R color cheatsheet

Finding a good color scheme for presenting data can be challenging. This color cheatsheet will help!

R uses hexadecimal to represent colors

Hexadecimal is a base-16 number system used to describe color. Red, green, and blue are each represented by two characters (#rrggbb). Each character has 16 possible symbols: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F:

"00" can be interpreted as 0.0 and "FF" as 1.0
i.e., red= #FF0000 , black=# 000000, white = #FFFFFF

Two additional characters (with the same scale) can be added to the end to describe transparency (#rrggbbaa)

R has 657 built in color names

To see a list of names:
colors()
These colors are displayed on P. 3.

Example:

peachpuff4

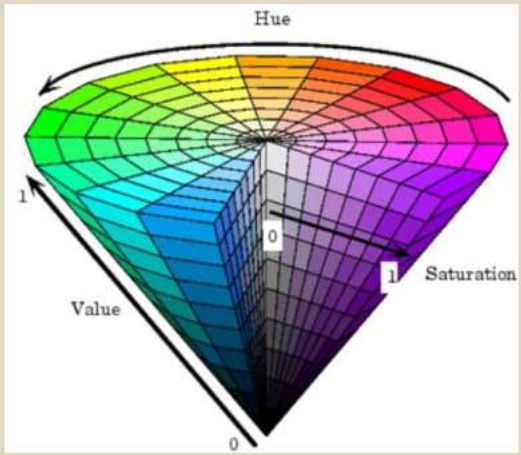
R translates various color models to hex, e.g.:

- RGB (red, green, blue): The default intensity scale in R ranges from 0-1; but another commonly used scale is 0-255. This is obtained in R using maxColorValue=255. *alpha* is an optional argument for transparency, with the same intensity scale.
  rgb(r, g, b, maxColorValue=255, alpha=255)
- HSV (hue, saturation, value): values range from 0-1, with optional alpha argument
  hsv(h, s, v, alpha)
- HCL (hue, chroma, luminance): hue describes the color and ranges from 0-360; 0 = red, 120 = green, blue = 240, etc. Range of chroma and luminance depend on hue and each other
  hcl(h, c, l, alpha)

A few notes on HSV/HLC

HSV is a better model for how humans perceive color. HCL can be thought of as a perceptually based version of the HSV model....blah blah blah...

Without delving into color theory: color schemes based on HSV/HLC models generally just look good.



R can translate colors to rgb (this is handy for matching colors in other programs)
col2rgb(c("#FF0000", "blue"))

R Color Palettes

This is for all of you who don't know anything about color theory, and don't care but want some nice colors on your map or figure....NOW!

TIP: When it comes to selecting a color palette **DO NOT** try to handpick individual colors! You will waste a lot of time and the result will probably not be all that great. R has some good packages for color palettes. Here are some of the options

Packages: grDevices and colorRamps

grDevices comes with the base installation and colorRamps must be installed. Each palette's function has an argument for the number of colors and transparency (*alpha*):

grDevices palettes
cm.colors
topo.colors
terrain.colors
heat.colors
rainbow
see P. 4 for options

heat.colors(4, alpha=1)
> #FF0000FF" "#FF8000FF" "#FFFF00FF" "#FFFF80FF"

For the rainbow palette you can also select start/end color (red = 0, yellow = *1/6*, green = *2/6*, cyan = *3/6*, blue = *4/6* and magenta = *5/6*) and saturation (s) and value (v):
rainbow(n, s = 1, v = 1, start = 0, end = max(1, n - 1)/n, alpha = 1)

Package: RColorBrewer

This function has an argument for the number of colors and the color palette (see P. 4 for options).
brewer.pal(4, "Set3")
> "#8DD3C7" "#FFFFB3" "#BEBADA" "#FB8072"

To view colorbrewer palettes in R: display.brewer.all(5)
There is also a very nice interactive viewer:
http://colorbrewer2.org/

My Recommendation

Package: colorspace

These color palettes are based on HCL and HSV color models. The results can be very aesthetically pleasing. There are some default palettes:

colorspace default palettes
diverge_hcl
diverge_hsl
terrain_hcl
sequential_hcl
rainbow_hcl

