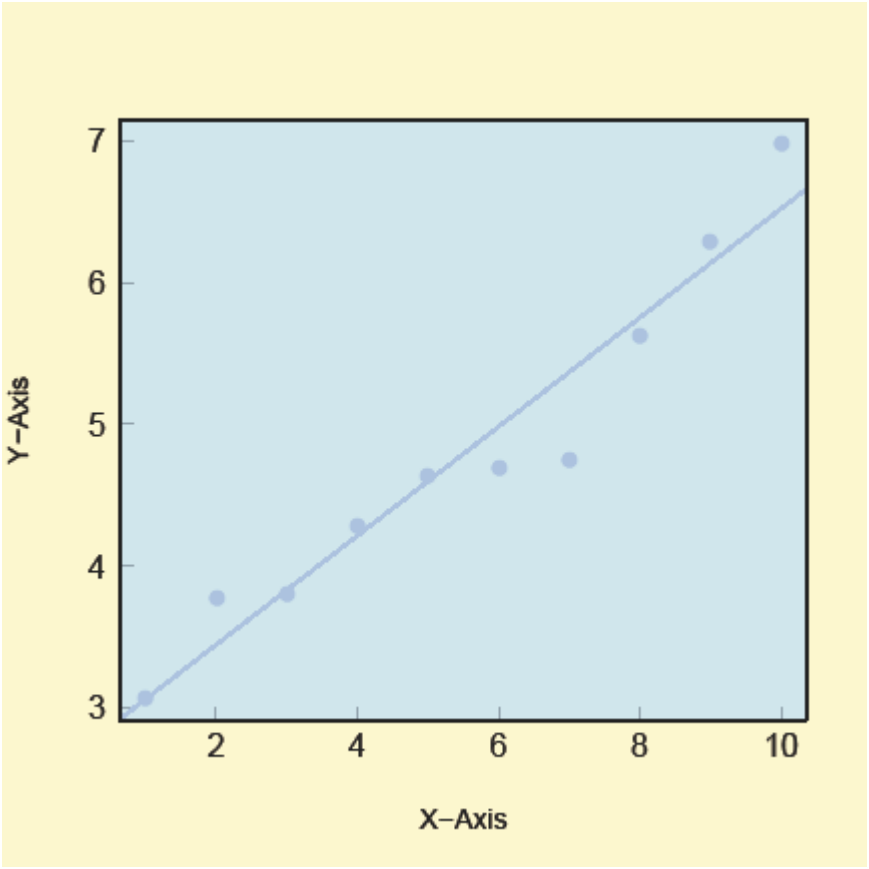
```
par(tck=0.02,las=1,cex.axis=1.3,cex.lab=1.2,mgp=c(3,0.5,0),
bty="o",bg="lemonchiffon")

plot(x,y,type="n",axes=F,xlab="",ylab="")
u <- par("usr")
rect(u[1],u[3],u[2],u[4],col=rgb(191,226,248,max=255))
points(x,y,pch=16,cex=1.3,col=rgb(102,78,145,max=255))
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-Axis")
abline(lm(y~x),lwd=2,col=rgb(102,78,145,max=255))
```



Here, colors are specified using **rgb()** values directly. **Transparency** of colored components of a plot can also be adjusted, but this is restricted to the **pdf()** device:

```
pdf("G:\\transparency.pdf",pointsize=12)
par(tck=0.02,las=1,cex.axis=1.3,cex.lab=1.2,mgp=c(3,0.5,0),
bty="o",bg="lemonchiffon")

plot(x,y,type="n",axes=F,xlab="",ylab="")
u <- par("usr")
points(x,y,pch=16,cex=1.3,col=rgb(102,78,145,max=255))
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-Axis")
abline(lm(y~x),lwd=2,col=rgb(102,78,145,max=255))
```

```
rect(u[1],u[3],u[2],u[4],
col=rgb(191/255,226/255,248/255,alpha=0.8),
border=F)
box(lwd=2)
dev.off()
```



Now what most of you will basically struggle with when creating R graphics are some of the basic questions such as:

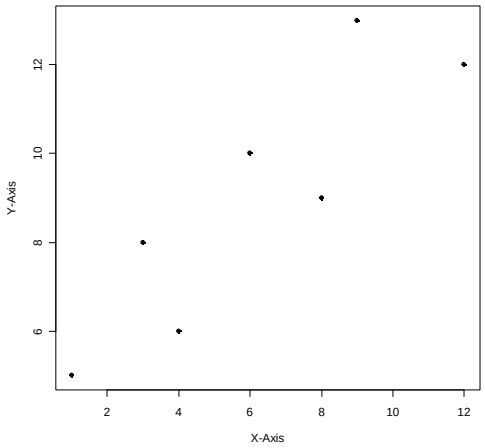
- How do I change the font size
- How do I make lines thicker or thinner, or change the line type
- How can I change plotting characters
- How do I change the distance between axis notations, axis title(s) and plot margins
- How do I control the size of the plot area?

All these questions are not difficult to answer, once you have carefully read through the par() help page.

Here are some solutions, again using our standard plot.

Here is the standard plot once more:

```
x<-c(1,3,4,6,8,9,12)
y<-c(5,8,6,10,9,13,12)
plot(x,y,type="n",axes=F,xlab
="",ylab="")
points(x,y,pch=16)
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-
Axis")
box()
```



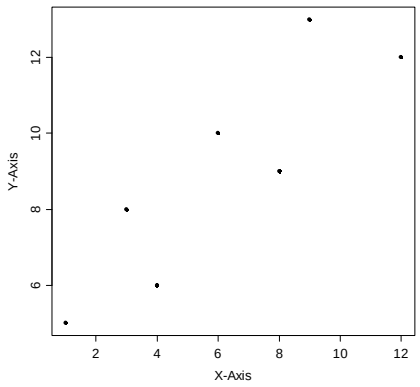
It has the basic look of any newly created R graphics object, so it is an ideal example of how to proceed when editing any R graph.

How do I change the font size?

