talleR peRcepcion del coloR gRupo R madRid

pedro.concejerocerezo at gmail.com

Martes 3 marzo 2015 - revisado 26 abril 2019

# GRACIAS!!

* A los asistentes a este talleR
* A los organizadores de estas nuestras (maravillosas) Jornadas de Usuarios R (primer lugar donde se impartió este taller)
* A los miembros del comité científico que (inexplicablemente) aprobaron el taller
* A José Antonio Collado, el experto sobre color autor de las recomendaciones y ejercicios y de la base teórica de este taller, pero que no ha podido asistir a las jornadas
* A José Luis Cañadas (<https://twitter.com/joscani>) por sus mapas (<http://rpubs.com/joscani/mapa_paro_andalucia>)
* A Miguel Fiandor (<https://twitter.com/mfiandor>) por sus ejemplos enviados a la lista R-hispano, en concreto esta que comentaremos (criticaremos; aunque lamentablemente ahora en 2019 no se ve bien): <http://www.elconfidencial.com/espana/2014-08-07/nueve-de-cada-diez-municipios-tienen-mas-paro-ahora-que-antes-de-comenzar-la-crisis_172665/>
* A Óscar Perpiñán (@oscarperpinan) por su ayuda sobre EBImage para convertir imagen a escala de grises.

# OBJETIVOS DE ESTE DOCUMENTO

Este documento es la documentación para el talleR de coloR en el gRupo madRid R:

<http://madrid.r-es.org/>

y que es una evolución del impartido dentro de las jornadas R-hispano 2014.

Dado que el taller está dirigido a la comunidad R, todo el código está en este lenguaje, aunque intentaremos que las recomendaciones sean suficientemente generales para cualquier lenguaje que permita manejar el color. De hecho se añade un apéndice con enlaces a información relevante para python.

# RESUMEN DEL TALLER

Resumen

El color es una más de entre las posibles características que pueden convertir un gráfico, una visualización, un sistema interactivo de análisis de datos en algo eficaz y atractivo o por el contrario hacer que esa misma visualización sea poco agradable, o, en algunos casos, más confusa que si se utilizan otras características visuales.

La disponibilidad actual de sistemas para manejar con práctica total libertad el color en las visualizaciones (por ejemplo en pantallas o proyectores) o impresiones no ha ido acompañada de recomendaciones sobre cómo optimizar el uso del mismo. De hecho, muchas tareas de manejo de los colores han pasado de auténticos expertos (por ejemplo diseñadores gráficos y técnicos de impresión) directamente a los generadores de los gráficos, con el riesgo de que una herramienta potentísima de comunicación no sea utilizada de la mejor manera.

En el mundo R hay multitud de tutoriales, guías, publicaciones de todo tipo para manejar el color desde un punto de vista técnico. De hecho, R es probablemente el entorno más potente para gestionar el color en un sistema informático dedicado al análisis de datos del que dispone un investigador en estos momentos. Sin embargo es difícil encontrar recomendaciones sobre el uso eficaz del color basadas en criterios científicos y ergonómicos.

La visión del color por los humanos es un campo multidisciplinar bien conocido que ha generado un amplio conocimiento, tanto desde el punto de vista científico como tecnológico. Nuestra propuesta es desarrollar un taller para trasladar este conocimiento a los expertos en R que desarrollan visualizaciones, de cualquier tipo, desde simples gráficos estáticos en una pantalla o proyector modernos, a gráficos interactivos en dispositivos móviles o impresiones -en este último caso con una simplificación fundamental, porque no podremos abordar la impresión profesional propiamente dicha.

# DOCUMENTACIÓN PREVIA

DOCUMENTACIÓN DISPONIBLE EN INTERNET Y QUE USAREMOS EN EL TALLER

* Tabla completa de colores y nombres en R <http://research.stowers.org/mcm/efg/R/Color/Chart/ColorChart.pdf>

obtenida a partir de esto: <http://research.stowers.org/mcm/efg/R/Color/Chart/ColorChart.R>

* Using Color in R (ver “Los números del color”) <http://research.stowers.org/mcm/efg/Report/UsingColorInR.pdf>
* Maravillosa “cheatsheet” que es mucho más moderna y eficaz: <https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/colorPaletteCheatsheet.pdf>
* Color Brewer web site <http://colorbrewer2.org/>

DOCUMENTACIÓN ADICIONAL QUE *NO* USAREMOS EN EL TALLER

* Color Scales, some further info <http://geog.uoregon.edu/datagraphics/color_scales.htm>
* Recopilación *completísima* de ciencia del color <http://www.efg2.com/Lab/Library/Color/Science.htm>

y en concreto sobre “Color and Computers”

<http://www.efg2.com/Lab/Library/Color/AndComputers.htm>

e info general inc. ceguera al color (“daltonismo”)

<http://www.efg2.com/Lab/Library/Color/index.html>

* Sobre ceguera al color:
* <http://colororacle.org/design.html>
* Regla 6 “Use Color Effectively” de “Ten Simple Rules for Better Figures”
* <http://www.ploscompbiol.org/article/info%3Adoi%2F10.1371%2Fjournal.pcbi.1003833#s7>

# PREPARACIÓN DE ENTORNO R

**MUY IMPORTANTE PON TU DIRECTORIO DE TRABAJO**

# Ojo pongo mi directorio de trabajo, tú pon el tuyo  
  
setwd("D:/2019/uned 2019/taller-color")

Si es necesario, instalemos librerías requeridas.

options(repos = c(CRAN = "http://cran.rstudio.com"))  
  
# Inicialización  
# Comprueba si están y si no instala librerías  
  
if ("sos" %in% row.names(installed.packages()) == FALSE) install.packages("sos", dependencies = T)  
if ("colortools" %in% row.names(installed.packages()) == FALSE) install.packages("colortools", dependencies = T)  
if ("dichromat" %in% row.names(installed.packages()) == FALSE) install.packages("dichromat", dependencies = T)  
if ("RColorBrewer" %in% row.names(installed.packages()) == FALSE) install.packages("RColorBrewer", dependencies = T)  
if ("vcd" %in% row.names(installed.packages()) == FALSE) install.packages("vcd", dependencies = T)  
if ("maptools" %in% row.names(installed.packages()) == FALSE) install.packages("maptools", dependencies = T)  
if ("sp" %in% row.names(installed.packages()) == FALSE) install.packages("sp", dependencies = T)  
if ("classInt" %in% row.names(installed.packages()) == FALSE) install.packages("classInt", dependencies = T)  
if ("ggmap" %in% row.names(installed.packages()) == FALSE) install.packages("ggmap", dependencies = T)  
if ("grDevices" %in% row.names(installed.packages()) == FALSE) install.packages("grDevices", dependencies = T)  
if ("colorspace" %in% row.names(installed.packages()) == FALSE) install.packages("colorspace", dependencies = T)  
  
  
# Cargamos librerías que vamos a usar  
library(colortools)

## Warning: package 'colortools' was built under R version 3.5.2

library(sos)

## Warning: package 'sos' was built under R version 3.5.3

## Loading required package: brew

## Warning: package 'brew' was built under R version 3.5.2

##   
## Attaching package: 'sos'

## The following object is masked from 'package:utils':  
##   
## ?

library(dichromat)

## Warning: package 'dichromat' was built under R version 3.5.2

library(RColorBrewer)  
library(grDevices)  
library(colorspace)  
  
# Para dibujar mosaicos  
library(vcd)

## Warning: package 'vcd' was built under R version 3.5.3

## Loading required package: grid

# Para dibujar los mapas  
library(maptools)

## Loading required package: sp

## Checking rgeos availability: FALSE  
## Note: when rgeos is not available, polygon geometry computations in maptools depend on gpclib,  
## which has a restricted licence. It is disabled by default;  
## to enable gpclib, type gpclibPermit()

library(sp)  
library(classInt)

## Warning: package 'classInt' was built under R version 3.5.3

library(ggmap) # también carga ggplot2

## Warning: package 'ggmap' was built under R version 3.5.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.3

## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.

# POR QUÉ R ES EL MEJOR SISTEMA ACTUAL (EN CAMPO ESTADÍSTICA) PARA MANEJO DE COLOR

Veamos qué librerías contienen color (sos levanta una página html fuera de RStudio, con enlaces a la documentación de las funciones encontradas)

sos::findFn("{color}")

## Warning in parseHTML(href): Too many documents hit. Ignored

## found Inf matches

## x has zero rows; nothing to display.

En 2015, cuando escribí estas líneas, la función anterior encontró 357 funciones en 159 paquetes CRAN para estudiar o manejar color en R.

