Table of Contents

Colours

(The R code used for this website is here)

Adding colours to the graphical output is a very effective way how to make it more attractive and possibly also more informative to the audience. In the end, the figure is often the very first thing the reader gets attracted to when reading the scientific (and not only scientific) text. On the other side, making a colourful figure with wrong colours can do more harm than good, so choosing the right colours is essential.

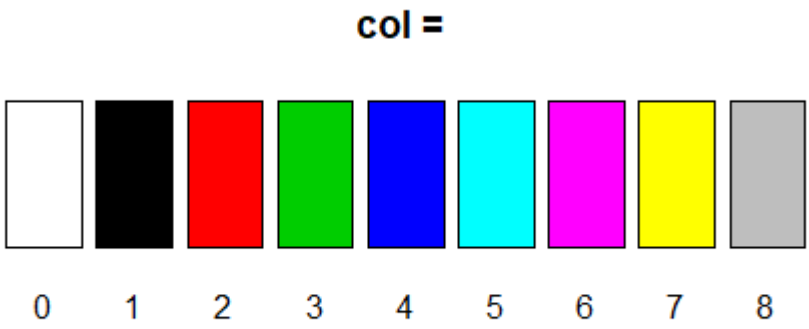
The rules about choosing the right colours can be simplified as the following:

- Choose colours which are not too saturated (this makes the eyes tired, and possibly causes the after image - after turning the eyes out of the figure, the reader will still see the image burned into the retina of her/his eyes.
- Choose colours which can also be distinguished by colorblind person, i.e. person with colour vision deficiency. For example, do not mix red and green colour in the same figure, since colourblind person often cannot recognize them (it is estimated that ~ 8% of males and ~ 0.5% of females has red-green colourblindness).
- Good colourful figure should be able to convey all the information, even if printed on the black-and-white printer (or displayed on the black-and-white screen).

Below, I will briefly summarize the ways how to define colours in R.

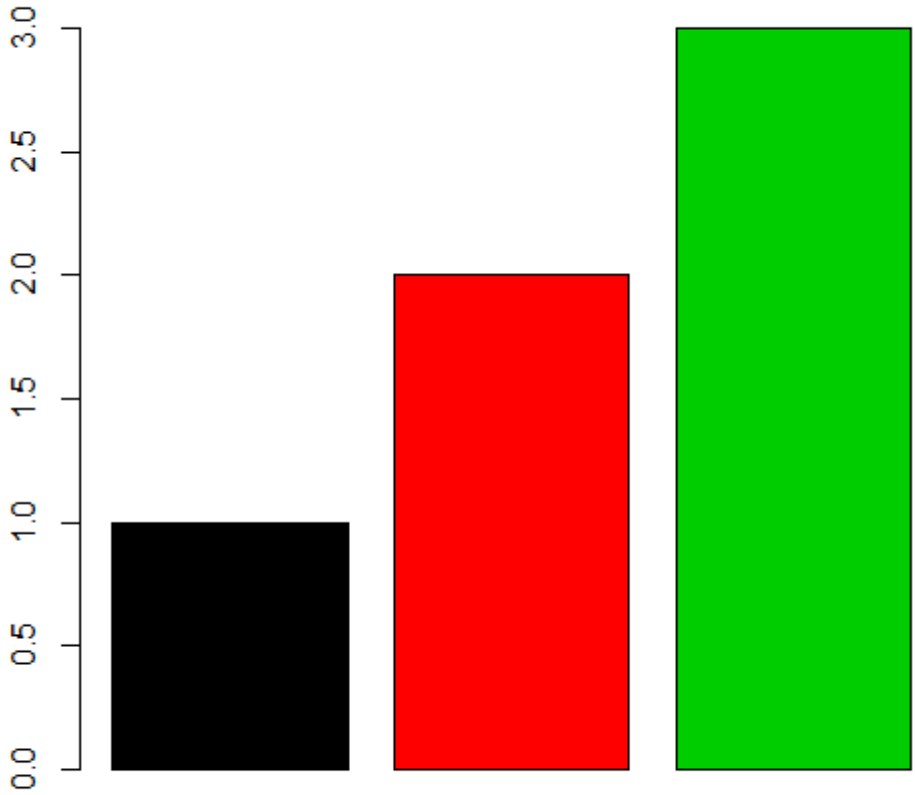
Define colours by the number

A simple way to define a colour is using the number. By default, R has 9 colours:



