[Top R Color Palettes to Know for Great Data Visualization](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/" \o "Top R Color Palettes to Know for Great Data Visualization)

[kassambara](https://www.datanovia.com/en/blog/author/kassambara/" \o "View all posts by )

|

[GGPLOT2 Graphical Parameters](https://www.datanovia.com/en/blog/tag/ggplot2-graphical-parameters/), [R Colors](https://www.datanovia.com/en/blog/tag/r-colors/)

 |

[ggplot2](https://www.datanovia.com/en/blog/category/ggplot2/)

 |

[0](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/#respond)

This article presents the top **R color palettes** for changing the default color of a graph generated using either the *ggplot2* package or the *R base plot* functions.

You’ll learn how to use the top 6 predefined color palettes in R, available in different R packages:

* Viridis color scales [viridis package].
* Colorbrewer palettes [RColorBrewer package]
* Grey color palettes [ggplot2 package]
* Scientific journal color palettes [ggsci package]
* Wes Anderson color palettes [wesanderson package]
* R base color palettes: rainbow, heat.colors, cm.colors.

Note that, the “rainbow” and “heat” color palettes are less perceptually uniform compared to the other color scales. The “viridis” scale stands out for its large perceptual range. It makes as much use of the available color space as possible while maintaining uniformity.

When [comparing these color palettes](https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html) as they might appear under various forms of colorblindness, the viridis palettes remain the most robust.



Contents:

* [Demo dataset](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/#demo-dataset)
* [Create a basic ggplot colored by groups](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/#create-a-basic-ggplot-colored-by-groups)
* [Viridis color palettes](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/#viridis-color-palettes)
* [RColorBrewer palettes](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/#rcolorbrewer-palettes)
* [Grey color palettes](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/#grey-color-palettes)
* [Scientific journal color palettes](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/#scientific-journal-color-palettes)
* [Wes Anderson color palettes](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/#wes-anderson-color-palettes)
* [R base color palettes](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/#r-base-color-palettes)
* [Conclusion](https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/#conclusion)

**Demo dataset**

We’ll use the R built-in iris demo dataset.

head(iris, 6)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species

## 1 5.1 3.5 1.4 0.2 setosa

## 2 4.9 3.0 1.4 0.2 setosa

## 3 4.7 3.2 1.3 0.2 setosa

## 4 4.6 3.1 1.5 0.2 setosa

## 5 5.0 3.6 1.4 0.2 setosa

## 6 5.4 3.9 1.7 0.4 setosa

**Create a basic ggplot colored by groups**

You can change colors according to a grouping variable by:

* Mapping the argument color to the variable of interest. This will be applied to points, lines and texts
* Mapping the argument fill to the variable of interest. This will change the fill color of areas, such as in box plot, bar plot, histogram, density plots, etc.

In our example, we’ll map the options color and fill to the grouping variable Species, for scatter plot and box plot, respectively.

Changes colors by groups using the levels of Species variable:

**library**("ggplot2")

# Box plot

bp <- ggplot(iris, aes(Species, Sepal.Length)) +

geom\_boxplot(aes(fill = Species)) +

theme\_minimal() +

theme(legend.position = "top")

bp

# Scatter plot

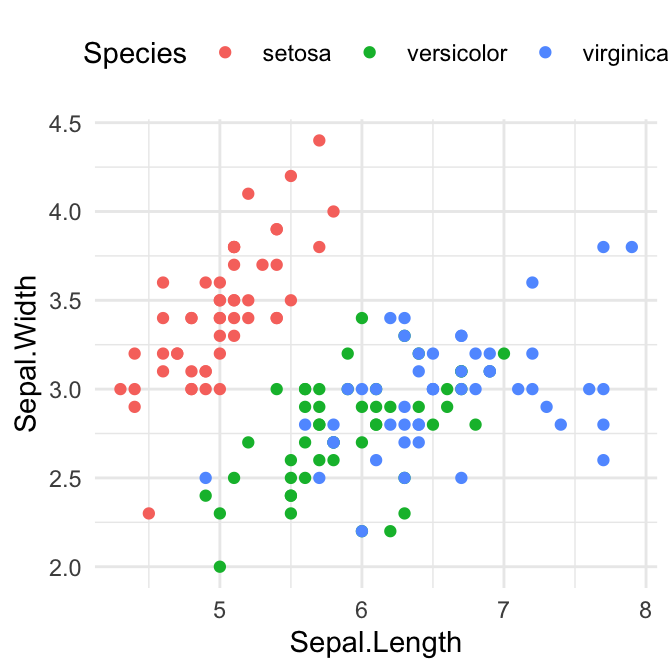
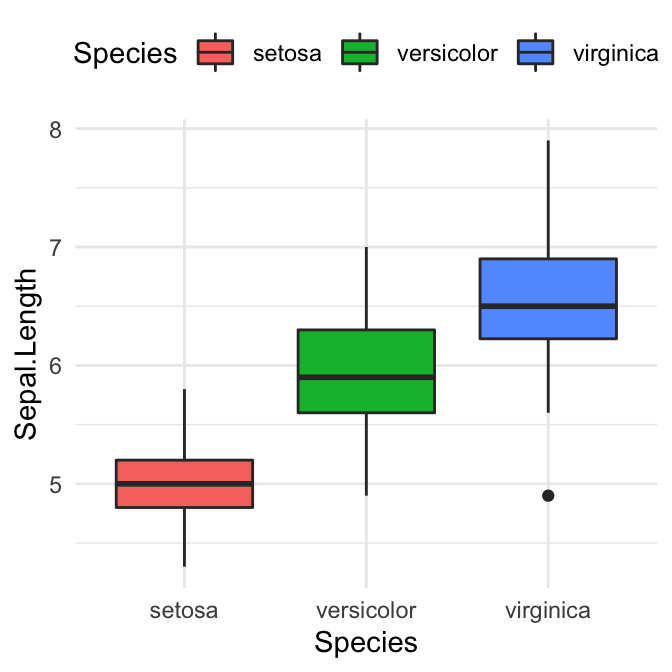
sp <- ggplot(iris, aes(Sepal.Length, Sepal.Width)) +

