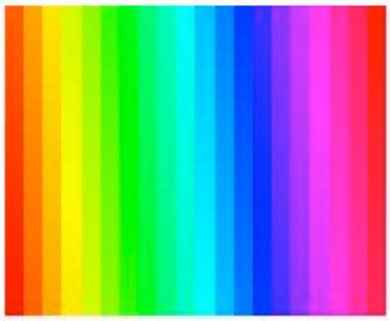
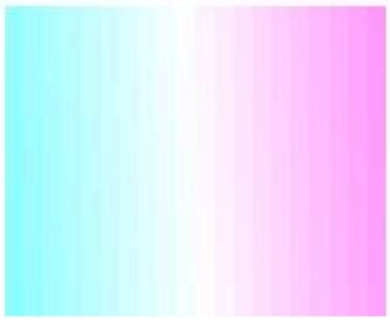
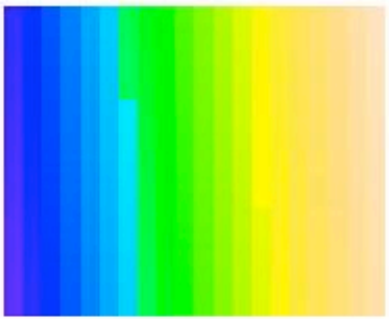
Colors

A color of R Chart

유 충현
블로그 모음 4탄(http://blog.naver.com/bdboys) • (주)오픈베이스 • 2010년 10월 3일







Colors

Chart를 그리기 위해서는 점, 선, 면, 문자 등의 가시적인 요소가 필연적이다. 이들 요소가 모여서, 범례를 만들고, 좌표축을 만들고, 타이틀 등과 그래프를 만든다. 여기에 각각의 객체들에 색상을 부여해서 가독성을 높일 수 있다. 단일 색상이 아니라 여러 색상을 적절하게 조합하면 보다 직관적인 Chart를 생성할 수 있다.

색채학이라는 학문이 있을 정도로 색에 대한 과학적인 정보는 무궁무진하다. 여기서는 기본적인 색에 대한 부분을 다루고자 한다. 구체적인 색의 원리 등은 다른 문서를 참고하기 바란다.

그러면 R Graphics에서의 색상에 대해서 알아보자.

1. colors/colours

R이 인식할 수 있는 내장된 색상 이름을 리턴하는 함수이다. 657개의 색상 이름을 반환하며, 두 함수는 이름만 다를 뿐 동일한 함수이다. 모두 colors() 함수를 Internal Call한다.

```
> colors
function ()
.Internal(colors())
<environment: namespace:grDevices>
> colours
function ()
.Internal(colors())
<environment: namespace:grDevices>
> length(colors())
[1] 657
> colors()[1:10]
 [1] "white"          "aliceblue"      "antiquewhite"   "antiquewhite1"
 [5] "antiquewhite2"  "antiquewhite3"  "antiquewhite4"  "aquamarine"
 [9] "aquamarine1"    "aquamarine2"
```

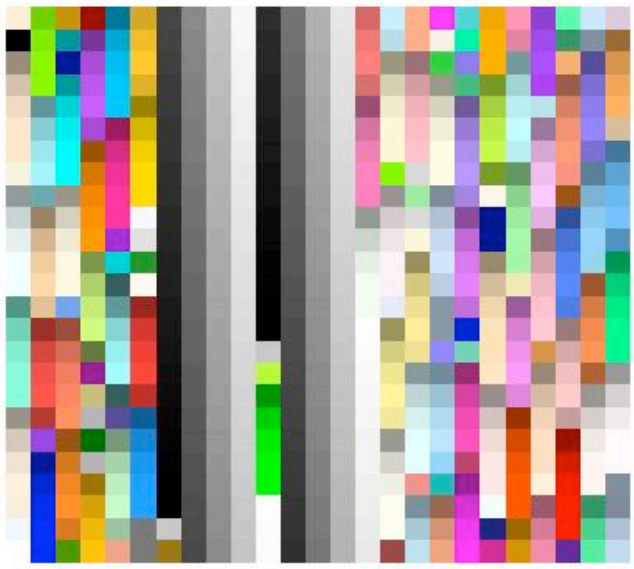
다음과 같은 함수를 만들었다. 이 함수는 그래프 좌표에 주어진 색깔들을 바둑판처럼 출력한다. 단, 가로와 세로의 동수를 만들기 위해서 몇 개의 색상은 출력이 되지

않을 수 있다. 만약 모든 입력 색상을 다 출력하려면 floow대신 round 함수를 사용하면 되지만 출력 모양이 사각형이 되지 않을 수 있다.

```
col.map <- function(cols = colours()) {
  n = length(cols)
  cnt = floor(sqrt(n))
  plot.new()
  plot.window(xlim = c(0,cnt),
              ylim = c(0,cnt))

  for (i in 1:cnt)
     for (j in 1:cnt)
       rect(i-1, j-1, i, j, col = cols[(i-1)*cnt+j], border = NA)
}
```

그러면 colors()를 출력해 보자.