Here, I will use a simple example of bar plot with bars of different colour, to illustrate the colour differences.

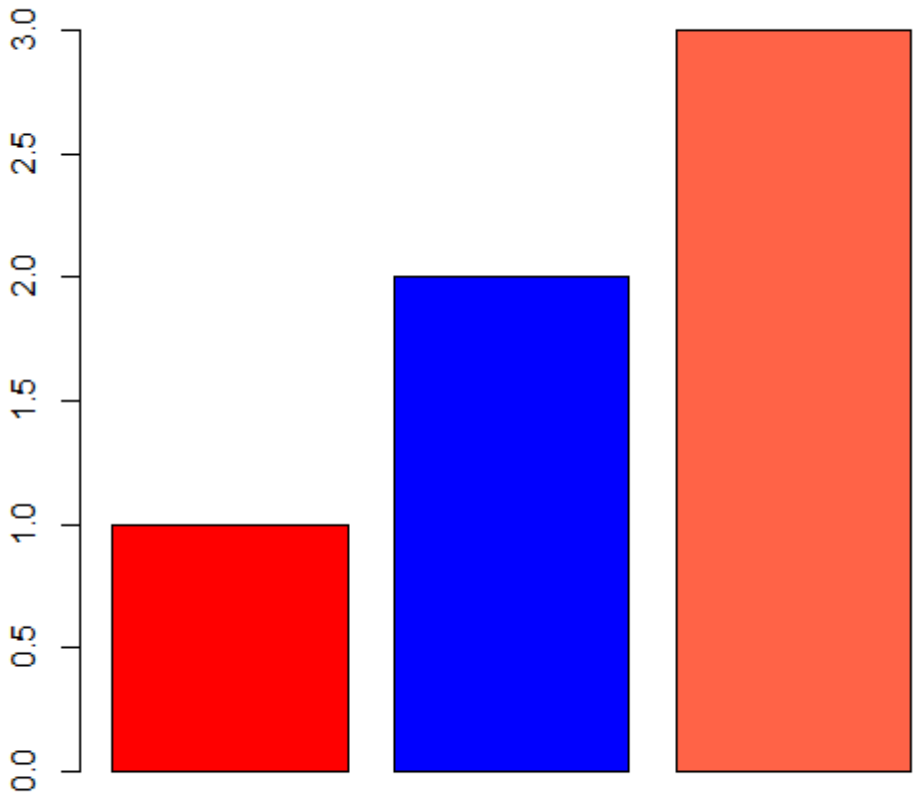
```
cols_num <- c(1, 2, 3)
barplot (1:3, col = cols_num)
```



Define colours by their name

There are 657 named colours in R; the function ‘‘colours ()’’ or ‘‘colors ()’’ returns a vector containing these names. For example:

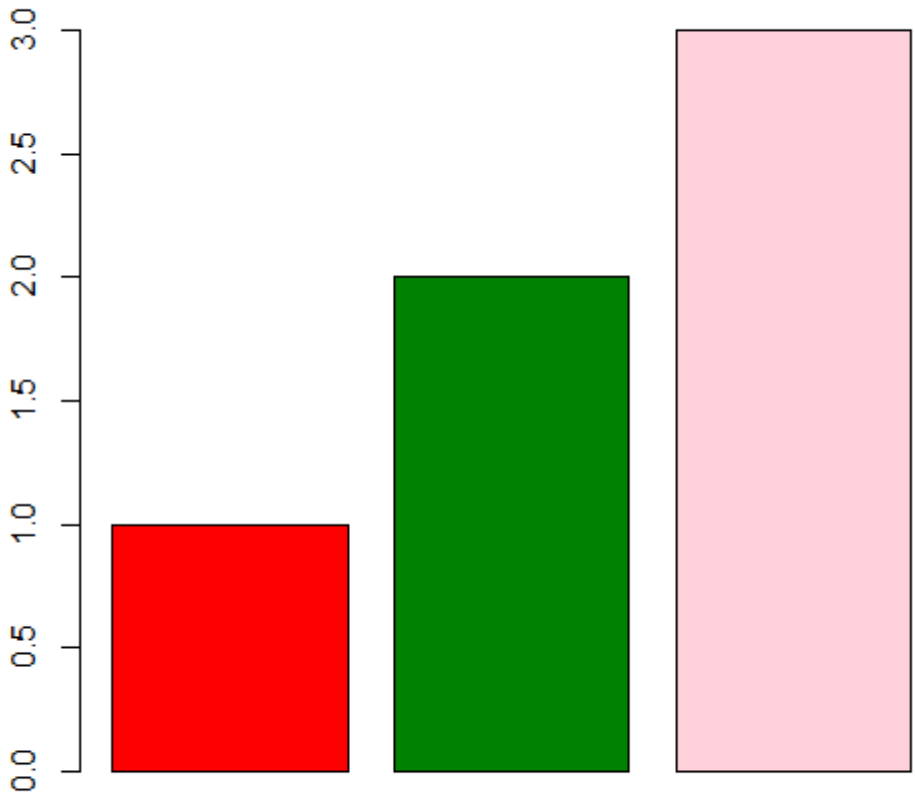
```
cols_name <- c("red", "blue", "tomato")
barplot (1:3, col = cols_name)
```



Define colour using rgb () function

We can also define the colour by mixing three colour channels - red, green and blue (that is why RGB). The function “rgb” in R offers this option. For example, pure red, green and blue colour, respectively, can be defined by setting given channel to full intensity, while setting the other two channels to zero intensity. If you want darker red, green, or blue, set the channel to lower value than 1 (maximum). Alternatively, set all three channels to different intensity to create any colour you wish:

```
cols_rgb <- c(rgb (red = 1, green = 0, blue = 0),  # red colour
              rgb (red = 0, green = 0.5, blue = 0),  # dark green colour
(lower intensity of green channel)
              rgb (red = 0.5, green = 0.2, blue = 0.8)) # mixing all three
chanels result in violet here
barplot (1:3, col = cols_rgb)
```



By the way, the last colour on the previous barplot is called Millenial Pink (I had no idea before there is something like this). Millennials, or Generation Y, is the generation born after 1990 and before 2010, and people born in this generation are comfortable with using the Internet and social media (while representants of previous Generation X not really).

All three channels set to high intensity produce white colour, all three channels set to zero produce black colour. The argument `maxColorValue` in the function `rgb` modifies the scale. For example, if you wish to specify the values for individual channels at the scale 0-255, which is the most efficient (because the colours are recoded into hex codes, see below), set the `maxColorValue = 255` Millennial Pink the will be coded as `rgb (red = 255, green = 209, blue = 220, maxColorValue = 255)`.