geom\_point(aes(color = Species)) +

theme\_minimal()+

theme(legend.position = "top")

sp



**Viridis color palettes**

The viridis R package (by Simon Garnier) provides color palettes to make beautiful plots that are: printer-friendly, perceptually uniform and easy to read by those with colorblindness.

Install and load the package as follow:

install.packages("viridis") # Install

**library**("viridis") # Load

The viridis package contains four sequential color scales: “Viridis” (the primary choice) and three alternatives with similar properties (“magma”, “plasma”, and “inferno”).



Key functions:

* scale\_color\_viridis(): Change the color of points, lines and texts
* scale\_fill\_viridis(): Change the fill color of areas (box plot, bar plot, etc)
* viridis(n), magma(n), inferno(n) and plasma(n): Generate color palettes for base plot, where n is the number of colors to returns.

Note that, the function scale\_color\_viridis() and scale\_fill\_viridis() have an argument named option, which is a character string indicating the colormap option to use. Four options are available: “magma” (or “A”), “inferno” (or “B”), “plasma” (or “C”), and “viridis” (or “D”, the default option).

1. Usage in ggplot2

**library**(ggplot2)

# Gradient color

ggplot(iris, aes(Sepal.Length, Sepal.Width))+

geom\_point(aes(color = Sepal.Length)) +

scale\_color\_viridis(option = "D")+

theme\_minimal() +

theme(legend.position = "bottom")

# Discrete color. use the argument discrete = TRUE

ggplot(iris, aes(Sepal.Length, Sepal.Width))+

geom\_point(aes(color = Species)) +

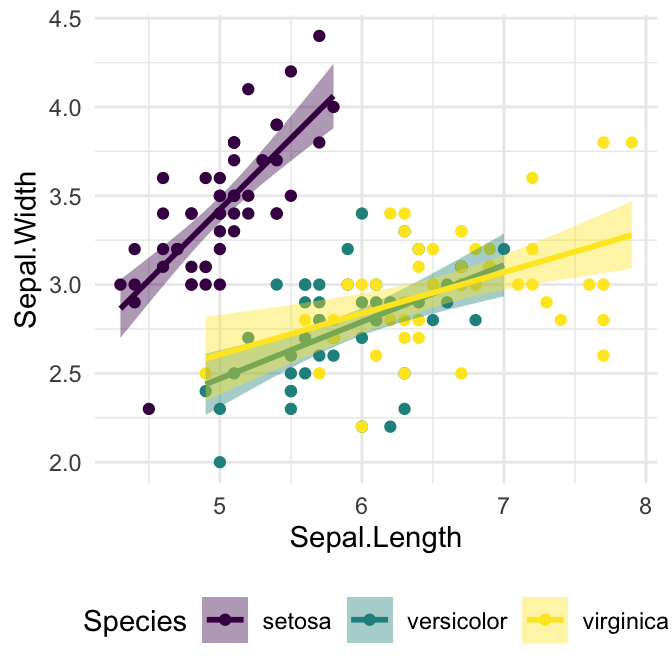
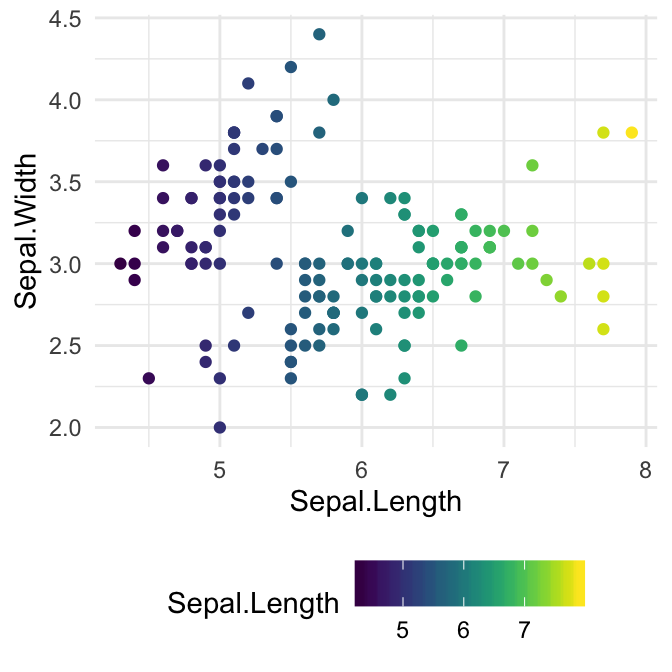
geom\_smooth(aes(color = Species, fill = Species), method = "lm") +

scale\_color\_viridis(discrete = TRUE, option = "D")+

scale\_fill\_viridis(discrete = TRUE) +

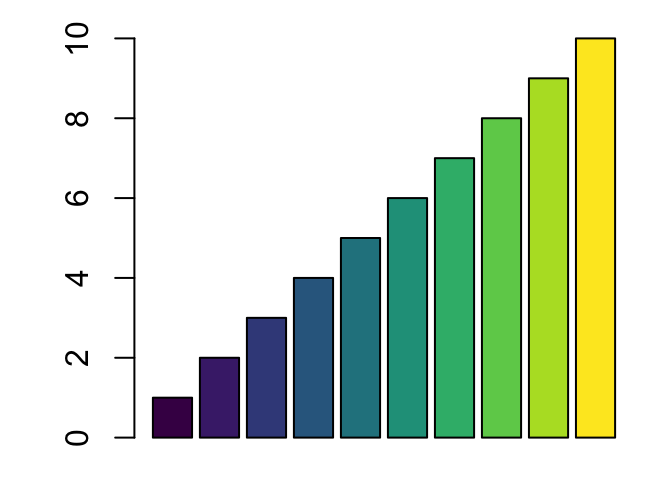
theme\_minimal() +

theme(legend.position = "bottom")



