

# La gestion avancée des couleurs sous R

Gaël Millot

UMR3244 et Université Pierre et Marie Curie  
Institut Curie  
26 rue d'Ulm  
75248 Paris Cedex 05  
FRANCE

Email : [gael.millot@curie.fr](mailto:gael.millot@curie.fr)  
<http://perso.curie.fr/Gael.Millot/index.html>

Issu du fichier "Code complet du livre Comprendre et Réaliser les tests statistiques à l'aide de R", paragraphe 1.9.6

Disponible à cette adresse : <http://perso.curie.fr/Gael.Millot/Telecharger.htm>

## Rappel des fonctions simples et limitées de R

`colors()` : 657 couleurs sous la forme d'une liste de caractères

`palette()` : attribue des chiffres aux caractères (pour l'argument `col` des fonctions)

Exemple :

```
colors()  
par(ann=FALSE, xaxt="n", yaxt="n", bty="n")  
plot(1, col="aquamarine", pch=16, cex=10)
```

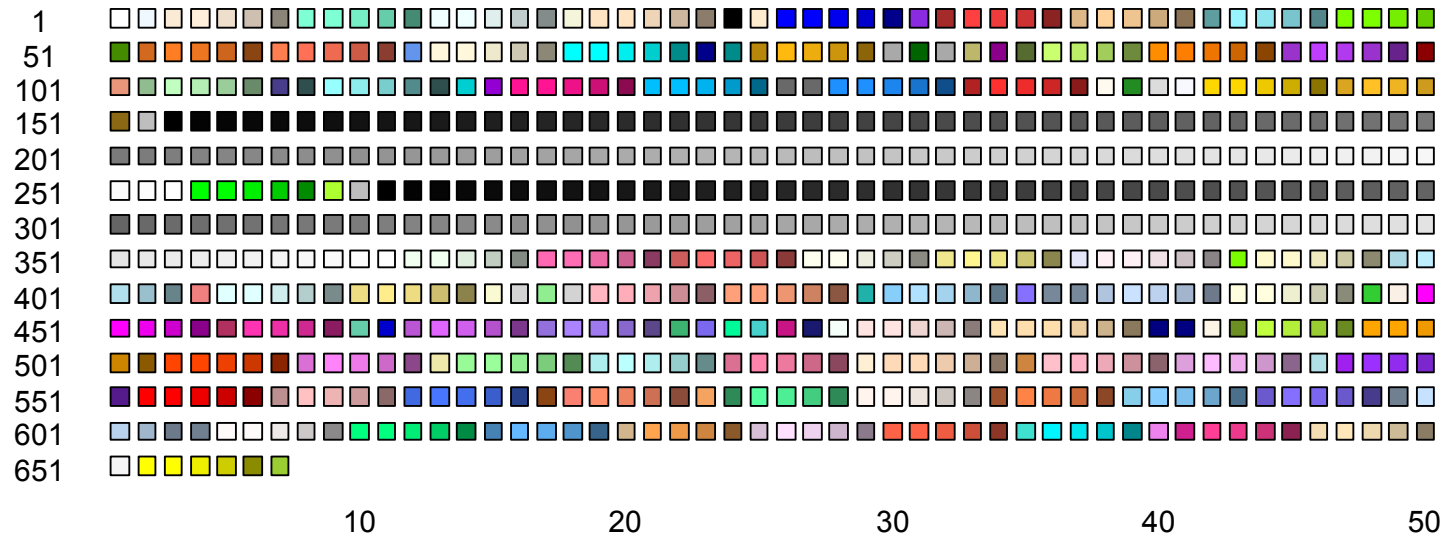


```
palette()  
plot(1, col=6, pch=16, cex=10)
```



# Astuce pour choisir une couleur de colors()

Exemple avec le fichier de code



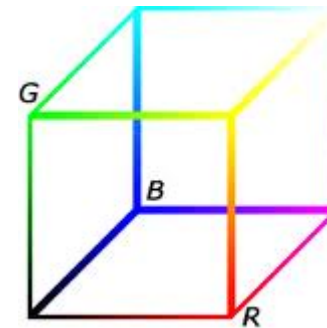
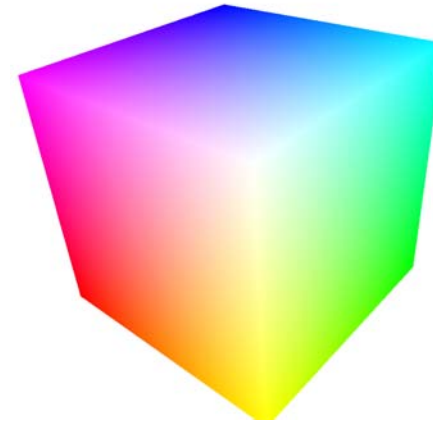
# L'espace colorimétrique RGB (Red Green blue)

