4. Graphs

Why graphs?

Graphs represent a crucial tool in statistical analysis**. They are used for exploratory data analysis, parameter comparisons between samples, and illustrations of associations between variables.** A number of graph types exist, which are suitable for different purposes. They are very useful for communication, including also data/analysis result presentation. Most people simply prefer seeing a graph to studying numbers presented in a table. Producing nice graphs is thus an important part of presentation of scientific results[1]. There are no universal rules how a nice graph should look like, but the good thing is that the quality of your graphs will quickly improve with practice and experience. Getting inspired by graphical presentations of other researchers is also very helpful.

An important aspect of the graphs is that they cannot display all the information contained in the raw data. An ideal graph should minimize this loss of information while efficiently depicting the patterns of interest. These requirements are however often in conflict. A reasonable solution often lies in providing the reader both the graph and the raw data (attached as supplementary material or deposited in a public repository such as Dryad: https://datadryad.org/stash). Many scientific journals nowadays require disclosure of the original data anyway, which is important for checking the integrity of the analyses presented. **By contrast, presenting the same descriptive statistics in both table and graph format is generally considered superfluous and should be avoided.**

Basic graph types

Table 4.1 Summary of basic graph types, their advantages and limitations

Graph type	Number of variables*	Preservation of information	Display of sample parameters	Visualization of dependence
Histogram	1	++	--	--
Boxplot	1 quantitative + 1 categorical	+	-	+
Barplot	1 quantitative + 1 categorical	--	+	++
Dotchart	1 quantitative + 1 categorical	--	++	++
Scatterplot	2 quantitative	++	-	++

++ excellent, + good, - (still) adequate, -- poor

* Refers to a minimum (typical) number. May be increased e.g., by combining multiple categorical predictors, or categorization of point in a scatterplot.

[1] Note here, that most readers of scientific papers only read the abstract and then look at the figures; and all of them do this before deciding whether the paper is worth of further reading. This applies also for journal editors and submitted manuscript. Figure quality and attractiveness may thus have a decisive effect on the editor's decision on publication.

Histogram

Histogram was already introduced in chapter 2. Construction of histograms is done in two steps. The range of the values is first divided into a number of intervals. These are plotted on the x-axis. Individual values are then assigned into them and the resulting frequencies of observations are plotted on the y-axis. Thus, histograms display the data with only minimal loss of information. They are a perfect tool for exploration of data distribution.

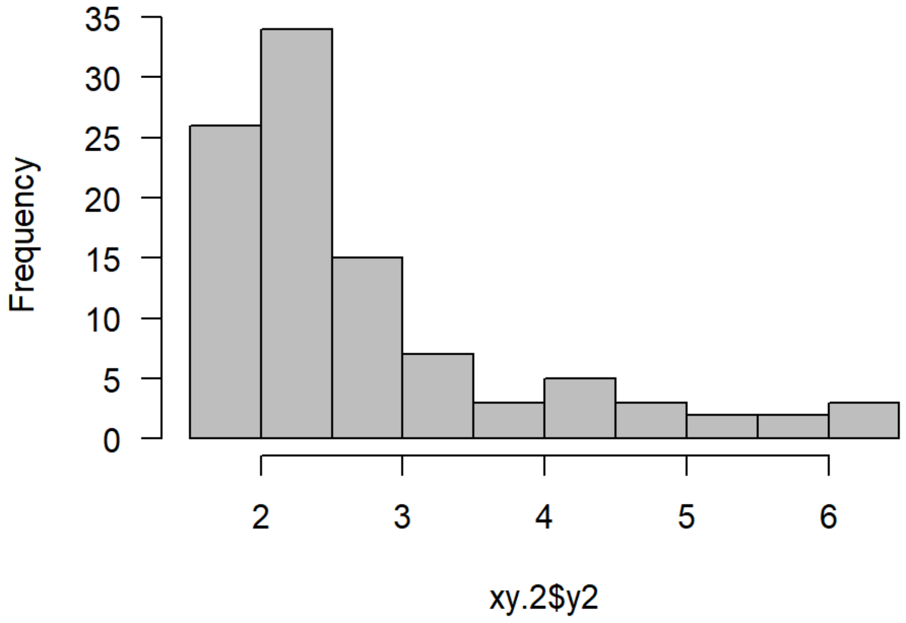


Fig. 4.1. Histogram of the variable xy.2$y2

`How to do in R`

`function **hist** applied on the variable to be plotted produces the histogram`

`Useful tip: Parameter col="grey" makes the histogram more readable/elegant; other colors may also be used.`