Using **par()**, you can do this using the commands
cex.axis()
cex.lab() and
cex.sub()

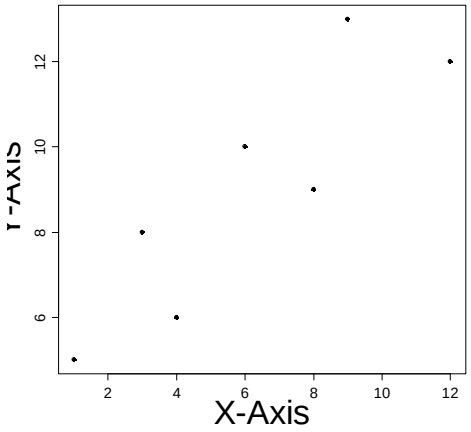
"**cex**" means "character expansion"; by default, the value is 1, so if we want to increase our font sizes by 1.3, we write **cex(1.3)**, or,
using our default graph again:

```
par(cex.axis=1.3,
cex.lab=1.3)
plot(x,y,type="n",axes=F,xlab
="",ylab="")
points(x,y,pch=16)
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-
Axis")
box()
```



Let´s see what happens if we exaggerate too much:

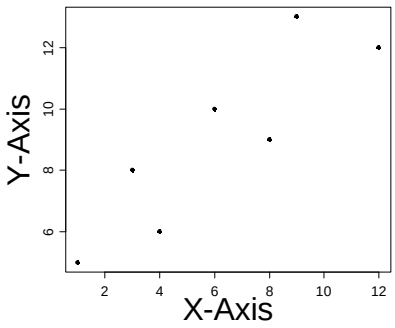
```
par(
cex.axis=1.3,
cex.lab=3)
plot(x,y,type="n",axes=F,xlab
="",ylab="")
points(x,y,pch=16)
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-
Axis")
box()
```



Now the y axis label has vanished, and we need to account for this by changing the size of the plotting device area. You would usually adjust this using the **mar()** command:
The standard values for **mar()** are **c(5, 4, 4, 2) + 0.1**, in the direction "bottom", "left","top", "right".

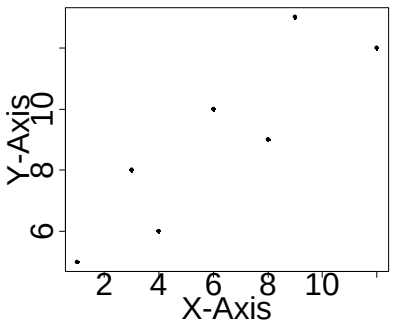
Let´s see what happens if we change these values:

```
par(mar=c(10,6,6,5),cex.axis=
1.3,cex.lab=3)
plot(x,y,type="n",axes=F,xlab
="",ylab="")
points(x,y,pch=16)
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-
Axis")
box()
```



This looks alright now; but what if we changed the cex.axis to 3, also?

```
par(mar=c(10,6,6,5),cex.axis=
3,cex.lab=3)
plot(x,y,type="n",axes=F,xlab
="",ylab="")
points(x,y,pch=16)
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-
Axis")
box()
```

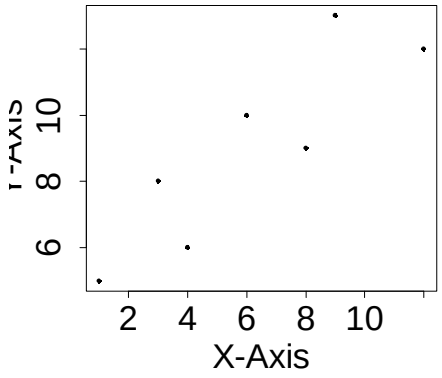


Well, now we need another command to change the distances between axis labels and axes. This is done using the **mgp()** command, which says: "How many lines of standard-sized text shall fit outside the plotting area?" Here, it is

(1) How many lines underneath the axis tick labels?
(2) How many lines between axis and tick label and
(3) How many lines shall the axis line itself be away from the plot?

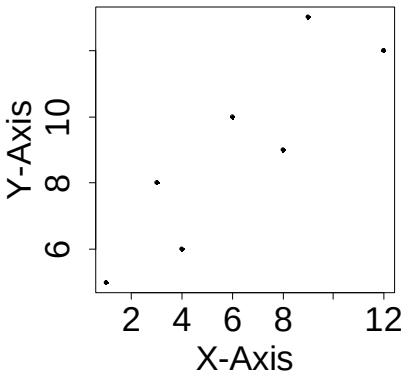
let us try the following:

```
par(mar=c(10,6,6,5),
mgp=c(5,2,0),
cex.axis=3,
cex.lab=3)
plot(x,y,type="n",axes=F,xlab
="",ylab="")
points(x,y,pch=16)
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-
Axis")
box()
```



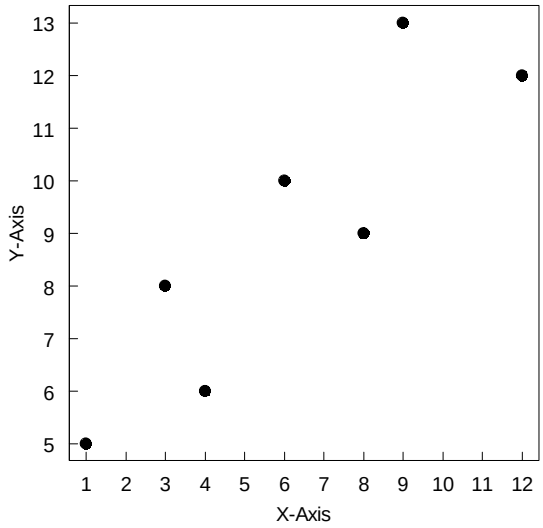
Now, again, we need to adjust the plotting area:

```
par(mar=c(10,10,6,5),
mgp=c(5,2,0),
cex.axis=3,
cex.lab=3)
plot(x,y,type="n",axes=F,xlab
="",ylab="")
points(x,y,pch=16)
axis(1)
axis(2)
title(xlab="X-Axis",ylab="Y-
Axis")
box()
```



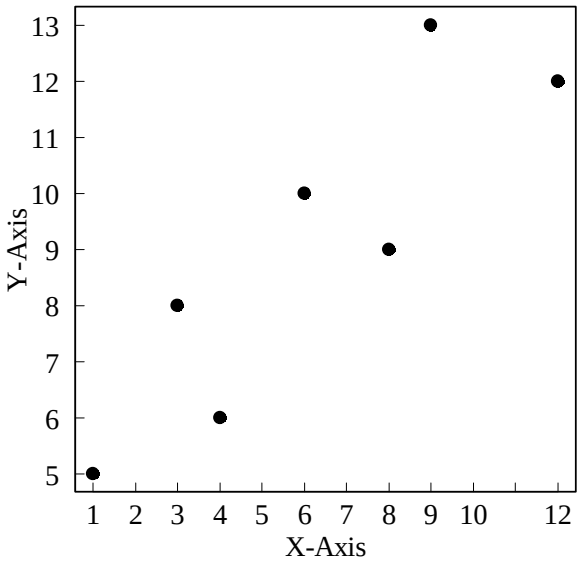
If you are not satisfied with the way the axis ticks are written, you can also change this; let us further change several other things such as the rotation of the tickmark labels using **las()**, the orientation of the tickmarks using **tck()**, and the sizes of the plotting characters using **cex()** inside the **points()** command:

```
par(mar=c(6,6,2,1),
mgp=c(3,1,0),
cex.axis=1.5,
cex.lab=1.5,
las=1, tck=0.02)
plot(x,y,type="n",axes=F,xlab
="",ylab="")
points(x,y,pch=16,cex=2)
axis(1,at=1:12)
axis(2,at=1:14)
title(xlab="X-Axis",ylab="Y-
Axis")
box()
```