RVB en français (Rouge Vert Bleu)

Cube tridimensionnel

Aspects remarquables :

- Un coin est rouge complet (Full red)
- Un coin est vert complet (Full green)
- Un coin est bleu complet (Full blue)
- Un coin est noir complet
- Un coin est blanc complet



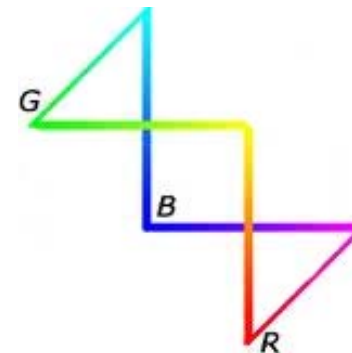
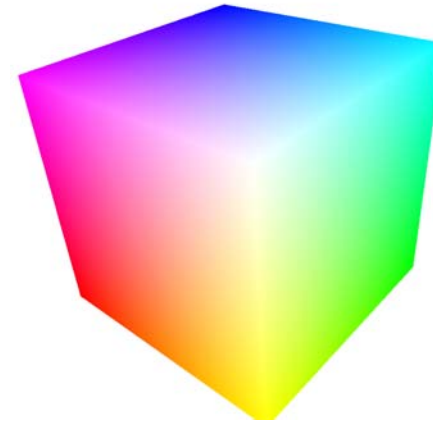
# L'espace colorimétrique RGB (Red Green blue)

RVB en français (Rouge Vert Bleu)

Cube tridimensionnel

Aspects remarquables :

- Un coin est rouge complet (Full red)
- Un coin est vert complet (Full green)
- Un coin est bleu complet (Full blue)
- Un coin est noir complet
- Un coin est blanc complet
- Les cotés qui ne touchent ni le coin blanc ni le noir forment l'ensemble des couleurs saturées



## Considérations techniques sur l'encodage des couleurs

Système 8-bits



$2^8 = 256$  possibilités

```
n<- 8
seed<-vector("list", n)
for(i in 1:n){seed[[i]]<-c("A", "O")}
expand.grid(seed)
```

Désignées par les valeurs 0 à 255

Un système 8-bits appliqué à chaque couleur primaire rouge, vert, bleu, offre  $256^3 = 16\,777\,216$  couleurs possibles

$256^3 = 2^{24}$  : on parle de représentation colorimétrique 24 bit ou Truecolor

C'est celle exploitée par la fonction `rgb()` de R

## Considérations techniques sur l'encodage des couleurs

Système hexadécimal

○ 0 1 2 3 4 5 6 7 8 9 A B C D E F

Equivalent d'un bit à 16 possibilités au lieu de 2

$16^2 = 256$  possibilités donc deux unités de base hexadécimale suffisent pour chaque couleur primaire rouge, vert ou bleu (8 unités de base pour le 8-bits)

Exemple du rouge le plus noir (= noir) : 00 (valeur 0 en 8-bits)

Exemple du rouge le plus rouge (= Full red) : **FF** (valeur 255 en 8-bits)

Six unités de base suffisent pour les 16 777 216 couleurs possibles : les 2 premières pour le rouge (du noir au rouge intense), les deux suivantes pour le vert (du noir au vert intense), les deux dernières pour le bleu (du noir au bleu intense).

Exemples :

Noir	000000
Full red	FF0000
Full green	00FF00
Full blue	0000FF
Blanc	FFFFFF

Dans R, le code hexadécimal commence par un #



## Les fonctions `rgb()` et `col2rgb()`

### `rgb()`

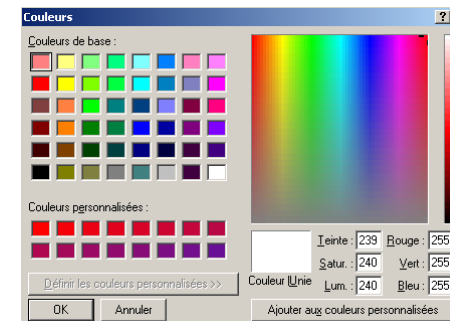
<code>red</code>	valeur de nuance de la couleur primaire rouge
<code>green</code>	valeur de nuance de la couleur primaire verte
<code>blue</code>	valeur de nuance de la couleur primaire bleue
<code>alpha</code>	valeur de transparence. Par défaut elle est égale à 1 (toujours opaque)
<code>maxColorValue</code>	valeur associée à l'intensité maximale du rouge, du vert, du bleu, ainsi qu'à l'opacité. Par défaut elle est égale à 1 (pas pratique)