Boxplot

Boxplot was also introduced in chapter 2. Boxplots display summary of descriptive statistics of samples: the median, quartiles, non-outlier range and outliers. Typically, they are used to study association between a categorical (factor) and a quantitative (numeric) variable, where they display differences between individual categories (levels). **Boxplots do not display means**, so it is not possible to use them for direct mean comparisons. However, crucial characteristics of the distributions are visible on the plots: variability, symmetry, presence of outliers. This makes boxplots an important tool of exploratory data analysis

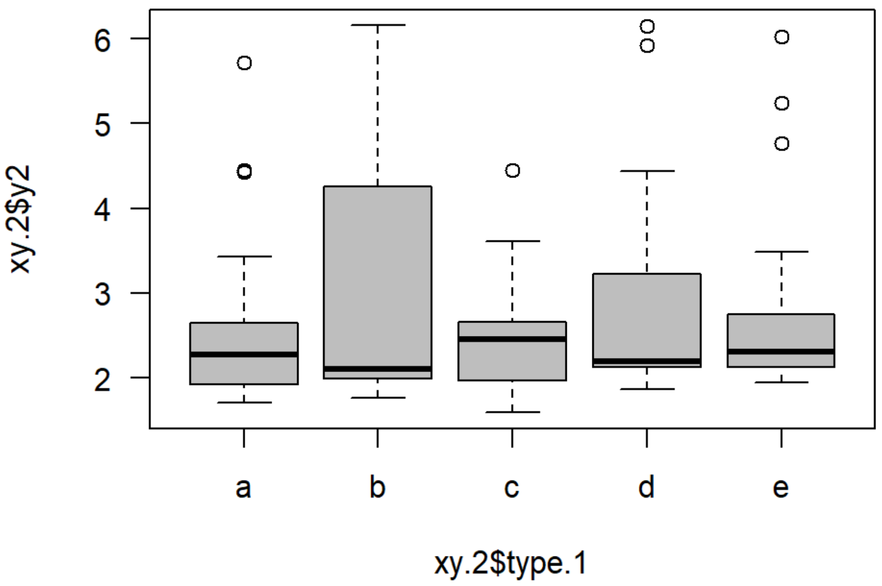


Fig 4.2. Boxplot displaying the values of the variable y2 for individual categories of type.1. Note the non-symmetric distributions and the outliers.

`How to do in R`

`function` **`boxplot`** `applied on formula numeric~factor produces the boxplot`

`Useful tips: Parameter col="grey" makes the boxplot more readable/elegant; other colors may be used. The midpoints of boxes of boxplots are located at integer numbers on the x-axis (unless changed by parameters).`

Modern alternatives to boxplots

Boxplots have many advantages, which makes them a standard plot type for displaying associations between a categorical and quantitative variable. However, there are also some issues. One obvious is that they do not display the mean values. In addition, they may provide misleading results if the underlying distribution is e.g. bimodal. For these reasons, alternative types were developed called Bean plot and Violin plot. While useful, their use is still rather limited in biological community. For more information, see e.g. https://cran.r-project.org/web/packages/beanplot/vignettes/beanplot.pdf .

