Colors in R

STAT 133

Gaston Sanchez

Department of Statistics, UC–Berkeley

gastonsanchez.com
github.com/gastonstat/stat133
Course web: gastonsanchez.com/teaching/stat133

Colors in R

Colors in plots

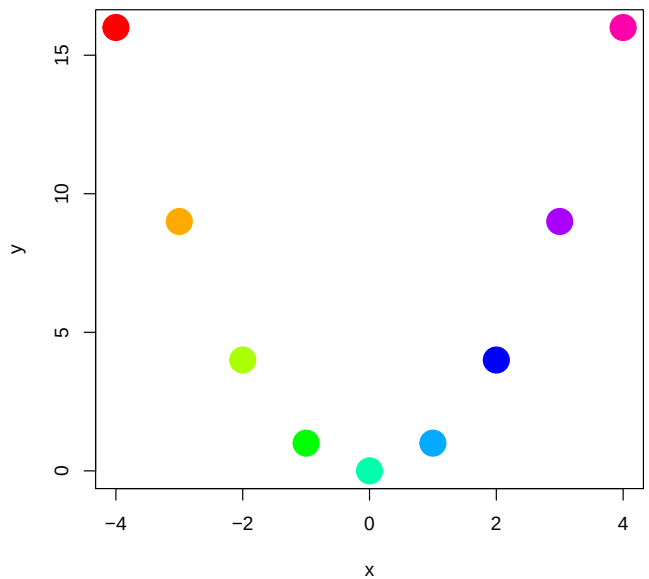
Colors on objects

In R plots, many objects can take on different colors

- points
- lines
- axes (and tick marks)
- filling areas
- borders
- text
- legends
- background

Colored points

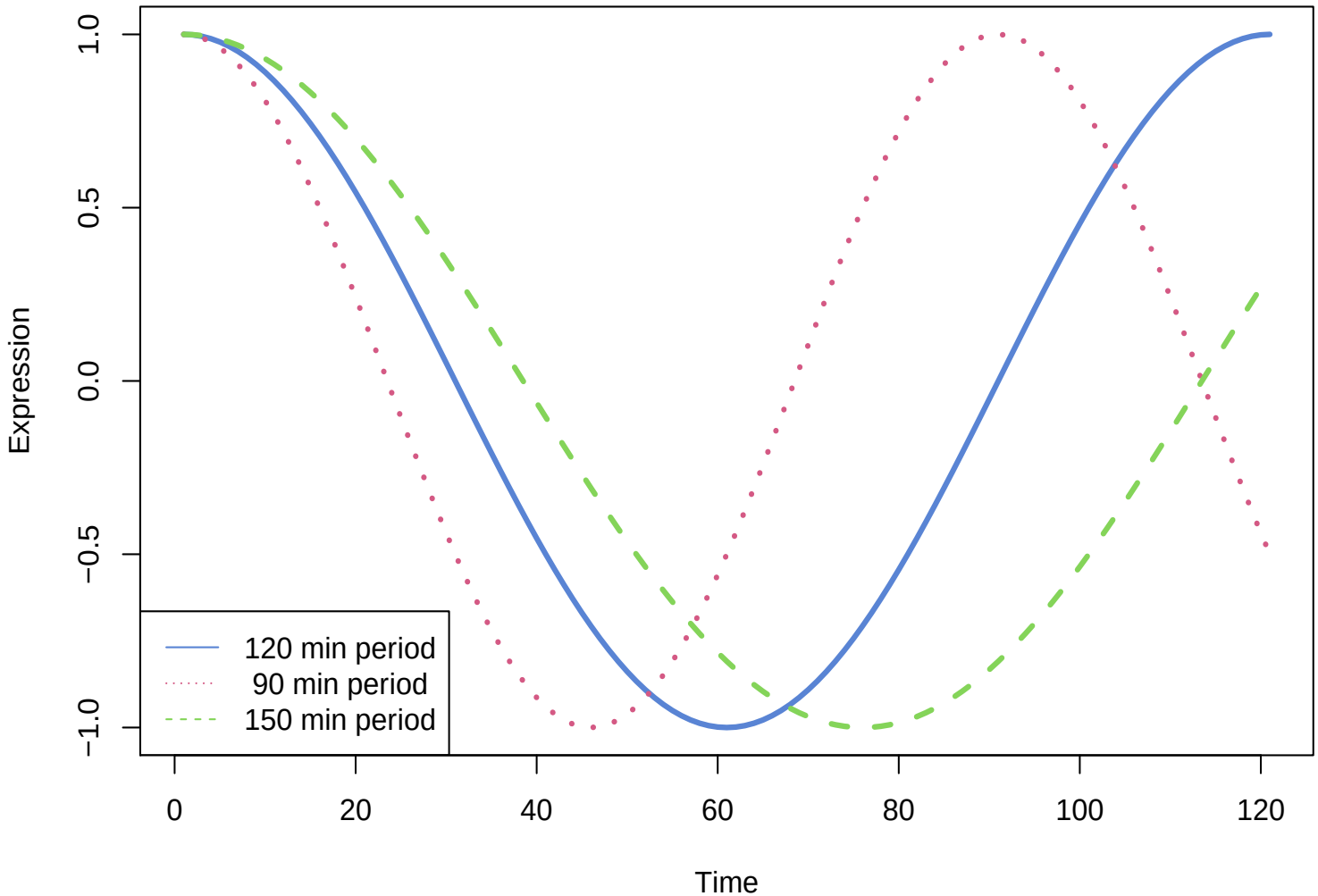
```
x <- -4:4
y <- x^2
plot(x, y, pch = 19, cex = 3, col = rainbow(length(x)))
```



Colored lines

```r
# data
Time <- 0:120
Period1 <- cos(2 * pi * Time/120)
Period2 <- cos(2 * pi * Time/90)
Period3 <- cos(2 * pi * Time/150)
Periods <- data.frame(
  Period1 = Period1,
  Period2 = Period2,
  Period3 = Period3)

# graphical parameters
line_cols <- c("#5984d4", "#d45984", "#84d459")
line_types <- c("solid", "dotted", "dashed")

# plot
matplot(Periods, type = "l", xlab = "Time", ylab = "Expression"
        , col = line_cols, lty = line_types, lwd = 3)
legend("bottomleft",
       c("120 min period", " 90 min period","150 min period"),
       col = line_cols, lty = line_types)
```

Colored lines



Colors can dramatically impact how we perceive a graphic and what we see in the data.

Why Colors?

Importance of Color

- Color isn't just about making your charts look pretty
- Color can serve as a visual cue just like the height of a bar or the position of a dot
- R provides a straighforward way to modify colors

Naming Colors

Naming colors

Specifying colors

There are various ways to specify colors in R

- by using the color's name (in English): e.g. `"turquoise"`
- by using a hexadecimal string: `"#FFAA00"`
- by using standard color space functions: e.g. `rgb()`

Function colors()

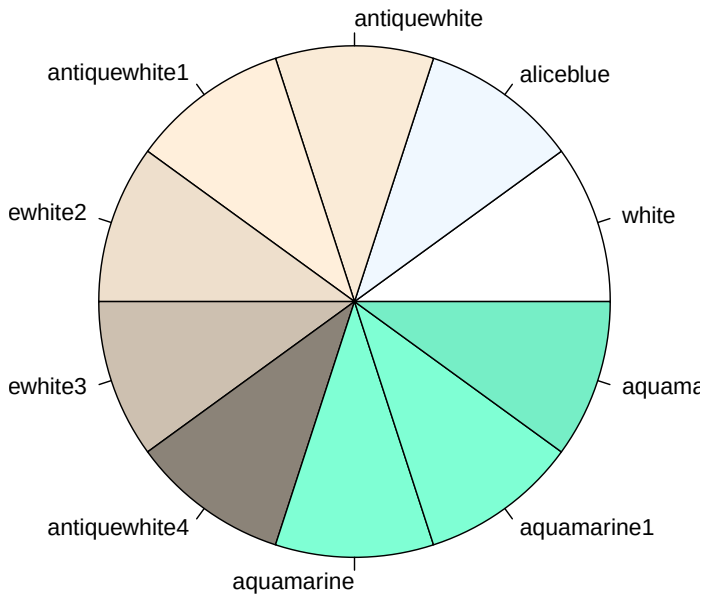
The easiest way to specify a color in R is simply to use the color's name. The R function `colors()` provides the names of 657 available colors

```
# first 30 colors
colors()[1:30]
```

```
##  [1] "white"           "aliceblue"       "antiquewhite"    "antiquewhite1"
##  [5] "antiquewhite2"   "antiquewhite3"   "antiquewhite4"   "aquamarine"
##  [9] "aquamarine1"     "aquamarine2"     "aquamarine3"     "aquamarine4"
## [13] "azure"           "azure1"          "azure2"          "azure3"
## [17] "azure4"          "beige"           "bisque"          "bisque1"
## [21] "bisque2"         "bisque3"         "bisque4"         "black"
## [25] "blanchedalmond" "blue"            "blue1"           "blue2"
## [29] "blue3"           "blue4"
```

Function colors()

```
# first 10 colors()
pie(rep(1, 10), col = colors()[1:10], labels = colors()[1:10])
```



More `colors()`

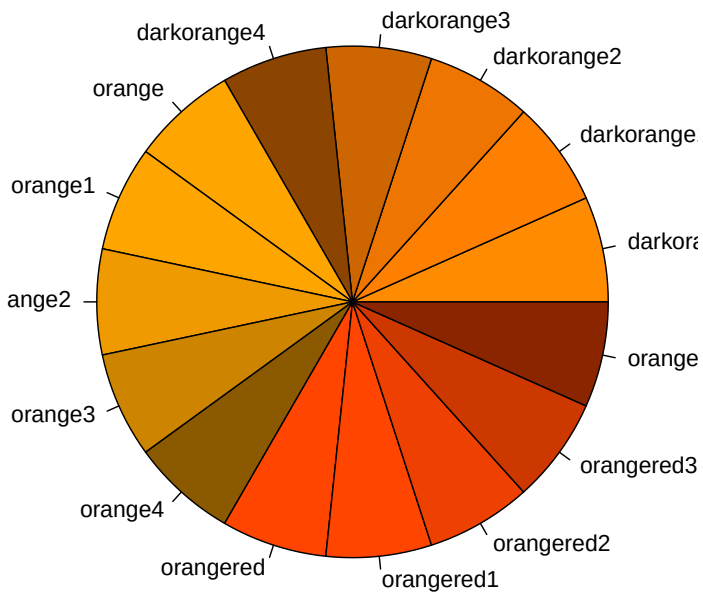
Use `grep()` to get colors of a given name

```
# orangey colors
colors()[grep("orange", colors())]
```