En 2019 comprobarás que devuelve: “Too many documents hit”.

Tenemos por ejemplo la librería colorscience

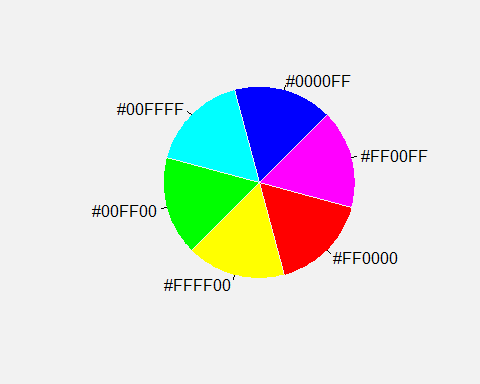
<http://finzi.psych.upenn.edu/R/library/colorscience/html/00Index.html>

y la colortools <http://gastonsanchez.com/colortools/>

“colortools is an R package designed to help users generate color schemes and color palettes. It provides a number of handy functions that will allow to select and play with *colors in an HSV (hue, saturation, value) color model*”.

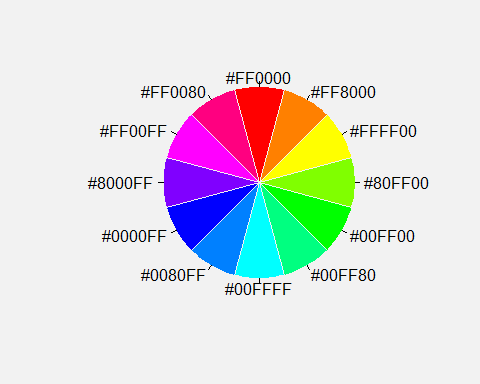
La librería colortools es fascinante. Por ejemplo, podemos hacer ruedas de color con un número variable de “gajos” o colores en la escala entre los extremos con la función wheel:

colortools::wheel("blue", num = 6)



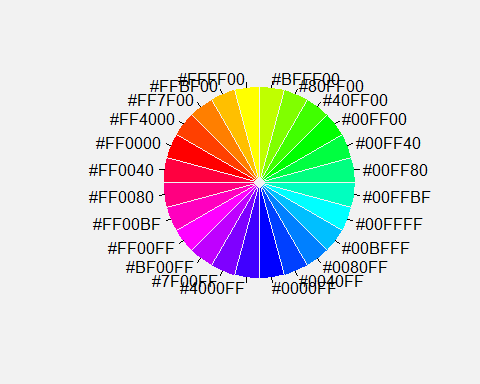
## [1] "#0000FF" "#FF00FF" "#FF0000" "#FFFF00" "#00FF00" "#00FFFF"

colortools::wheel("red", num = 12)



## [1] "#FF0000" "#FF8000" "#FFFF00" "#80FF00" "#00FF00" "#00FF80" "#00FFFF"  
## [8] "#0080FF" "#0000FF" "#8000FF" "#FF00FF" "#FF0080"

colortools::wheel("yellow", num = 24)



## [1] "#FFFF00" "#BFFF00" "#80FF00" "#40FF00" "#00FF00" "#00FF40" "#00FF80"  
## [8] "#00FFBF" "#00FFFF" "#00BFFF" "#0080FF" "#0040FF" "#0000FF" "#4000FF"  
## [15] "#7F00FF" "#BF00FF" "#FF00FF" "#FF00BF" "#FF0080" "#FF0040" "#FF0000"  
## [22] "#FF4000" "#FF7F00" "#FFBF00"

# 1. COLOR Y ENTORNO FÍSICO

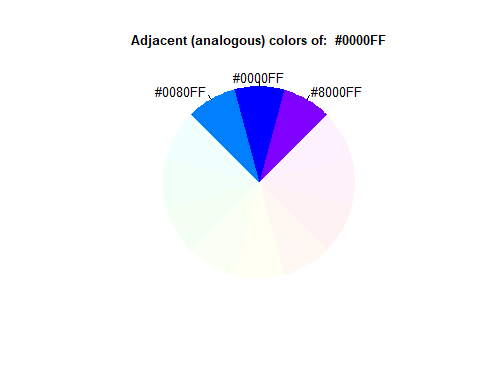
En la percepción del color, el entorno físico y dispositivo que genera el color son clave.

Muy importante:

* La visión de color en humanos se realiza por la activación de los fotorreceptores de la retina conocidos como conos (<http://en.wikipedia.org/wiki/Color_vision#Cone_cells_in_the_human_eye>), que se activan con la luz intensa como la luz del día. Rápidamente: para poder distinguir el color la luz debe ser parecida a la del día: intensa. Si no lo es veremos en monocromo (con los bastones, los otros fotorreceptores).
* En la práctica: distinguir luz emitida (p.ej. monitor LCD-TFT) vs. luz reflejada (p.ej., proyector)
* Color y contraste y constancia. Veremos más detalle más adelante.

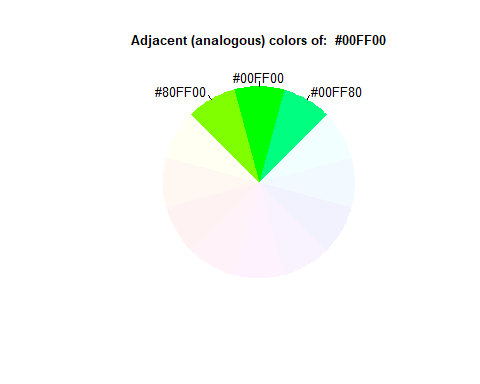
La función colortools::adjacent permite generar esquemas de color que están próximos unos a otros.

colortools::adjacent("blue")



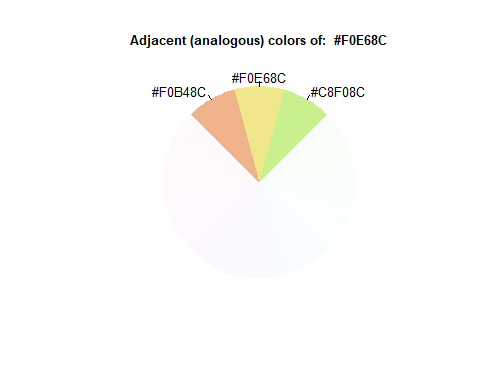
## [1] "#0000FF" "#8000FF" "#0080FF"

colortools::adjacent("green")



## [1] "#00FF00" "#00FF80" "#80FF00"

colortools::adjacent("khaki")



## [1] "#F0E68C" "#C8F08C" "#F0B48C"

# 2. LOS NÚMEROS DEL COLOR

Ver excelente presentación aquí: <http://research.stowers.org/mcm/efg/Report/UsingColorInR.pdf>

Pero hay una maravillosa “cheatsheet” que es mucho más moderna y eficaz: <https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/colorPaletteCheatsheet.pdf>

Excelente consejo! “Cuando se trata de seleccionar una paleta de color, NO INTENTES escoger los colores A MANO, por tu cuenta. Perderás un montón de tiempo y el resultado no será probablemente tan bueno. Usa uno de los paquetes especializados de R!”

Podemos utilizar 3 modelos de codificación de color en R: RGB - Red Green Blue, modelo en el que se basa la representación de color por dispositivos pantallas etc. HSV - Hue Saturation Value -años 70- modelo más relacionado con cómo percibimos color las personas, pero con sus problemas HCL - Hue Chroma Luminance - modelo más reciente basado en CIELUV

¡¡Mirad las diferencias de escalas entre ellos!!