La fonction `col2rgb()` transforme un code de couleur hexadécimal en système 8-bits

Exemples avec le fichier de code

Un moyen simple de connaître la correspondance entre une couleur et les valeurs RGB :

```
require(tcltk)
tclvalue(tcl("tk_chooseColor"))
```



# L'espace colorimétrique HSV (Hue Saturation Value)

TSV en français (Teinte Saturation Valeur)

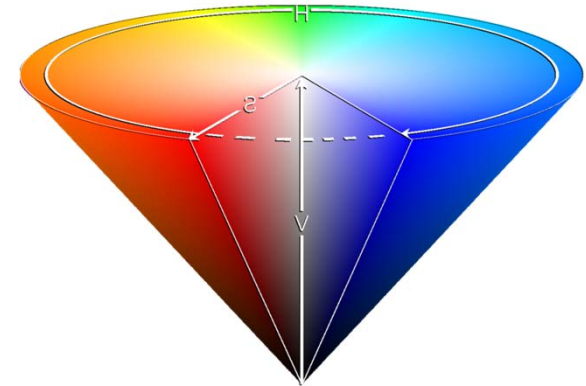
Cône tridimensionnel

Aspects remarquables :

- Les couleurs saturées sont sur le bord du grand disque
- Full red :  $0^\circ$  ou  $360^\circ$
- Full green :  $120^\circ$
- Full blue :  $240^\circ$
- Le centre du grand disque est le blanc (saturation nulle)
- La pointe du cône est le noir (brillance nulle)
- Le dégradé de gris est sur l'axe central

Souvent plus pratique d'utilisation que RGB :

- Teinte saturée sur le cercle
- Blanchir la teinte avec  $s$  (blanc  $0 \leq s \leq 1$ )
- Foncer la teinte avec  $v$  (noir  $0 \leq v \leq 1$ )



### hsv()

h	teinte (rouge $0 \leq s \leq 1$ rouge, tournant dans le sens des aiguilles d'une montre, par défaut 1)
s	s (blanc $0 \leq s \leq 1$ saturé, par défaut 1)
v	s (noir $0 \leq v \leq 1$ brillant, par défaut 1)
alpha	valeur de transparence. Par défaut 1 (toujours opaque)
gamma	contraste (clair $0 \leq \text{gamma} \leq \infty$ foncé, par défaut 1 (pas de contraste))

Exemples avec le fichier de code

## *gamma : facteur de contraste*

Correction qui fait se déplacer les couleurs soit vers le clair, soit vers le foncé.

En RGB, c'est la puissance des trois valeurs  $x$  de rouge, vert et bleu lorsqu'elles varient entre 0 et 1

Propriétés sachant que  $0 \leq \text{gamma} \leq \infty$  :

Si  $0 \leq x \leq 1$  alors  $0 \leq x^{\text{gamma}} \leq 1$

$0^{\text{gamma}} = 0$

$1^{\text{gamma}} = 1$

Lorsque  $\text{gamma} \rightarrow 0$ ,  $x^{\text{gamma}} \rightarrow 1$

Lorsque  $\text{gamma} \rightarrow \infty$ ,  $x^{\text{gamma}} \rightarrow 0$

Exemples (également avec le fichier de code) :

```
gamma<-1
rgb(((0:255)/255)^gamma, 0^gamma, 0^gamma, maxColorValue=1)
[1] "#000000" "#010000" "#020000" "#030000" "#040000" "#050000" "#060000"
[8] "#070000" "#080000" "#090000" "#0A0000" "#0B0000" "#0C0000" "#0D0000"
```

```
gamma<-2
rgb(((0:255)/255)^gamma, 0^gamma, 0^gamma, maxColorValue=1)
[1] "#000000" "#000000" "#000000" "#000000" "#000000" "#000000" "#000000"
[8] "#000000" "#000000" "#000000" "#000000" "#000000" "#010000" "#010000"
```

Avec la fonction `hsv()`, il suffit d'utiliser l'argument `gamma`

convertit des valeurs RGB en HSV

### `rgb2hsv()`

<code>red</code>	valeur de nuance de la couleur primaire rouge
<code>green</code>	valeur de nuance de la couleur primaire verte
<code>blue</code>	valeur de nuance de la couleur primaire bleue
<code>gamma</code>	facteur de contraste. Par défaut il est égal à 1
<code>maxColorValue</code>	valeur associée à l'intensité maximale du rouge, du vert et du bleu. Par défaut elle est égale à 255 (différent de <code>rgb()</code> )

Exemple :