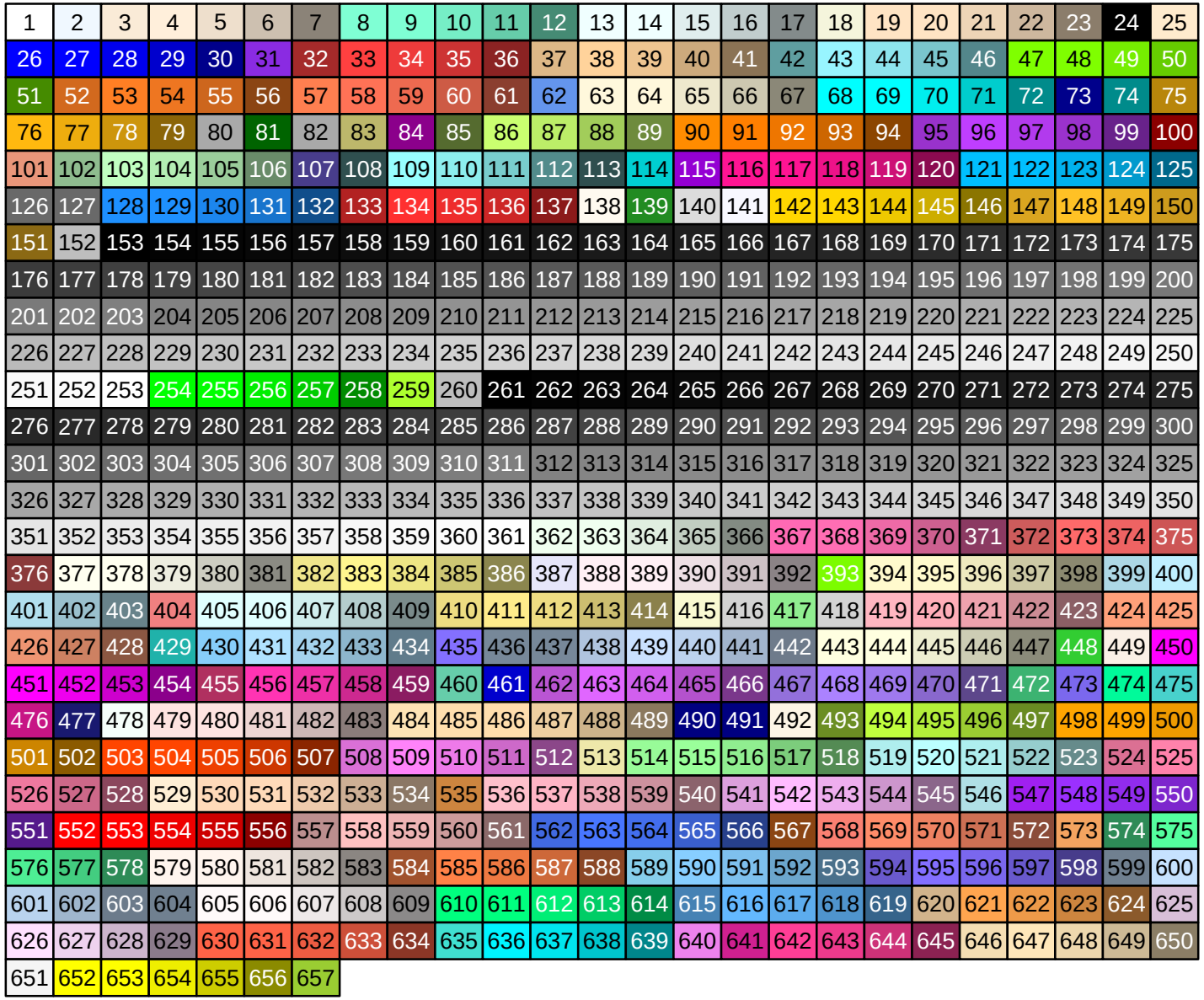
```
##  [1] "darkorange"  "darkorange1" "darkorange2" "darkorange3" "darkorange4"
##  [6] "orange"      "orange1"     "orange2"     "orange3"     "orange4"
## [11] "orangered"   "orangered1"  "orangered2"  "orangered3"  "orangered4"
```

Orangey colors()

```
# orangey colors()
oranges <- colors()[grep("orange", colors())]
pie(rep(1, length(oranges)), col = oranges, labels = oranges)
```



657 R built−in colors



```r
# Code by Earl F. Glynn
SetTextContrastColor <- function(color) {
  ifelse( mean(col2rgb(color)) > 127, "black", "white")
}
TextContrastColor <- unlist(lapply(colors(), SetTextContrastColor))

colCount <- 25 # number per row
rowCount <- 27

op <- par(mar = c(0, 0, 4, 0))
plot(c(1, colCount), c(0, rowCount), type = "n", axes=FALSE,
     ylab = "", xlab = "", ylim = c(rowCount, 0))
title("657 R built-in colors")
for (j in 0:(rowCount-1))
{
  base <- j * colCount
  remaining <- length(colors()) - base
  RowSize <- ifelse(remaining < colCount, remaining, colCount)
  rect((1:RowSize)-0.5, j-0.5, (1:RowSize)+0.5, j+0.5,
       border = "black", col = colors()[base + (1:RowSize)])
  text((1:RowSize), j, paste(base + (1:RowSize)), cex = 0.7,
       col = TextContrastColor[base + (1:RowSize)])
}
par(op)
```

http://research.stowers-institute.org/efg/R/Color/Chart/index.htm

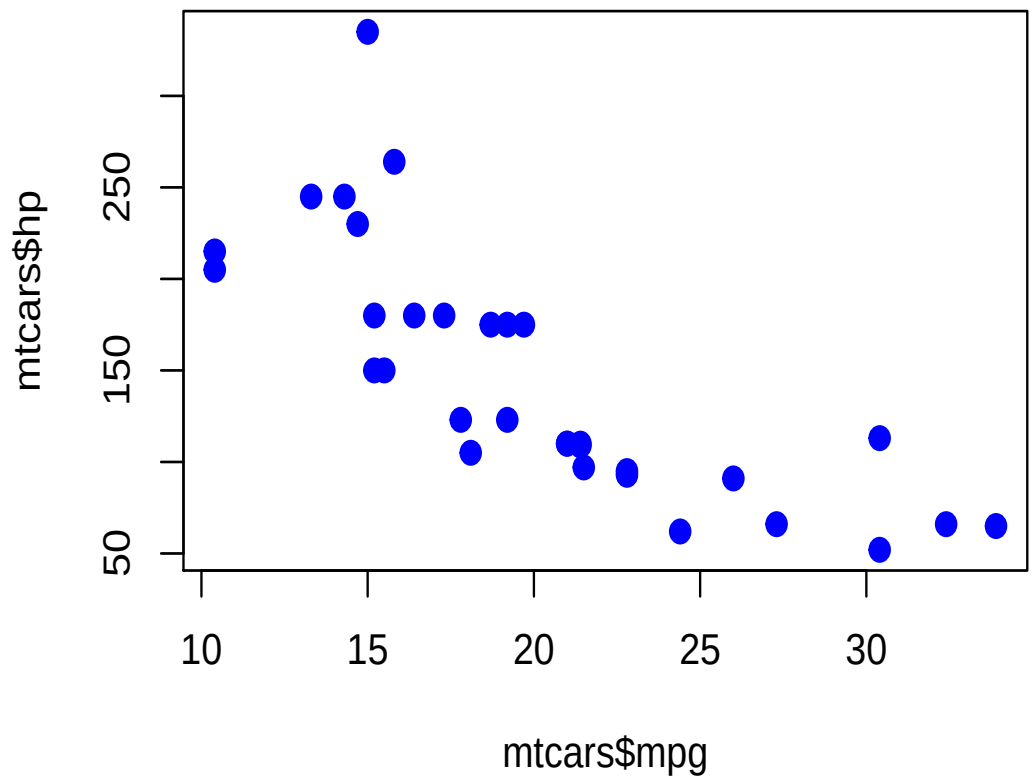
Gray colors()

Note that there is a wide range of gray (grey) colors:

```
# gray and grey colors
grays <- colors()[grep("gr[a|e]y", colors())]
length(grays)

## [1] 224

head(grays, 10)

##  [1] "darkgray"       "darkgrey"       "darkslategray"  "darkslategray1"
##  [5] "darkslategray2" "darkslategray3" "darkslategray4" "darkslategrey"
##  [9] "dimgray"        "dimgrey"
```

Example

```r
# color argument 'col'
plot(mtcars$mpg, mtcars$hp, pch = 19, col = "blue", cex = 1.2)
```



RGB Color Model

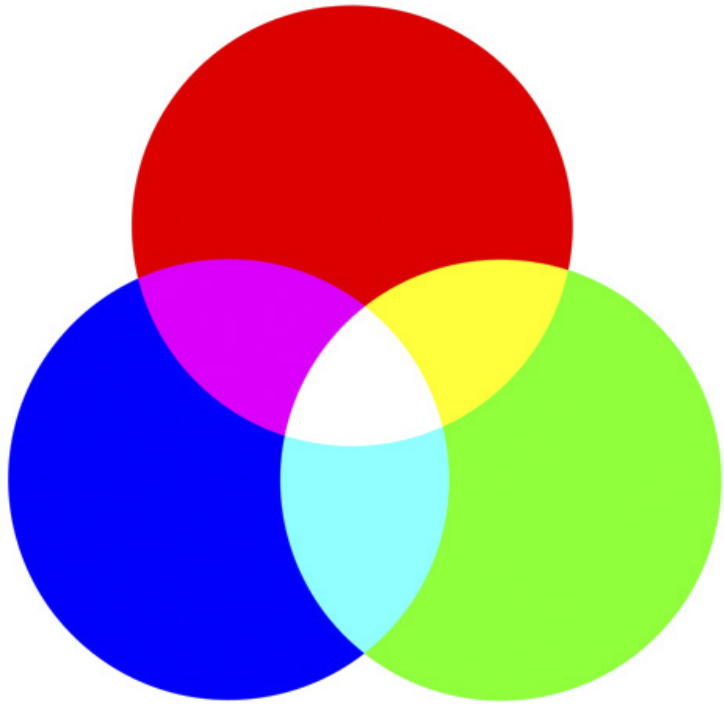
RGB Colors

About RGB color

- Computers create the colors we see on a monitor by combining 3 primary colors of light:
  - red
  - green
  - blue
- This combination is known as **RGB color model**
- Each color light is also referred to as a **channel**

Red-Green-Blue

A computer screen displays a color by combining red light, green light and blue light, the so-called RGB model.