Now, finally, let us change the fonts used for overall labelling (**family()** and **font()**), the overall font sizes onsce more using **cex.axis()** and **cex.lab()**, and the line widths using **lwd():**

```
par(mar=c(6,6,2,1),
mgp=c(3,1,0),
cex.axis=2,
cex.lab=2,
las=1,
font=2,
family="serif",
lwd=2,tck=0.02
)
plot(x,y,type="n",axes=F,xlab
="",ylab="")
points(x,y,pch=16,cex=2)
axis(1,at=1:12)
axis(2,at=1:14)
title(xlab="X-Axis",ylab="Y-
Axis")
box()
```



Another way of dealing with all these above-mentioned things quite conveniently is to use the **pdf** device and directly use the command:

```
pdf(file="C:\\sample.pdf",pointsize=12)
```

followed by additional **par()** commands and in the end a **dev.off()** command like written below:

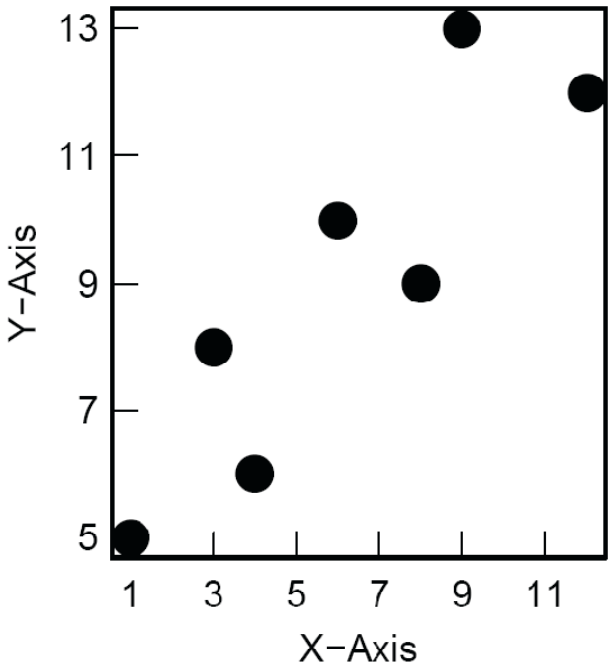
```
par(mar=c(6,6,2,1),mgp=c(3,1,0),las=1,lwd=2)
plot(x,y,type="n",axes=F,xlab="",ylab="")
points(x,y,pch=16,cex=2)
```

```
axis(1,at=1:12)
axis(2,at=1:14)
title(xlab="X-Axis",ylab="Y-Axis")
box()
dev.off()
```

where **pointsize()** gives the size of the font to be used.

Using the pdf device also gives you the option to specify the font family to be used, and the size you would like to have for your document; usually, it is also useful to state explicitly what character encoding you would wish to use. To create our following last figure, several other options have been adjusted, because the sizes of the figure were explicitly given to be 4 inches wide and 5 inches in height:

```
pdf("G:\\sample.pdf",width=4,
height=5,
pagecentre=T,pointsize=14,
fonts=c("Helvetica"),encoding
="WinAnsi")

par(las=1,lwd=2,tck=0.05,mgp=
c(1.5,0.3,0))
plot(x,y,type="n",axes=F,xlab
="",ylab="")
points(x,y,pch=16,cex=2)
axis(1,at=seq(1,12,2))
axis(2,at=seq(1,14,2))
title(xlab="X-Axis",ylab="Y-
Axis")
box()
dev.off()
```



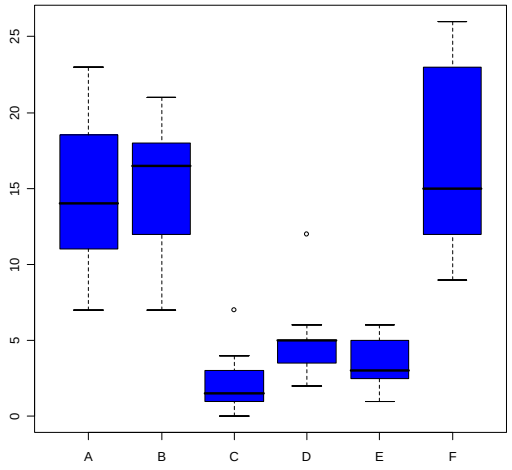
That´s all for now. Let´s now switch to other plot types.

Boxplots can be drawn using several options. One of the most interesting ones is notch=T, which draws notches that show confidence intervals for the medians:

```
par(tck=0.02,las=1,cex.axis=1.3,cex.lab=1.2,mgp=c(3,0.5,0),
bty="o",bg="lemonchiffon")
```

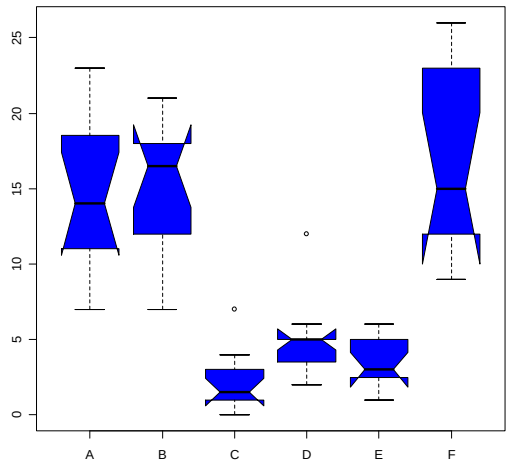
a) standard boxplot:

```
boxplot(count ~ spray, data = InsectSprays, notch = F,
col = "blue")
```



b) with notches:

```
boxplot(count ~ spray, data = InsectSprays, notch = TRUE,
col = "blue")
```