En HSV/HCL el color se puede representar “casi” como una circunferencia: <http://www.handprint.com/HP/WCL/IMG/conesps3.gif>

Siempre podemos pasar de color RGB a color HSV

rgb2hsv(col2rgb("blue"))

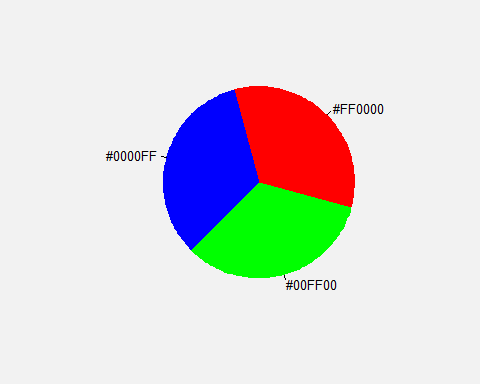
## [,1]  
## h 0.6666667  
## s 1.0000000  
## v 1.0000000

rgb2hsv(col2rgb("khaki"))

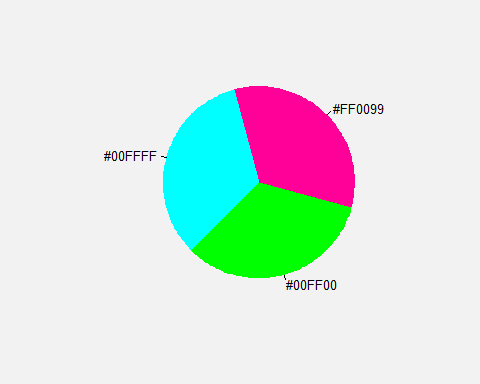
## [,1]  
## h 0.1500000  
## s 0.4166667  
## v 0.9411765

HSV usa valores 0-1, al igual que R hace con RGB por defecto a no ser que especifiquemos maxColorValue = 255 (!). HCL usa valores para H (color) de 0-360, con 0= rojo, 120 = verde, 240 = azul. Ojo, es más complicado de manejar que HSV (sin herramientas)

# A Color Triad (Maximal Color Differences) in HSV and HCL  
  
pizza(c(hsv(1, 1, 1), hsv(1/3, 1, 1), hsv(2/3, 1, 1)))



pizza(hcl(h = c(0, 120, 240), c = rep(400, 3)))



* Espacio HCL (Hue, Chroma, Luminance):

Escaping RGBLand <http://statmath.wu.ac.at/~zeileis/papers/Zeileis+Hornik+Murrell-2009.pdf>

<http://www.r-bloggers.com/choosing-colour-palettes-part-ii-educated-choices/>

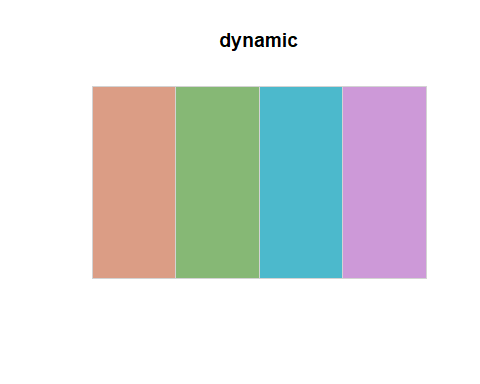
“Unfortunately, the dimensions in HSV space map poorly to perceptual properties and the use of HSV colors encourages the use of highly saturated colors. A perceptually-based color model that mitigates these problems involves Hue-Chroma-Luminance (HCL) triplets (see Ihaka, 2003), resulting from a transformation of CIELUV color space (see Wikipedia 2007a)”

<http://cran.r-project.org/web/packages/colorspace/vignettes/hcl-colors.pdf> (Paquete colorspace, de Ross Ihaka)

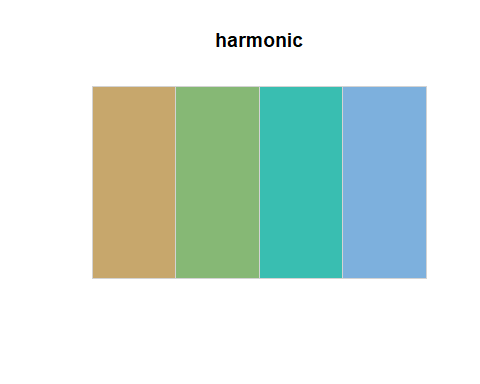
# en la viñeta de colorspace se propone esta función para dibujar los colores en mosaicos  
  
pal <- function(col, border = "light gray", ...)  
 {  
 n <- length(col)  
 plot(0, 0, type="n", xlim = c(0, 1), ylim = c(0, 1),  
 axes = FALSE, xlab = "", ylab = "", ...)  
 rect(0:(n-1)/n, 0, 1:n/n, 1, col = col, border = border)  
 }

colorspace::PALETAS CUALITATIVAS

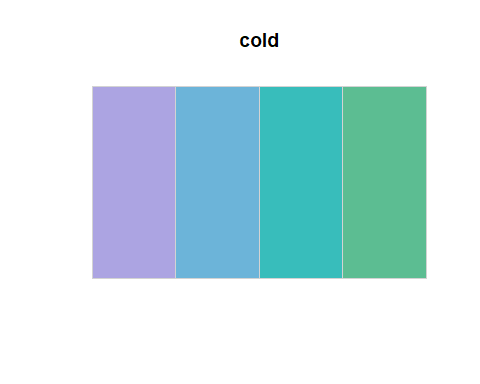
pal(rainbow\_hcl(4, start = 30, end = 300), main = "dynamic")



pal(rainbow\_hcl(4, start = 60, end = 240), main = "harmonic")



pal(rainbow\_hcl(4, start = 270, end = 150), main = "cold")

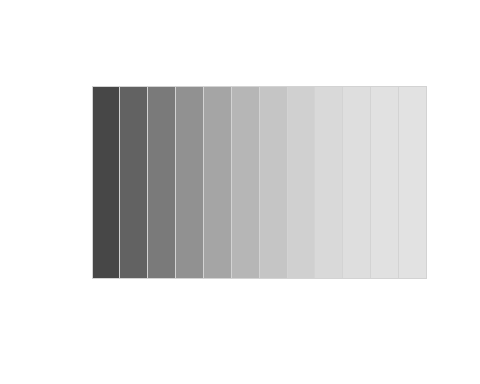


pal(rainbow\_hcl(4, start = 90, end = -30), main = "warm")

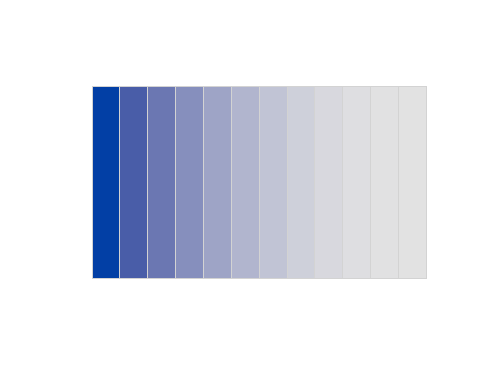


colorspace::PALETAS SECUENCIALES

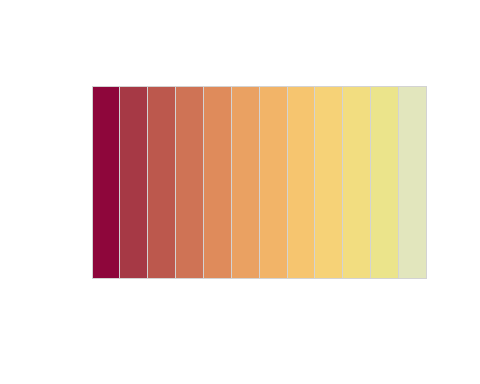
pal(sequential\_hcl(12, c = 0, power = 2.2))



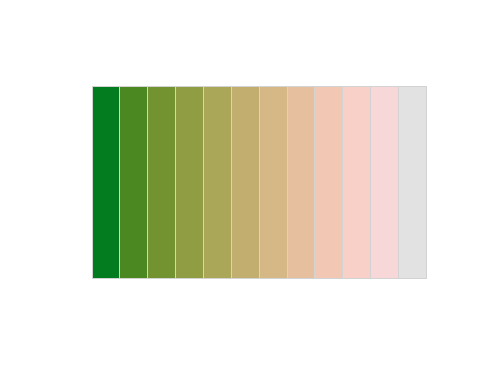
pal(sequential\_hcl(12, power = 2.2))



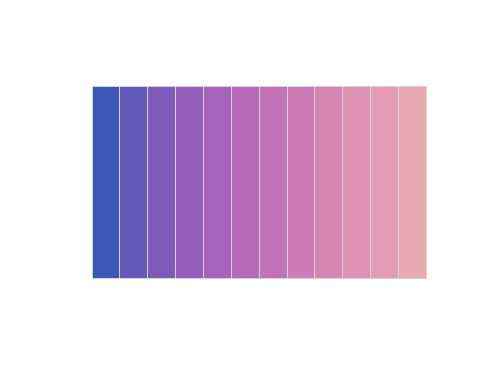
pal(heat\_hcl(12, c = c(80, 30), l = c(30, 90), power = c(1/5, 2)))



pal(terrain\_hcl(12, c = c(65, 0), l = c(45, 90), power = c(1/2, 1.5)))