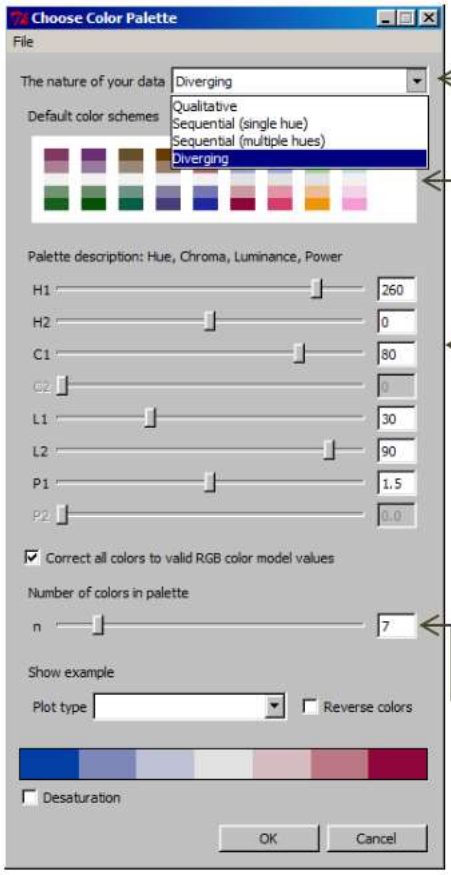
rainbow_hcl(4)
"#E495A5" "#ABB065" "#39BEB1" "#ACA4E2"

However, all palettes are fully customizable:
diverge_hcl(7, h = c(246, 40), c = 96, l = c(65, 90))
Choosing the values *would* be daunting. But there are some recommended palettes in the colorspace documentation. There is also an interactive tool that can be used to obtain a customized palette. To start the tool:
pal <- choose_palette()

R color cheatsheet

Overview of colorspace palette selector

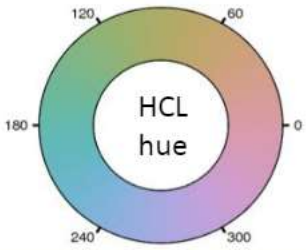
```
library("colorspace")
pal <- choose_palette()
```



Select the type of color scheme based on the type of data

Default color schemes – can be used "as is" or as a starting point for modification

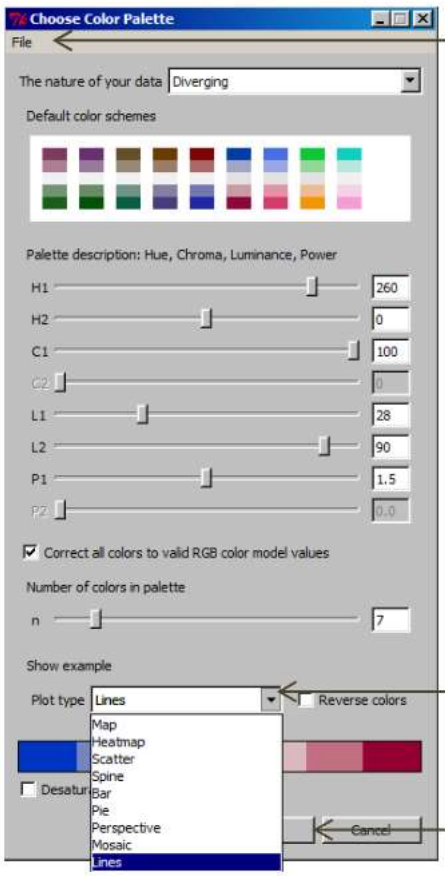
Interactively select:
- hue: color
- chroma: low chroma = gray
- luminance: high luminance = pastel
- power: how the color changes along a gradient



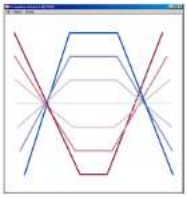
Select # of colors in palette

Save palette for future R sessions:
- .txt file with hex codes
- .R file with a function describing how to generate the palette. source can be used to import the function into R, but one complication is that you have to open the .R file and name the function to use it.
- Copy values into relevant colorspace functions.

Diverging color schemes:
`diverge_hcl(7, h = c(260, 0), c = 100, l = c(28, 90), power = 1.5)`
Sequential color schemes:
`sequential_hcl(n, h, c.= c(), l=c(), power)`
Qualitative color schemes.
`rainbow_hcl(n, c, l, start, end)` (for qualtitative schemes; start/ end refer to the H1/H2 hue values)



Display color scheme with different plot types



When "OK" is selected, the color palette will be saved in the R session. To return 7 hex color codes from the selected palette:

```
pal <- choose_palette()
pal(7)
```

[NOTE: These values are not saved if you don't save the session]