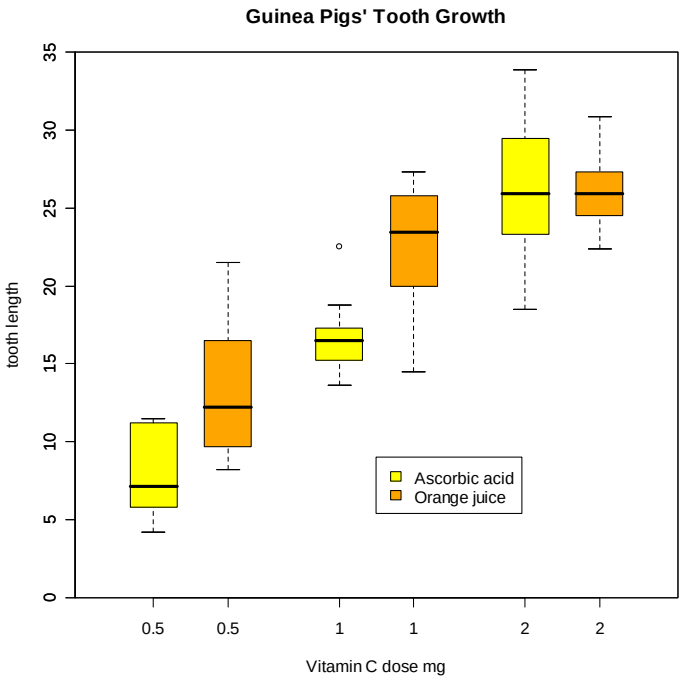
Another useful feature of boxplots is to have several boxes grouped at the x axis; this is something that most other stats packages will fail to produce:

```
boxplot(len ~ dose, data = ToothGrowth,
        boxwex = 0.25, at = 1:3 - 0.2,
        subset = supp == "VC", col = "yellow",
        main = "Guinea Pigs' Tooth Growth",
        xlab = "Vitamin C dose mg",
        ylab = "tooth length", ylim = c(0, 35), yaxs = "i")
boxplot(len ~ dose, data = ToothGrowth, add = TRUE,
        boxwex = 0.25, at = 1:3 + 0.2,
        subset = supp == "OJ", col = "orange")
legend(2, 9, c("Ascorbic acid", "Orange juice"),
       fill = c("yellow", "orange"))
```

The trick here is to use the

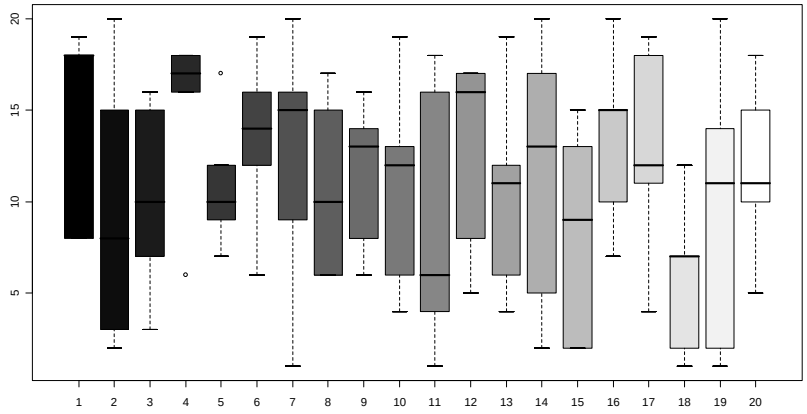
- "**add**" command, the
- "**at**" command and the
- "**subset**"

command when specifying boxplots.



Working with different shadings of grey can be shown using a quick example dataset:

```
boxplot(
     sample(1:20,100,replace=T)~
     sort(rep(1:20,5)),
col=gray(seq(from=0,to=1,length=20)))
```