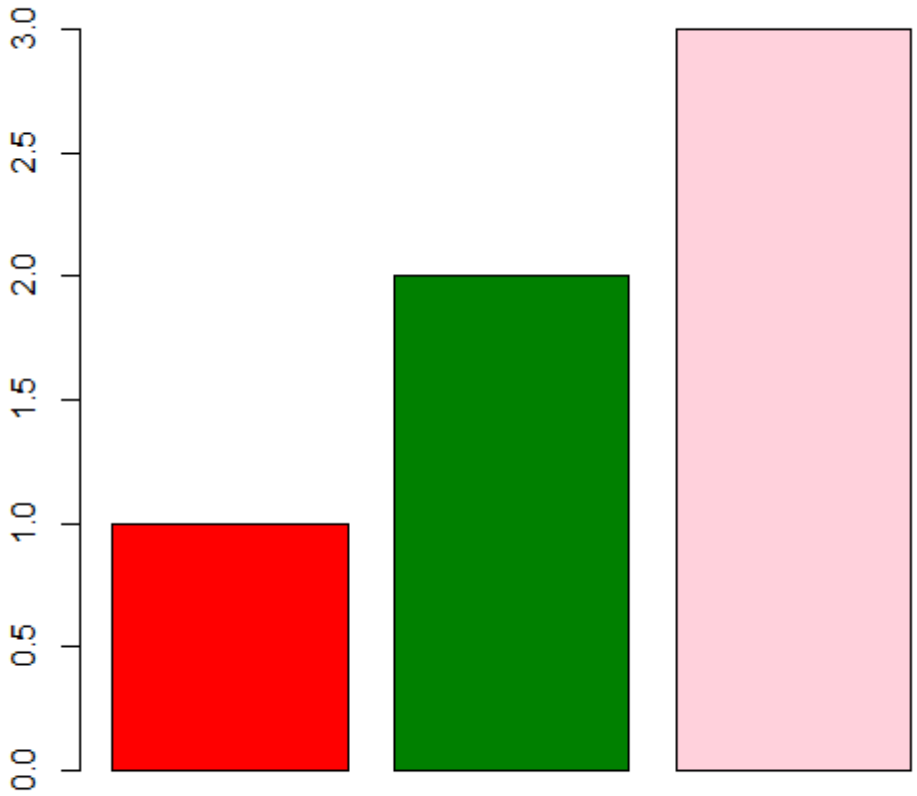
Note that `rgb` function has also the argument `alpha`, which sets the **colour transparency**. The lower the `alpha` value is, the more transparent the colour is (transparent = you can see through), while the higher it is, the more opaque it is (opaque = you cannot see through).

Define colours by the hexadecimal colour code

There is a universal hexadecimal colour code linked to RGB colour concept. The code is starting with "#" sign, and is followed by 6 or 8 digits. These digits encode intensity of red (RR), green (GG) and

blue (BB) channel: #RRGGBB. While the decimal system is based on ten symbols (0 - 9), the hexadecimal system is using 16 symbols (0 - 9 and A - F). In this way it can represent the values between 0 and 255 using only two digits (decimal 0 = hex 0, decimal 10 = hex A, decimal 15 = hex F, decimal 16 = hex 10, decimal 255 = hex FF, etc.; see here for full table). It is also possible to set the transparency (channel alpha) of the colour; in that case, the code has eight digits: #RRGGBBTT (where TT is hex code for transparency).

```
cols_hex <- c("#FF0000", "#008000", "#FFD1DC") # red, dark green, and
Millenial Pink
barplot (1:3, col = cols_hex)
```