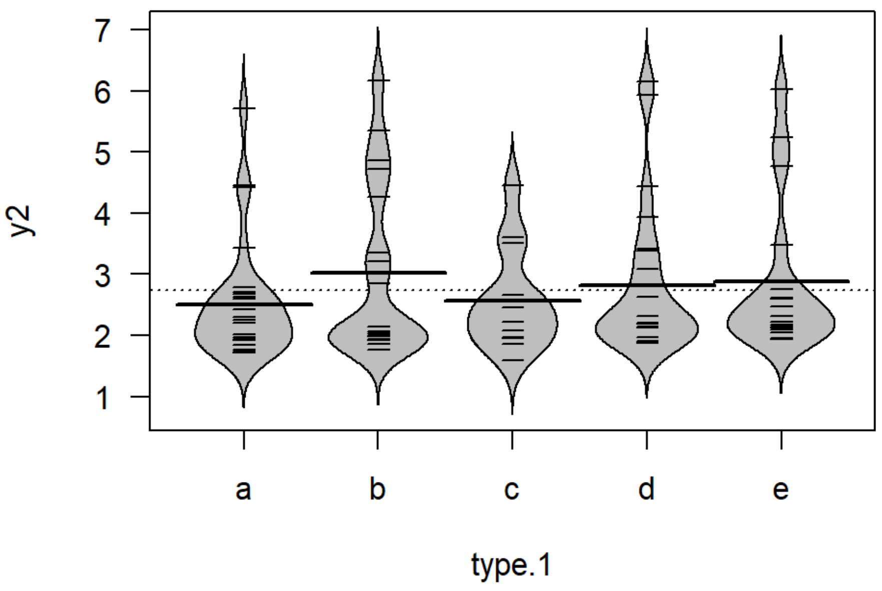


Fig 4.3. Beanplot displaying the values and densities of the variable y2 for individual categories of type.1. Dotted line, bold lines and short narrow lines indicate global mean, group means and individual observations respectively.

Barplot

Barplots mostly display means of quantitative variables, in particular differences between means of individual categories (factor levels). To judge on difference between means, it is necessary to display also a characteristic of uncertainty of mean estimate or variability. Therefore, barplots are usually supplied by error bars displaying standard errors, confidence intervals or standard deviation. Of these, the general best choice is probably the confidence intervals, which indicates the range of values within which the population mean lies with 95% probability (more on that in chapter 7). In any case, **specification of error bars (what they display) must always be included in graph caption**. The strong aspect of barplots is that they allow judging on difference between means. However, this comes with substantial loss of information: barplots do not display the distribution at all and may even be misleading in this respect. The y-axis range in barplots should always start at 0. Displaying negative values (or combination of negative and positive values) may be awkward.

Barplots may also be used to display counts, where they display the raw data without any loss of information.

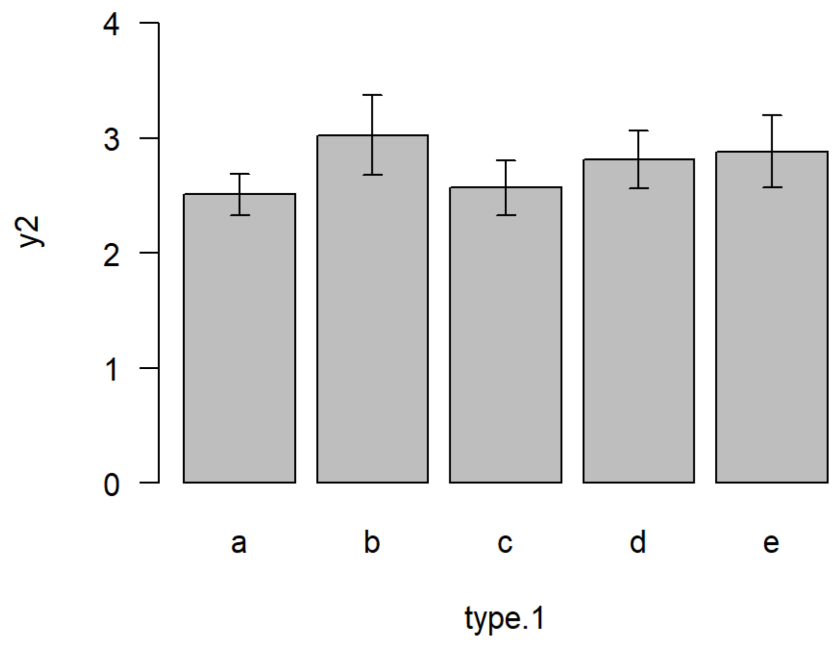


Fig. 4.3. Barplot displaying mean values of variable y2 for individual categories of type.1. Error bars indicate 1 standard error.