RGB Colors

Values of RGB colors

- Any color you see on a monitor can be described by a series of 3 numbers (in the following order):
  - a red value
  - a green value
  - a blue value
- The amount of light in each color channel is typically described on a scale from 0 (none) to 255 (full-blast)
- Alternatively, scales can be provided as percent values from 0 (none) to 1 (100%)

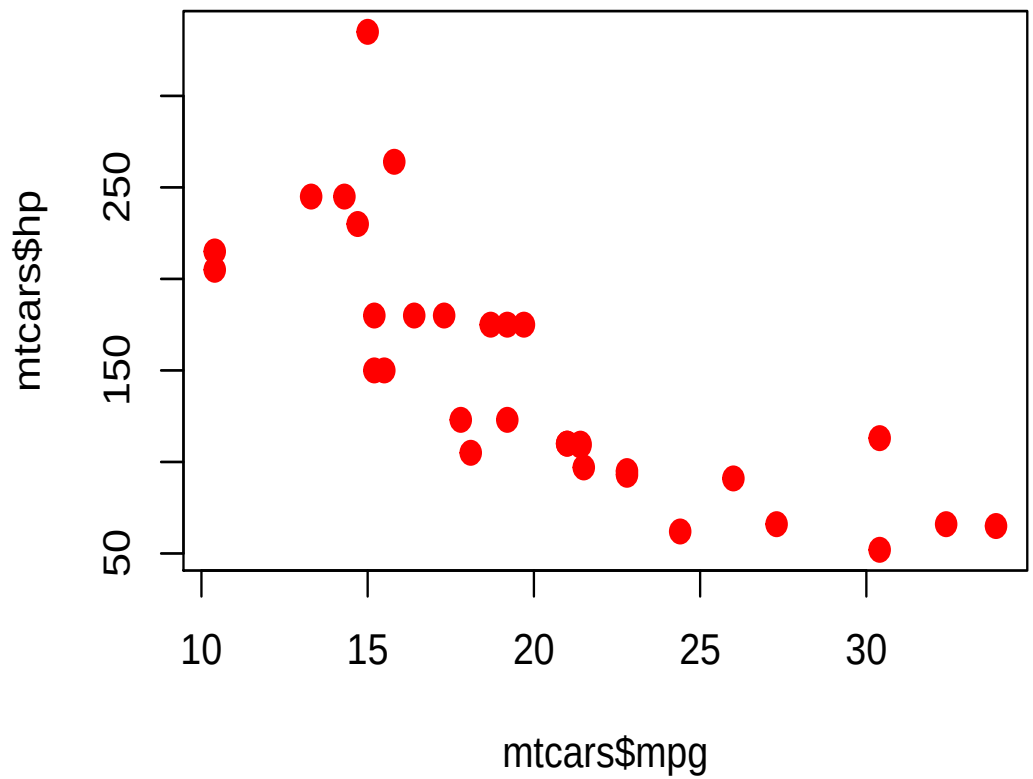
rgb() colors

R provides the function rgb() to specify RGB colors

```
# 0 to 1 (default scale)
rgb(red = 1, green = 0, blue = 0)

## [1] "#FF0000"

# 0 to 255
rgb(red = 255, green = 0, blue = 0, maxColorValue = 255)

## [1] "#FF0000"
```

Example

```
# color argument 'col'
plot(mtcars$mpg, mtcars$hp, pch = 19, cex = 1.2,
     col = rgb(255, 0, 0, max = 255))
```



RGB Colors

Some reference colors:

RGB Values	Color
`(255, 0, 0)`	red
`(0, 255, 255)`	green
`(0, 0, 255)`	blue
`(0, 0, 0)`	black
`(255, 255, 255)`	white

The closer the three values get to 255 (%100), the closer the resulting color gets to white

Colors in Hexadecimal Notation

Hex Colors

Storing RGB colors in decimal notation would require 9 digits:

255 | 255 | 255

Hex Colors

Storing RGB colors in decimal notation would require 9 digits:

255 | 255 | 255

In order to have a more efficient storage system, computers use **hexadecimal digits**

Hex Colors

Colors in hexadecimal notation

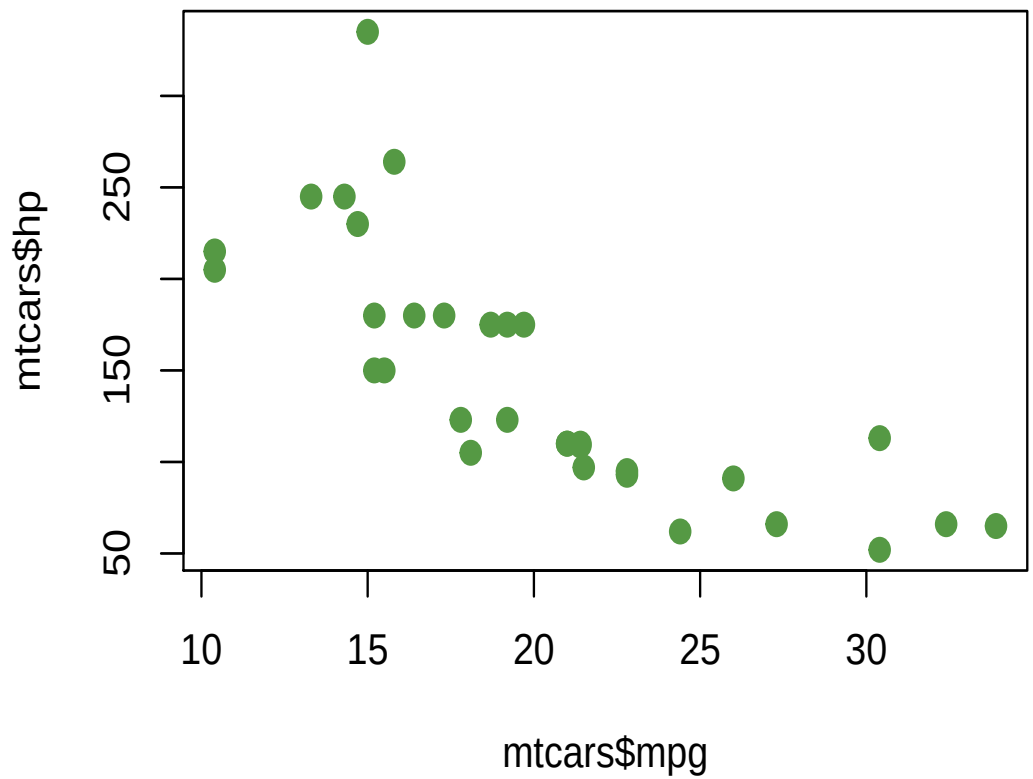
A color can also be specified as a string beginning with a hash symbol "#" and followed by six hexadecimal digits

- `"#FF0000"` (red)
- `"#00FF00"` (green)
- `"#0000FF"` (blue)
- `"#FF6347"` (tomato)

This is actually the output format of `rgb()`

Example

```
# color argument 'col'
plot(mtcars$mpg, mtcars$hp, pch = 19, cex = 1.2,
     col = "#559944")
```



Hexadecimal code

R uses the hexadecimal system to name colors

- Hexadecimal: numeral system with base 16, or *hex*
- The hexadecimal representation uses 16 different symbols
- The 16 symbols are all 10 digits (0-9) and 6 first letters (A-F)
- The digits 0-9 represent values zero to nine
- The letters A, B, C, D, E, F (or a, b, c, d, e, f) represent values ten to fifteen

Hexadecimal Notation

Decimal and Hexadecimal

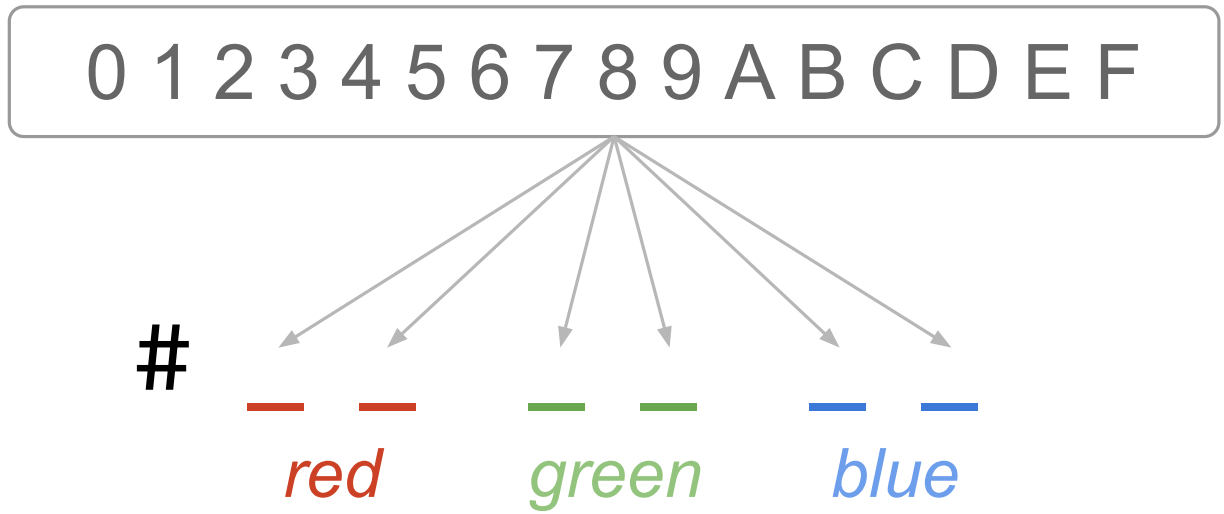
```
##                1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
## decimal        0 1 2 3 4 5 6 7 8  9 10 11 12 13 14 15
## hexadecimal 0 1 2 3 4 5 6 7 8  9  A  B  C  D  E  F
```

Two hexadecimal digits together can make $16 \times 16 = 256$ different values (from 0 to 255)
Six hexadecimal digits together can make $16^6 = 16,777,216$ different values.

Hexadecimal code

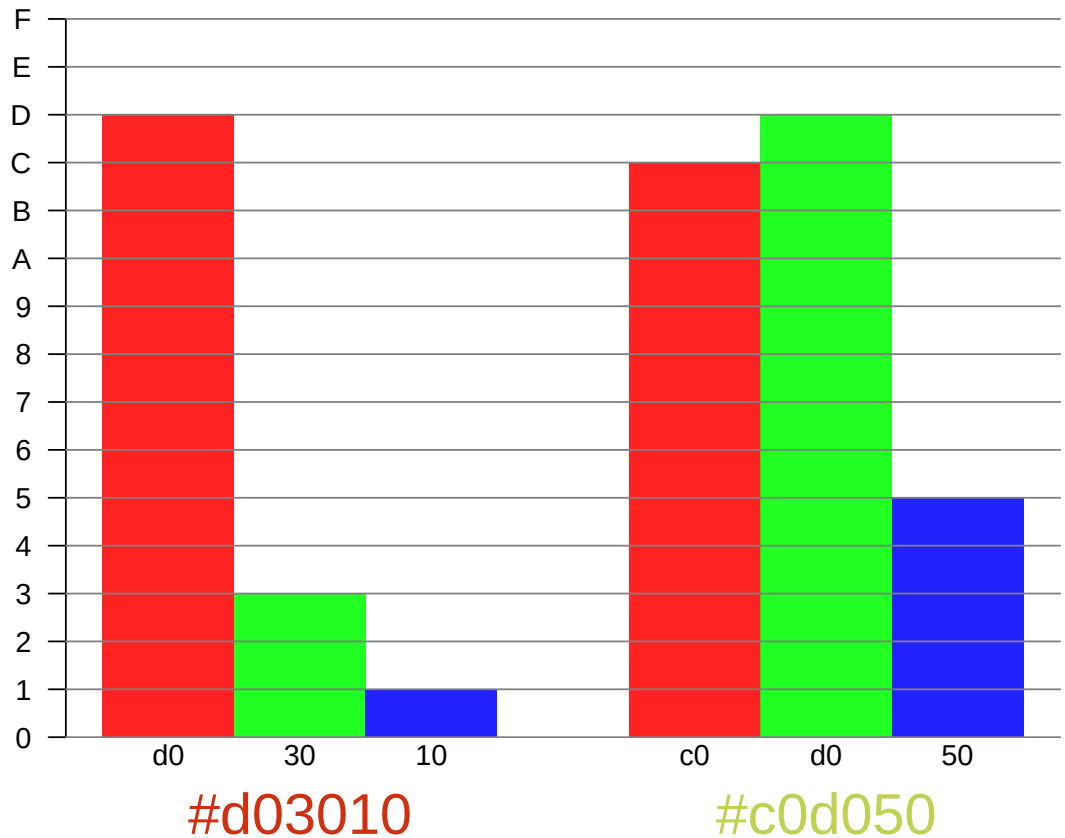
Hexadecimal color: #975015

- # declares that this "is a hex number"
- The other six are really three sets of pairs
- Each pair controls one primary additive color
- The first pair corresponds to red
- The second pair corresponds to green
- The last pair corresponds to blue