1. Usage in base plot. Use the function viridis() to generate the number of colors you want:

barplot(1:10, col = viridis(10))



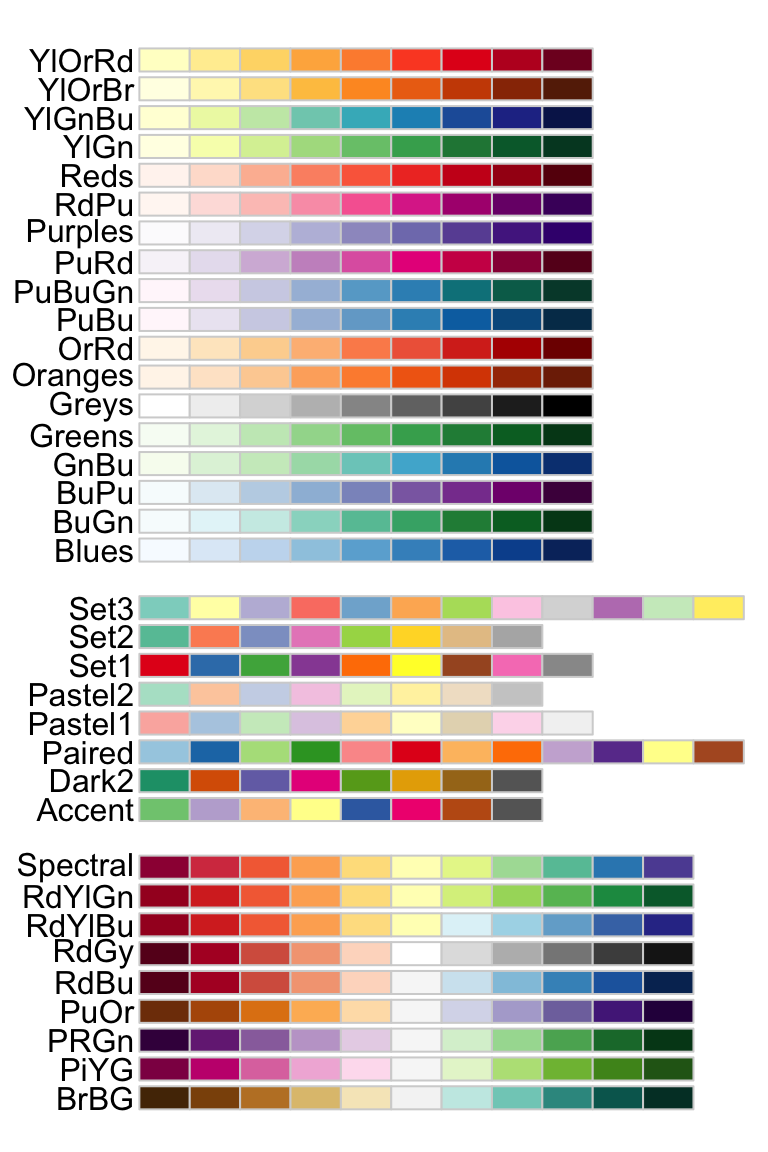
**RColorBrewer palettes**

The RColorBrewer package creates a nice looking color palettes. You should first install it as follow: install.packages("RColorBrewer").

To display all the color palettes in the package, type this:

**library**(RColorBrewer)

display.brewer.all()



The package contains 3 types of color palettes: sequential, diverging, and qualitative.

1. **Sequential palettes** (first list of colors), which are suited to ordered data that progress from low to high (gradient). The palettes names are : Blues, BuGn, BuPu, GnBu, Greens, Greys, Oranges, OrRd, PuBu, PuBuGn, PuRd, Purples, RdPu, Reds, YlGn, YlGnBu YlOrBr, YlOrRd.
2. **Qualitative palettes** (second list of colors), which are best suited to represent nominal or categorical data. They not imply magnitude differences between groups. The palettes names are : Accent, Dark2, Paired, Pastel1, Pastel2, Set1, Set2, Set3.
3. **Diverging palettes** (third list of colors), which put equal emphasis on mid-range critical values and extremes at both ends of the data range. The diverging palettes are : BrBG, PiYG, PRGn, PuOr, RdBu, RdGy, RdYlBu, RdYlGn, Spectral

The RColorBrewer package include also three important functions:

# 1. Return the hexadecimal color specification

brewer.pal(n, name)

# 2. Display a single RColorBrewer palette

# by specifying its name

display.brewer.pal(n, name)

