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 (for counts)	1 or 2 categorical	+	--	+
Dotchart (with errorbars)	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.

Graph plotting in R

You may take several ways how to plot graphs in R. Two most common include the **R base graphics** and **the package ggplot2**. These approaches have their advantages and disadvantages. The R base graphics uses the same script grammar as the rest of R. Thus, you do not need to study another specialized package. However, plotting complicated plots may require quite a lot of programming. Producing graphically nice outputs may also require adjustments of many parameters in some (albeit not all) graph types. The ggplot2 package uses its own script grammar, which is quite different from R base. This means that you need to study another programming language. However, with this language you can easily plot complicated graphs with just two lines of code (instead of 20 in base graphics). In this material, I take things pragmatically. Generally, ggplot is in the focus as a modern tool of graph plotting. However, if it easier to to produce a particular graph type with base graphics, I choose that way.

The ggplot grammar

Each definition of a ggplot graph starts with the ggplot function, which defines the data (i.e. data frame where to look for variables) and so called aesthetics mappings, which are definition of variables used for plotting.

For a data frame called df with variables x and y used to define a scatterplot, it is:

ggplot2(data, mapping=aes(x=x, y=y))

The scatterplot is then plotted by a geom function:

geom_point()

The ggplot2 definition, geoms and possible further functions are separated by ”+”. In our case, it would be: ggplot2(data, aes(x=x, y=y))+geom_point()

Other elements or arrangements of plot are specified by additional functions added with another “+”. The most essential among these are:

- theme() – setting visual attributes; There are preset graphical themes. I really prefer theme_classic() or theme_bw() to the default theme_gray().
- xlab(), ylab(), or labs() – specification of axis label text
- facet_wrap() – faceting graphs, i.e. defining multipanel plots with panels based on a variable
- grid.arrange() – creating general multipanel plots (package gridExtra)
- ggsave(file.name, width, height) – saves the most recent plot to hard drive (height and width are specified in cm). File type (pdf, svg, png) is automatically set by file extension.

For ggplot2, there is abundant reference available, such as a free online book (https://ggplot2-book.org/index.html ; printed version for $$) and ggplot2 cheat sheet (https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-visualization.pdf) . You can also find valuable reference/guidelines by Google searches such as “scatterplot in

ggplot". The Google searches are very efficient way how to get the guidelines, generally more efficient than using the R help like ?ggplot_function.

R base graphics

The R base graphics uses the common R grammar, i.e.

plot(dfx, dfy, further parameters) or plot(y~x, data=df, further parameters).

For the base R graphics, there is also a cheat sheet available (http://publish.illinois.edu/johnrgallagher/files/2015/10/BaseGraphicsCheatsheet.pdf) as well as abundant online resources. R help is in general quite informative. Info on parameters of graphical functions can be called by ?par.

To export plots produced by R graphics, you need to define a graphical device in a file, draw the plot there and save the file. I.e.:

```
pdf("filename.pdf", width, height) # Creating the file – width and height are in inches (vector graphics – pdf, svg) or in pixels (raster graphics – png, jpg, tif)

plot(y~x, data=df, further parameters)

dev.off() #Closes and saves the file. This is essential.
```

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.

