S-PLUS vs R

그래픽에 대해서

유 충현
블로그 모음 11탄(http://blog.naver.com/bdboys) • (주)오픈베이스 • 2012년 10월 2일

Color Table by terrain.colors(20)

Color Table by topo.colors(20)

Colors (S-PLUS vs R)

80년대 말 PC를 처음 접했을 때, 사용하던 IBM XT는 허큘레스 그래픽 카드가 탑재되었고 모니터는 모노크롬 모니터였다. 모니터 색상은 녹색의 단색이었지만 어떤 이는 이것을 보고 칼라 모니터라고도 부르기도하였다. 흑백의 반대로서 칼라만 생각한 것이다. 흑백이 아니니 칼라라고 하였던 것이다. 이후, CGA, VGA, Super VGA 등의 비약적인 발전을 거듭하여 지금과 같은 컴퓨팅 환경을 이룩한 것이다.

다른 언어에 비해 R/S-PLUS는 이러한 컴퓨터의 디스플레이의 발전을 덕을 많이 보았다. 그 이유는 S System의 장점 중에 강력한 Graphic 솔루션으로서의 Chart가 자리잡고 있기 때문이다.

"잘 그린 Chart하나가 열 기술통계 Report보다 낫다."

단변량의 Chart를 위해서는 굳이 다른 색깔로 드로잉할 필요는 없다. 그러나 여러 변량이나, 아니면 보여주고자 하는 내용이 여러 개일 경우에는 서로 다른 색이나 심볼로 보여주는 것이 변별력이 높아 사용자의 이해를 쉽게 한다. 그러기 때문에 Chart를 그리는 함수에서는 색상의 선택을 지원하는 인수를 가지고 있다. 일반적으로 col이라는 인수가 그것이다.

이번에는 이 col의 인수에 사용되는 color에 대해서 R과 S-PLUS의 차이점을 알아보기로 한다. R은 2.1.0, S-PLUS는 6.2 버젼을 이용하였다.

지난번에 col.map이라는 함수는 R에서는 사용이 가능한데 S-PLUS에서 사용할 수 없어서, 양쪽 시스템에 공통적으로 사용할 수 있는 col.table 이라는 함수를 만들어 보았다.

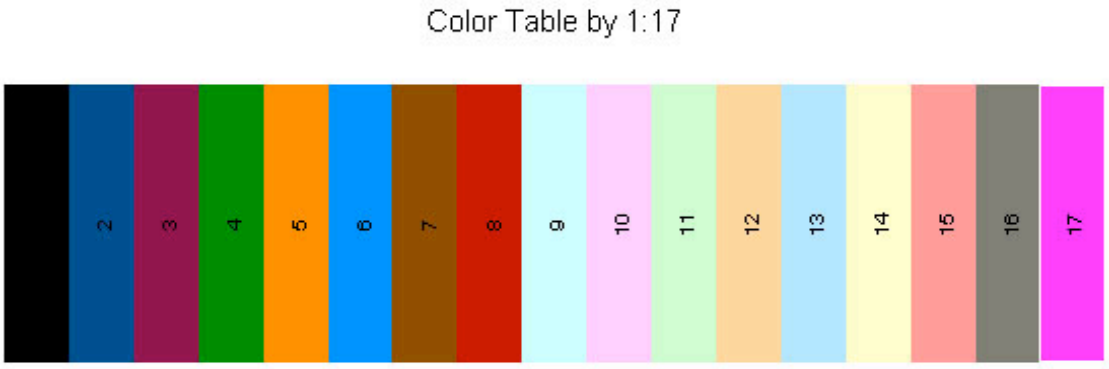
```
col.table <- function(cols)
{
    n = length(cols)

    plot(seq(n), rep(1, n), xlim = c(0, n), ylim = c(0, 1), type = "n", xlab = "",
            ylab = "", axes = F)
    title(paste("Color Table by",deparse(substitute(cols))))
```