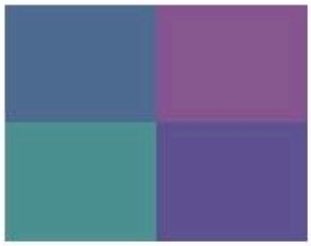


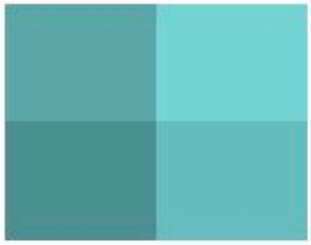
이 색상으로 우리는 좌표를 그릴 수 있고, 문자를 출력하는 등 Graphic Device 자유롭게 활용 수 있다.

2. hsv

색의 세가지 속성인 h=색상, s=채도, v=명도를 사용해서 색깔을 지정한다. 이외에 감마나, 알파도 지정할 수 있다. R에서의 많은 색상관련 함수들이 이 함수를 사용해서 구현하였다.

```
> par(mfrow=c(2,2))
> col.map(c(hsv(.5,.5,.5),hsv(.6,.5,.5),hsv(.7,.5,.5),hsv(.8,.5,.5)))
> col.map(c(hsv(.5,.5,.5),hsv(.5,.6,.5),hsv(.5,.7,.5),hsv(.5,.8,.5)))
> col.map(c(hsv(.5,.5,.5),hsv(.5,.5,.6),hsv(.5,.5,.7),hsv(.5,.5,.8)))
> col.map(c(hsv(.5,.5,.5),hsv(.6,.6,.6),hsv(.7,.7,.7),hsv(.8,.8,.8)))
```





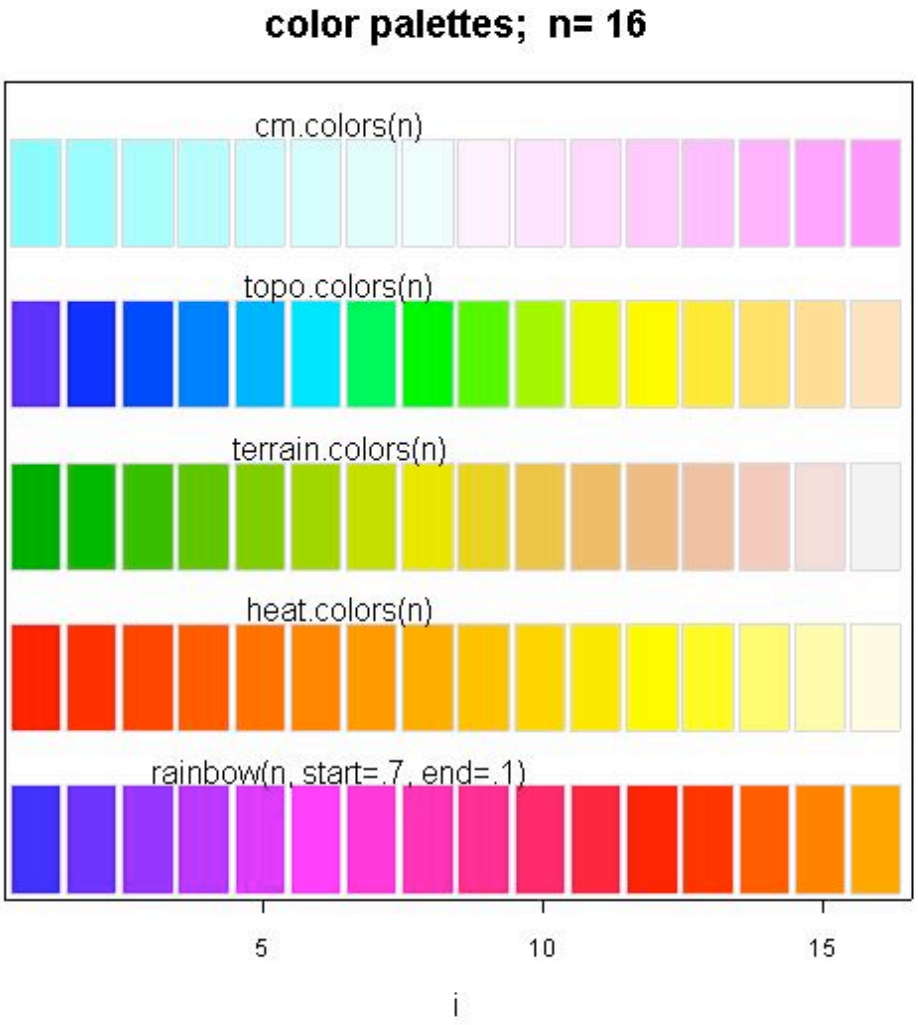
3. rainbow Family 함수

R에서 hsv 함수를 이용해서 근접한 연속선상의 색상들을 n개의 벡터로 반환하는 함수를 다음과 같이 재공한다.

```
rainbow(n, s=1, v=1, start=0, end=max(1, n - 1)/n, gamma=1)
heat.colors(n)
terrain.colors(n)
topo.colors(n)
cm.colors(n)
```