# 3. Display all color palette

display.brewer.all(n = NULL, type = "all", select = NULL,

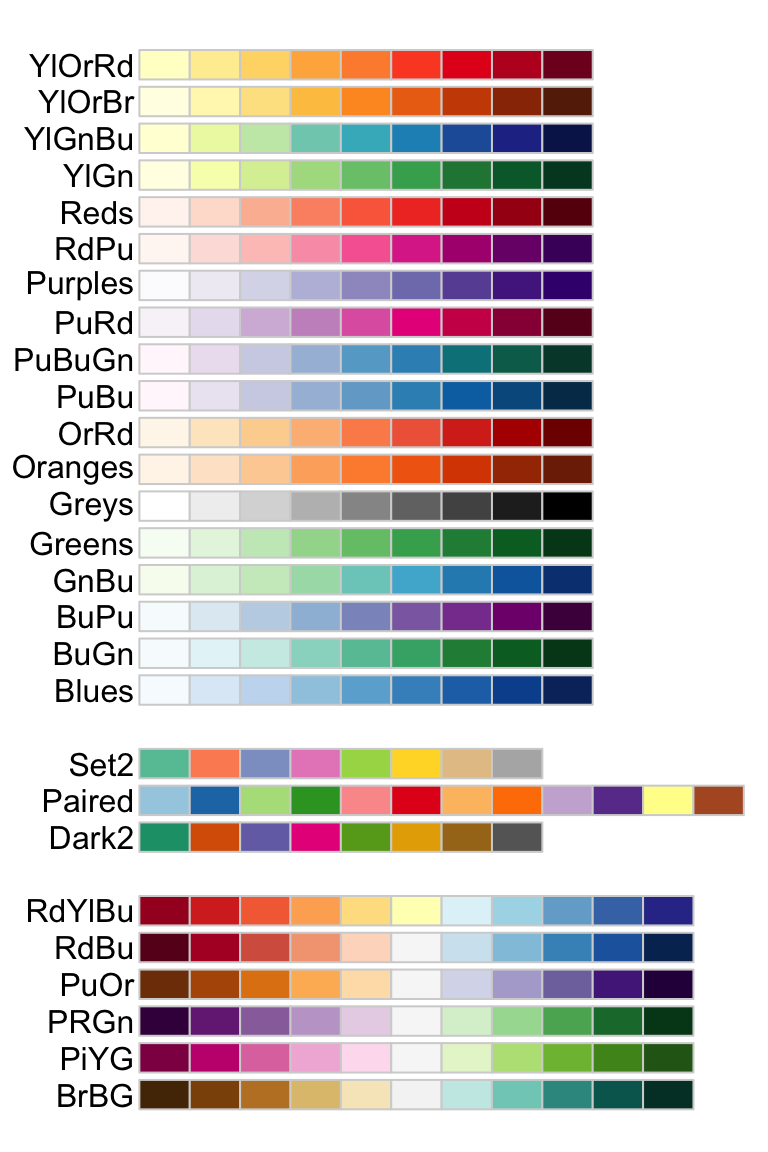
colorblindFriendly = FALSE)

Description of the function arguments:

* n: Number of different colors in the palette, minimum 3, maximum depending on palette.
* name: A palette name from the lists above. For example name = RdBu.
* type: The type of palette to display. Allowed values are one of: “div”, “qual”, “seq”, or “all”.
* select: A list of palette names to display.
* colorblindFriendly: if TRUE, display only colorblind friendly palettes.

To display only colorblind-friendly brewer palettes, use this R code:

display.brewer.all(colorblindFriendly = TRUE)



You can also view a single RColorBrewer palette by specifying its name as follow :

# View a single RColorBrewer palette by specifying its name

display.brewer.pal(n = 8, name = 'Dark2')



# Hexadecimal color specification

brewer.pal(n = 8, name = "Dark2")

## [1] "#1B9E77" "#D95F02" "#7570B3" "#E7298A" "#66A61E" "#E6AB02" "#A6761D"

## [8] "#666666"

Usage in ggplot2. Two color scale functions are available in ggplot2 for using the colorbrewer palettes:

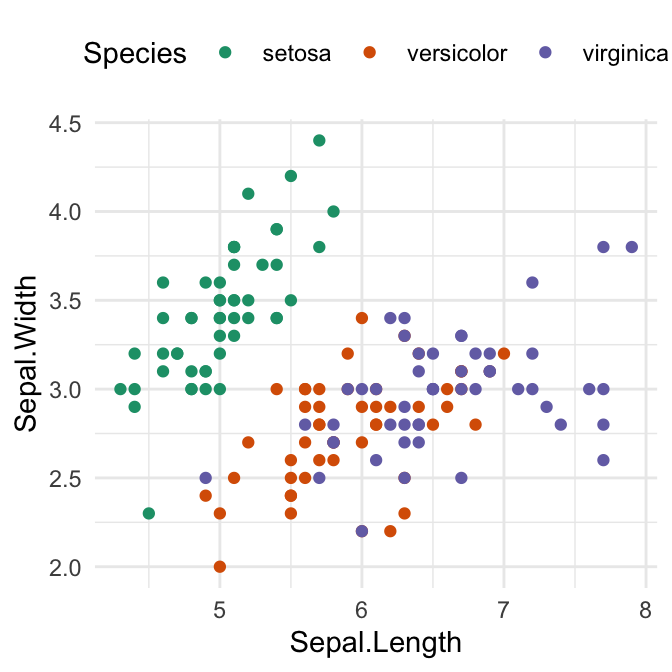
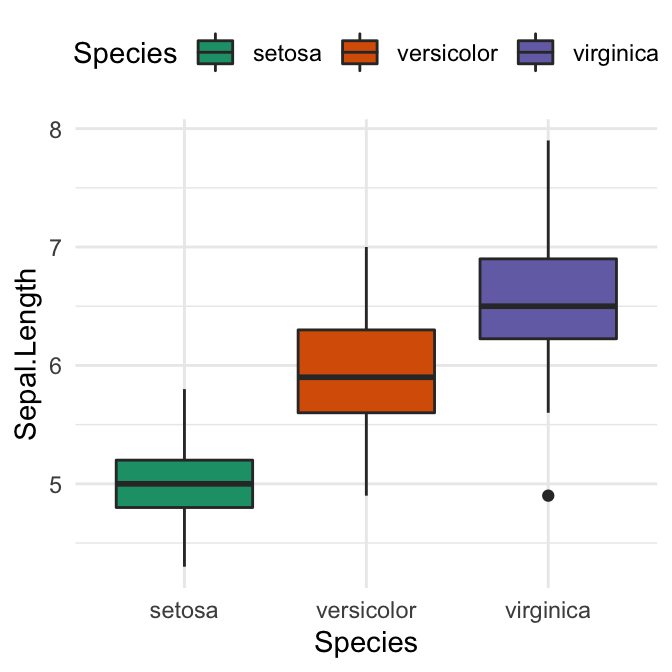
* scale\_fill\_brewer() for box plot, bar plot, violin plot, dot plot, etc
* scale\_color\_brewer() for lines and points