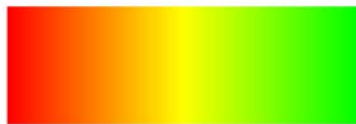
```
rgb2hsv(255,0,0)  
[ ,1 ]  
h    0  
s    1  
v    1
```

Transition des valeurs HSV vers RGB: fonction `hsv()` qui produit une valeur hexadécimale de couleur puis `col2rgb()` qui convertit en trois valeurs RGB

## Créer une palette de couleurs

Exemple d'une palette pour l'analyse de puces à ADN :

```
chip.colors <- hsv(seq(0, 1/3, length.out=256), 1, 1)
for(i in 1:256){stripchart(i, col= chip.colors[i], pch="|", cex=10, add=TRUE)}
stripchart(257, col="white", pch="|", cex=10, add=TRUE)
```

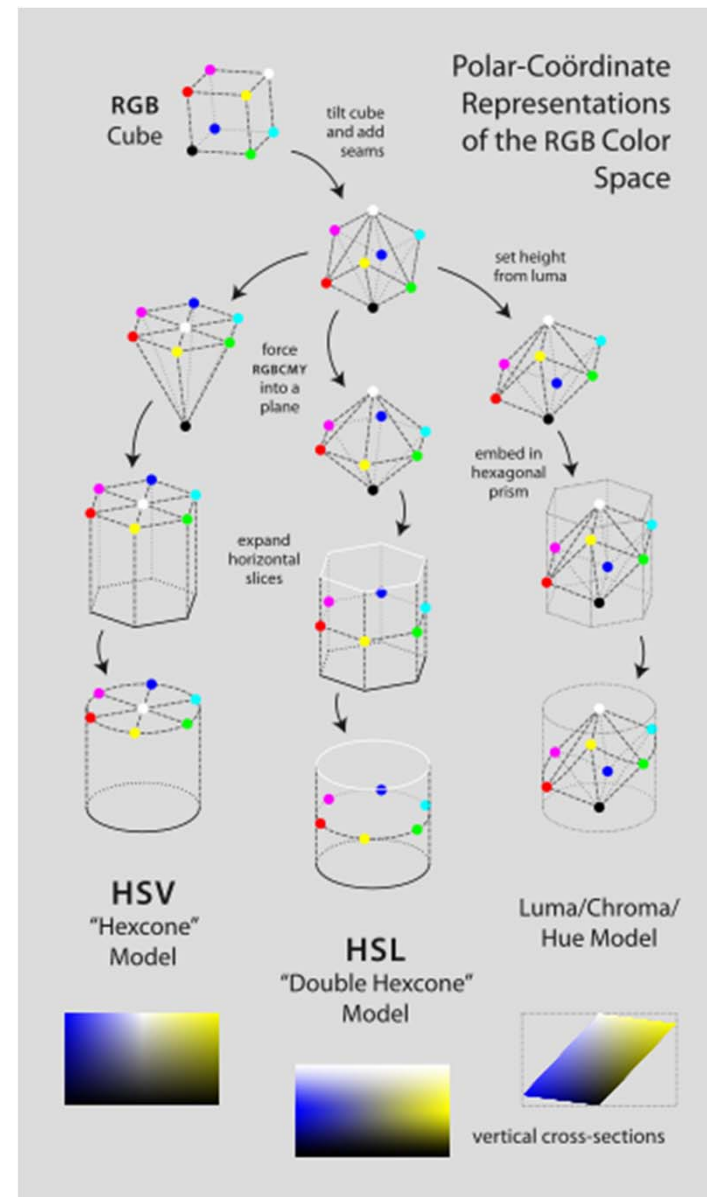


La gestion des couleurs est un vaste sujet, plutôt complexe, qui intègre des techniques de manipulation des couleurs et des modèles de perception de celles-ci.

Les systèmes colorimétriques sont nombreux :

Le RGB est dit "naturel" car il reflète la façon dont la couleur est perçue par les trois types de cône dans la rétine.

HSV et HSL sont appelés "systèmes perceptuels". Le HSL (Hue, Saturation, Luminance) est proche du HSV. Les deux dérivent d'une transformation de l'espace RGB. La fonction hcl() existe sous R (Hue, Chroma, Luminance).



D'autres systèmes plus difficiles à assimiler car plus mathématiques, ont vu le jour, comme le système tridimensionnel CIEXYZ et son dérivé, le CIExyY.

Voir l'aide de la fonction `convertColor()`

On trouve quelques informations online sur la gestion très très avancée de la couleur sous R :

<http://webcache.googleusercontent.com/search?q=cache:sJD3hHQlgXQJ:casoilresource.lawr.ucdavis.edu/drupal/node/201+%22r+package%22+ciexyy&cd=1&hl=fr&ct=clnk&gl=fr>