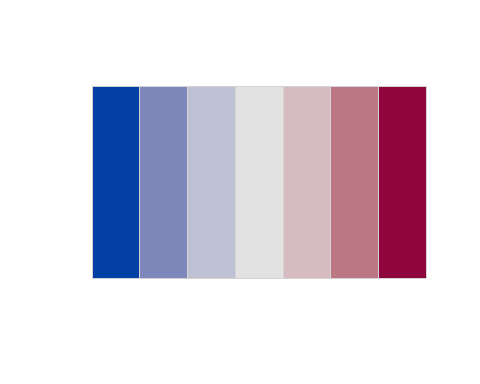


pal(rev(heat\_hcl(12, h = c(0, -100), c = c(40, 80), l = c(75, 40), power = 1)))

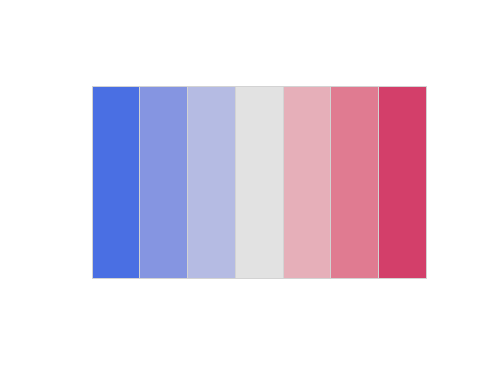


colorspace::PALETAS DIVERGENTES

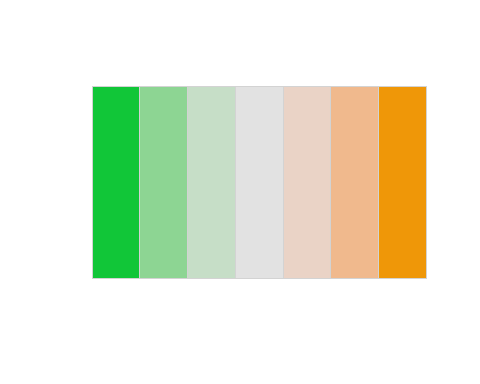
pal(diverge\_hcl(7))



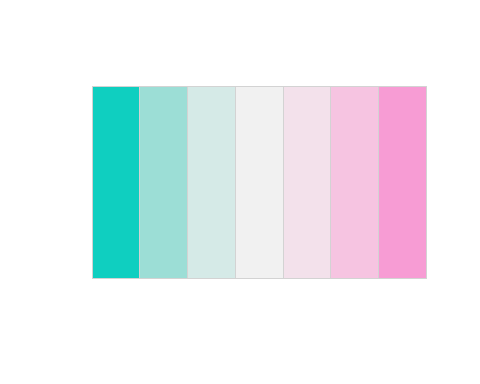
pal(diverge\_hcl(7, c = 100, l = c(50, 90), power = 1))



pal(diverge\_hcl(7, h = c(130, 43), c = 100, l = c(70, 90)))



pal(diverge\_hcl(7, h = c(180, 330), c = 59, l = c(75, 95)))



colorspace::choose\_palette() UNA INTERFAZ GRÁFICA PARA ELEGIR PALETA

kk <- choose\_palette()

# 3. PERCEPCIÓN DEL COLOR - COLOR Y PSICOFÍSICA Y PSICOLOGÍA

¿Cómo percibimos ese atributo físico que es el color? Antes de nada, ¿el color existe en la naturaleza? ¿Necesitamos el color para poder manejarnos en el mundo?

Para manejarnos en el mundo lo que necesitamos es contraste. El color *no* es imprescindible.

<http://en.wikipedia.org/wiki/Contrast_%28vision%29>

Respuesta normalizada de fotorreceptores a longitudes de onda -“los 3 colores básicos RGB” (con una precisión importante, cada curva cubre y *se solapa* a un rango de frecuencias, no a un único color):

<http://en.wikipedia.org/wiki/Color_vision#mediaviewer/File:Cone-fundamentals-with-srgb-spectrum.svg>

Muy importante: sensibilidad relativa del Sistema Visual Humano en f. longitudes de onda:

<http://en.wikipedia.org/wiki/Color_vision#mediaviewer/File:Eyesensitivity.svg>

El color es un constructo humano, la luz tiene unas propiedades físicas y los objetos en los que se refleja otras y eso permite una estimulación en los receptores humanos, la interpretación de lo que llega a los receptores, su transporte y procesamiento en las zonas visuales del cerebro es lo que realmente percibimos como color, es en suma una interpretación de lo que nos llega.

En la percepción de color tanto retina como córtex están involucrados, y gobernados por principios generales como la “constancia del color”:

<http://en.wikipedia.org/wiki/Color_constancy>

Recomendaciones:

* Antes de fijar los colores en una visualización: maximizar el contraste -> elegir fondo!

-> MUY A MENUDO UN FONDO CON UN GRIS MEDIO PERMITE MAXIMIZAR EL CONTRASTE Y POR TANTO LA VISUALIZACIÓN DE FORMAS Y TEXTOS.

* Color *no puede ser la única clave* (veremos más adelante razones adicionales)
* Humanos somos relativamente malos para distinciones finas de color (matiz)
* No todos los colores se ven igual: recomendaciones sobre azul y rojo en concreto
* Evitar que canales entre en conflicto. Efecto “cromoestereopsis”
* <http://en.wikipedia.org/wiki/User:Karam91/Chromostereopsis>
* aunque ocasional y muy cuidadosamente se puede usar con beneficio

<http://www.luminanze.com/writings/chromostereopsis_in_ux_design.html>

# 4. COLOR Y MEMORIA Y SEMÁNTICA

**Aquí haremos un ejercicio práctico de memoria que por motivos obvios no se puede publicar previamente. Después de haberlo hecho subiremos el código**

La memoria humana es relativamente bien conocida, y hay un principio muy bien establecido (muy a menudo demasiado simplificado) que describe la denominada “memoria a corto plazo”, una especie de “buffer” de entrada a partir de los sentidos antes de que se le dé significado y se incorpore a procesos cognitivos de más alto nivel. Pues bien, esta memoria es muy limitada, hasta el punto de conocerse como la “el mágico número 7 +/- 2”:

<http://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two>

<http://www.musanim.com/miller1956/>

<http://www.musanim.com/miller1956/fig6.gif>

La memoria visual humana es baja <http://en.wikipedia.org/wiki/Visual_memory#Short_term_visual_memory>, y es crucial dotar a los estímulos visuales de significado. El color, por su naturaleza cambiante y arbitraria, *no es la mejor clave visual*.

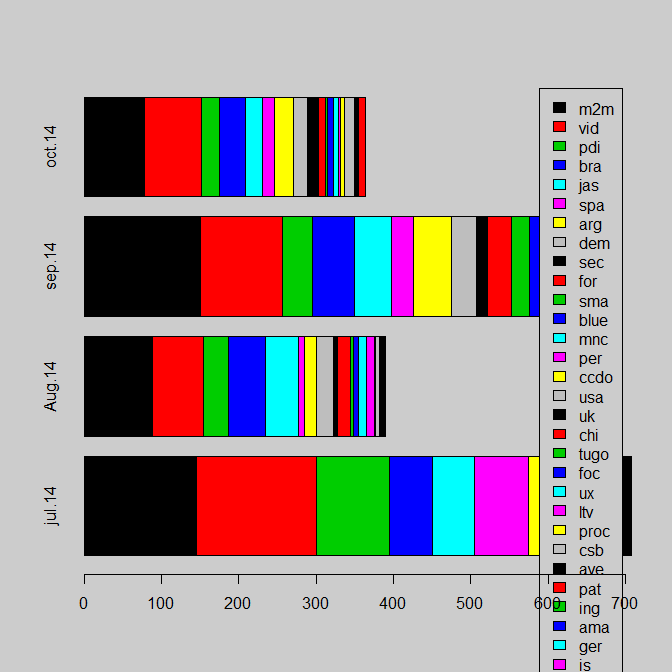
# EJEMPLO APLICADO

Veamos un gráfico de barras acumuladas por categorías para 4 meses (4 barras). Primero preparamos un poquito los datos.