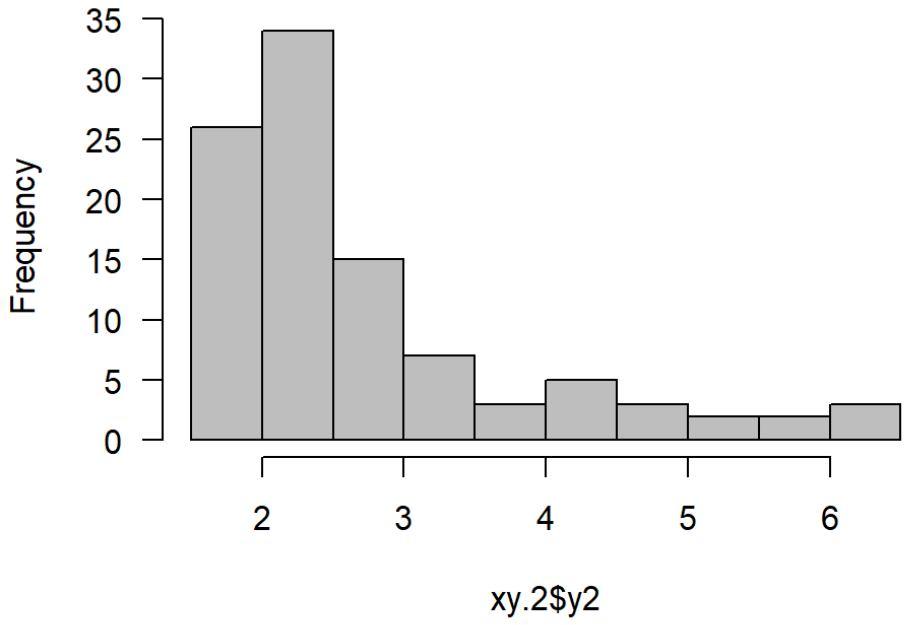

Fig. 4.1. Histogram of the variable xy.2$y2 plotted by base R.

How to do in R

The R base function **hist** applied on the variable to be plotted produces the histogram.

In ggplot, plotting the histogram is rather complicated.

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

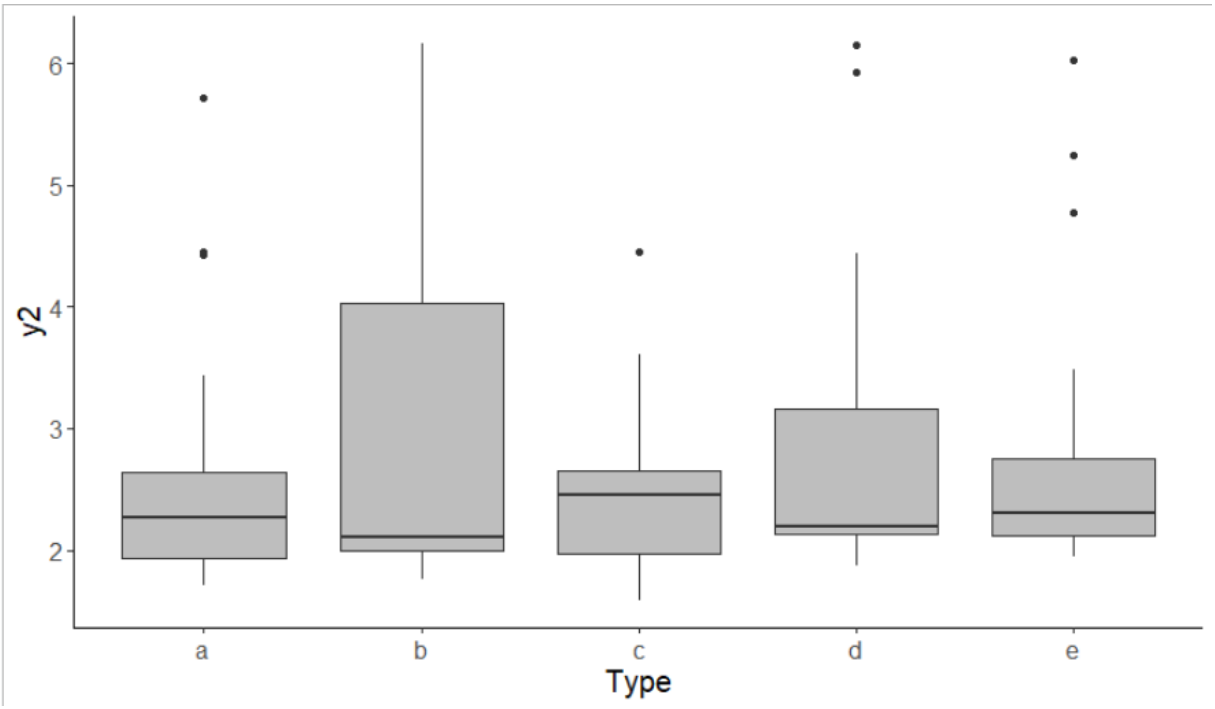

Fig 4.2. Boxplot displaying the values of the variable y2 for individual categories of type.1. Note the non-symmetric distributions and the outliers. Plotted by ggplot.

How to do in R

base R: function **boxplot** applied to formula numeric~character produces the boxplot

```
ggplot: ggplot(data=xy.2,mapping=aes(x=type.1, y=y2))+
geom_boxplot(fill="grey")+theme_classic()+
theme(text=element_text(size=15))+xlab("Type")
```

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 .