# Box plot

bp + scale\_fill\_brewer(palette = "Dark2")

# Scatter plot

sp + scale\_color\_brewer(palette = "Dark2")



Usage in base plots. The function brewer.pal() is used to generate a vector of colors.

# Barplot using RColorBrewer

barplot(c(2,5,7), col = brewer.pal(n = 3, name = "RdBu"))

**Grey color palettes**

Key functions:

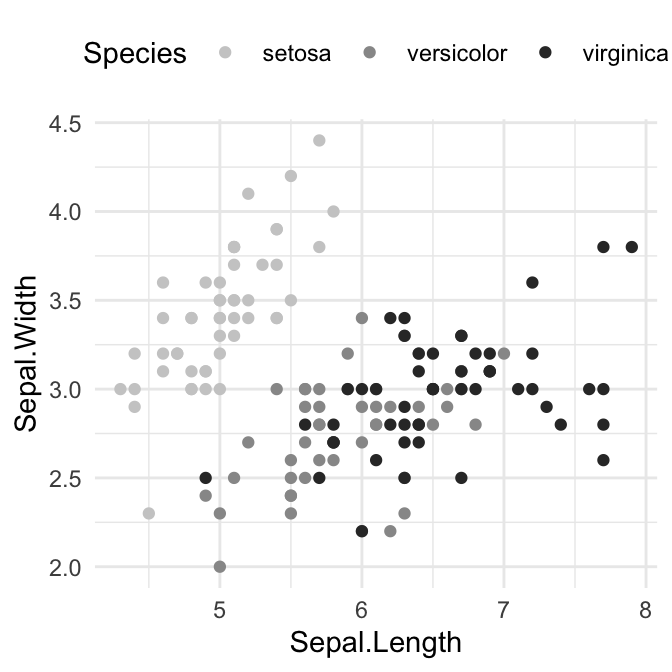
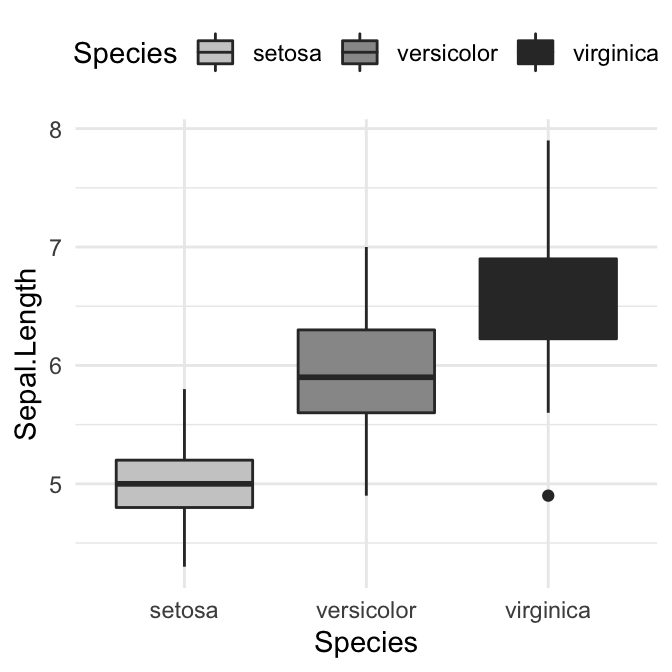
* scale\_fill\_grey() for box plot, bar plot, violin plot, dot plot, etc
* scale\_colour\_grey() for points, lines, etc

# Box plot

bp + scale\_fill\_grey(start = 0.8, end = 0.2)

# Scatter plot

sp + scale\_color\_grey(start = 0.8, end = 0.2)



**Scientific journal color palettes**

The R package ggsci contains a collection of high-quality color palettes inspired by colors used in scientific journals, data visualization libraries, and more.

The color palettes are provided as ggplot2 scale functions:

* scale\_color\_npg() and scale\_fill\_npg(): Nature Publishing Group color palettes
* scale\_color\_aaas() and scale\_fill\_aaas(): American Association for the Advancement of Science color palettes
* scale\_color\_lancet() and scale\_fill\_lancet(): Lancet journal color palettes
* scale\_color\_jco() and scale\_fill\_jco(): Journal of Clinical Oncology color palettes
* scale\_color\_tron() and scale\_fill\_tron(): This palette is inspired by the colors used in Tron Legacy. It is suitable for displaying data when using a dark theme.