There are 256 possible shades each of red, green, and blue

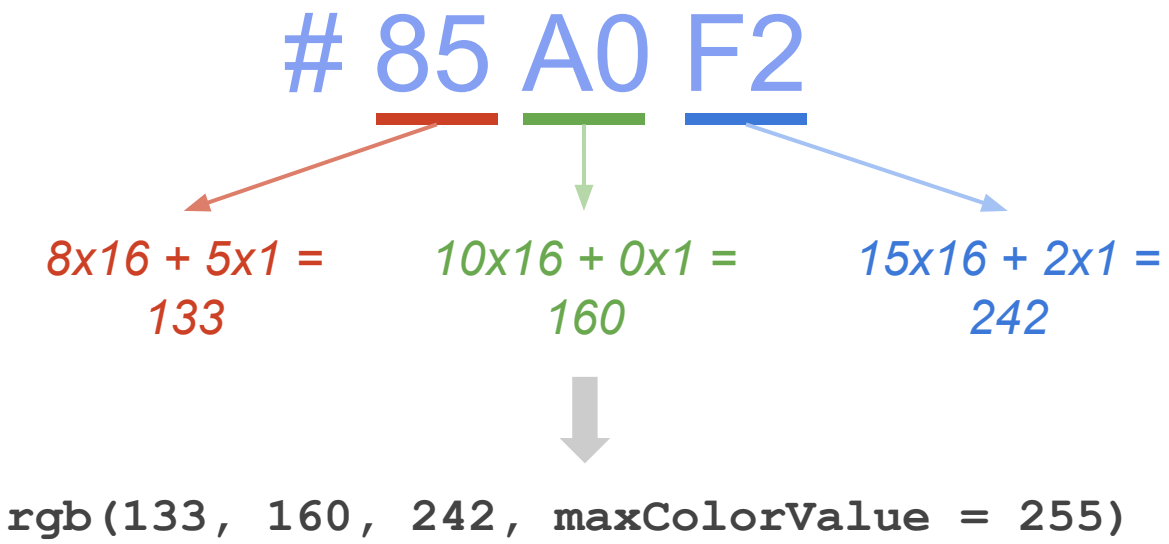
RGB Hexadecimal Notation



Hexadecimal code

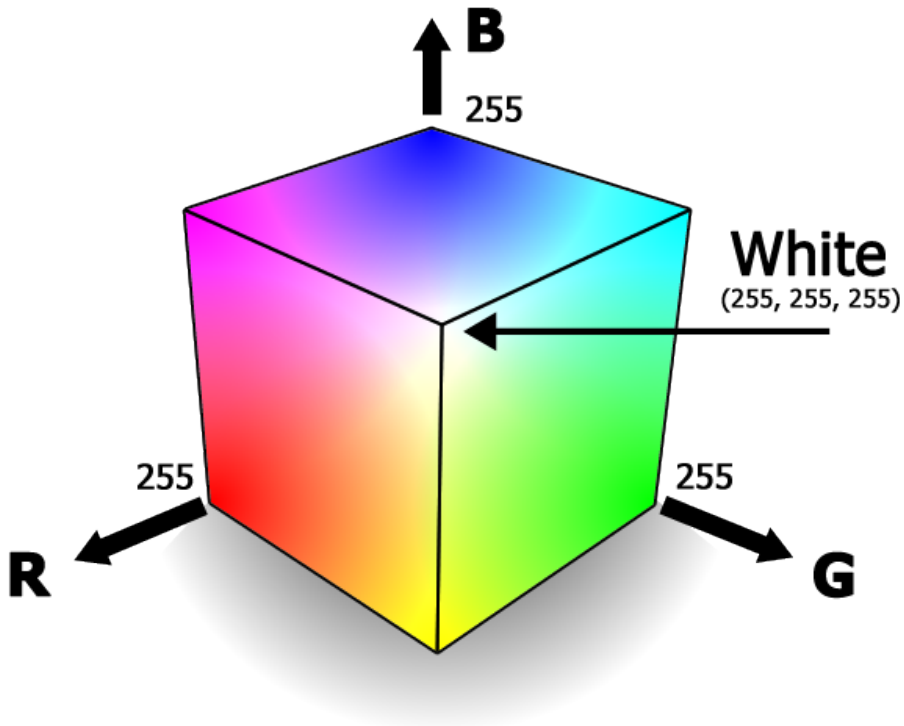
Hexadecimal digits

- **0** is the smallest representation of a color (absence of color)
- **F** is 15 times the intensity of color 0
- **00** is equal to zero hue
- **FF** is equal to a pure color
- **#000000** black
- **#FFFFFF** white
- equal digits produce a shade of gray




```
rgb(133, 160, 242, maxColorValue = 255)
```

RGB Cube Representation



http://drmoron.org/is-black-a-color/

RGB Inconvenience

RGB drawbacks

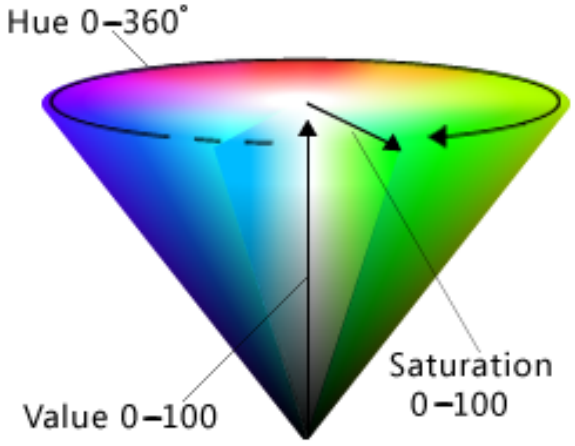
- The RGB color model is the most commonly used
- However, specifying RGB colors in not intuitive
- It is no straightforward how to make a color stronger, darker or lighter with RGB values
- It is also hard to "read" RGB values (and being able to identify the corresponding hue)

Cylindrical-coordinate Representations

Cylindrical Models

- An alternative to RGB values is the **HSV** model
- HSV: Hue (color), Saturation, Value
- HSV rearranges the geometry of RGB following cyclindrical coordinates

HSV representation



Cylindrical-coordinate Representations

HSV Models

- Hue values are measured in degrees around the circle
  - Red at 0 degrees
  - Green at 120°
  - Blue at 240°
  - other colors in between
- Saturation is a percentage value from 0 (gray) to 1 (full blast)
- Value is also a percentage value from 0 (darkest) to 1 (lightest)

Function `hsv()`

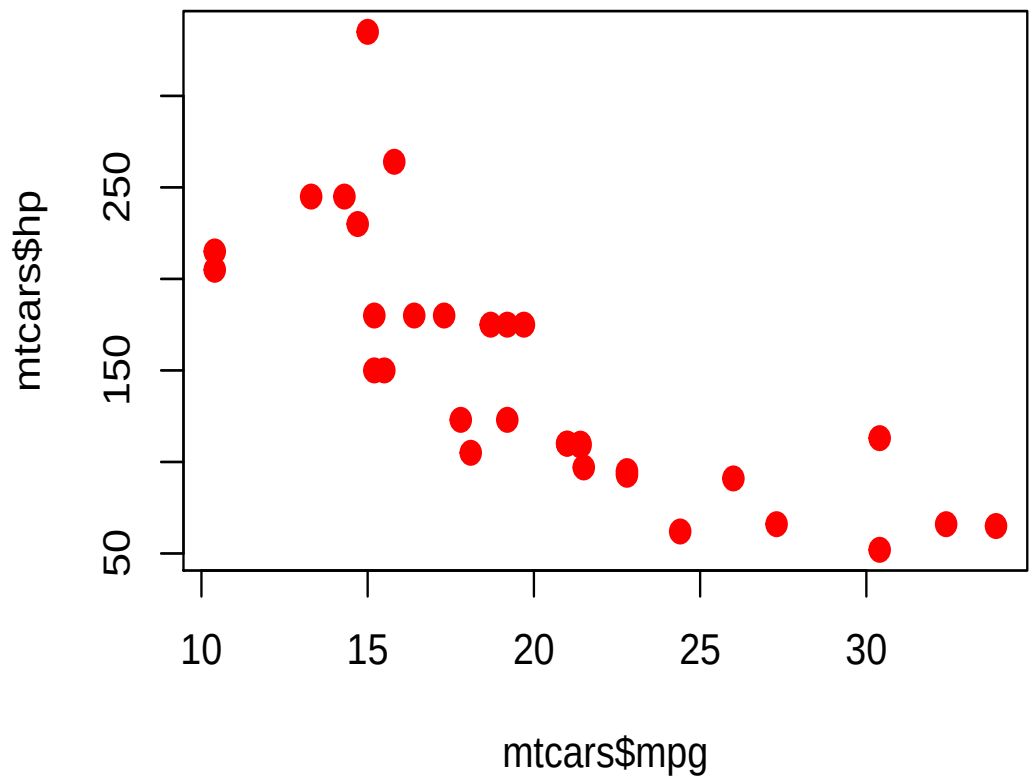
R provides the function `hsv()` which takes an HSV triplet

```
hsv(h = 1, s = 0.8, v = 0.9)
```

Note that the Hue value ranges from 0 (0 degrees) to 1 (360 degrees)

Example

```r
# hsv color
plot(mtcars$mpg, mtcars$hp, pch = 19, cex = 1.2,
     col = hsv(h = 1, s = 1, v = 1))
```



HSV

About HSV

- HSV is a more intuitive system
- HSV is also more perceptually relevant
- Once you select a hue, it is easy to make it stronger, darker, or lighter

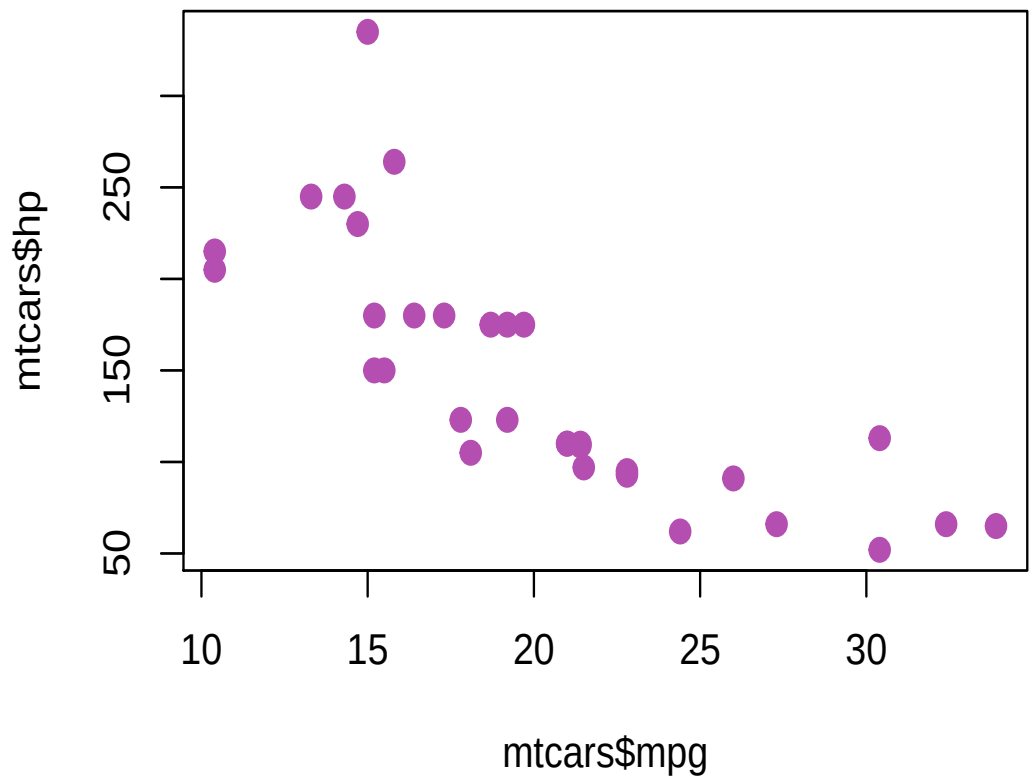
Other Representations

HCL Model

- Another model is **HCL**
- HCL: Hue, Chroma, Luminance
- Hue values are measured in degrees around the circle
  - Red at 0 degrees
  - Green at 120°
  - Blue at 240°
  - other colors in between
- Luminance is also a percentage value from 0 (darkest) to 100 (lightest)
- Chroma depends on Hue and Luminance

Example

```
# hcl color
plot(mtcars$mpg, mtcars$hp, pch = 19, cex = 1.2,
     col = hcl(h = 3550, c = 75, l = 50))
```



Color Space

HSV Colors by space model

R provides the functions `hsv()` and `hcl()`

- HSV (Hue, Saturation, Value) triplet:
  `hsv(h = 1, s = 0.8, v = 0.9)`
- HCL (Hue, Chroma, Luminance) triplet:
  `hcl(h = 1, c = 35, l = 85)`

Color Space Models

Color Model

A **color model** is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components.