`How to do in R`

function **barplot** requires a numeric vector of bar height (i.e. means)

Errorbars are supplied by function **arrows**

```
arrows(x0=x.coords, y0=means-err.b, y1= means+err.b, code=3,
length=0.05)
```

where err.b is the errorbar parameter (standard error or confidence interval). May also be range but then, the interval is not symmetric.

The midpoints of bars are **not** located at integer numbers on the x-axis. To get their coordinates, you need to save the output of barplot in a vector, like:

`x.coords<-barplot(x);` this plots the barplot and saves the midpoints in the x.coords vector.

Useful tips: The barplot function does not allow the formula input and data parameter.

Parameter col=”grey” makes the boxplot more readable/elegant; other colors may be used.

R does not have a dedicated function for plotting a barplot with errorbars. I have made one for you: see barplotN.R in IS (Study Materials/Learning Materials/Rfunctions). **The barplotN function also implements automatic calculation of means; just the classifying factor and the numeric variable need to be supplied.** The type of error bars is specified by parameters.

Dotchart

Dotcharts are closely similar to barcharts. They are also very suitable to comparisons between means. They however better display negative values and allow adjustment of y-axis range. Thus they are considered generally superior options to barplots.

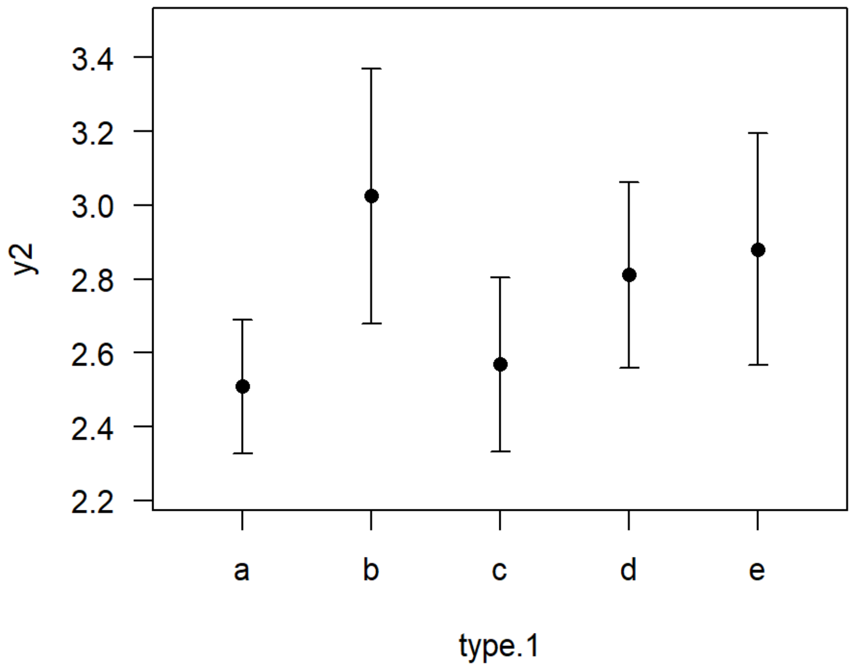


Fig. 4.4. Barplot displaying mean values of variable y2 for individual categories of type.1. Error bars indicate 1 standard error.

How to do in R

function **dotchart** requires a numeric vector of means. Using this function is however quite awkward. A better option is to use simple plot

plot(1:N, means) where N is the number of categories and means is the vector containing the means. Errorbars are supplied by function **arrows**

```
arrows(x0=1:N, y0=means-err.b, y1= means+err.b, code=3,
length=0.05)
```

where err.b is the errorbar parameter (standard error or confidence interval). May also be range but then, the interval is not symmetric.