You can find more examples in the [ggsci package vignettes](https://cran.r-project.org/web/packages/ggsci/vignettes/ggsci.html).

Note that for base plots, you can use the corresponding palette generator for creating a list of colors. For example, you can use: pal\_npg(), pal\_aaas(), pal\_lancet(), pal\_jco(), and so on.

1. Usage in ggplot2. We’ll use JCO and the Tron Legacy color palettes.

**library**("ggplot2")

**library**("ggsci")

# Change area fill color. JCO palette

ggplot(iris, aes(Species, Sepal.Length)) +

geom\_boxplot(aes(fill = Species)) +

scale\_fill\_jco()+

theme\_classic() +

theme(legend.position = "top")

# Change point color and the confidence band fill color.

# Use tron palette on dark theme

ggplot(iris, aes(Sepal.Length, Sepal.Width)) +

geom\_point(aes(color = Species)) +

geom\_smooth(aes(color = Species, fill = Species)) +

scale\_color\_tron()+

scale\_fill\_tron()+

theme\_dark() +

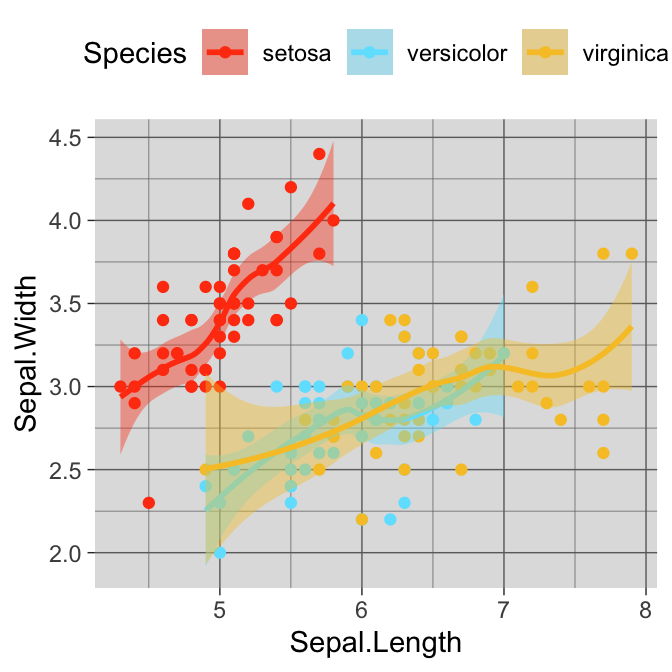
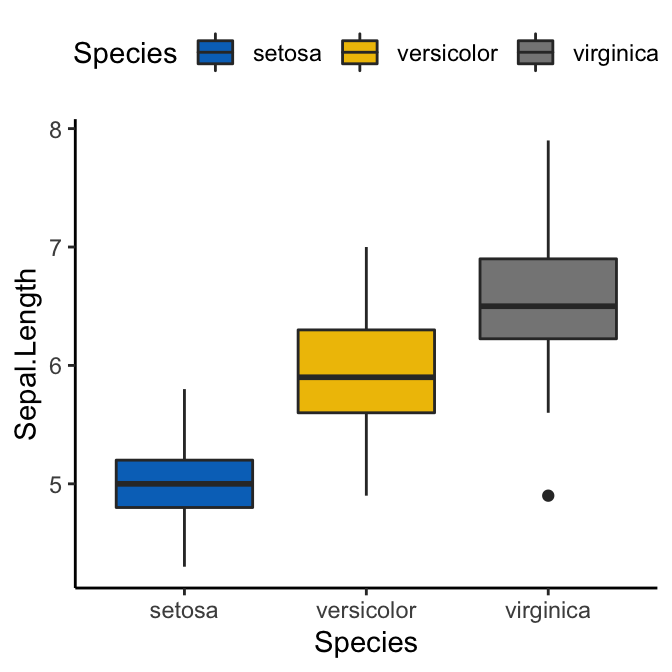
theme(

legend.position = "top",

panel.background = element\_rect(fill = "#2D2D2D"),

legend.key = element\_rect(fill = "#2D2D2D")

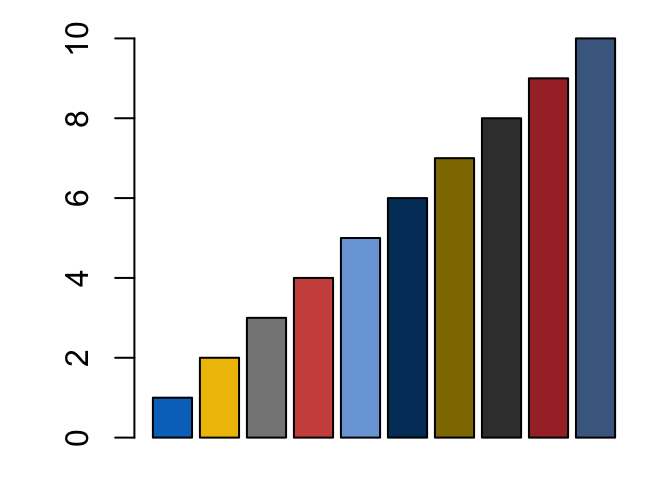
)



1. Usage in base plots

par(mar = c(1, 3.5, 1, 1))

barplot(1:10, col = pal\_jco()(10))



**Wes Anderson color palettes**

Install the latest developmental version from Github (devtools::install\_github("karthik/wesanderson")) or install from CRAN (install.packages("wesanderson")).