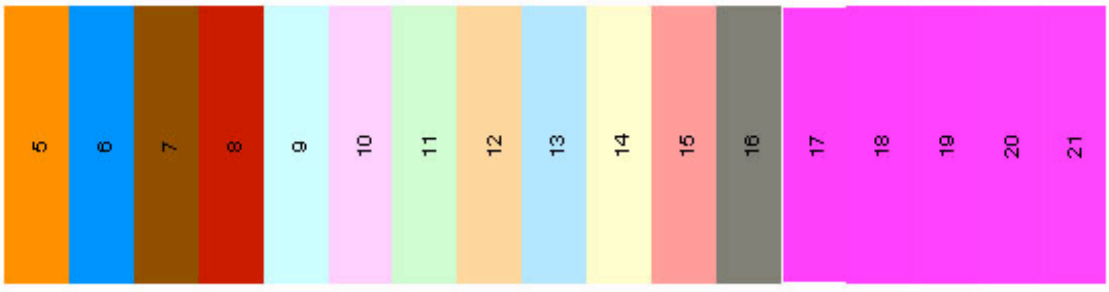
```
    for(i in 1:n) {
        polygon(c(i - 1, i - 1, i, i), c(0.05, 1, 1, 0.05), col = cols[i])
        text(mean(c(i - 1, i)), 0.52, labels = cols[i], srt=90, adj=0.5)
    }
}
```

S-PLUS

S-PLUS에서의 Color인자 값은 1부터 17까지의 정수를 사용한다. 즉, 17색깔을 사용할 수가 있다. 만약에 17보다 큰 정수를 사용하면, 17에 해당하는 색상이 출력된다. 다음 명령어를 입력해서 출력물을 비교해 보자.

```
> graphsheet(format='JPG', file="color_s.jpg")
> par(mfrow=c(2,1))
> col.table(1:17)
> col.table(5:21)
> dev.off()
```



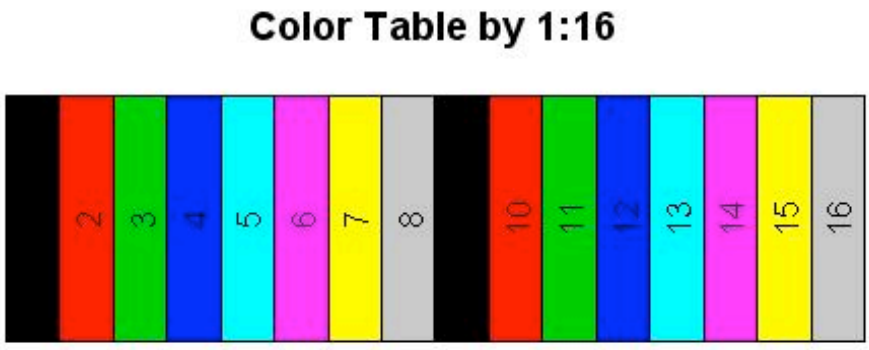


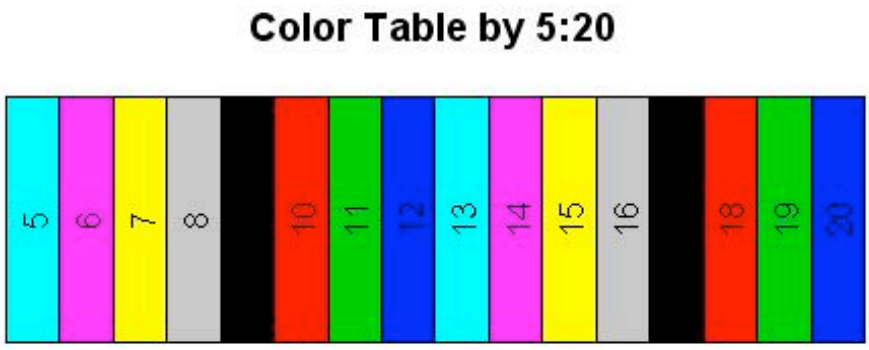
위의 그림은 1부터 16까지의 정수의 색상을 표현한 것이고, 아래의 그림은 5부터 20까지의 정수의 색상을 표현하였다. 17부터 20까지는 17의 색상으로 출력됨을 알 수 있다.