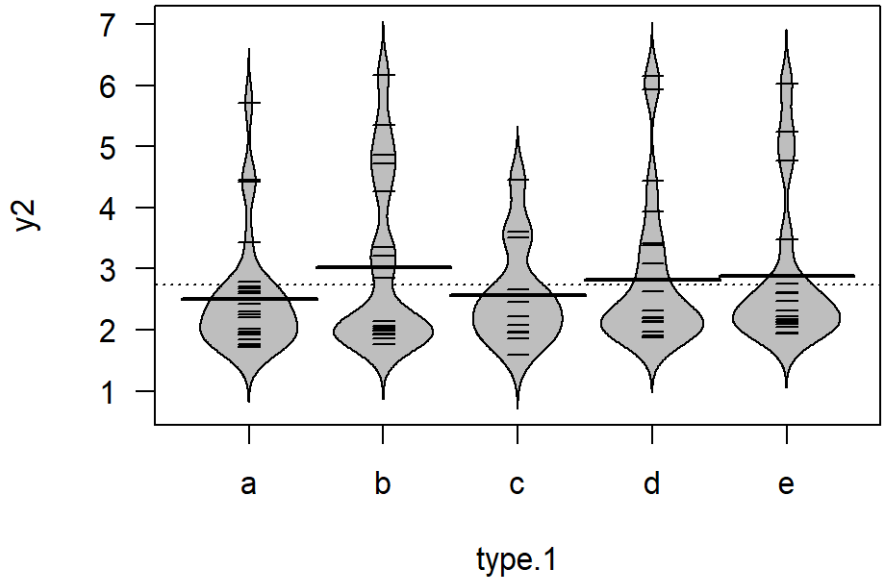

Fig 4.3. Beanplot displaying the values and densities of the variable y2 for individual categories of type.1. Dotted line, bold lines and short narrow lines indicate global mean, group means and individual observations respectively.

Barplot

Barplot is an efficient graphical tool to display counts of a categorical variable. Multiple categorical variables may also be combined to define types of observations.