About colors

RGB Model

The **RGB model** uses an additive color mixing with primary colors of red, green, and blue, each of which stimulates one of the three types of the eye's color receptors.

RGB usefulness

Mixtures of light of these primary colors cover a large part of the human color space and thus produce a large part of human color experiences.

Semitransparent Colors

Semitransparent Colors

Color transparency

All R colors are stored with an **alpha** transparency channel.

- An alpha value of 0 means fully transparent
- An alpha value of 1 means fully opaque

Semitransparent Colors

Alpha values

When using any of the color space functions, transparency is indicated with the parameter `alpha`:

- `rgb(1, 0, 0, alpha = 0.5)`
- `hsv(h = 1, s = 0.8, v = 0.8, alpha = 0.5)`
- `hcl(h = 0, c = 35, l = 85, alpha = 0.5)`

Semitransparent Colors

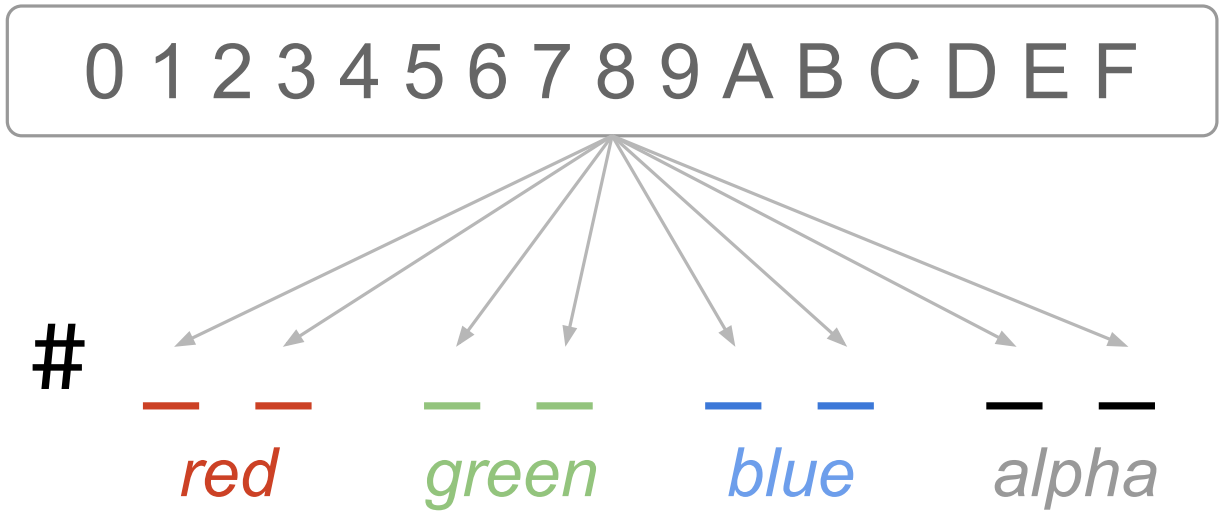
Hex color transparency

When using hexadecimal notation, transparency is indicated by using a hexadecimal string of eight `digits`. The last two digits indicate transparency:

- a hex digit `"00"` indicates an alpha value of 0 (fully transparent)
- a hex digit `"FF"` indicates an alpha value of 1 (fully opaque)

For example, `"#FFA50080"` specifies a semitransparent orange. Note that `"#FFA500"` is equivalent to `"#FFA500FF"`

Hex notation with alpha channel



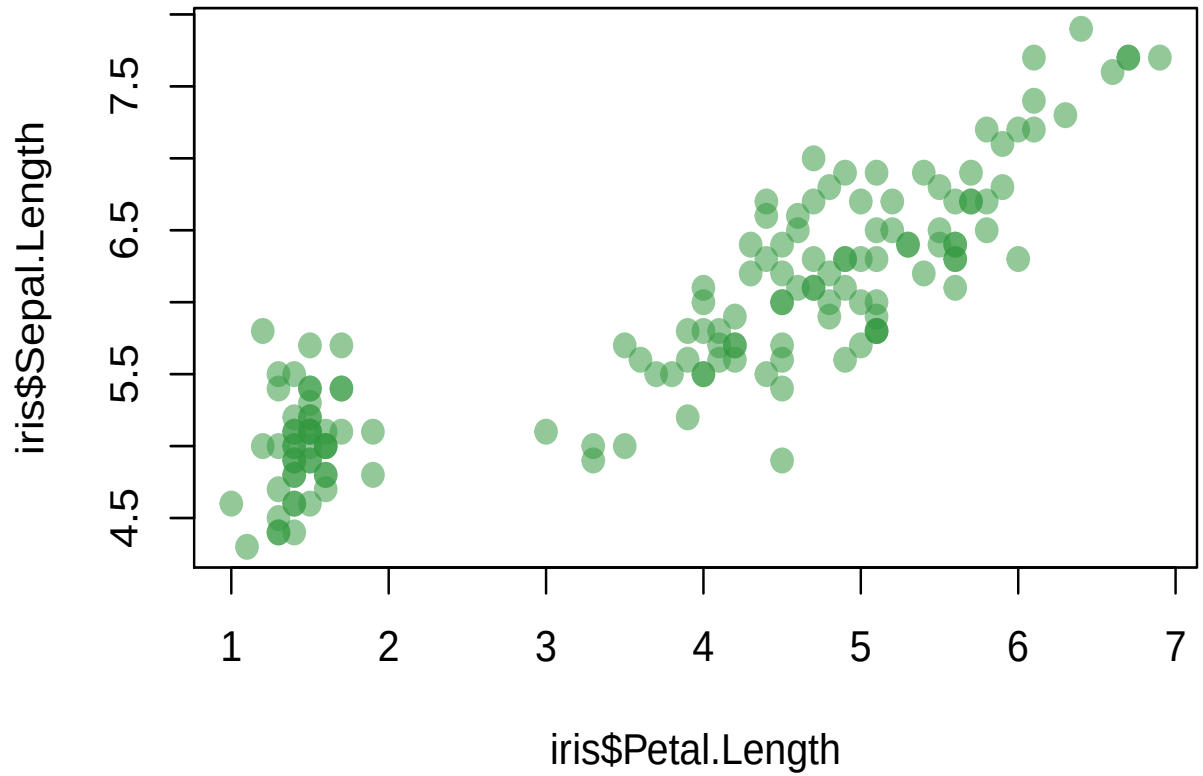
Transparency

Use transparency for overlapping data

- The number of lines or points on a graph can obscure other values that lie behind them.
- One option to present many overlapping values is to make the color values partially transparent

Example

```
# transparent color
plot(iris$Petal.Length, iris$Sepal.Length,
     pch = 19, cex = 1.2, col = "#33994088")
```



Converting Colors

Converting colors functions

R provides a set of functions for converting between different color spaces

function	package
`col2rgb()`	`"grDevices"`
`hex2RGB()`	`"colorspace"`
`convertColor()`	`"grDevices"`

```r
# turquoise into RGB
col2rgb("turquoise")
```

```
##       [,1]
## red     64
## green  224
## blue   208
```

```r
# turquoise (hex) into RGB
library("colorspace")
hex2RGB("#40E0D0")
```

```
##                R         G         B
## [1,] 0.2509804 0.8784314 0.8156863
```

```r
# color 'red' sRGB into Luv
convertColor(t(col2rgb("red")/255), from = "sRGB", to = "Luv")
```

```
##               L        u        v
## [1,] 53.48418 175.3647 37.80017
```

Color Palettes

Color Sets

Converting colors functions

Usually more than one color is required within a single plot, and in such cases we need to select colors that are aesthetically pleasing or related in some way. For this, R provides several ways to specifcy or create color schemes and palettes.

R Default Palettes

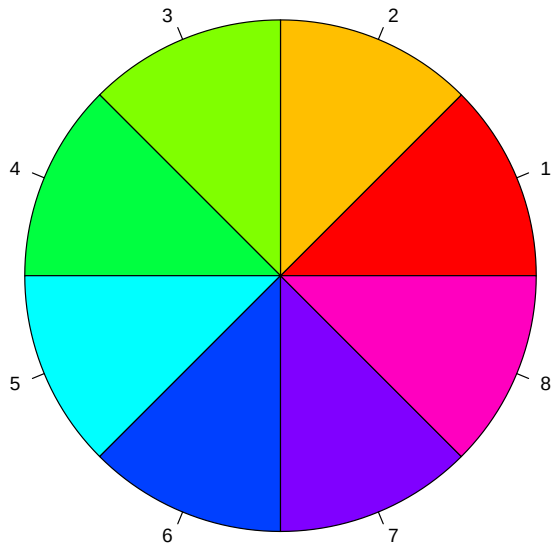
Basic Color Palettes

R provides a set of functions to generate basic color sets.