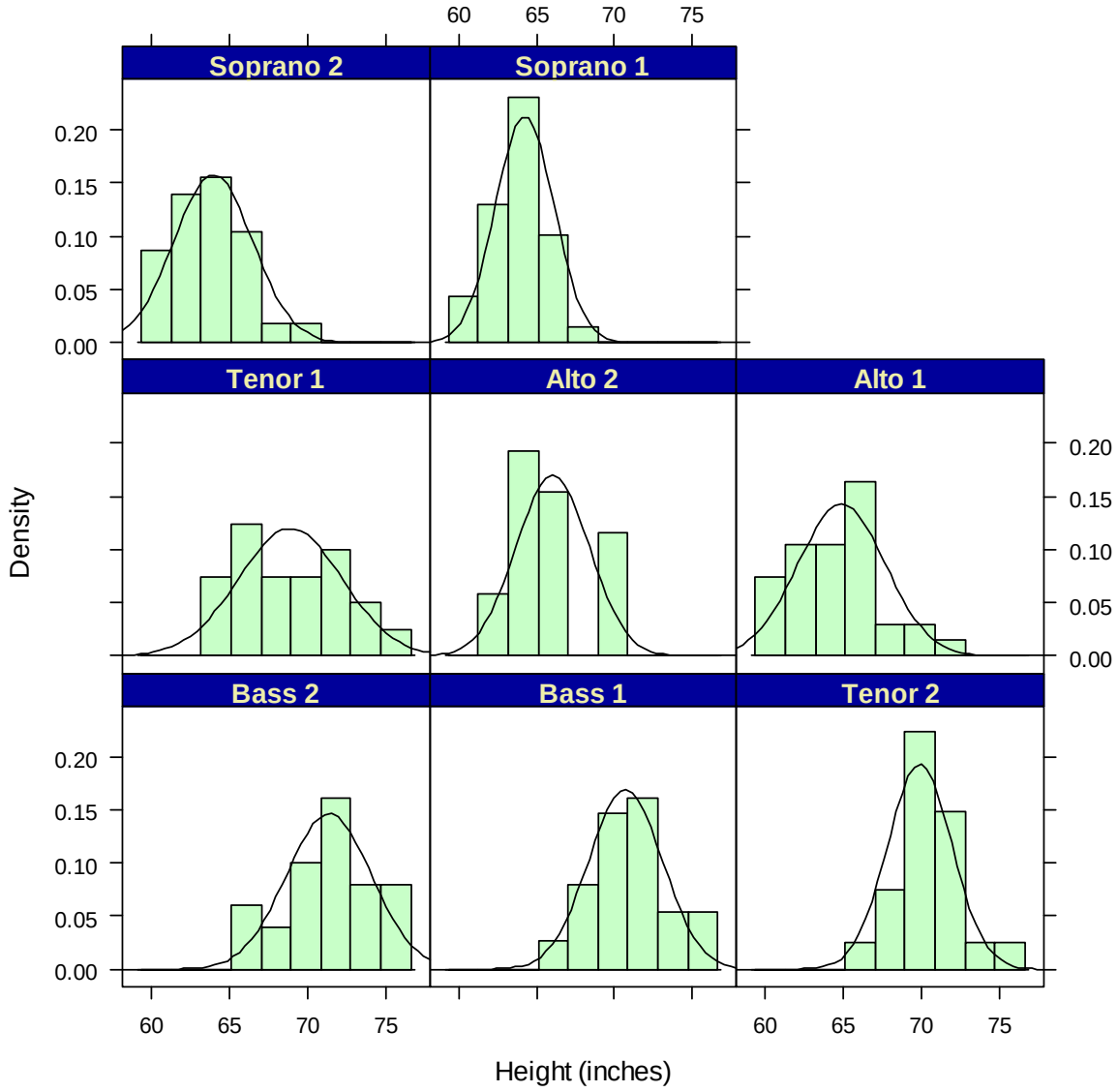


2. Trellis plots

Here are some introductory examples of what can be done using trellis plots:

```
     require(lattice)

     sundar.theme <- function() {
            par <- col.whitebg()
            par$strip.background$col <- rep("#000099", 7)
            par$add.text$col <- "#eeeeaa"
            par$add.text$font <- 2
            par$background$col <- "#ffffff"
            par$superpose.line$lty <- rep(1, 7)
            par$superpose.line$col[1:2] <- c("#880000",
     "#008800")
            par$superpose.symbol$col[1:2] <- c("#880000",
     "#008800")
            par
          }

     trellis.par.set(sundar.theme())

     print(  # necessary if the file is source()'d
```

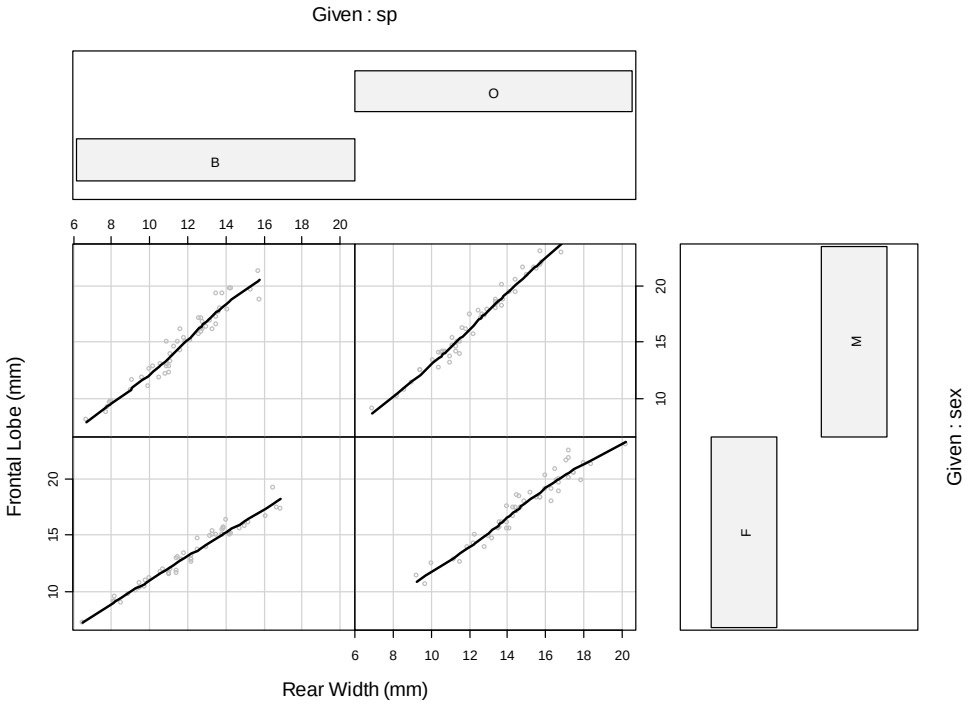
```
      histogram( ~ height | voice.part, data = singer,
              xlab = "Height (inches)", type = "density",
              panel = function(x, ...) {
                  panel.histogram(x, ...)
                  panel.mathdensity(dmath = dnorm, col =
    "black",
                                    args =
    list(mean=mean(x),sd=sd(x)))
              } )
 )
```



Conditioning plots (coplots):

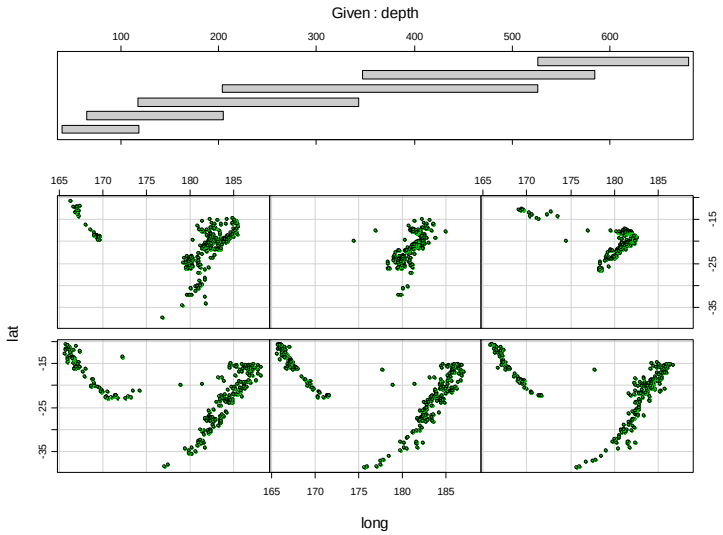
```
    library(MASS)

    with(crabs,
         coplot(FL ~ RW | sp * sex,
                subscripts = T,
```

```
                xlab = "Rear Width (mm)",
                ylab = "Frontal Lobe (mm)",
                panel = function(x, y, subscripts, ...){
                        points(x, y, col = "gray75")

lines(smooth.spline(RW[subscripts], FL[subscripts]), lwd
= 2,col = "black")
                }
        )
)
```