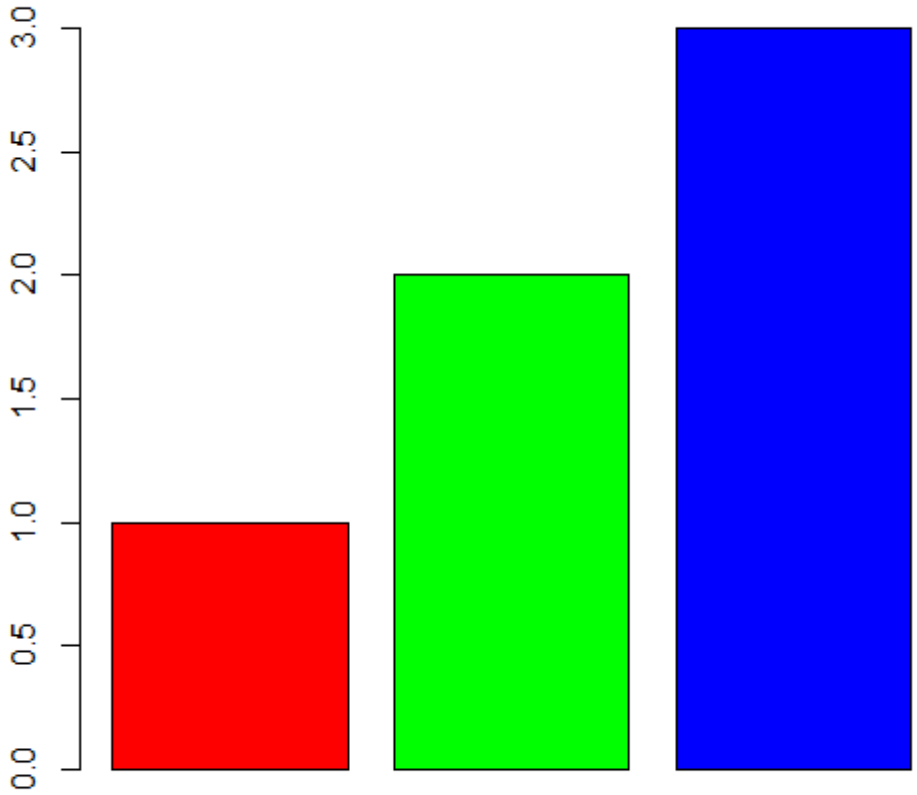


Choose colours from colour palette

Colour palette is a set of colours, assembled following some logic. R has several built-in functions with colour palettes. One is `rainbow (n)`, which takes rainbow continuum (colours from red to violet) and chops them into *n* segments. Another is `heat.colors (n)`, which is the sequence of colours from yellow to red (the higher = warmer, the redder). For other palettes, check `?heat.colors` help file.

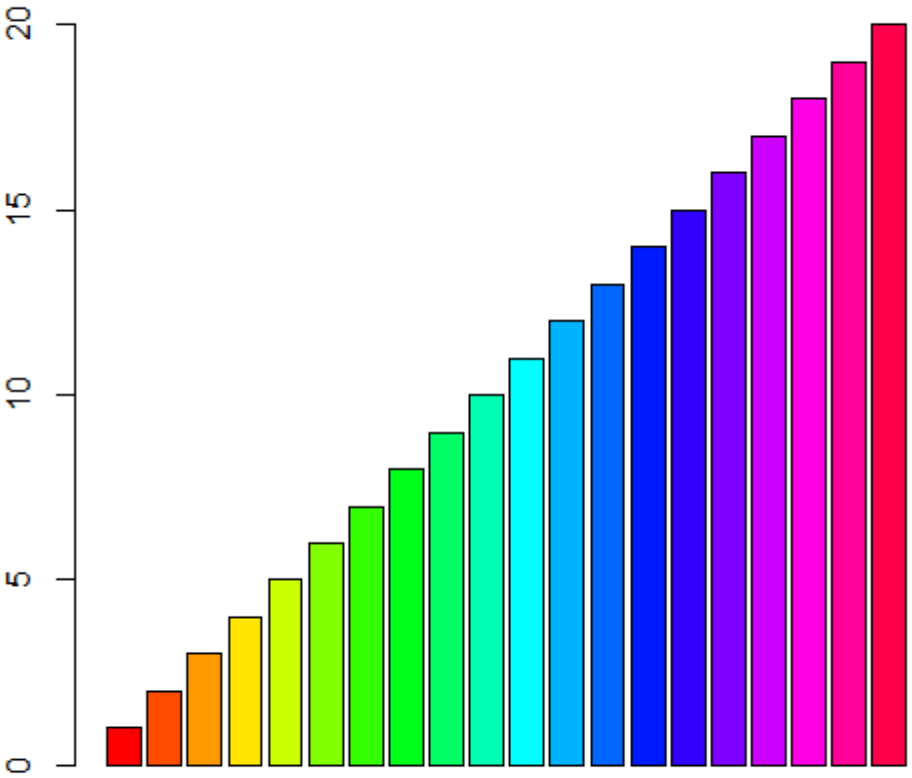
```
cols_rb1 <- rainbow (3)
```

```
barplot (1:3, col = cols_rb1)
```