R

R에서의 Color인자 값은 1부터 8까지의 정수를 사용한다. 즉, 8가지 색깔을 사용할 수가 있다. 만약에 8보다 큰 정수를 사용하면, Rotation된 색상을 출력한다. 즉, 9는 1번의 색상을 10은 2번의 색상을 출력한다.

```
> jpeg(file="color_r.jpg", 550,550)
> par(mfrow=c(2,1))
> col.table(1:16)
> col.table(5:20)
> dev.off()
```





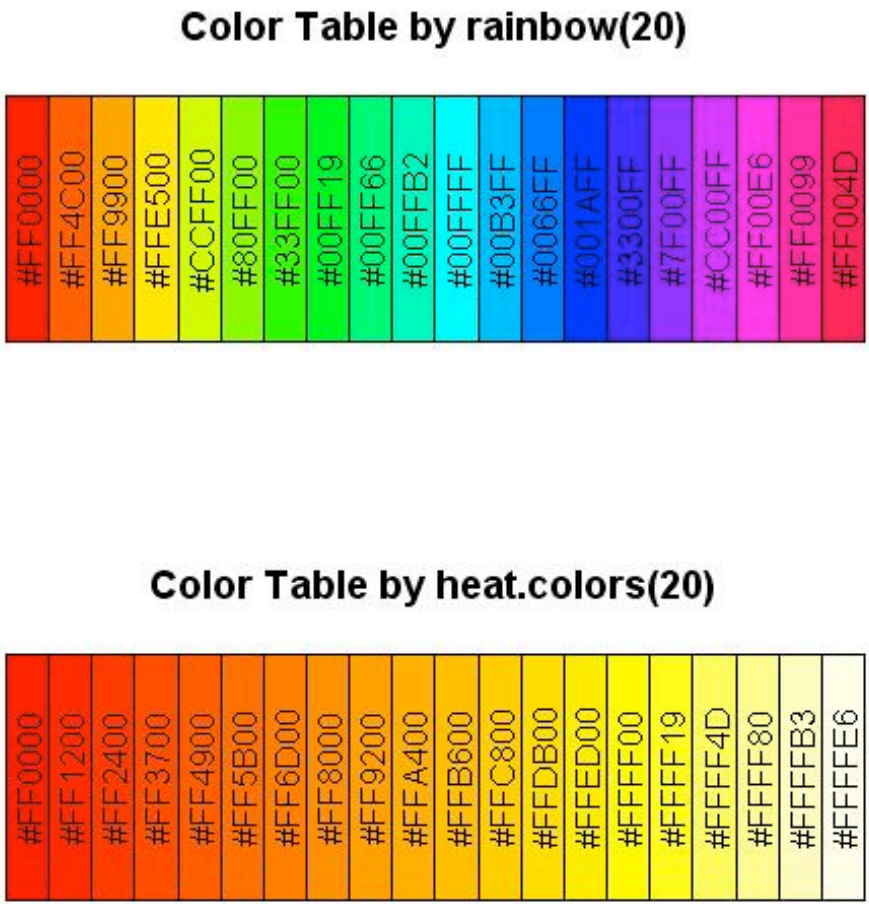
col의 인자값으로 S-PLUS가 16가지 색상을, R이 8가지 색상만 지정한다고 생각하면 안된다.
R에서는 col 인자에 RGB로 구성된 color도 지정할 수 있기 때문에 그 색상은 무궁무진한다.
(RGB의 색상 조합의 수는 256*256*256 = 16,777,216 가지의 색상을 정의할 수 있다.)

RGB란 빛의 삼원색인 R(Red), G(Green), B(Blue)의 조합으로 정의되는 색상이다. 이 값의 형식은 "#RRGGBB"이다. 각각의 영역의 두자리는 00(십진수 0)부터 FF(십진수 255)까지의 16진수가 올 수 있다. 00은 해당 색상이 전혀 없는 것이고, FF는 해당 색상이 충만함을 의미한다.
그러므로 "#FF0000"은 빨간색(Red)이 된다. "#00FF00"은 초록색(Green), "#0000FF"은 파란색(Blue)이 된다.