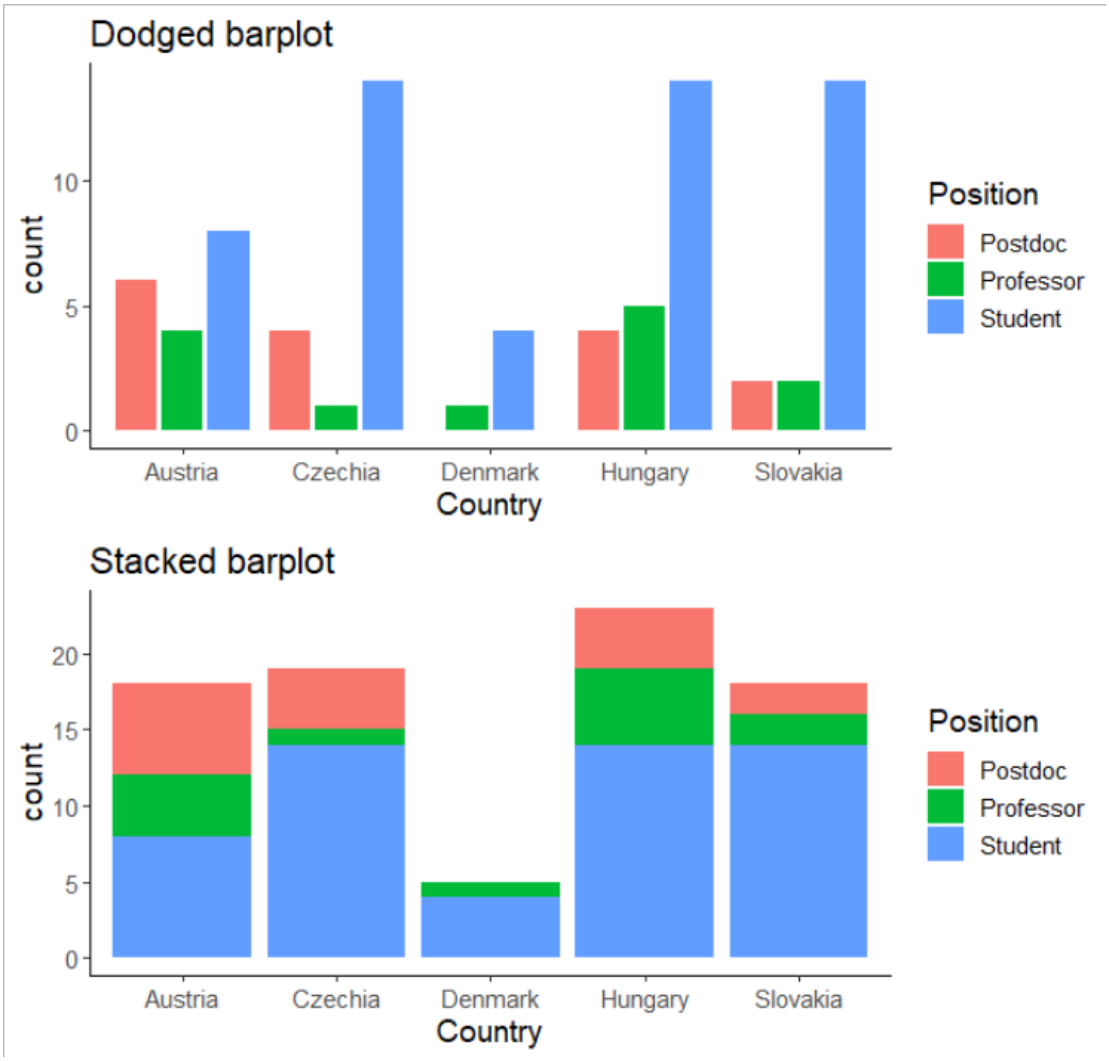

Fig 4.4. Dodged and stacked barplots displaying counts of conference participants categorized by their country of origin and position held.

How to do in R

```
ggplot: ggplot(data=conf,mapping=aes(x=Country,
fill=Position))+geom_bar(position="stack")+
theme_classic()+theme(text=element_text(size=14))+
xlab("Country")+labs(title="Stacked barplot")
```

Note that the combination of the two categorical variables is done by defining one as x and the other as fill. The position parameter in geom_bar may be “stacked”, “dodge” or “dodge2”. Producing the upper panel of Fig. 4.4 however required a bit more complicated geom_bar(position = position_dodge2(preserve = "single")) because of the zero postdocs from Denmark

Dotchart

Dotchart can be used to 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, dotcharts should be supplied by error bars displaying standard errors, confidence intervals or standard deviations. 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 dotcharts is that they allow judging on difference between means. However, this comes with substantial loss of information: dotcharts do not display the distribution at all and may even be misleading in this respect.