How to use hex codes to define color using the plot function

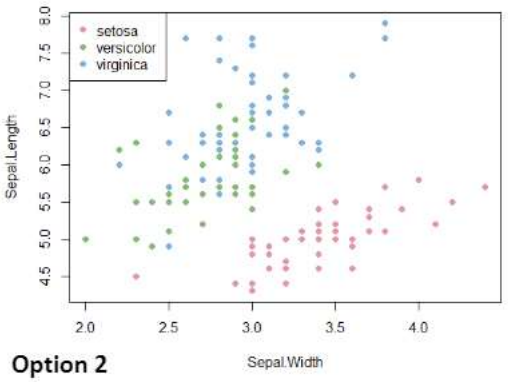
Discrete variables

Option 1

If you don't need to control which colors are associated with each level of a variable:

```
plot(Sepal.Length ~ Sepal.Width,
col=rainbow_hcl(3)[c(Species)],
data=iris, pch=16)

legend("topleft", pch=16, col=rainbow_hcl(3),
legend=unique(iris$Species))
```



Option 2

If you want to control which colors are associated with the levels of a variable, I find it easiest to create a variable in the data:

```
iris$color <- factor(iris$Species,
levels=c("virginica", "versicolor", "setosa"),
labels=rainbow_hcl(3))

plot(Sepal.Length ~ Sepal.Width,
col=as.character(color), pch=16, data=iris)
```

Continuous variables

Option 1

Break into categories and assign colors:

```
iris2 <- subset(iris, Species=="setosa")

color <- cut(iris2$Petal.Length,
breaks=c(0,1.3,1.5,2), labels=sequential_hcl(3))
```

Or, break by quantiles (be sure to include 0 & 1):

```
color <- cut(iris2$Petal.Length,
breaks=quantile(iris$Petal.Length, c(0, 0.25, 0.5,
0.75, 1)), labels=sequential_hcl(3))

plot(Sepal.Width ~ Sepal.Length, pch=16,
col=color, data=iris2)
```

Option 2

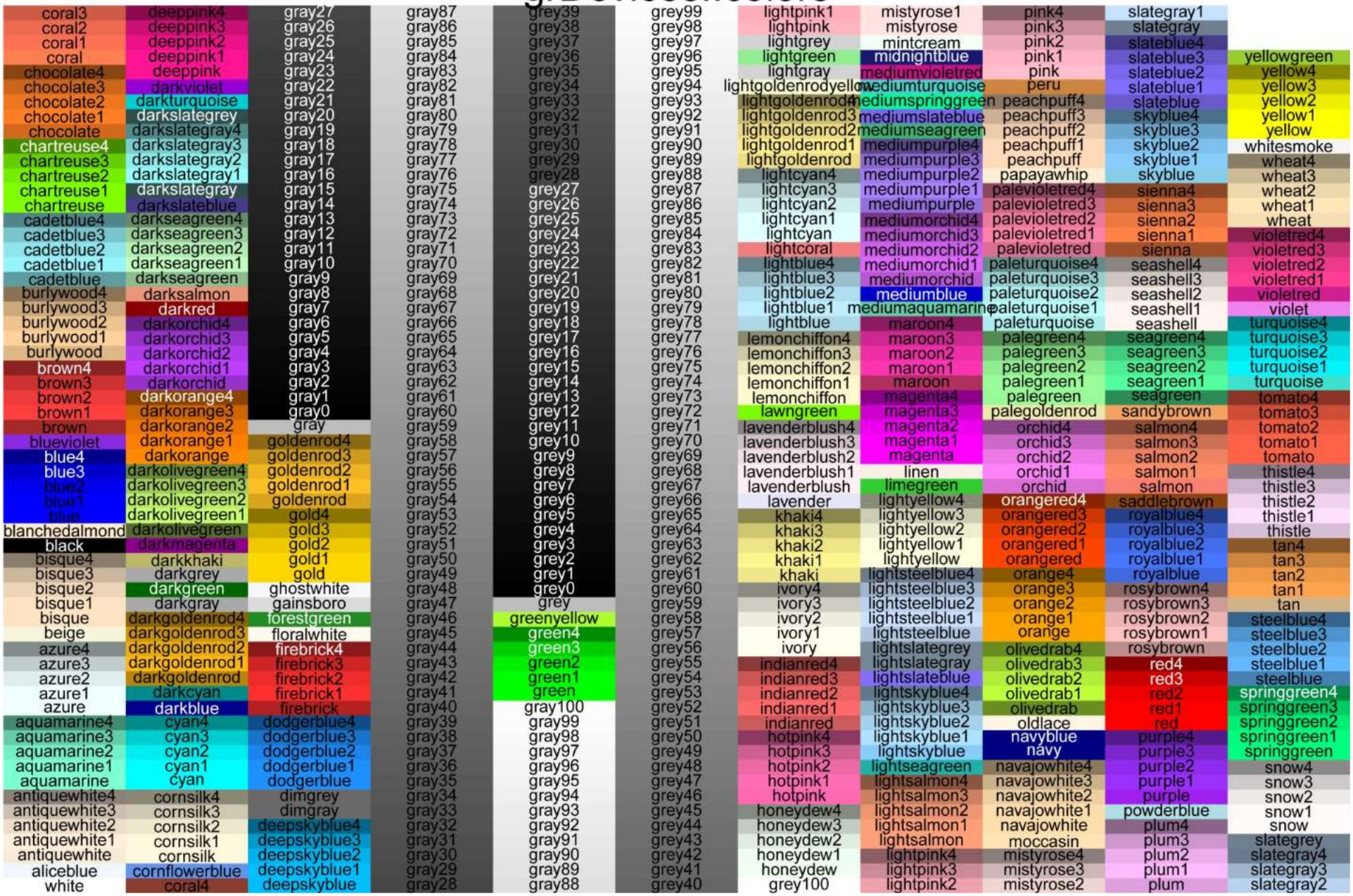
Fully continuous gradient:

```
data <- data.frame("a"=runif(10000),
"b"=runif(10000))

color=diverge_hcl(length(data$a))[rank(data$a)]
plot(a~b, col=color, pch=16, data=data)
```