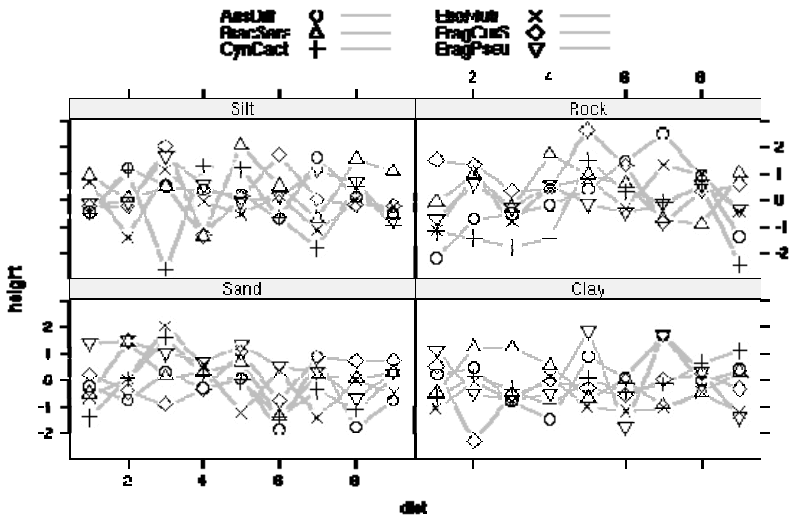
Coplot() can also be used with numerical conditioning variables:

```
coplot(lat ~ long | depth, data = quakes, pch = 21, bg =
"green3")
```



Here is another example showing the usage of xyplot():

```
library(lattice)

lattice.options(default.theme = canonical.theme(color =
FALSE))

tmp <- expand.grid(geology =
c("Sand","Clay","Silt","Rock"),species = c("ArisDiff",
"BracSera", "CynDact", "ElioMuti", "EragCurS",
"EragPseu"),dist = seq(1,9,1) )

tmp$height <- rnorm(216)

sp <- list(superpose.symbol = list(pch = 1:6, cex = 1.2),
           superpose.line = list(col = "grey", lty = 1))

print(xyplot(height ~ dist | geology, data = tmp,
       groups = species,
       layout = c(2,2),
       panel = function(x, y, type, ...) {
         panel.superpose(x, y, type="l", ...)
         lpoints(x, y, pch=16, col="white", cex=2)
         panel.superpose(x, y, type="p",...)
       },
       par.settings = sp,
  auto.key = list(columns = 2, lines = TRUE)))
```



3. Examples from special graphics packages

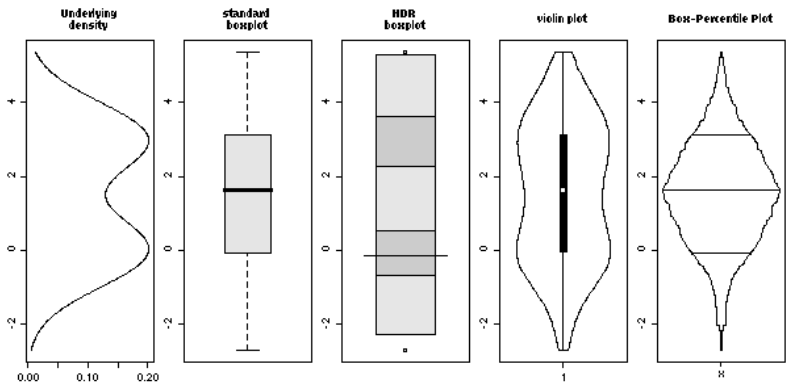
The examples below are from the "R Graph Gallery" maintained by Francois Romain at

http://addictedtor.free.fr/graphiques/

The Hmisc package

To create the following graph (showing variations of boxplots), several libraries will need to be downloaded and installed:

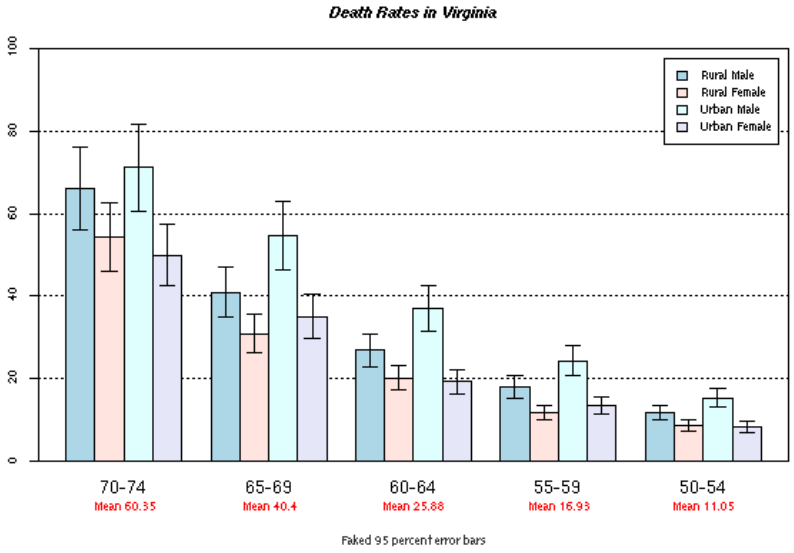
```
require(hdrcde)
require(vioplot)
require(Hmisc)

x <- c(rnorm(100,0,1), rnorm(100,3,1))
opar <- par(mfrow=c(1,5), mar=c(3,2,4,1))

xxx <- seq(min(x), max(x), length=500)
yyy <- dnorm(xxx)/2 + dnorm(xxx, mean=3)/2

plot(yyy, xxx, type="l",main="Underlying\ndensity")

boxplot(x, col="gray90", main="standard\nboxplot")
hdr.boxplot(x, main="HDR\nboxplot")
vioplot(x)
title("violin plot")
bpplot(x)
par(opar)
```



gplots package

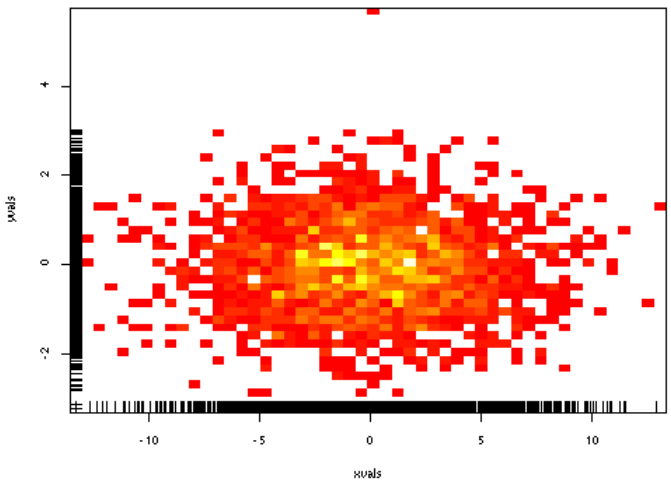
This package is a fusion of conventional and lattice graphics. It contains very useful features, but some are still in development.

```
library(gplots)
# Example with confidence intervals and grid
hh <- t(VADeaths)[, 5:1]
mybarcol <- "gray20"
ci.l <- hh * 0.85
ci.u <- hh * 1.15
mp <- barplot2(hh, beside = TRUE,
        col = c("lightblue", "mistyrose",
                "lightcyan", "lavender"),
        legend = colnames(VADeaths), ylim = c(0, 100),
        main = "Death Rates in Virginia", font.main = 4,
        sub = "Faked 95 percent error bars", col.sub =
mybarcol,
        cex.names = 1.5, plot.ci = TRUE, ci.l = ci.l, ci.u =
ci.u,
        plot.grid = TRUE)
mtext(side = 1, at = colMeans(mp), line = 2,
      text = paste("Mean", formatC(colMeans(hh))), col =
"red")
box()
```