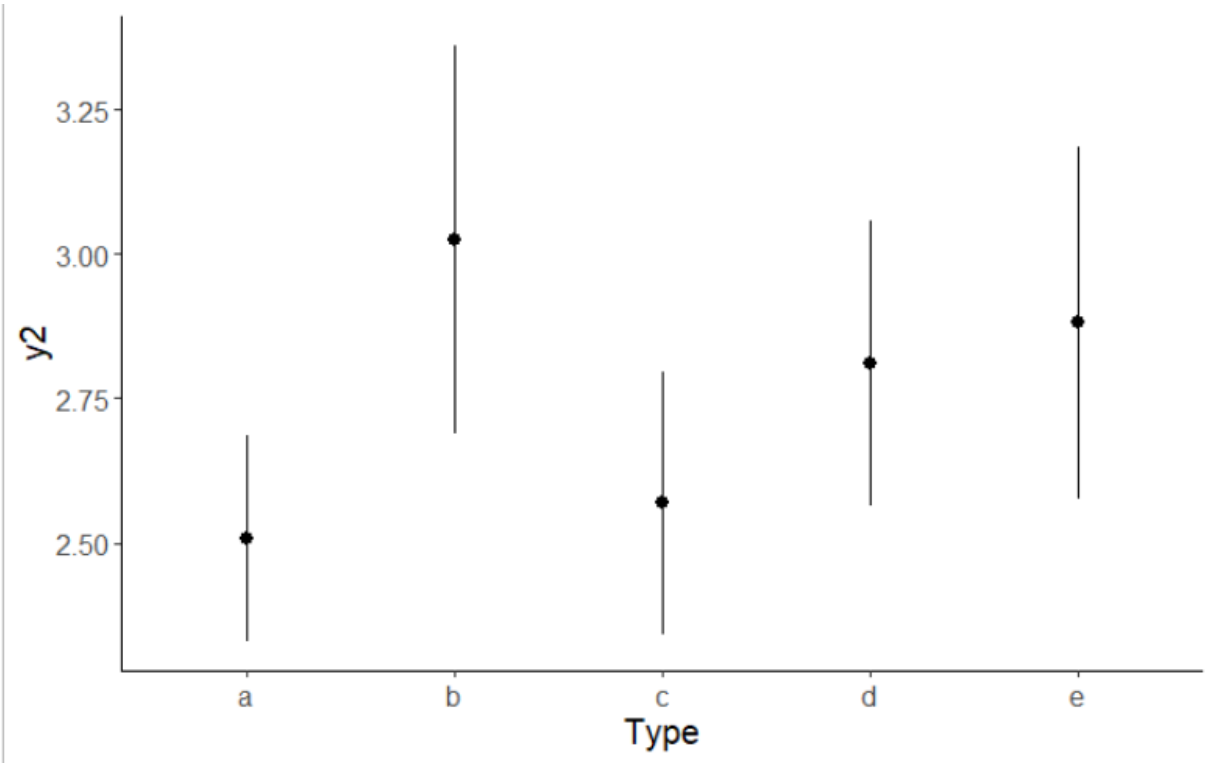

Fig. 4.5. Dotchart displaying mean values of variable y2 for individual categories of type.1. Error bars indicate 1 standard error.

How to do in R

ggplot: The layer is defined by a stat function here, which produces a statistical summary of the data (as opposed to geom_xx functions which need to be supplied by the data for direct plotting). In this case (which is applicable in general):

```
stat_summary(fun.data=mean_se, geom="pointrange")
```

```
fun.data may be mean_se (mean +- standard error),
mean_cl_normal(95%-confidence interval), or smean_sdl
(standard deviation).

geom="pointrange" defines the geom function used for plotting.


The full script for fig. 4.5. is:

ggplot(data=xy.2,mapping=aes(x=type.1, y=y2))+
stat_summary(fun.data=mean_se, geom="pointrange")+
theme_classic()+theme(text=element_text(size=15))+xlab("Type")
```

Scatterplot

Scatterplot is a simple point-based plot illustrating the association between two quantitative variables. The points in 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).

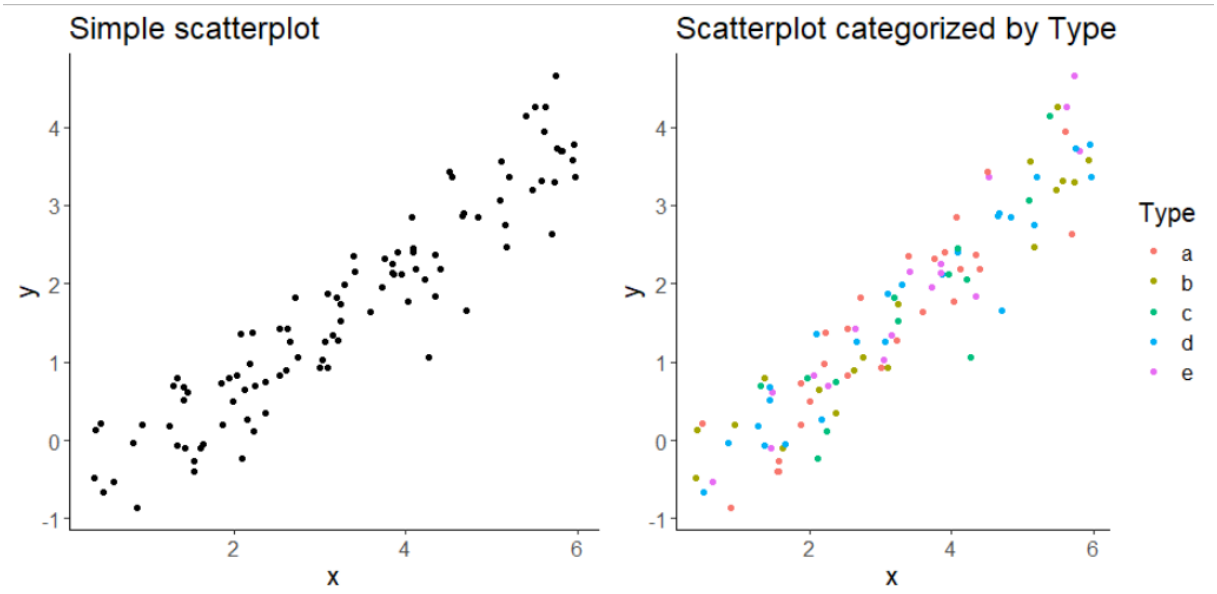

Fig 4.6. Scatterplots displaying the relationships between x and y. An additional categorical variable may be displayed by point colors.