# Cargamos los datos  
t <- get(load("tabla.rda"))  
   
# podemos ver qué contiene  
t

## jul.14 Aug.14 sep.14 oct.14  
## m2m 145 88 151 78  
## vid 155 66 105 74  
## pdi 95 32 39 23  
## bra 56 48 54 33  
## jas 54 43 48 22  
## spa 70 8 29 16  
## arg 48 15 49 25  
## dem 54 22 32 18  
## sec 31 6 15 14  
## for 17 31 9  
## sma 6 3 23 2  
## blue 21 7 23 8  
## mnc 20 10 16 7  
## per 12 10 11 2  
## ccdo 7 2 6  
## usa 6 5 10 13  
## uk 21 7 6 5  
## chi 11 1 2 9  
## tugo 13   
## foc 16 7 2 6  
## ux 2 1 6 1  
## ltv 10  
## proc 8   
## csb 7 6  
## ave 6 1  
## pat 1 3   
## ing 3 2   
## ama 3   
## ger 1 2   
## is 2   
## tms 1 1 1  
## pay 1

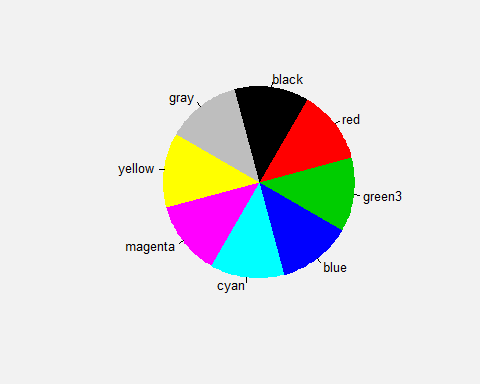
#png(filename = "barplot\_1.png",  
# width = 1024,  
# height = 768)  
par(bg = "gray80")  
  
barplot(t, col=palette(),   
 legend = rownames(t),  
 horiz = T)



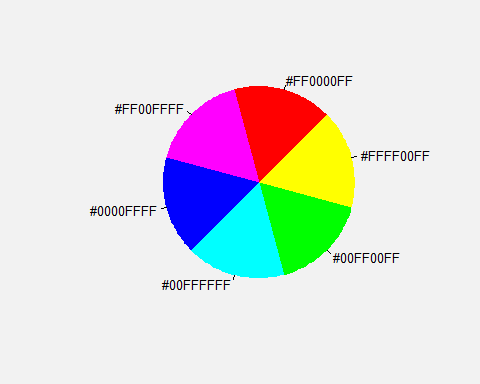
# palette contiene 32 colores muy saturados variando sobre los colores básicos   
  
#dev.off()

Un poco horrible no? Es lo que sucede siguiendo las recomendaciones al pie de la letra. O quizás por no haber hecho la recomendación adecuadamente. Es un problema de elegir la saturación adecuada para nuestros colores. Veamos cómo.

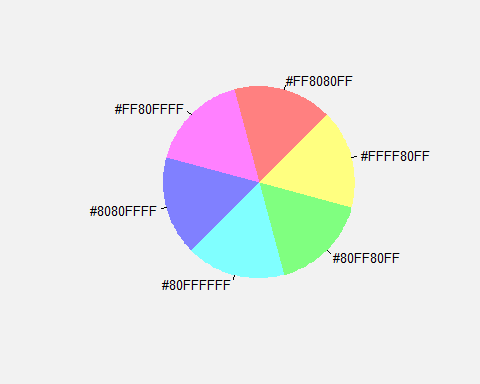
# palette contiene 32 colores muy saturados variando sobre los colores básicos   
  
pizza(palette())



# es lo mismo que si le pides rainbow  
  
pizza(rainbow(6))

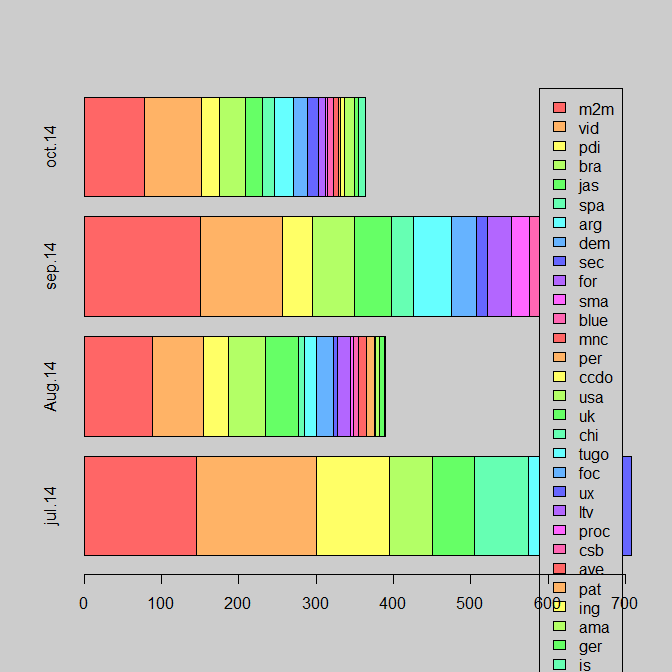


# pero puedes variar los valores de saturación para evitar que "canten" tanto  
  
pizza(rainbow(6, s = 0.5))



Y usando s = 0.5 tendremos

par(bg = "gray80")  
barplot(t, col=rainbow(12, s = 0.6),   
 legend = rownames(t),  
 horiz = T)



* Cuando sólo puedas utilizar color para categorizar, y quieres facilitar al máximo la asociación color a categoría, estarás *muy limitado*, en general *no más de 5 categorías*.

Una vez dicho esto, no tienes por qué utilizar los colores perfectamente saturados (a no ser por ejemplo que estés en un contexto de muy bajo contraste). Puedes utilizar gamas de colores con suficiente separación y mucho más atractivos.

Una herramienta maravillosa para prototipar estas “paletas de color” o escalas es el sitio web <http://colorbrewer2.org/> (totalmente independiente de lenguaje de programación).

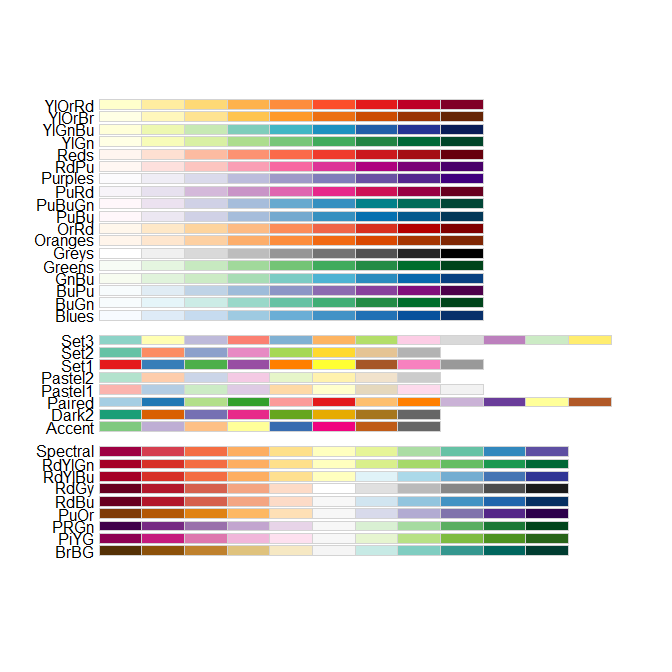
**Ahora mostraremos brevemente el manejo del sitio colorbrewer**

También hay otro sitio similar aquí <https://color.adobe.com/es/> muy vistoso pero en mi opinión menos práctico que colorbrewer.

La librería más frecuentemente usada para generar este tipo de paletas en R es RColorBrewer.