Similarly, plotting a two-dimensional histogram is also possible using the gplot library:

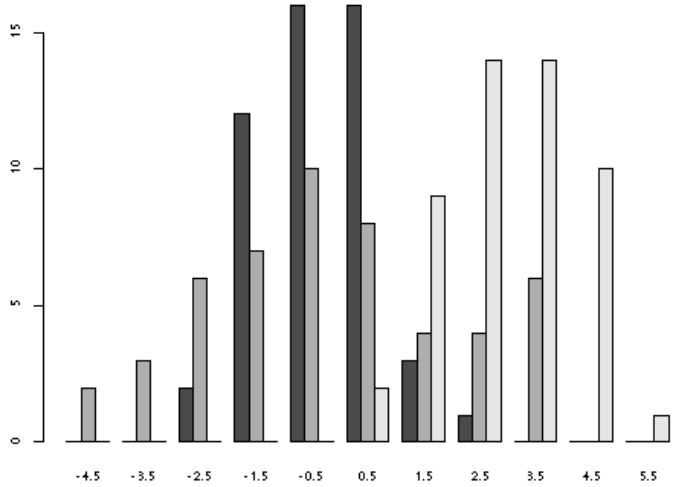
```
require(gplots)

# example data, bivariate normal, no correlation
x <- rnorm(2000, sd=4)
y <- rnorm(2000, sd=1)

# separate scales for each axis, this looks circular
hist2d(x,y, nbins=50, col = c("white",heat.colors(16)))
rug(x,side=1)
rug(y,side=2)
box()
```



plotrix package

The following short command allows the drawing of a double histogram side-by-side; the only thing required is to have the different datapoints concatenated in a list with as many sub-lists as there shall be histograms:

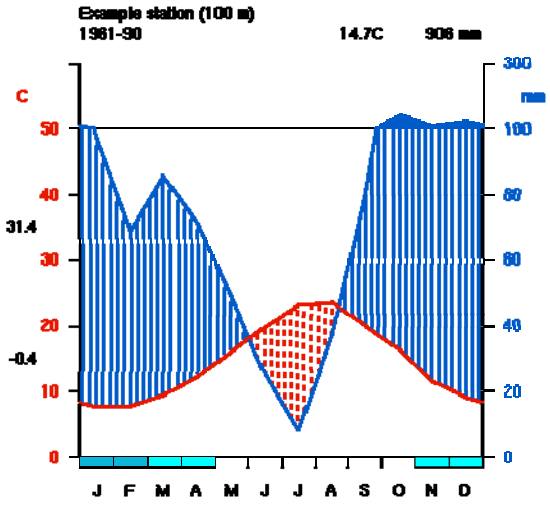
```
require(plotrix)

l <- list(rnorm(50),rnorm(50,sd=2),rnorm(50,mean=3))
multhist(l)
```



Climatol package

This package allows the drawing of classical climate diagrams.

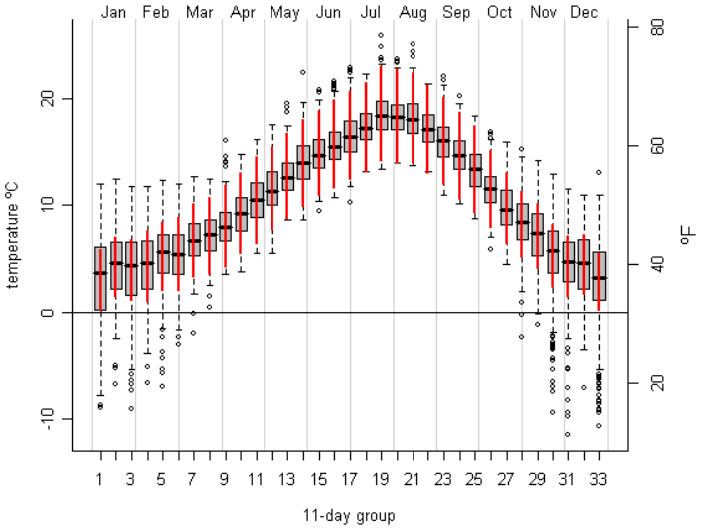
```
require(climatol)

  data(cli.dat)
  diagwl(cli.dat,est="Example station",alt=100,per="1961-
90",mlab="en")
```



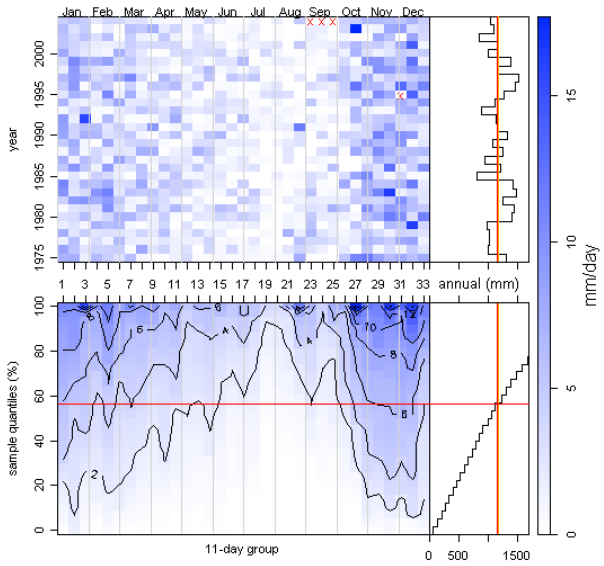
Seas package

This package provides plotting options for seasonal data.

```
require(seas)
data(mscdata)
par(cex=0.8)
plot.seas.temp(mscdata,id="1108447",add.alt=TRUE,style=c(0,1))
```