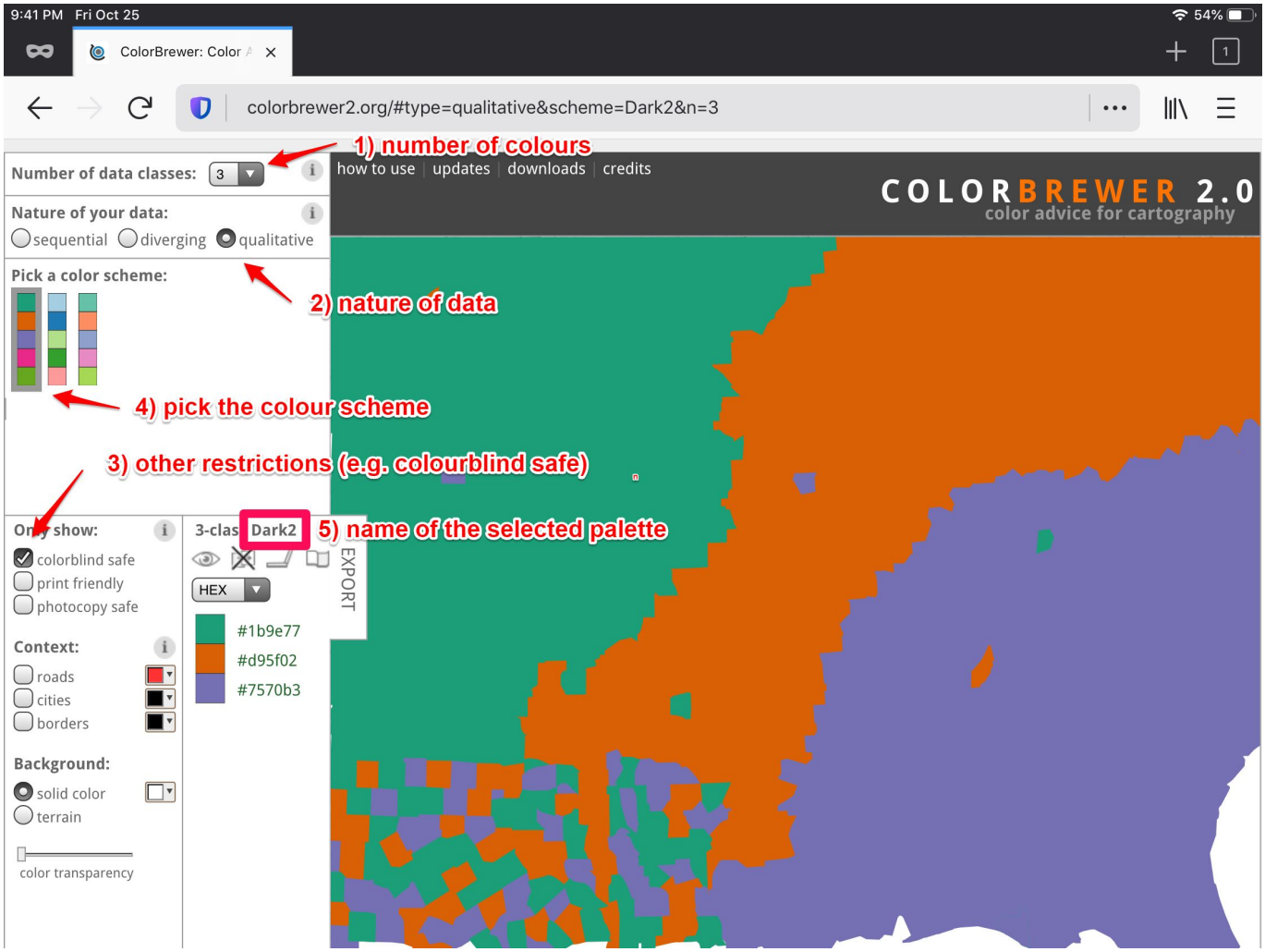


To get the feeling of the rainbow, separate it into more (here 20) segments:

```
cols_rb2 <- rainbow (20)
barplot (1:20, col = cols_rb2)
```

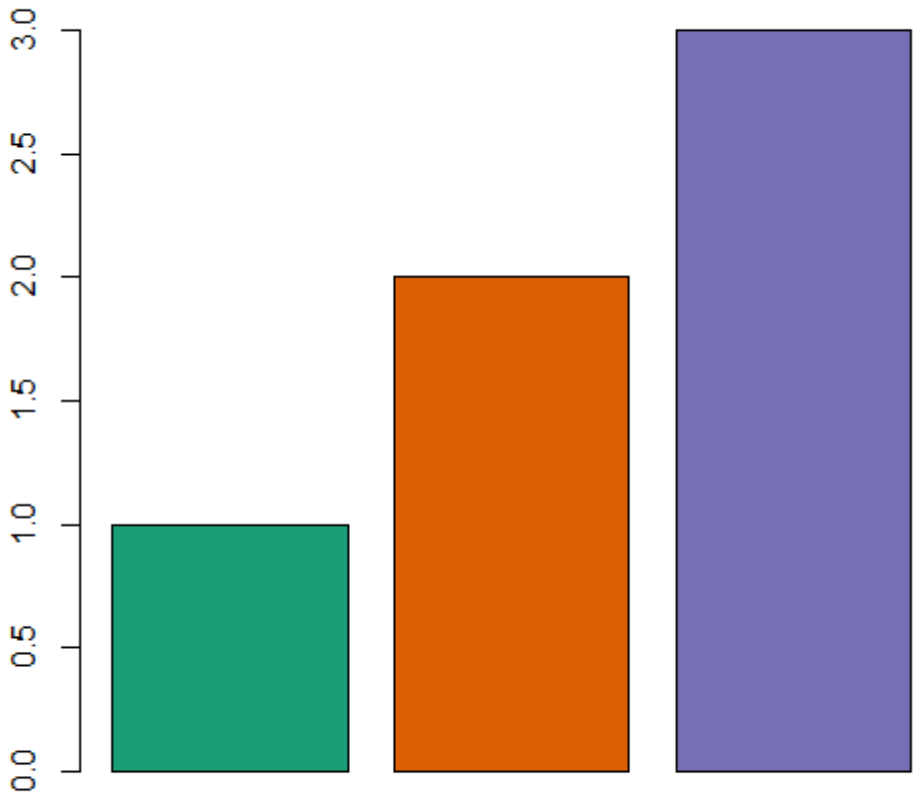


R offers several packages that are defining colour palettes according to a set of rules. One is “RColorBrewer”, which is implementing the palettes from the website www.colorbrewer2.org, where you can make several choices to pick the colour palette optimal for your purpose:



Then, record the name of the palette (e.g. *Dark2*), and use it in the function `brewer.pal` as the argument `type = 'Dark2'`, together with the number of colours n:

```
library (RColorBrewer)
cols_rcb <- brewer.pal (n = 3, name = 'Dark2')
barplot (1:3, col = cols_rcb)
```



There are plenty of other options on how to choose colours. For example, a truly advanced package with many options to select colour palettes is the `library (pals)` (see the introduction and examples here).