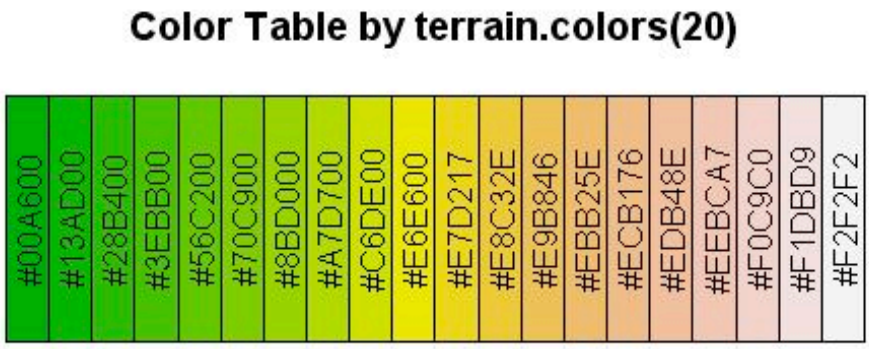
그러면 "#FFFFFF"은 무슨 색일까? 빛의 삼원색을 동일하게 섞으면 흰색이 된다고 하였으니 흰색이라고 할 수 있다. 그리고 "#000000"은 세가지 색깔의 빛이 아무 것도 없으니 검은색이 되겠다. 빛이 없으면 검은 것은 당연하다.

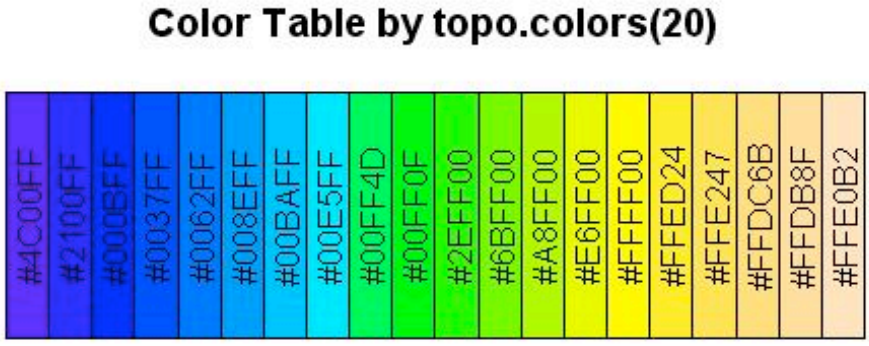
R은 여러 방법으로 다양한 색상을 선택할 수 있다. rainbow, heat.colors 등의 함수를 이용하는데, 이 함수의 결과값은 "#RRGGBB"형식의 색상이다. 이미 앞서 다루었던 주제이기 때문에 개별 방법에 대한 설명은 생략하고 몇 가지 예제로 이번 이야기를 마친다.

```
> jpeg(file="color_r1.jpg", 550,550)
> par(mfrow=c(2,1))
> col.table(rainbow(20))
> col.table(heat.colors(20))
> dev.off()
```