Veamos primero todas las paletas que tenemos en RColorBrewer

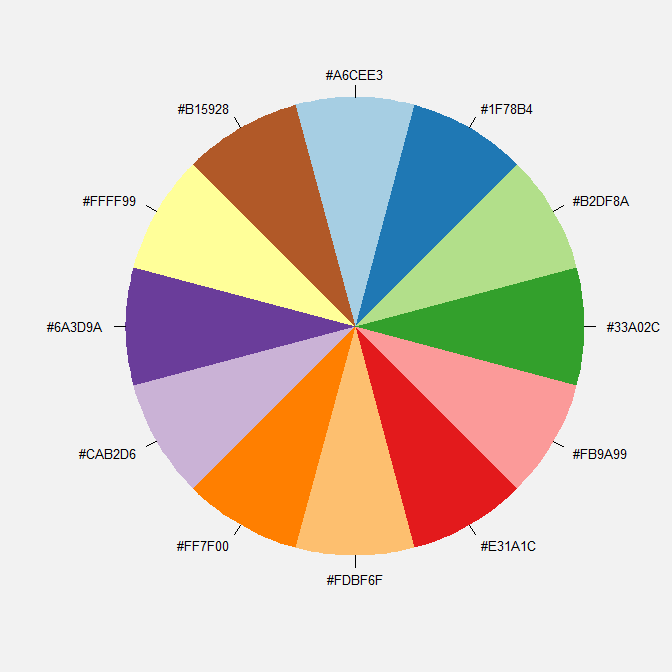
library(RColorBrewer)  
  
display.brewer.all()



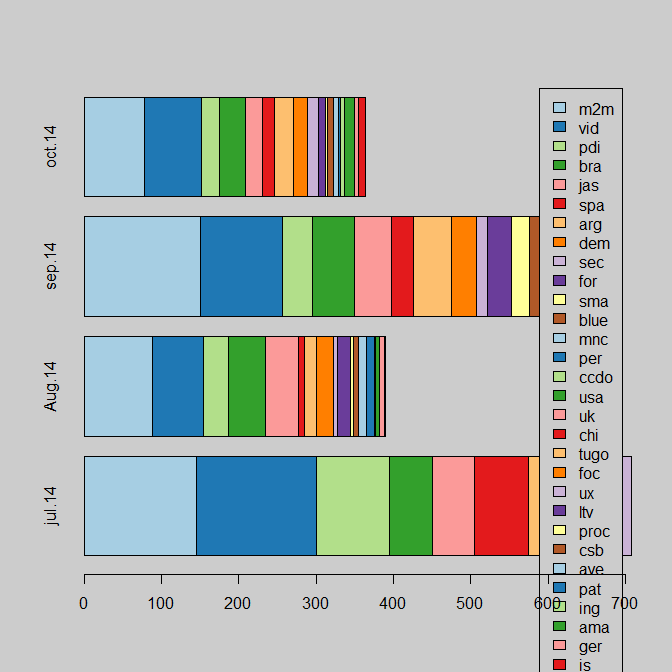
Seleccionemos una paleta que cumpla unos requisitos:

* que tenga cuantos más colores diferentes mejor
* que parta de colores con bajo contraste para permitir el máximo contraste con los objetos más pequeños

cols <- brewer.pal(12,"Paired")  
  
pizza(cols)



par(bg = "gray80")  
barplot(t, col = cols,   
 legend = rownames(t),  
 horiz = T)

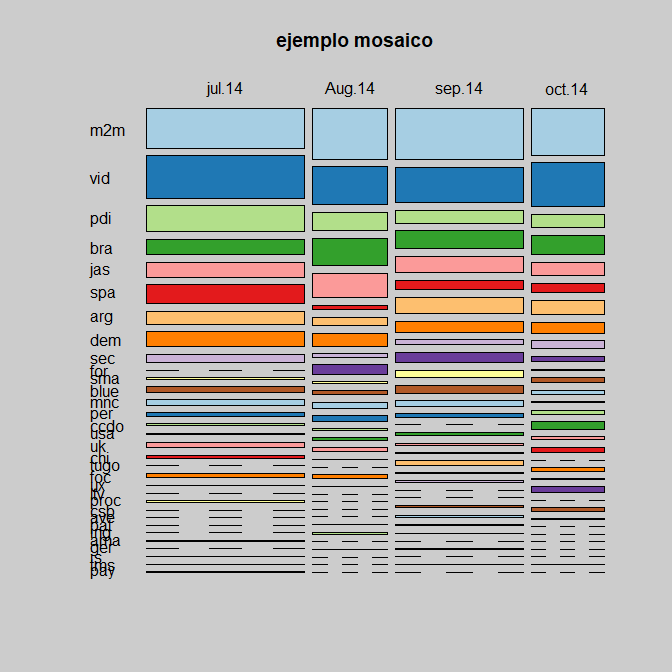


Aunque la apariencia sea mucho más agradable, no quiere decir que el gráfico se entienda o se perciba mejor, porque quizás el mecanismo de visualización no es el más adecuado.

Veamos otras posibilidades, entre ellas el mosaico.

mosaicplot es mucho mejor que mosaic pero debemos pasar los NA a 0

t2 <- as.matrix(t)  
   
t2[is.na(t2)] <- 0  
  
t2 <- as.table(t(t2))  
  
par(bg = "gray80")  
mosaicplot(t2,  
 main = "ejemplo mosaico",  
 color = cols,   
 cex = 1,   
 las = 1)



# DIBUJAMOS UN MAPA DE COLOR Y VAMOS APLICANDO RECOMENDACIONES

José Luis Cañadas tiene publicado un código R para la preparación de datos de la EPA aquí: <http://rpubs.com/joscani/mapa_paro_andalucia>. Nosotros aprovecharemos directamente el objeto generado. Veamos lo que contiene:

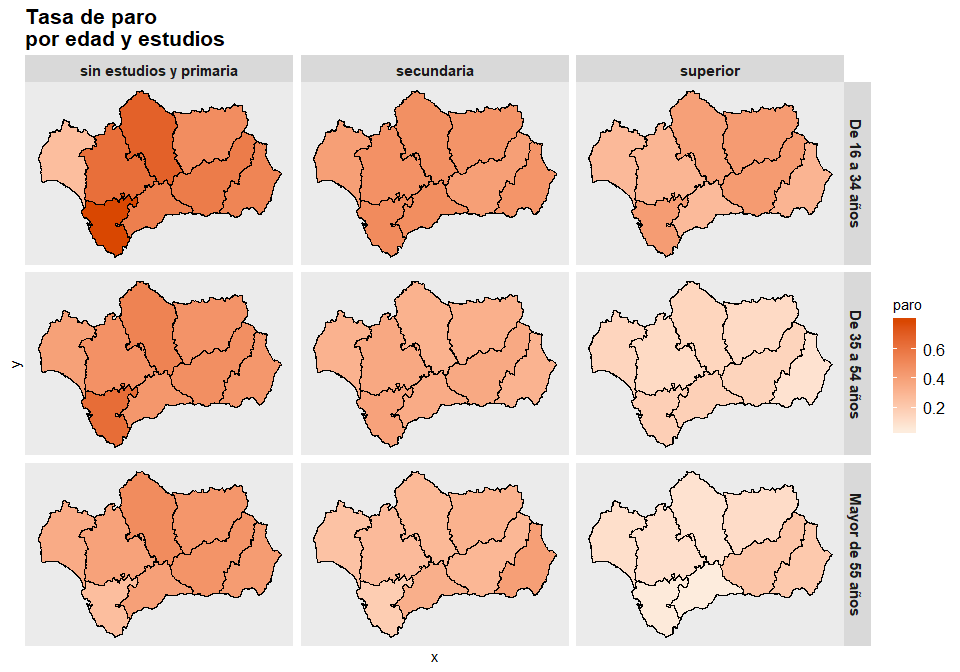
load("tasas\_paro\_andalucia.rda")  
  
# Objeto generado da problemas en sistemas no-windows  
  
Encoding(levels(tasa.paro.and.provincial$prov)) <- "latin1"  
Encoding(levels(tasa.paro.and.provincial$gedad)) <- "latin1"  
  
#tasa.paro.and.provincial

Necesitamos también los datos para dibujar las provincias andaluzas.

load("prov\_map.rda")  
  
load("and.ggmap.rda")

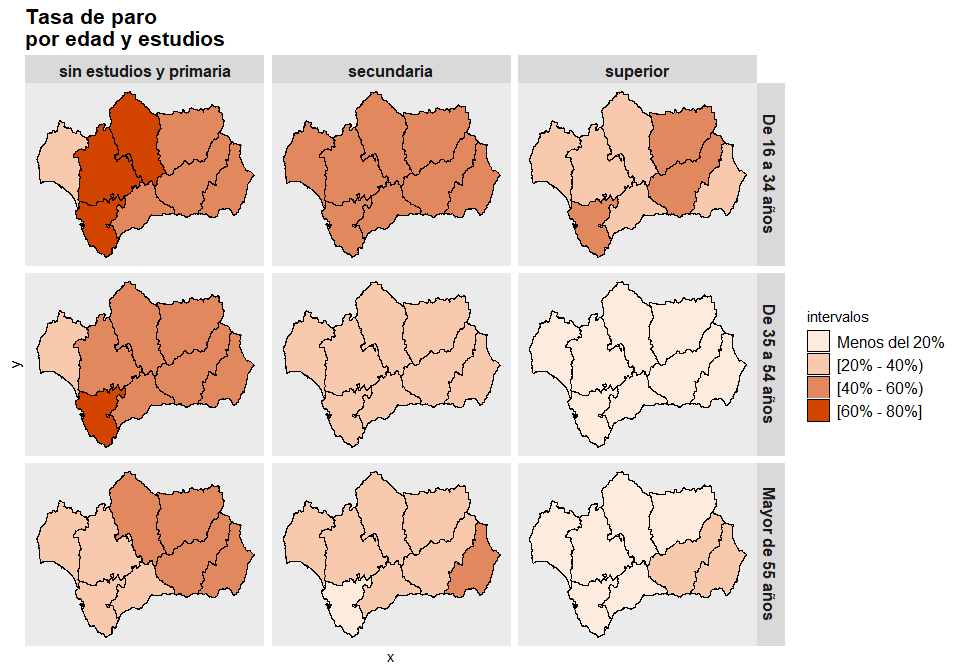
Ahora ya podemos crear el mapa, utilizando ggplot y geom\_map. Como facets utilizamos gedad y nforma3