- `rainbow()`
- `heat.colors()`
- `topo.colors()`
- `terrain.colors()`
- `cm.colors()`
- `gray.colors()`

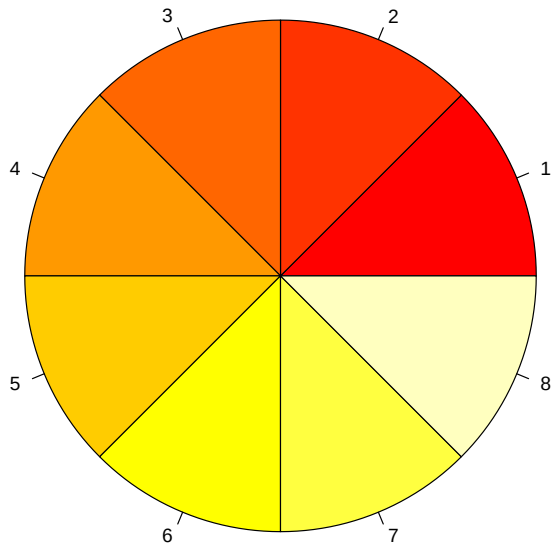
Function rainbow()

```
# 8 colors rainbow
n <- 8
pie(rep(1, n), col = rainbow(n))
```



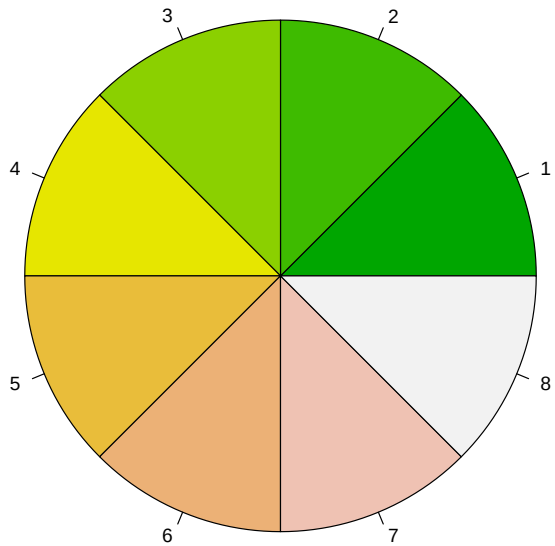
Function heat.colors()

```
# heat colors (8 values)
n <- 8
pie(rep(1, n), col = heat.colors(n))
```



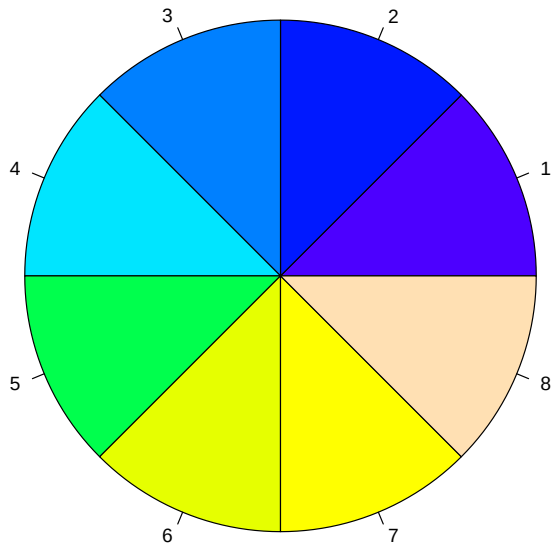
Function terrain.colors()

```r
# terrain colors (8 values)
n <- 8
pie(rep(1, n), col = terrain.colors(n))
```



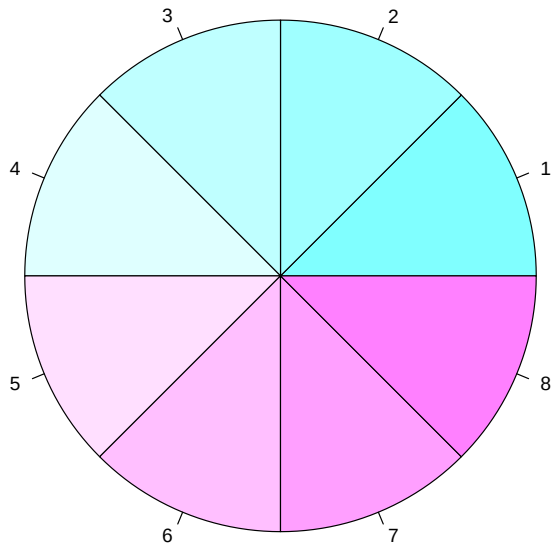
Function topo.colors()

```r
# topo colors (8 values)
n <- 8
pie(rep(1, n), col = topo.colors(n))
```



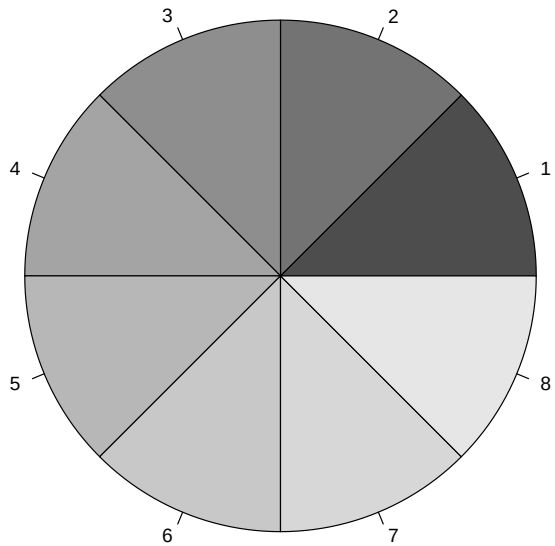
Function cm.colors()

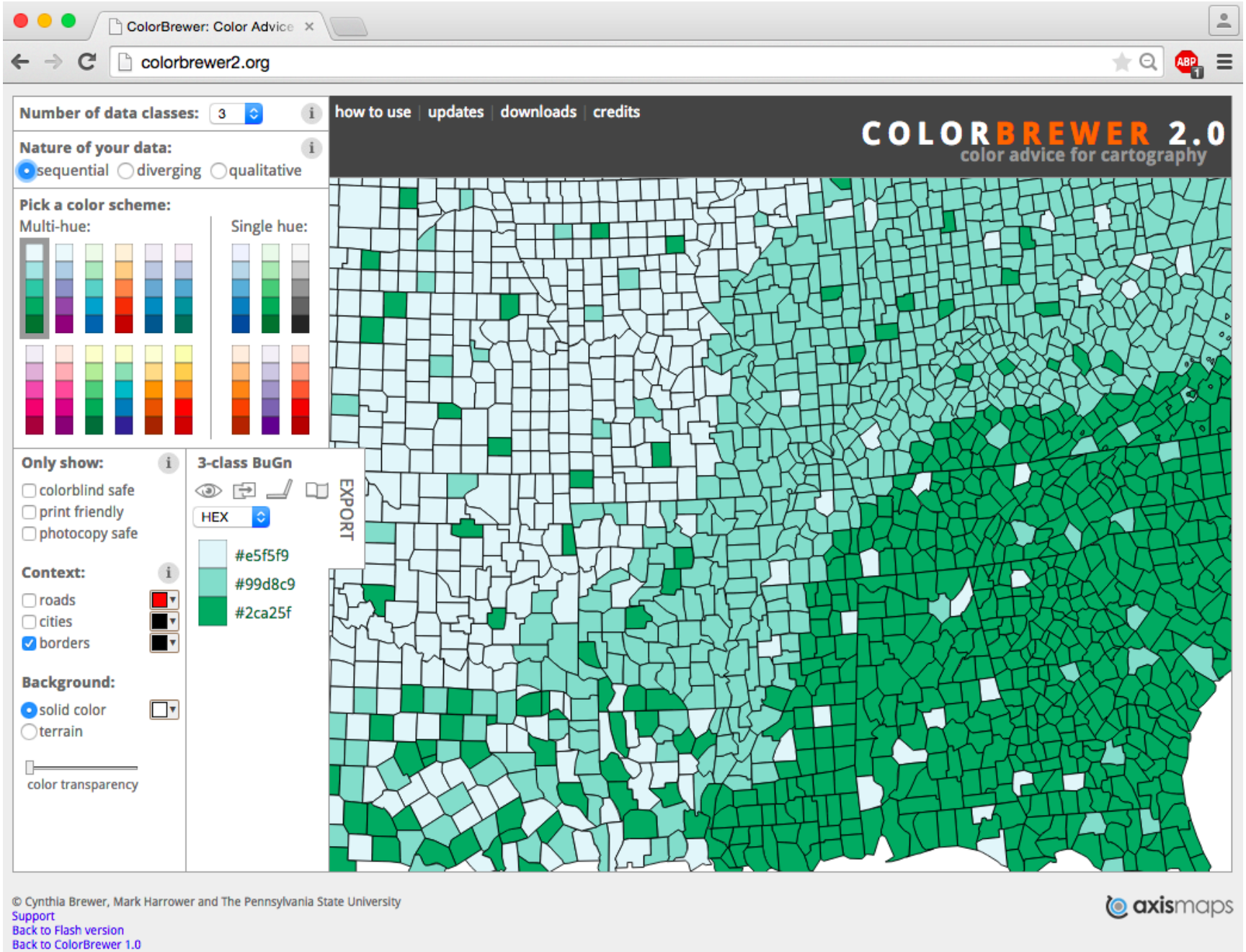
```
# cyan-magenta colors (8 values)
n <- 8
pie(rep(1, n), col = cm.colors(n))
```



Function gray.colors()

```r
# gray colors (8 values)
n <- 8
pie(rep(1, n), col = gray.colors(n))
```



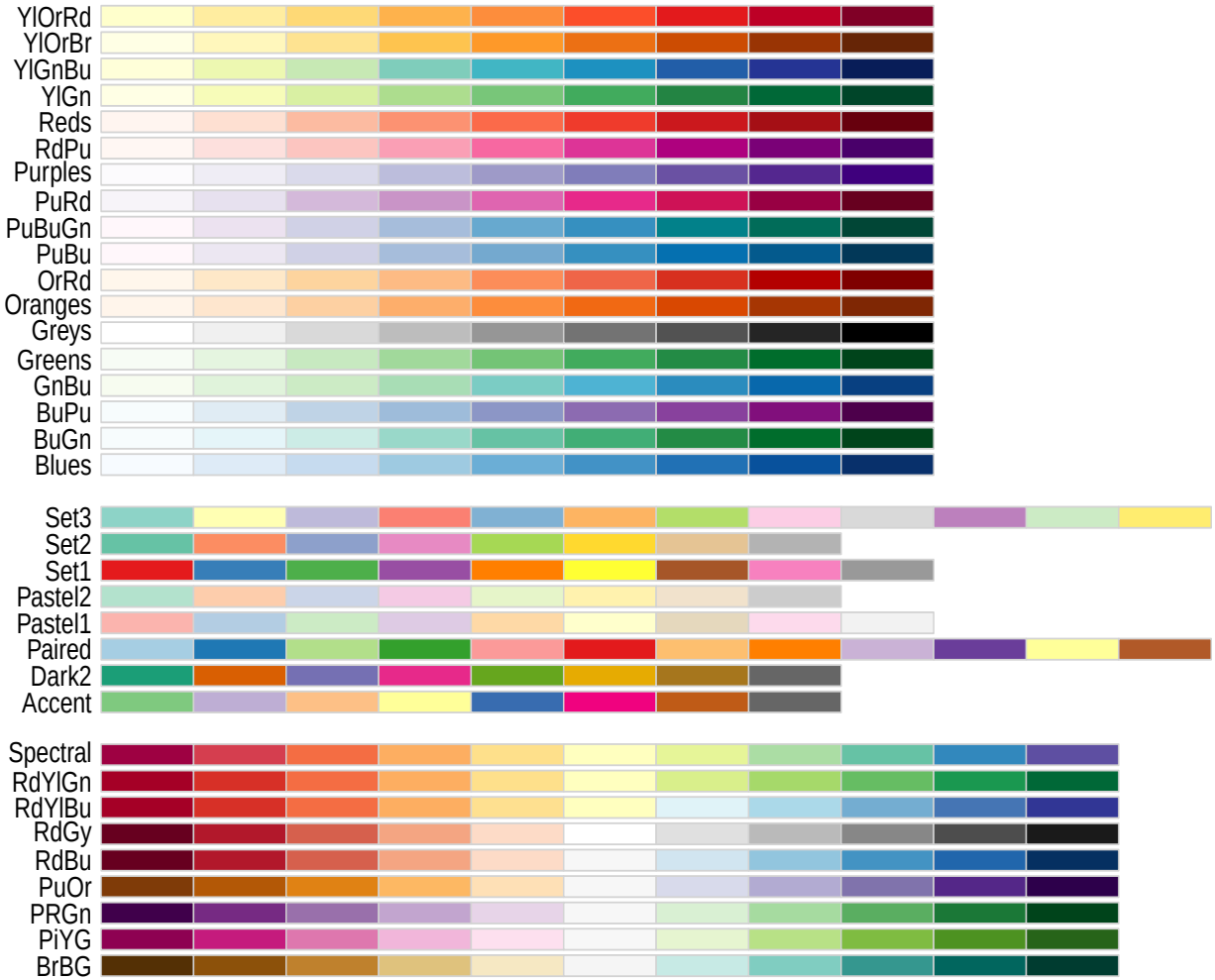

ColorBrewer: Color Advice
colorbrewer2.org
Number of data classes: 3
Nature of your data:
sequential diverging qualitative
Pick a color scheme:
Multi-hue:
Single hue:
Only show:
colorblind safe
print friendly
photocopy safe
Context:
roads
cities
borders
Background:
solid color
terrain
color transparency
3-class BuGn
HEX
#e5f5f9
#99d8c9
#2ca25f
EXPORT
how to use | updates | downloads | credits
COLORBREWER 2.0
color advice for cartography
© Cynthia Brewer, Mark Harrower and The Pennsylvania State University
Support
Back to Flash version
Back to ColorBrewer 1.0
axismaps