How to do in R

```
base R: plot(y~x, data=df) – quick and simple, the graphics is
reasonable; adding colors of faceting quite complicated

ggplot: geom_point is the function plotting the scatterplot.

color=color.variable must be included in aes for colored
points following the valued of the color.variable
```

The full script for fig. 4.6 (note grid.arrange function used to combine the two panels in a single plot.

```
library(gridExtra)

p1<-ggplot(data=xy.2,mapping=aes(x=x, y=y))+
geom_point()+theme_classic()+
theme(text=element_text(size=15))+labs(title="Simple
scatterplot")

p2<-ggplot(data=xy.2,mapping=aes(x=x, y=y, color=type.1))+
geom_point()+ theme_classic()+
theme(text=element_text(size=15))+labs(color="Type")+
labs(title="Scatterplot categorized by Type")

grid.arrange(p1, p2, nrow=1)
```

Facetting

Facetting is a powerful tool display dependence of two variables at different levels of a factor. It may often be better that displaying colors.

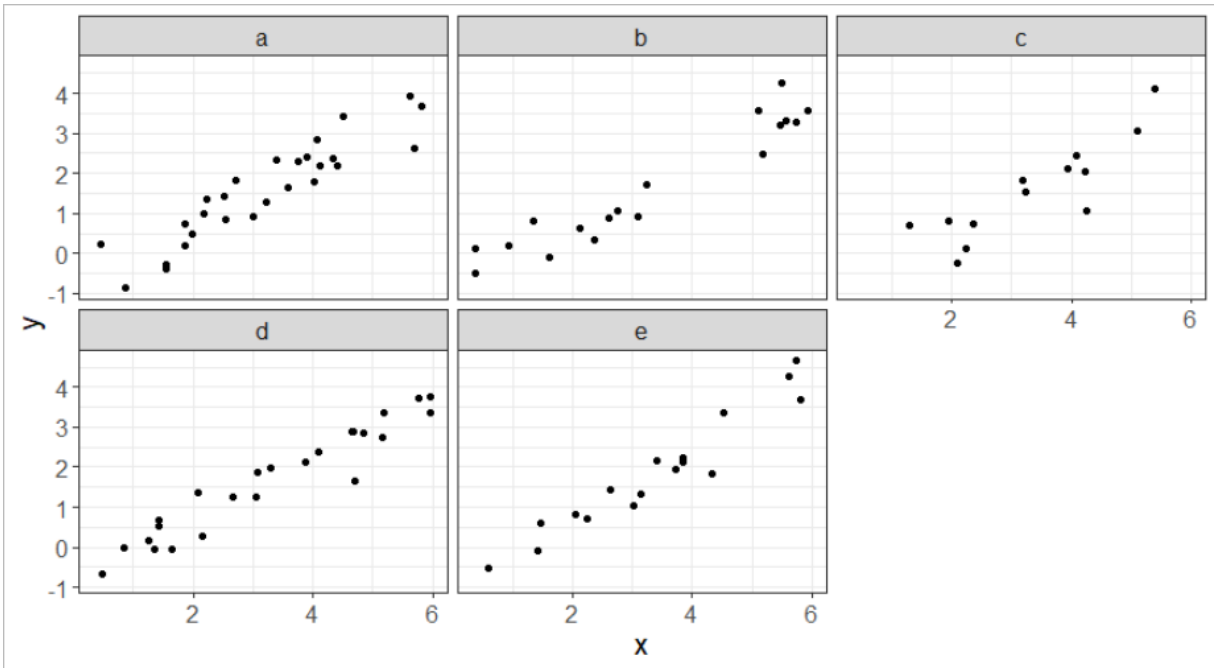

Fig. 4.7. Facetted scatterplot showing the relationships between x and y at different levels of Type.

How to do in R

ggplot: function facet_wrap

```
ggplot(data=xy.2,mapping=aes(x=x, y=y))+geom_point()+
facet_wrap(~type.1)+theme_bw()+
theme(text=element_text(size=15))
```