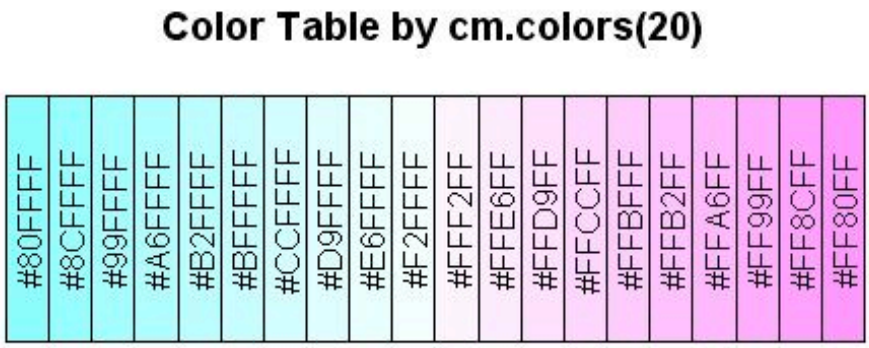


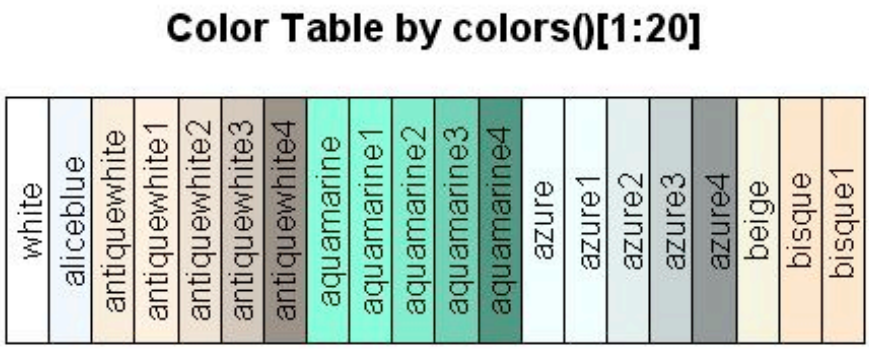
```
> jpeg(file="color_r2.jpg", 550,550)
> par(mfrow=c(2,1))
> col.table(terrain.colors(20))
> col.table(topo.colors(20))
> dev.off()
```





```
> jpeg(file="color_r3.jpg", 550,550)
> par(mfrow=c(2,1))
> col.table(cm.colors(20))
> col.table(colors()[1:20])
> dev.off()
```





Line Type (S-PLUS vs R)

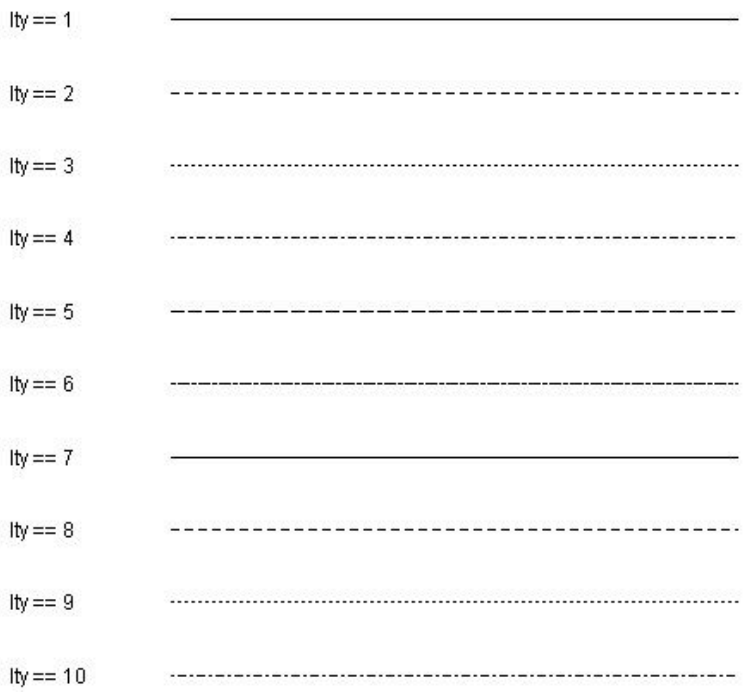
S-PLUS와 R의 Graphics에서 사용되는 Line Type에 대해서 차이점을 살펴 본다.

두 가지 이상의 집단의 그래프를 구분하는데 변별력을 높히는 방법에 색깔의 차이 못지 않게 선의 모양의 차이도 중요하게 사용된다. 실선, 파선, 괘선 등의 선의 모양을 두 시스템을 비교해서 설명하기로 한다.