ggplot(tasa.paro.and.provincial) +   
 geom\_map(aes(map\_id = cod\_prov, fill = paro),   
 map = and.ggmap, colour = "black") +   
 expand\_limits(x = and.ggmap$long, y = and.ggmap$lat) +   
 facet\_grid(gedad ~ nforma3) +   
 scale\_fill\_gradient(low = "#FDECDD", high = "#D94701") +   
 scale\_x\_continuous(breaks = NULL) +   
 scale\_y\_continuous(breaks = NULL) +   
 theme(axis.text.y = element\_blank(), axis.text.x = element\_blank(),   
 plot.title = element\_text(face = "bold",   
 size = rel(1.4)),   
 legend.text = element\_text(size = rel(1.1)),   
 strip.text = element\_text(face = "bold", size = rel(1))) +   
 labs(list(x = "", y = "", fill = "")) +   
 ggtitle("Tasa de paro\npor edad y estudios")



Mapa con tasa de paro categorizada.

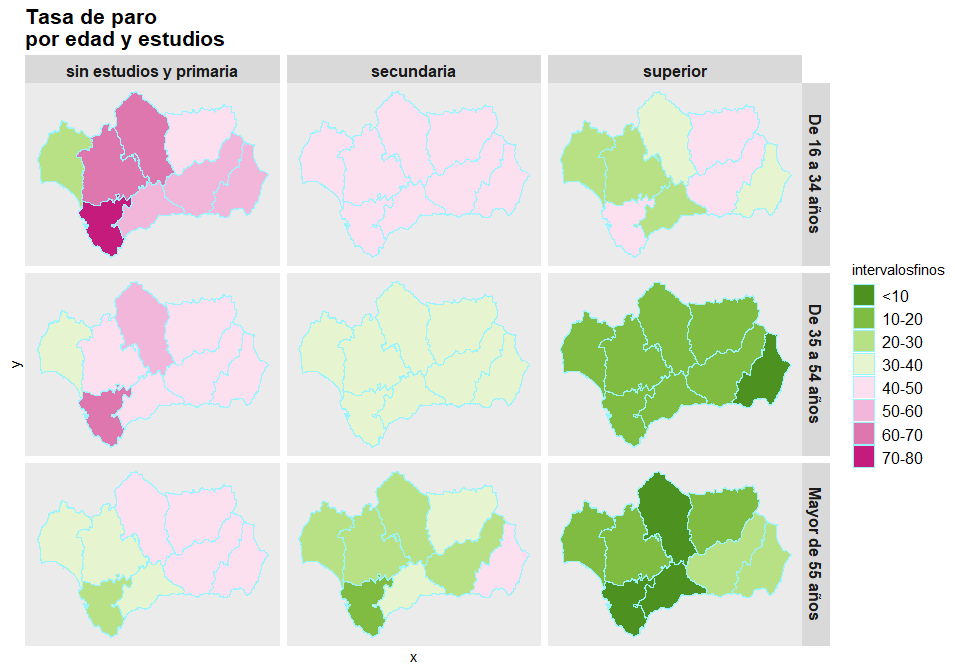
ggplot(tasa.paro.and.provincial) +   
 geom\_map(aes(map\_id = cod\_prov, fill = intervalos),   
 map = and.ggmap, colour = "black") +   
 expand\_limits(x = and.ggmap$long, y = and.ggmap$lat) +   
 facet\_grid(gedad ~ nforma3) +   
 scale\_fill\_manual(values = c("#FDEBDD",   
 "#F9C9AD",  
 "#E1885F",   
 "#D34400"),   
 labels = c("Menos del 20%",   
 "[20% - 40%)",   
 "[40% - 60%)",   
 "[60% - 80%]")) +   
 scale\_x\_continuous(breaks = NULL) +   
 scale\_y\_continuous(breaks = NULL) +   
 theme(axis.text.y = element\_blank(),   
 axis.text.x = element\_blank(),   
 plot.title = element\_text(face = "bold",   
 size = rel(1.4)),   
 legend.text = element\_text(size = rel(1.1)),   
 strip.text = element\_text(face = "bold",   
 size = rel(1.1))) +   
 labs(list(x = "", y = "", fill = "")) +   
 ggtitle("Tasa de paro\npor edad y estudios")



Mapa con tasa de paro con alguna mejora

* optimización de colores de líneas y fondo
* incremento de categorías
* uso de una escala divergente

cols <- brewer.pal(8,"PiYG")  
  
  
  
ggplot(tasa.paro.and.provincial) +   
 geom\_map(aes(map\_id = cod\_prov, fill = intervalosfinos),   
 map = and.ggmap, colour = "cadetblue1") +   
 expand\_limits(x = and.ggmap$long, y = and.ggmap$lat) +   
 facet\_grid(gedad ~ nforma3) +   
 scale\_fill\_manual(values = rev(cols),   
 labels = c("<10",   
 "10-20",   
 "20-30",   
 "30-40",  
 "40-50",  
 "50-60",  
 "60-70",  
 "70-80")) +   
 scale\_x\_continuous(breaks = NULL) +   
 scale\_y\_continuous(breaks = NULL) +   
 theme(axis.text.y = element\_blank(),   
 axis.text.x = element\_blank(),   
 plot.title = element\_text(face = "bold",   
 size = rel(1.4)),   
 legend.text = element\_text(size = rel(1.1)),   
 strip.text = element\_text(face = "bold",   
 size = rel(1.1))) +   
 labs(list(x = "", y = "", fill = "")) +   
 ggtitle("Tasa de paro\npor edad y estudios")

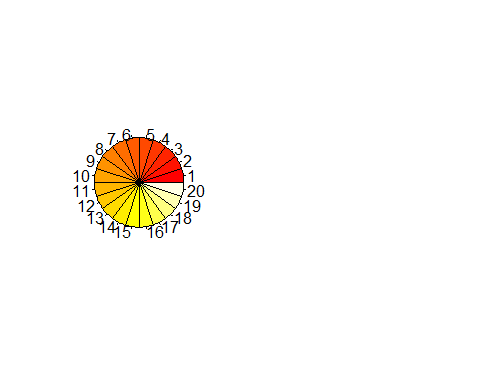


# POR QUÉ ES IMPORTANTE CONSIDERAR LA CEGUERA AL COLOR

dichromat es una librería que incorpora funciones que cambian las distinciones de color azul-verde a aproximadamente los efectos de las dos formas más comunes de ceguera a color: protanopia y deuteranopia.