For ggplot2, I think the most flexible color scales are:
scale_colour_manual
scale_colour_gradient
for discrete and continuous variables, respectively

grDevices::colors



code to produce R color chart from: http://www.biecek.pl/R/R.pdf and http://bc.bojanorama.pl/2013/04/r-color-reference-sheet

colorRamps and grDevices

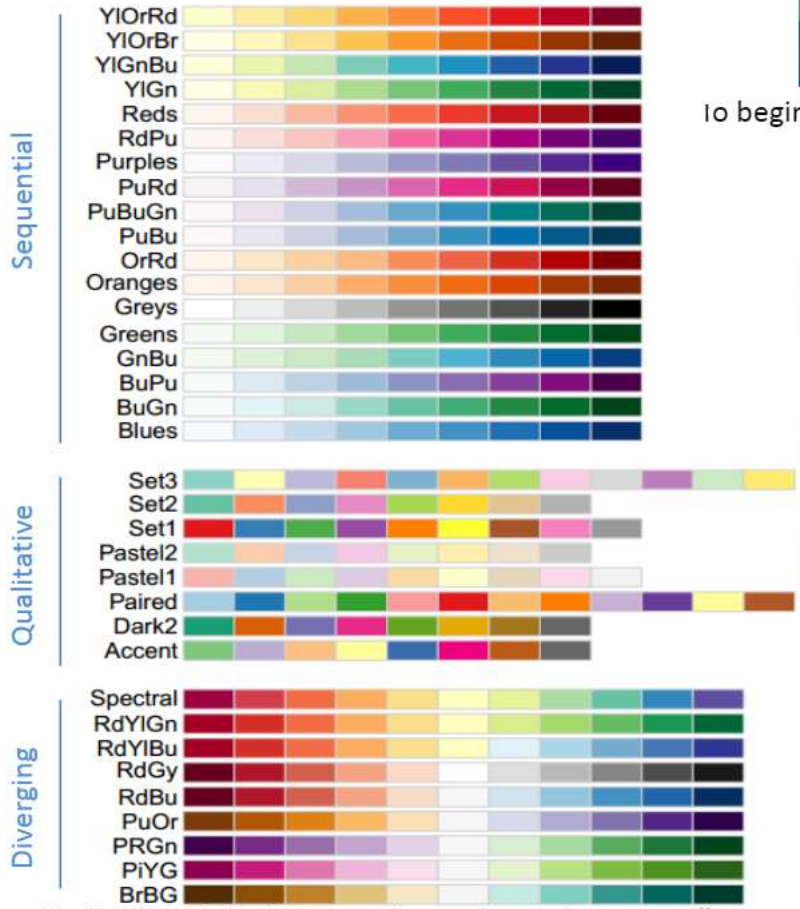
colorRamps and grDevices color palette, display from:
http://bc.bojanorama.pl/2013/04/r-color reference sheet/

colorspace defaults

colorspace useful palette examples

To begin interactive color selector: pal <- choose_palette()

RColorBrewer



To display RColorBrewer palette: display.brewer.all()
For interactive color selector: http://colorbrewer2.org/

Useful Resources:

A larger color chart of R named colors:
http://research.stowers-institute.org/efg/R/Color/Chart/ColorChart.pdf

Nice overview of color in R:
http://research.stowers-institute.org/efg/Report/UsingColorInR.pdf

http://students.washington.edu/mclarkso/documents/colors Ver2.pdf

A color theory reference:
Zeileis, A. K. Hornik P. Murrell. 2009. Escaping RGBland: selecting colors for statistical graphics. Computational and Statistics & Data Analysis 53:3259-3270