일반적으로 High Level 그래픽 함수에서 선의 종류를 결정하는 인수는 lty이다. 이 lty의 값은 S-PLUS와 R이 조금 다른데 다음의 Code를 이용해서 결과를 출력한 후 비교해 보자.

```
> plot(1,1, xlim=c(1,10), ylim=c(1,10), type="n",xlab="",ylab="",axes=F)
> for (i in 1:10) {
    text(1,11-i,label=paste("lty ==",i),adj=0)
    lines(c(3,10), c(11-i,11-i), lty=i, lwd=1.5)
  }
```

1부터 8까지의 8가지 선의 종류가 있다. 8을 넘수 정수는 Recycle Rule에 의해서 1부터 다시 반복된다.



1부터 6까지의 6가지 선의 종류가 있다. 마찬가지로 6을 넘수 정수는 Recycle Rule에 의해서 1부터 다시 반복된다. 그런데 1을 제외한 정수에 대해서는 모양이 S-PLUS와 R이 다름을 알 수 있다.

그런데 엄밀히 말하면 7가지라고 할 수 있다. 0일 경우에는 blank line이 만들어진다. 즉, R은 "blank", "solid", "dashed", "dotted","dotdash", "longdash","twodash"의 7가지 Line Type이 있다고도 할 수 있다. 그런데 S-PLUS에서는 0이 1과 같은 "solid"이다.

색깔에서 R은 S-PLUS와 달리 RGB의 값으로 색상을 지정하는 방법이 있었다. Line Type에서도 R은 S-PLUS에 없는 확장 기능이 있다. 그것은 문자열로 선의 형태를 지정하는 방법이다. 형식은 다음과 같다.

lty="BW", lty="BWBW"

B는 Black, W는 White를 의미하며 16진수가 온다. B는 선의 길이를 W는 공백의 길이를 의미한다.

파선일 경우는 "33" 정도로 표시할 수 있다. 선의 길이가 3, 공백의 길이가 3으로 이루어진 Line을 정의한 것이다. 실제로 선의 길이보다 공백의 길이가 짧다. 같은 3이라고 하지만 공백의 폭이 더 짧은 것이다. 그러므로 기하학적인 의미보다는 패턴으로 이해하기 바란다.

네자리로 이루어진 패턴은 "dotdash"와 같은 것을 정의할 때 상용한다. 예를 들면 긴 선, 짧은 선으로 이루어진 패턴을 정의할 때 사용할 수 있다. 앞의 패턴과 뒤의 패턴이 Recycle되어서 Line Type을 결정하는 것이다.

다음 Script를 실행시켜 결과를 보면 쉽게 이해가 될 것이다.

```
> plot(1,1, xlim=c(1,10), ylim=c(1,10), type="n", xlab="",ylab="",axes=F)
> lines(c(3,10),c(10,10), lty="22")
> lines(c(3,10),c(9,9), lty="33")
> lines(c(3,10),c(8,8), lty="24")
> lines(c(3,10),c(7,7), lty="42")
> lines(c(3,10),c(6,6), lty="F2")
> lines(c(3,10),c(5,5), lty="2F")
> lines(c(3,10),c(4,4), lty="3313")
> lines(c(3,10),c(3,3), lty="FF52")
> lines(c(3,10),c(2,2), lty="F252")
> lines(c(3,10),c(1,1), lty="FF29")
>
> text(1,10, "lty==₩"22₩"", adj=0)
> text(1,9, "lty==₩"33₩"", adj=0)
> text(1,8, "lty==₩"24₩"", adj=0)
> text(1,7, "lty==₩"42₩"", adj=0)
> text(1,6, "lty==₩"F2₩"", adj=0)
> text(1,5, "lty==₩"2F₩"", adj=0)
> text(1,4, "lty==₩"3313₩"", adj=0)
> text(1,3, "lty==₩"FF52₩"", adj=0)
> text(1,2, "lty==₩"F252₩"", adj=0)
> text(1,1, "lty==₩"FF29₩"", adj=0)
```

결과는 다음 그림과 같다.


lty=="22"
lty=="33"
lty=="24"
lty=="42"
lty=="F2"
lty=="2F"
lty=="3313"
lty=="FF52"
lty=="F252"
lty=="FF29"