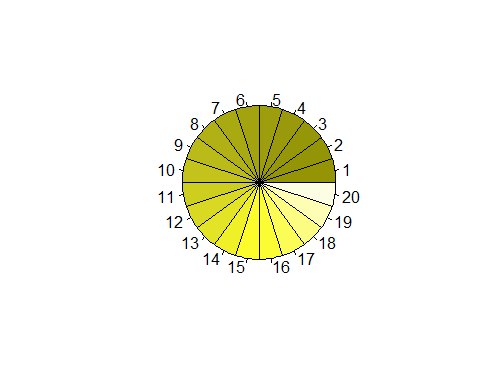
Un ejemplo de heatmap.

par(mfcol=c(1, 2))  
N <- 20  
pie(rep(1, N),  
 col = heat.colors(N))



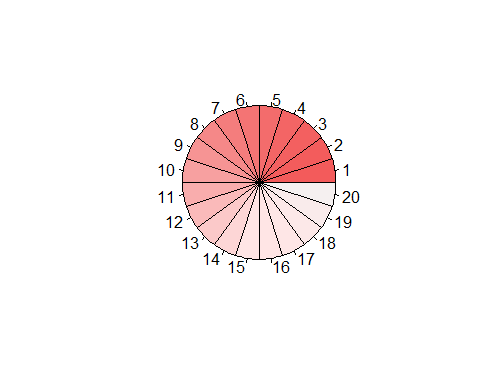
Cómo lo vería una persona con deuteranopia:

pie(rep(1, N),  
 col = dichromat(heat.colors(N), type = "deutan"))



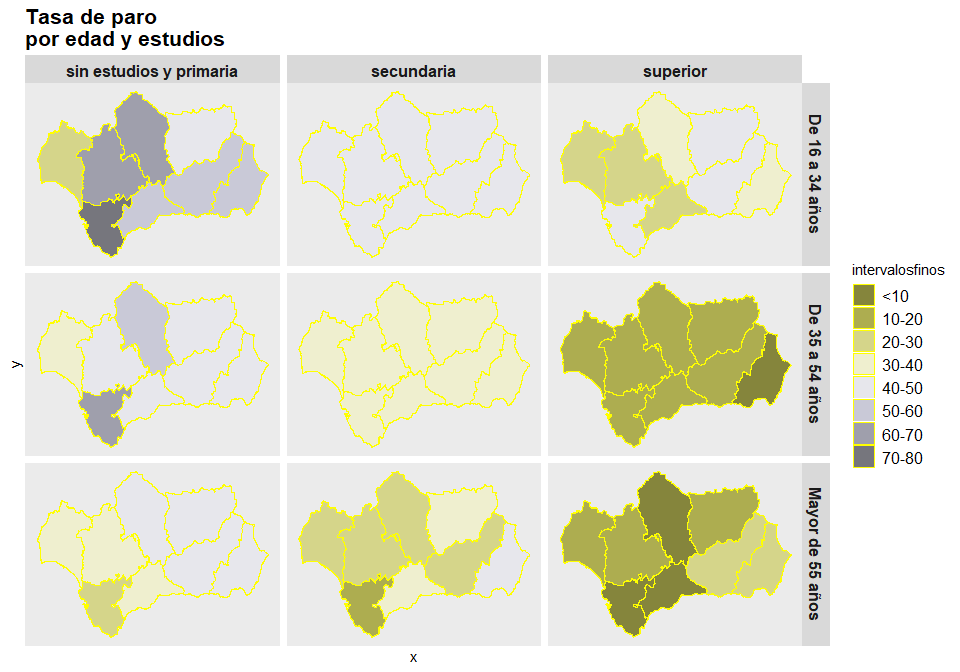
Y con protanopia

pie(rep(1, N),  
 col = dichromat(heat.colors(N), type = "tritan"))



Veamos la simulación de nuestro último mapa para una persona con deuteranopia (y aprovechamos para variar el color de las líneas de separación):

cols <- brewer.pal(8,"PiYG")  
  
  
  
ggplot(tasa.paro.and.provincial) +   
 geom\_map(aes(map\_id = cod\_prov, fill = intervalosfinos),   
 map = and.ggmap, colour = "yellow") +   
 expand\_limits(x = and.ggmap$long, y = and.ggmap$lat) +   
 facet\_grid(gedad ~ nforma3) +   
 scale\_fill\_manual(values = dichromat(rev(cols), type = "deutan"),   
 labels = c("<10",   
 "10-20",   
 "20-30",   
 "30-40",  
 "40-50",  
 "50-60",  
 "60-70",  
 "70-80")) +   
 scale\_x\_continuous(breaks = NULL) +   
 scale\_y\_continuous(breaks = NULL) +   
 theme(axis.text.y = element\_blank(),   
 axis.text.x = element\_blank(),   
 plot.title = element\_text(face = "bold",   
 size = rel(1.4)),   
 legend.text = element\_text(size = rel(1.1)),   
 strip.text = element\_text(face = "bold",   
 size = rel(1.1))) +   
 labs(list(x = "", y = "", fill = "")) +   
 ggtitle("Tasa de paro\npor edad y estudios")



A propósito de daltonismo y heatmaps, mirad comentarios en el foro de a propósito de los mapas del cáncer en España:

<http://elpais.com/elpais/2014/09/30/ciencia/1412091987_955227.html>

Guillermo López <http://eskup.elpais.com/glopllu>

heber <http://eskup.elpais.com/C1412009164-b3defc80060c0a3a32fa0a1b8803d2eb/1412173041-45d715fa118fefc62185df50eeef8120>

Gracias por la explicación. Normalmente en ciencia se utiliza un código de colores común a lo que se conoce como la codificación semafórica donde verde significa poco, amarillo moderado y rojo peligroso. Por eso utilizan este tipo de codificación. Cierto es que a los daltónicos esa codificación no se les puede aplicar. Pero a los no daltónicos, los colores verde y rojo son los que mejor se distinguen entre sí por las características de las células de la retina.

Julio Fernández <http://eskup.elpais.com/juli0>

heber <http://eskup.elpais.com/C1412009164-b3defc80060c0a3a32fa0a1b8803d2eb/1412174451-120c5b04818bff4f681b6de84a271ff5>

Muchas gracias, esa sería una solución, pues la mayoría de daltónicos (en mi caso ausencia aguda de percepción de la frecuencia del verde) desarrollamos un sentido para diferenciar tonos en relación a su luminosidad, hasta el punto de que “coloreamos” lo que en realidad es gris (¿cuánto tiempo ha de pasar hasta que me de cuenta de que la película en la TV no es en color sino en B/N?). El cambiar los gráficos hacia colores visibles o B/N (jamás soy capaz de leer un gráfico con cólores) sería un detalle, por no hablar del tema de los semáforos (distintos tonos en distintas ciudades…) Foro EL PAÍS

# APENDICE - COLOR EN PYTHON

* Escalas o paletas de color en python <http://web.stanford.edu/~mwaskom/software/seaborn/tutorial/color_palettes.html>
* Más color y python, muy buena discusión del tema <https://jakevdp.github.io/blog/2014/10/16/how-bad-is-your-colormap/>

# APENDICE

¿Necesitas software más especializado? Consulta

<http://visionscience.com/documents/strasburger/strasburger.html>

Quien por cierto menciona R así:

“Modeling Psychophysical Data in R:

However, given the emphasis on freely available software, the – freely available – general purpose statistical package R must be mentioned here. Ken Knoblauch from Lyon, France has written three libraries for R that are aimed at psychophysics, psyphy, MLDS, and MLCM, explained above and below. In particular there is now a comprehensive book on modeling psychophysical data by Knoblauch and Maloney. (Here is a preview and Chapter 1) "

# APENDICE HOJAS DE ESTILO PERSONALIZADAS PARA GGPLOT

Es posible preparar hojas de estilo sobre todo para ggplot. En concreto estas se han hecho bastante populares, reproducen los estilos del sitio de Nate Silver

Hoja de estilo: <https://github.com/alex23lemm/theme_fivethirtyeight>

Explicación de la misma: <http://austinclemens.com/blog/2014/07/03/fivethirtyeight-com-style-graphs-in-ggplot2/>

El sitio de Nate Silver <http://fivethirtyeight.com/>

# APENDICE COLORS IN THE COMPLEX DOMAIN

Como propone Antonio Sánchez Chinchón en su blog <https://fronkonstin.com/2014/10/01/complex-domain-coloring/>

require(ggplot2)  
  
f = function(x) (1+1i)\*log(sin((x^3-1)/x))  
  
z=as.vector(outer(seq(-5, 5, by =.01),1i\*seq(-5, 5, by =.01),'+'))  
  
z=data.frame(x=Re(z),  
 y=Im(z),  
 h=(Arg(f(z))<0)\*1+Arg(f(z))/(2\*pi),  
 s=(1+sin(2\*pi\*log(1+Mod(f(z)))))/2,  
 v=(1+cos(2\*pi\*log(1+Mod(f(z)))))/2)

## Warning in sin(2 \* pi \* log(1 + Mod(f(z)))): Se han producido NaNs

## Warning in cos(2 \* pi \* log(1 + Mod(f(z)))): Se han producido NaNs

z=z[is.finite(apply(z,1,sum)),]  
  
opt=theme(legend.position="none",  
 panel.background = element\_blank(),  
 panel.grid = element\_blank(),  
 axis.ticks=element\_blank(),  
 axis.title=element\_blank(),  
 axis.text =element\_blank())  
  
ggplot(data=z, aes(x=x, y=y)) + geom\_tile(fill=hsv(z$h,z$s,z$v))+ opt