It contains 16 color palettes from Wes Anderson movies:

**library**(wesanderson)

names(wes\_palettes)

## [1] "BottleRocket1" "BottleRocket2" "Rushmore1" "Royal1"

## [5] "Royal2" "Zissou1" "Darjeeling1" "Darjeeling2"

## [9] "Chevalier1" "FantasticFox1" "Moonrise1" "Moonrise2"

## [13] "Moonrise3" "Cavalcanti1" "GrandBudapest1" "GrandBudapest2"

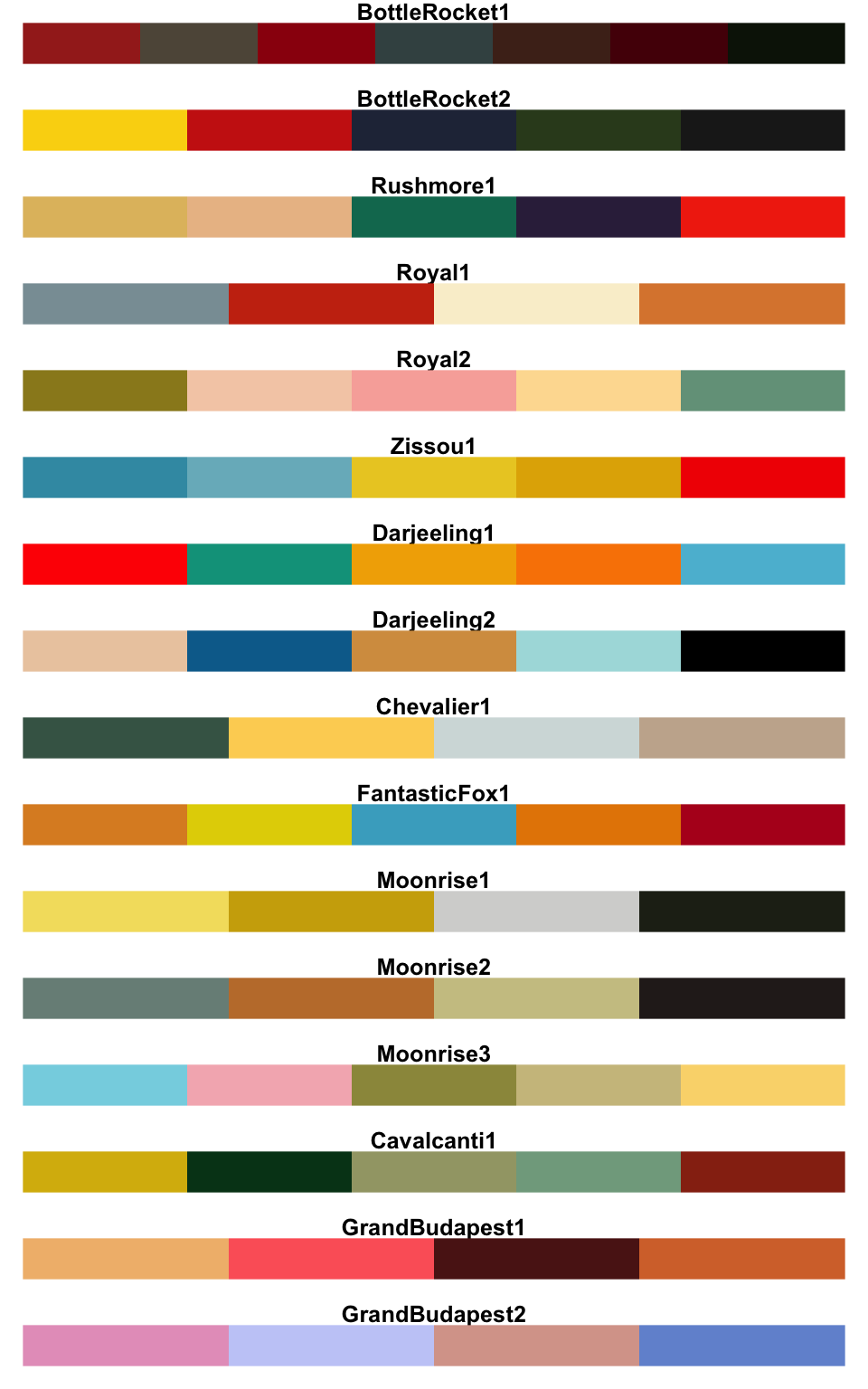
The key R function in the package, for generating a vector of colors, is

wes\_palette(name, n, type = c("discrete", "continuous"))

* name: Name of desired palette
* n: Number of colors desired. Unfortunately most palettes now only have 4 or 5 colors.
* type: Either “continuous” or “discrete”. Use continuous if you want to automatically interpolate between colours.

If you need more colours than normally found in a palette, you can use a continuous palette to interpolate between existing colours.