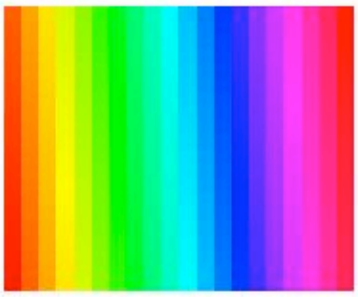
```
> demo.pal <-
   function(n, border = if (n<32) "light gray" else NA,
            main = paste("color palettes;  n=",n),
            ch.col = c("rainbow(n, start=.7, end=.1)", "heat.colors(n)",
                       "terrain.colors(n)", "topo.colors(n)", "cm.colors(n)"))
{
   nt <- length(ch.col)
   i <- 1:n; j <- n / nt; d <- j/6; dy <- 2*d
   plot(i,i+d, type="n", yaxt="n", ylab="", main=main)
   for (k in 1:nt) {
      rect(i-.5, (k-1)*j+ dy, i+.4, k*j,
           col = eval(parse(text=ch.col[k])), border = border)
      text(2*j,  k * j +dy/4, ch.col[k])
   }
}
n <- if(.Device == "postscript") 64 else 16
    # Since for screen, larger n may give color allocation problem
demo.pal(n)
```

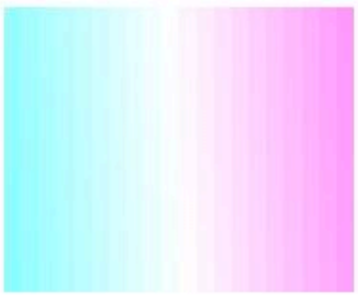
이 예제는 다음과 같은 결과를 출력한다.

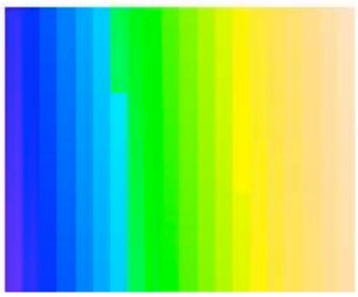


그러면 앞서 만든 col.map를 이용해서 몇가지 연속선상의 색깔을 출력해 보자.

```
> par(mfrow=c(2,2))
> col.map(rainbow(400))
> col.map(heat.colors(400))
> col.map(cm.colors(400))
> col.map(topo.colors(400))
```





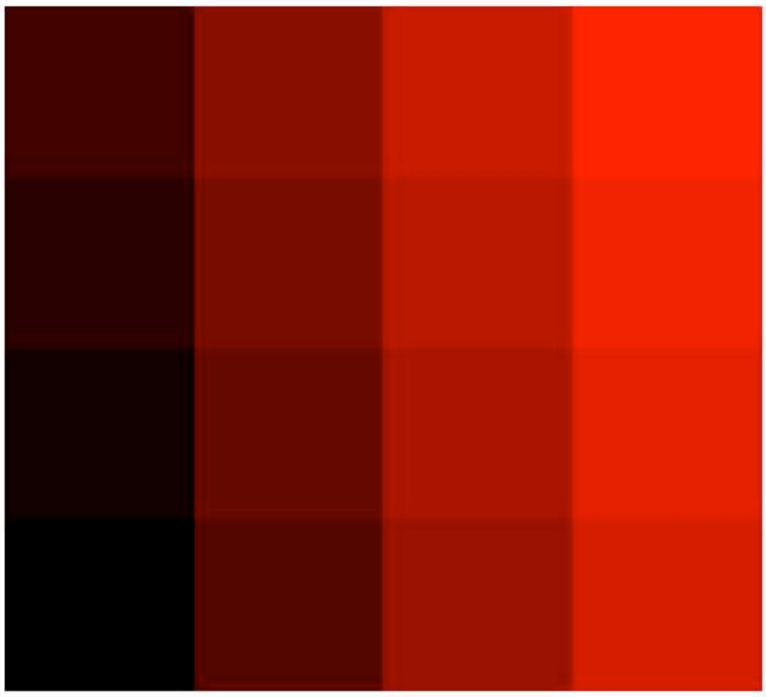


4. rgb

빛의 3원색인 Red, Green, Blue를 이용해서 색상을 나타낸다.

rgb(red, green, blue, alpha, names = NULL, maxColorValue = 1)

```
> reds <- rgb((0:15)/15, g=0,b=0, names=paste("red",0:15,sep="."))
> reds
 [1] "#000000" "#110000" "#220000" "#330000" "#440000" "#550000"
 [7] "#660000" "#770000" "#880000" "#990000" "#AA0000" "#BB0000"
 [13] "#CC0000" "#DD0000" "#EE0000" "#FF0000"
> col.map(reds)
```



5. gray

회색계열의 색상을 만든다. level은 0과 1사이의 숫자이며, 1이면 흰색, 0이면 검정색이다.

gray(level)

```
> gray(0:8 / 8)
[1] "#000000" "#202020" "#404040" "#606060" "#808080" "#9F9F9F"
[7] "#BFBFBF" "#DFDFDF" "#FFFFFF"
> col.map( gray(0:8 / 8))
```