Color Brewer

The R package "RColorBrewer" (by Erich Neuwirth) provides nice color schemes designed by Cynthia Brewer and described at http://colorbrewer2.org

```
# remember to install RColorBrewer first!
library(RColorBrewer)

# display available schemes
display.brewer.all()
```

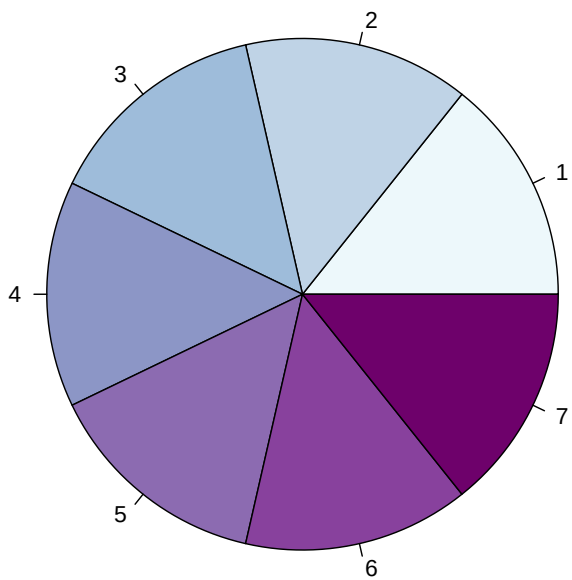
R Color Brewer Schemes



Brewer palette

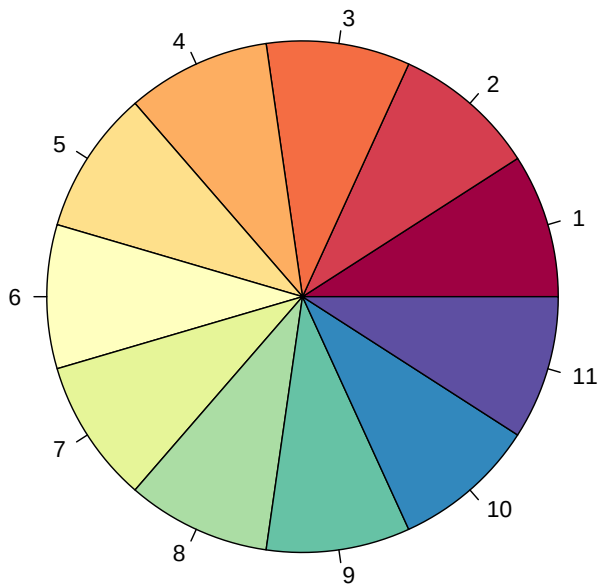
The main function of `"RColorBrewer"` is `brewer.pal()` that allows you to select a color palette by specifying the name and size of the palette.

```
# palette "BuPu" (blue-purple)
n <- 7
pie(rep(1, n), col = brewer.pal(n, "BuPu"))
```



Brewer palette

```r
# palette "Spectral"
n <- 11
pie(rep(1, n), col = brewer.pal(n, "Spectral"))
```



More about color spaces

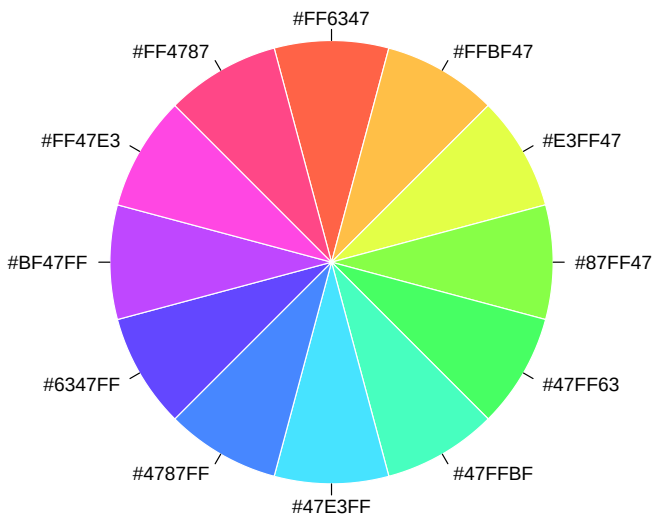
- The package `"colorspace"` provides functions to create colors in a variety of color spaces, plus functions to convert between color spaces.
- Another useful package is `"munsell"`, which is designed on the Munsell color space.
- There is also the `"dichromat"` package that provides a series of palettes that are suitable for people with color blindness deficiencies.

Color Wheel

Color Wheel

Understanding the Color Wheel

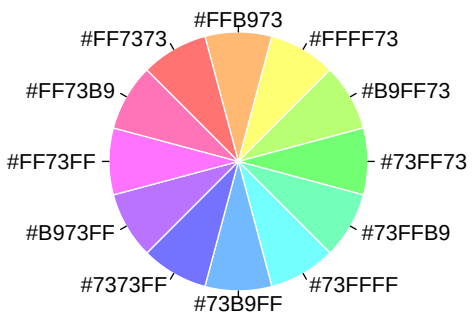
The color wheel helps you visualize the relationships that colors have to one another. The wheel shows different colors evenly apart.



Color Wheel

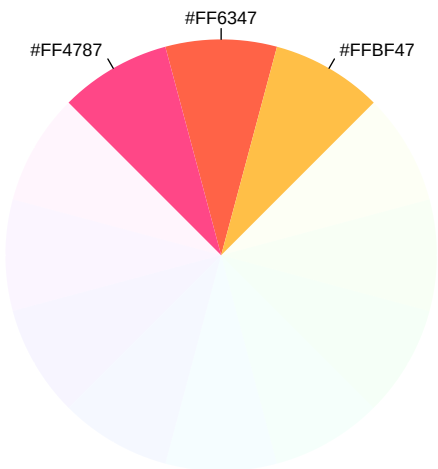
The R package "colortools" provides the function wheel(). The function takes the name of a color, and produces a corresponding color wheel with the specified number of slices.

```
# remember to install colortools first!
library(colortools)
# color wheel for 'tomato'
wheel("#FFB973", bg = "white")
```



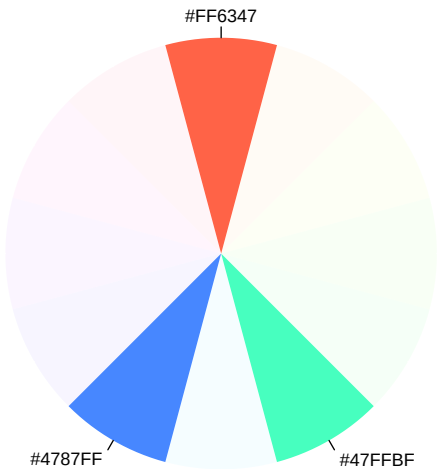

```
##  [1] "#FFB973" "#FFFF73" "#B9FF73" "#73FF73" "#73FFB9" "#73FFFF" "#73B9FF"
##  [8] "#7373FF" "#B973FF" "#FF73FF" "#FF73B9" "#FF7373"
```

```r
# Adjacent (analogous)
adjacent("tomato", title = FALSE)
```

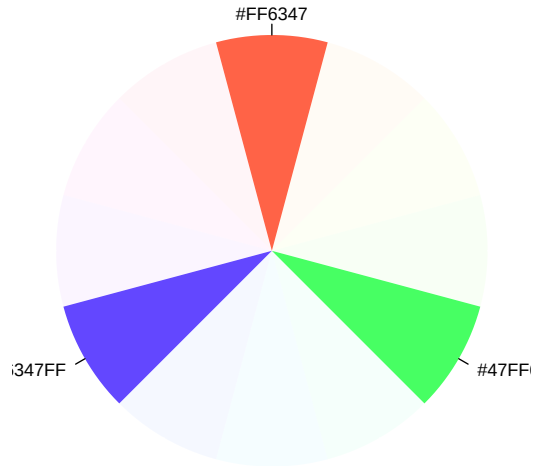



```r
# Complementary
complementary("tomato", title = FALSE)
```

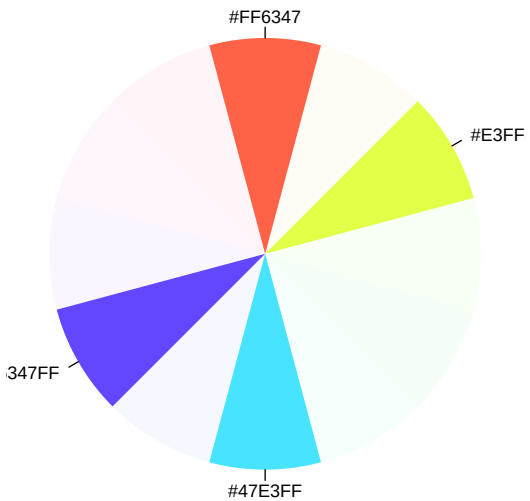
```r
# split complementary
splitComp("tomato", title = FALSE)
```



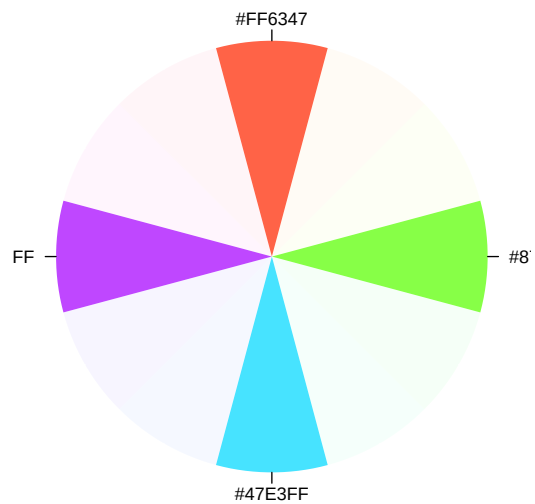

```r
# triadic
triadic("tomato", title = FALSE)
```



```r
# tetradic
tetradic("tomato", title = FALSE)
```




```r
# square
square("tomato", title = FALSE)
```



Ramp Palettes

`colorRampPalette()`

R also offers the functions `colorRamp()` and `colorRampPalette()`. They are not color set generators, but color set **function** generators.