Useful tip:

You may change the point symbol of the mean by parameter pch

pch = 16 creates a filled point
pch = 15 makes a filled box

Because R does not have a dedicated function for plotting a dotchart with errorbars, I have made one for you: see dotchartN.R in IS (Study Materials/Learning Materials/Rfunctions). **The dotchartN function also implements automatic calculation of means; just the classifying factor and the numeric variable need to be supplied.** The type of error bars is specified by parameters.

Scatterplot

Scatterplot is a simple point-based plot illustrating the association between two quantitative variables. The point sin scatterplot usually represent original data, thus there is little loss of information, if any. Scatterplot is perfect for exploration of interdependence between two variables. Regression line (with confidence) intervals may also be added to the raw scatterplot to visualize a regression model (see chapter 10 for details).

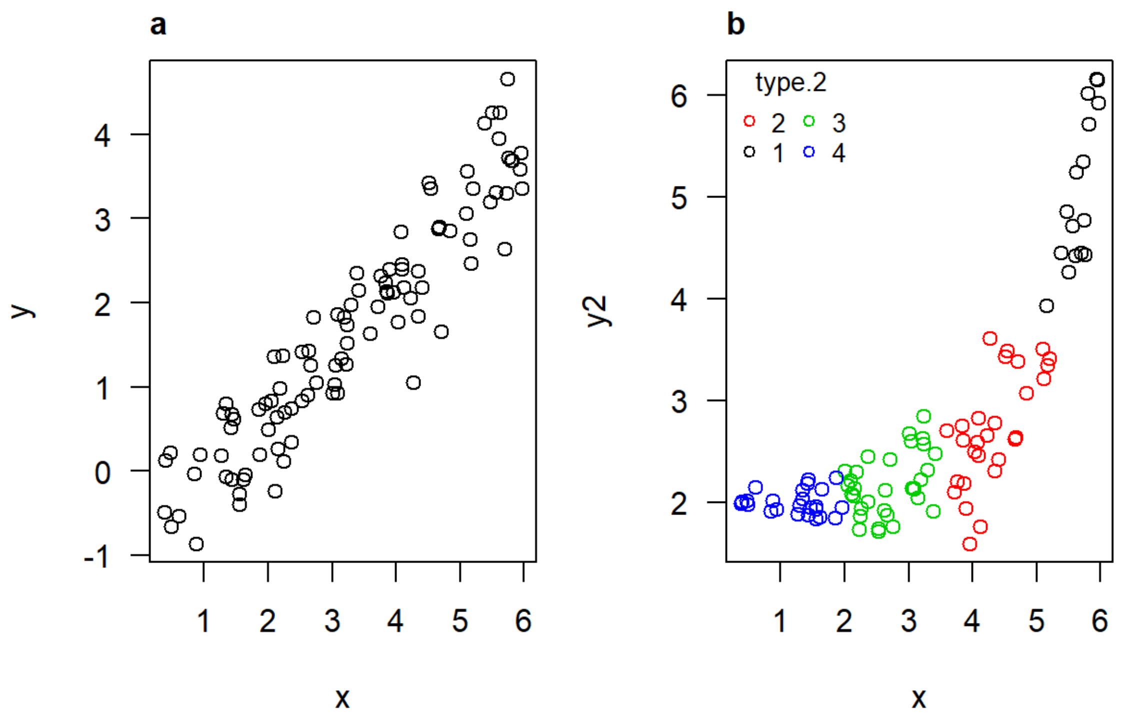


Fig 4.4. Scatterplots displaying the relationships between x and y (a) and x and y2 with indication of assignment into categories of type.2 (b). Note the different types of relationships between variables: linear (a) and exponential (b).

How to do in R

scatterplot is produced by function **plot**(x, y) if both x and y are numeric variables. Alternatively, plot also accepts the formula and data parameters.