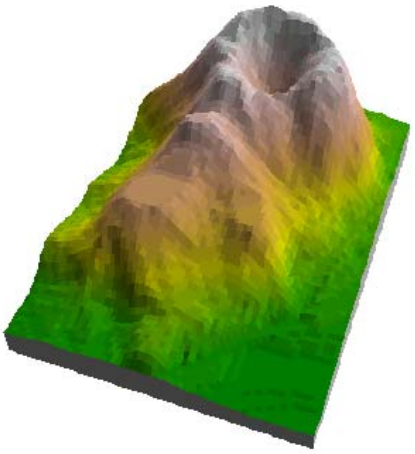



```
require(seas)
data(mscdata)
par(cex=0.8)
image(seas.sum(mscdata,id="1108447"),style=c(0,1))
```



Lastly, here are is a 3D surface plot that have also been plotted using R:

```
z <- 2 * volcano        # Exaggerate the relief
x <- 10 * (1:nrow(z))   # 10 meter spacing (S to N)
y <- 10 * (1:ncol(z))   # 10 meter spacing (E to W)

z0 <- min(z) - 20
z <- rbind(z0, cbind(z0, z, z0), z0)
x <- c(min(x) - 1e-10, x, max(x) + 1e-10)
y <- c(min(y) - 1e-10, y, max(y) + 1e-10)

fill <- matrix("green3", nr = nrow(z)-1, nc = ncol(z)-1)
fill[ , i2 <- c(1,ncol(fill))] <- "gray"
fill[i1 <- c(1,nrow(fill)) , ] <- "gray"

fcol <- fill
zi <- volcano[ -1,-1] + volcano[ -1,-61] +
      volcano[-87,-1] + volcano[-87,-61]  ## / 4
fcol[-i1,-i2] <-
    terrain.colors(20)[cut(zi, quantile(zi, seq(0,1, len =
     21)), include.lowest = TRUE)]
par(mar=rep(.5,4))
persp(x, y, 2*z, theta = 110, phi = 40, col = fcol, scale =
FALSE, ltheta = -120, shade = 0.4, border = NA, box = FALSE)
```



Index

R commands and topics covered in this manual

abline 5, 20
adj 6
ann 6
arrows 3, 4
ask 6
auto.key 32
axis 3, 4, 6, 7, 8, 10, 11, 12, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 35
barplot2 34
bg 6, 20, 27, 31
border 18, 21, 38
box 3, 4, 6, 12, 21, 22, 23, 24, 25, 26, 34, 35, 38
boxplot 27, 28, 29, 33
Boxplots 27
bpplot 33
bty 3, 4, 6, 20, 27
canonical.theme 32
cex 3, 4, 6, 7, 10, 19, 20, 22, 23, 24, 25, 26, 27, 32, 34, 37
cex.axis 3
cex.lab 3, 4, 6, 19, 20, 22, 23, 24, 25, 27
climatol 36
Climatol 36
col 4, 7, 17, 18, 20, 21, 27, 28, 29, 30, 31, 32, 33, 34, 35, 38
colMeans 34
Color Specification 13
colors 6, 7, 13, 17, 18, 20, 35, 38
Colors 13
Conditioning 30
coplot 30, 31
coplots 30
crt 7, 11
dev.off 19, 21, 25, 26
diagwl 36
dnorm 30, 33
family 7, 8, 25, 26
fg 7
fig 7
fill 28, 38
fin 7
font 7, 8, 9, 21, 22, 25, 26, 29, 34
function 13, 18, 29, 30, 31, 32
gamma 8
gplots 34, 35
graphics parameters 5, 6
hdrcde 33
hist2d 35
histogram 30, 35, 36
Hmisc 33
image 37
lab 7, 8, 12
las 3, 4, 8, 19, 20, 24, 25, 26, 27
lattice 6, 29, 32, 34
legend 28, 34
lend 8
lheight 8
library 30, 32, 34, 35
Line Type Specification 9, 13
lines 3, 7, 9, 10, 11, 13, 21, 24, 31, 32
ljoin 8
lm 5, 20
lmitre 9
lty 4, 9, 13, 29, 32
lwd 9, 13, 19, 20, 21, 25, 26, 31
mai 9, 11
mar 9, 11, 23, 24, 25, 33, 38
mathdensity 30
matrix 38
mean 8, 9, 11, 30, 33, 36
mex 9, 10
mfcol 10
mfg 10
mfrow 10, 33
mgp 3, 4, 10, 19, 20, 24, 25, 26, 27
min 33, 38
mkh 10
multhist 36
ncol 38
new 6, 7, 10, 12
notch 27
oma 9, 10, 11
omd 10
omi 9, 10, 11
opar 33
options 26, 27, 32, 37
panel 30, 31, 32

par 3, 4, 5, 6, 7, 8, 11, 12, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29, 32, 33, 37, 38
pch 3, 4, 10, 19, 20, 22, 23, 24, 25, 26, 31, 32
pdf 13, 19, 20, 25, 26
persp 38
pie 17
pin 10
plot 3
plot(x,y) 3
plot.ci 34
plot.grid 34
plotrix 36
plotting characters 18, 19, 21, 24
Plotting characters 18
plt 10
points 3, 4, 6, 10, 13, 20, 22, 23, 24, 25, 26, 31
pointsize 11, 19, 20, 25, 26
polygon 4
ps 11
pty 11
rainbow 13, 17, 18
rbind 38
rect 18, 20, 21
rep 17, 19, 29, 38
rgb 13, 20, 21
rnorm 32, 33, 35, 36
sample 25, 26, 29
sd 30, 35, 36
seas 37
Seas package 37
seq 26, 29, 32, 33, 38
smooth.spline 31
sort 29
srt 7, 8, 11
standard plot 21
subscripts 30, 31
subset 28
superpose 29, 32
tck 3, 4, 11, 19, 20, 24, 25, 26, 27
tcl 11
terrain.colors 18
theme 29, 32
title 3, 4, 10, 20, 21, 22, 23, 24, 25, 26, 33
Trellis 29
usr 4, 11, 12, 20
vioplot 33
volcano 38
xaxp 8, 11, 12
xaxs 12
xaxt 12
xlab 3, 4, 19, 20, 22, 23, 24, 25, 26, 28, 30, 31
xlog 11, 12, 13
xpd 4, 12
yaxp 8, 12
yaxs 12, 28
yaxt 12, 18
ylab 3, 4, 18, 19, 20, 22, 23, 24, 25, 26, 28, 31
ylog 12