Ramp Palettes

colorRamp()

colorRamp() produces a function for creating colors based on a sequence of values in the range 0 to 1. The output is a numeric matrix of RGB color values

```
red_green <- colorRamp(c("red", "green"))
red_green( (0:4)/4 )
```

```
##        [,1]   [,2] [,3]
## [1,] 255.00   0.00    0
## [2,] 191.25  63.75    0
## [3,] 127.50 127.50    0
## [4,]  63.75 191.25    0
## [5,]   0.00 255.00    0
```

Ramp Palettes

`colorRampPalette()`

`colorRampPalette()` produces a function that generates `n` colors

```
orange_blue <- colorRampPalette(c("orange", "blue"))
orange_blue(5)

## [1] "#FFA500" "#BF7B3F" "#7F527F" "#3F29BF" "#0000FF"
```

Color appearance

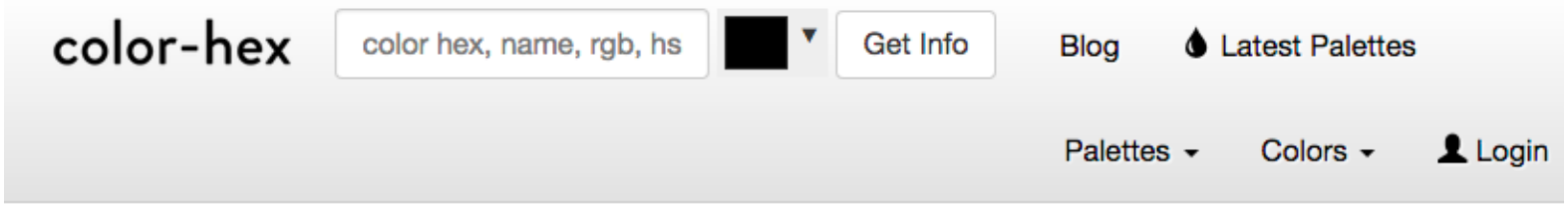
The final appearance of a color can vary considerably depending on whether it appears on a screen, printed on paper, or displayed through a projector.

Device Dependency of Color

The colors that R sends to a graphics device are **sRGB** colors; the reason is that most computer monitors are set up to work with sRGB model.

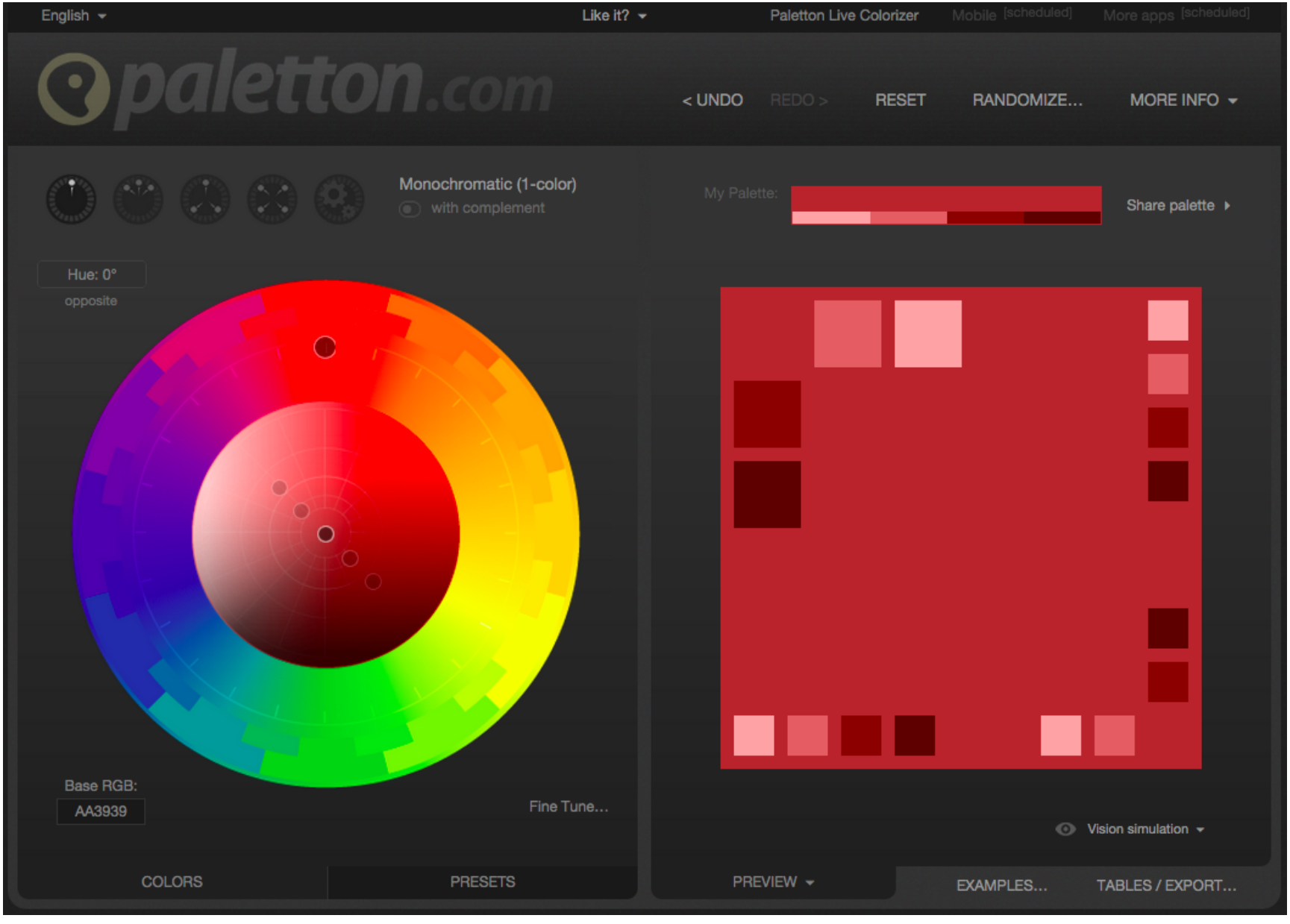
Some Resources

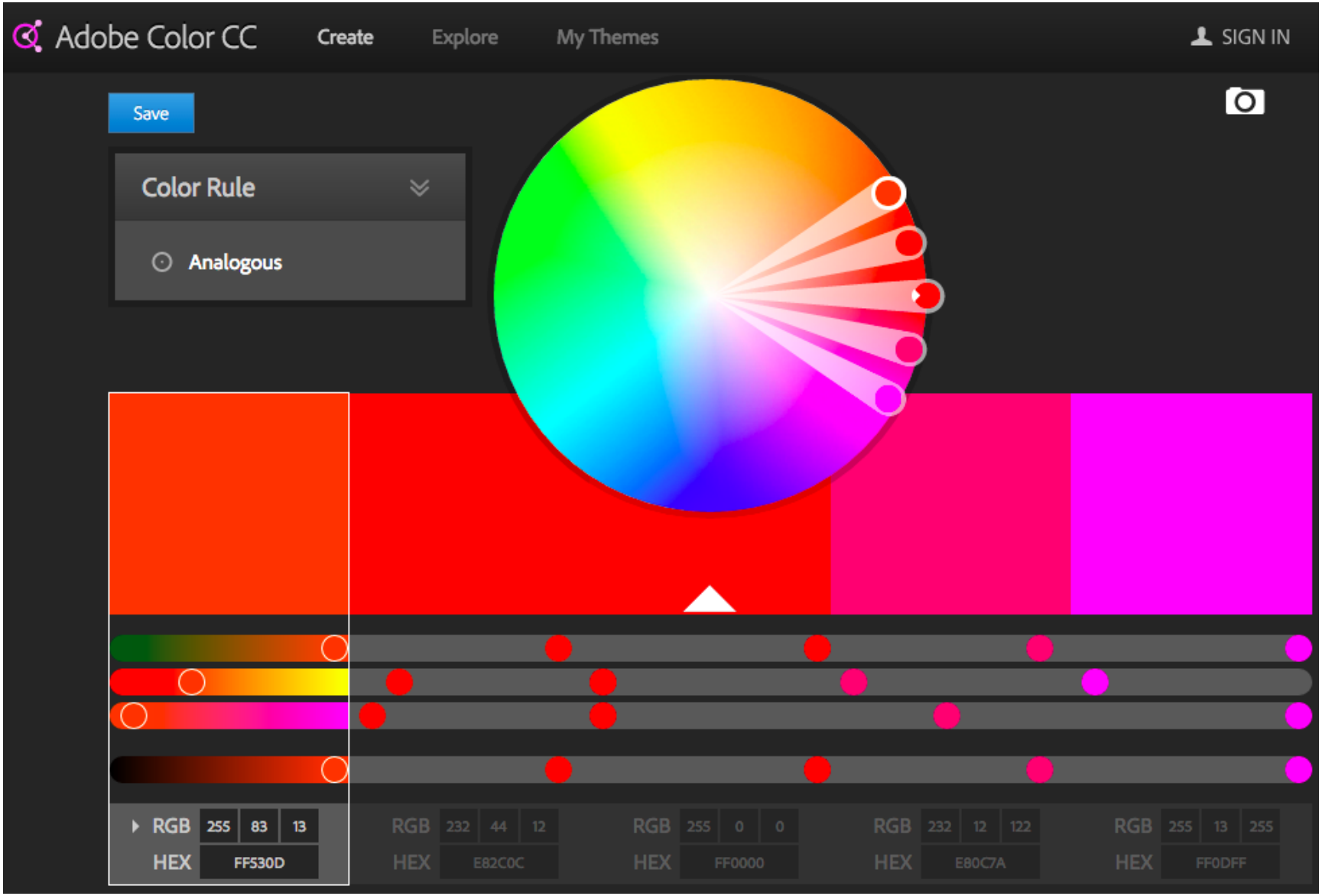
- Color-Hex
  http://www.color-hex.com/
- Paletton
  http://paletton.com
- Adobe color scheme
  https://color.adobe.com/create/color-wheel/



Color-hex gives information about **colors** including color models (RGB,HSL,HSV and CMYK), Triadic colors, monochromatic colors and analogous colors calculated in color page. Color-hex.com also generates a simple css code for the selected color. Html element samples are also shown below the color detail page. Simply type the 6 digit color code in the box above and hit enter.

Users Latest Favorite Colors


English
Like it?
Paletton Live Colorizer
Mobile [scheduled]
More apps [scheduled]
paletton.com
< UNDO
REDO >
RESET
RANDOMIZE...
MORE INFO
Monochromatic (1-color)
with complement
My Palette:
Share palette
Hue: 0°
opposite
Base RGB:
AA3939
Fine Tune...
Vision simulation
COLORS
PRESETS
PREVIEW
EXAMPLES...
TABLES / EXPORT...


Adobe Color CC
Create
Explore
My Themes
SIGN IN
Save
Color Rule
Analogous
RGB 255 83 13
HEX FF530D
RGB 232 44 12
HEX E82C0C
RGB 255 0 0
HEX FF0000
RGB 232 12 122
HEX E80C7A
RGB 255 13 255
HEX FF0DFF