Useful tips: With large data, or data, where values are limited to few integers, overlapping points may occur in a scatterplot, which are not visible. A solution to this is to use semitransparent point color. The number overlapping points is then indicated by color intensity. Semitransparent color is specified by parameter alpha in color-specifying function rgb, e.g.

col=rgb(0.2,0.2,0.2,alpha=0.5) produces semitransparent grey.

Function **scatterplot** of **car** package provides many additional functionalities for enhanced scatterplots like adding a regression line, possibility to draw categorized scatterplots (i.e. with different types of points), etc. However, its default settings is not very nice.

My friend Pavel Fibich has also scripted a scatterplot function which plots a scatterplot together with regression line and its confidence intervals. It is called **lmconf** and is available in IS (Study Materials/Learning Materials/Rfunctions).

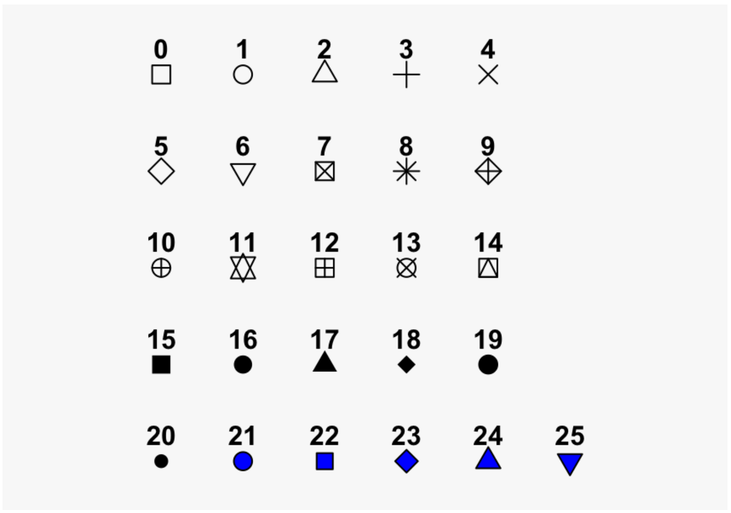
General tips for graph creation/adjustments in R

1. **Exporting graphs** is best done by saving them as separate files. These files may be raster or vector graphics (see e.g. here for explanation https://vector-conversions.com/vectorizing/raster_vs_vector.html)
   a. vectors: functions **pdf**, **svg** (svg can easily be post-processed in InkScape https://inkscape.org/).
   b. rasters: functions **png**, **jpg**
   c. general syntax is e.g.
      pdf("file.name.pdf", width.in.inches, height.in.inches)# In rasters, the width and height are specified in pixels. In addition, raster resolution (in dpi) can also be specified.
      plot(x,y)
      dev.off() # Closes the file and saves it to disk.
2. **Graphical parameters** are set by function **par**
   a. **?par** provides info on all graphical parameters used also in other functions like plot
   b. Parameter setting done by par affects the plot, which is produced afterwards. e.g.

par(mar=c(2,2,2,2)) sets all plot margins to 2 text lines. A graph produced by plot afterwards will have such margins.

c. most important parameters used directly in par:
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.
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.

d. most important graphical parameters used mostly in other functions (like **plot**)
xlim: the x limits (x1, x2) of the plot. The same with **ylim** for y-axis
xlab, ylab: axis labels
main: graph headline
cex: A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. This starts as 1 when a device is opened, and is reset when the layout is changed.
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.
pch: plotting 'character', i.e., symbol to use. This can either be a single character or an integer code for one of a this set of graphics symbols.



lty: line type (2 for dashed)
lwd: line width

log: produces log-scaled axis; use log=’x’, log=’y’ or log=’xy’ for horizontal, vertical or both axes respectively.

3. **Other useful functions**
   a. **legend:** this function adds legend to an existing plot. It is very useful e.g. in scatterplots with multiple point types.
   b. **text:** adds a text to a specified place within the plotting region; **mtext** adds text onto graph margins