The available color palettes are :



Usage in ggplot2:

**library**(wesanderson)

# Discrete color

bp + scale\_fill\_manual(values = wes\_palette("GrandBudapest1", n = 3))

# Gradient color

pal <- wes\_palette("Zissou1", 100, type = "continuous")

ggplot(heatmap, aes(x = X2, y = X1, fill = value)) +

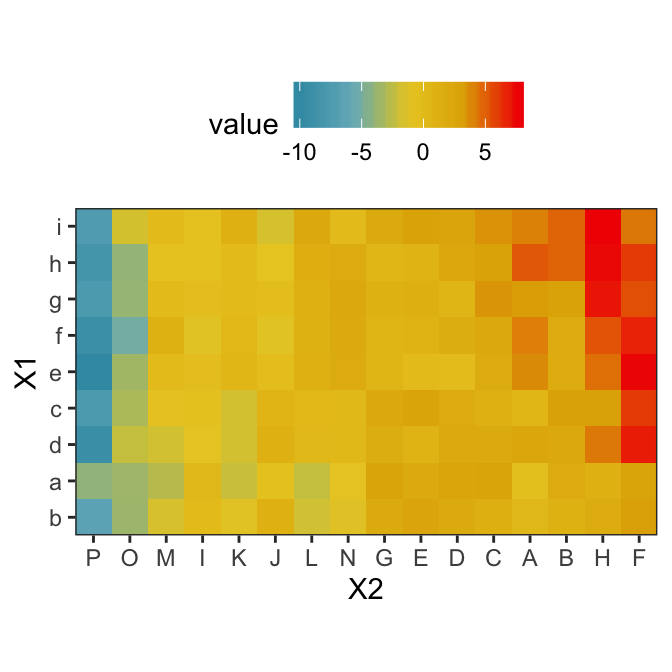
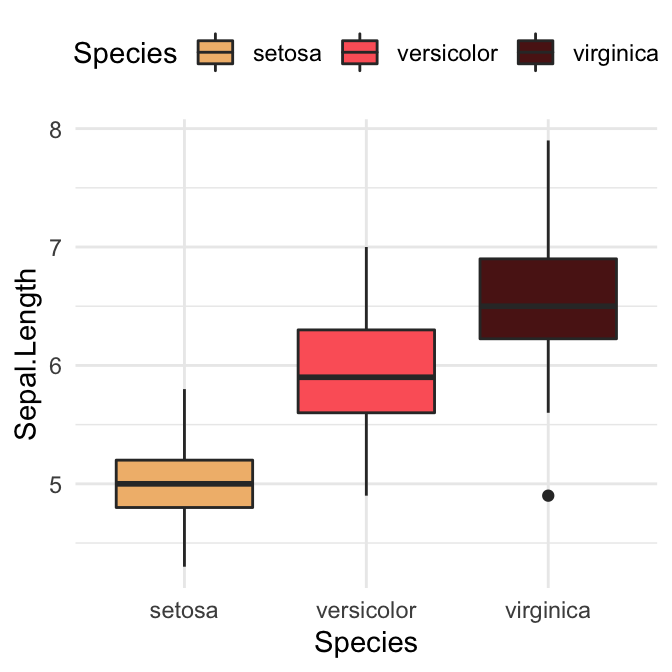
geom\_tile() +

scale\_fill\_gradientn(colours = pal) +

scale\_x\_discrete(expand = c(0, 0)) +

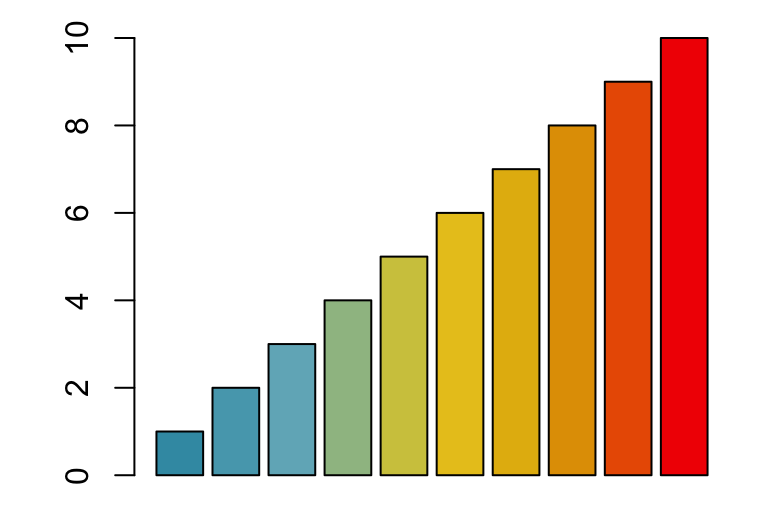
scale\_y\_discrete(expand = c(0, 0)) +

coord\_equal()



Usage in base plots:

barplot(1:10, col = wes\_palette("Zissou1", 10, type = "continuous"))



**R base color palettes**

There are 5 R base functions that can be used to generate a vector of n contiguous colors: rainbow(n), heat.colors(n), terrain.colors(n), topo.colors(n), and cm.colors(n).



Usage in R base plots:

barplot(1:5, col=rainbow(5))

# Use heat.colors

barplot(1:5, col=heat.colors(5))

# Use terrain.colors

barplot(1:5, col=terrain.colors(5))

# Use topo.colors

barplot(1:5, col=topo.colors(5))

# Use cm.colors

barplot(1:5, col=cm.colors(5))

**Conclusion**

We present the top R color palette to customize graphics generated by either the ggplot2 package or by the R base functions. The main points are summarized as follow.

* Create a basic ggplot. Map the color argument to a factor or grouping variable.

p <- ggplot(iris, aes(Sepal.Length, Sepal.Width))+

geom\_point(aes(color = Species))

p

* Set the color palette manually using a custom color scale:

p + scale\_color\_manual(values = c("#00AFBB", "#E7B800", "#FC4E07"))

* Use color blind-friendly palette:

cbp1 <- c("#999999", "#E69F00", "#56B4E9", "#009E73",

"#F0E442", "#0072B2", "#D55E00", "#CC79A7")

p + scale\_color\_manual(values = cbp1)

* Use RColorBrewer palettes:

p + scale\_color\_brewer(palette = "Dark2")

* Use viridis color scales:

**library**(viridis)

p + scale\_color\_viridis(discrete = TRUE)

­­­­