Palettes de couleurs

Table of contents

 Lorsqu’ils sont générés, les graphiques appliquent un thème/style par défaut:

* Ces thèmes sont appelés scheme.
* Les options de la commande graphique visent à modifier les paramètres du thème.
* Les paramètres du style/thèmes sont appliqués aux couleurs, tailles/épaisseurs, positions des éléments de texte, définitions des axes….
* Stata dispose de 5 thèmes officiels. Des thèmes externes ont été programmés et peuvent être installés. La série de commandes associées à **grstyle** (*Ben Jann*) permet de paramétrer très facilement un thème, a minima en 2-3 lignes avec un nombre très réduit d’arguments.
* Collection **schemepack** : récemment (2021), *Asjad Naqvi* a programmé une suite de thèmes à partir du paquet grstyle. Cette série offre une alternative très séduisante au thèmes usines vieillissants de Stata.
* Les thèmes utilisent des palettes de couleurs. Il existe 3 types de palettes selon la nature des données.
	+ ***Palettes qualitatives*** (variables discrètes).
	+ ***Palettes séquentielles*** (variables ordonnées/quantitatives).
	+ ***Palettes divergentes*** (variables ordonnées/quantitatives avec deux sens implicites et une valeur pivot
* Des commandes externes permettent d’augmenter le nombre de palettes mises à disposition par Stata, de les modifier ou de les créer.
* Pour la cartographie, la commande spmap (*Maurizio Pisatti*) intègre une quarantaine de palettes, principalement séquentielles (commande à visée cartographique).
* La librairie brewscheme (William Buchanan) dispose de plusieurs commandes pour générer des palettes de couleurs. Son utilisation est assez complexe.
* La commande \*\*colorpalette (Benn Jann) charge plusieurs dizaines de palettes de couleurs, de tout type, permet de les modifier, d’en créer, et de les utiliser facilement dans la création de graphiques. Elle est indispensable dès lors qu’on souhaite utiliser efficacement des couleurs différentes de celles fournies par le logiciel. Cette commande, à mon sens incontournable, sera longuement traitée dans ce chapitre, ainsi que la commande grstyle qui reprend sa syntaxe pour générer de manière rapide un thème.

# 1. **Les palettes usines**

Stata dispose de 5 palettes de couleurs usines qui sont associées à des thèmes graphiques. Par exemple la *palette s2* est associée au *thème s2color*.

Les palettes associées aux thèmes Stata sont de type qualitative. L’ordre d’utilisation des couleurs dans un graphique composé de plusieurs objets est pensé de tel sorte que des couleurs même type, par exemple le bleu, ne se succède pas.

|  |
| --- |
| Palette Stata |

Les numéros sur l’abscisse correspondent à l’ordre par défaut d’utilisation des couleurs.Les couleurs usines ont un nom, auquel correspond un code RGB.

|  |
| --- |
| Palette s2 |

La palette **mono** d’avère très utile pour modifier les couleurs des titres, labels, grids ou du background. Elle est ambivalente, l’ordre par défaut de la palette est plutôt qualitatif, mais le nom des couleurs correspond à une palette séquentielle appelé « grey scale » (gs# avec # allant de 0 – noir – à 16 – blanc).

|  |
| --- |
| Palette mono |

|  |
| --- |
| Palette mono rgb |

*Remarque*: il est facile de générer une couleur appartenant à une palette correspondant à une échelle de gris: les trois valeurs du code RGB étant identique (compris entre 0 et 255).

 On est vite limité avec les palettes usines de Stata. Malgré les efforts de plusieurs utilisateurs pour augmenter leur nombre (M.Pissatti, F.Briatte, D.Bischoff…), la commande **colorpalette** (Ben Jann) a récemment permis de démultiplier quasiment à l’infini les possibilités en terme de manipulation des couleurs.

# 2. **Colorpalette (Ben Jann)**

***Installation***

ssc install palettes [, replace force]
ssc install colrspace [, replace force]

***Syntaxe générique***

 colorpalette nom\_palette/collection [, nom\_sous\_palette] [options]

*Remarque*: le nom d’une palette peut être une couleur ou une série de couleur

\* Charge et affiche la palette s1 de Stata
colorpalette s1

\* Charge et affiche la palette hue
colorpalette hue

\* Charge et affiche la palette viridis
colorpalette viridis

\* Charge et affiche la palette summer de la collection matplotlib
colorpalette matplotlib, summer

\* Charge et affiche une palette composée des couleur bleu et rouge
colorpalette blue red

Pour visualiser les palettes disponibles : **help colorpalette**.

**Exemple : la palette plasma**

* Appartient à la ***collection viridis****, palettes Python visant à remplacer les palettes de la* collection matplotlib\* (intégrée à colorpalette)
* Longueur par défaut : 15 couleurs
* Type : séquentielle

colorpalette plasma

|  |
| --- |
| Palette plasma |

## 2.1 **Principales options et altération d’une palette**

**Modifier la taille de la palette: n(#)**

Important car la réduction de la taille pour certaines palettes ne consiste pas à prendre les # premiers éléments de la palette par défaut (cf palettes *hu*e ou *viridis*), mais forcer une palette non qualitative, donc plutôt séquentielle, à produire une palette réduite avec des types de couleurs contrastées plutôt qualitative.

**Modifier l’intensité ou l’opacité : intensity(#), opacity(#) et ipolate(#)**

* On peut modifier l’intensité (saturation) ou l’opacité d’une palette avec **intensity(#) ou** intensity(numlist)\*\* et **opacity(#)**.
* On peut créer une palette séquentielle à partir d’une couleur de départ avec **intensity(#1(delta)#2)**.
* On peut créer une palette divergente avec des couleurs de départ, d’arrivée et de transition avec **ipolate(#)**.

**Select – reverse - nograph**

* On peut sélectionner des couleurs d’une palette avec **select(numlist)**. L’option autorise la répétition de couleurs, utile lorsque plusieurs objets graphiques partage une même couleur au sein d’un graphique (exemple nuages de point et courbes pour différentes valeurs d’une variable additionnelle : voir exemple avec grstyle).
* On peut inverser l’ordre des couleurs de la palette: **reverse**.
* Ne pas afficher la palette: **nograph** [voir *utilisation de colorpalette pour générer un graphique*]. Après avoir séléctionné les couleurs, à utiliser systématiquement pour charger la palette avant son utilisation dans un graphique. Cela réduit considérablement le temps d’exécution.

***Création d’une palette de 4 couleurs***

colorpalette plasma, n(4)

|  |
| --- |
| Palette plasma 5 couleurs |

***Répétition d’une liste de couleur avec select(numlist)***

colorpalette plasma, select(1 8 15 1 8 15)

|  |
| --- |
| Palette plasma 2 cycles |

***Création d’une palette séquentielle avec augmentation de l’intensité***

A partir de la couleur 3 (RGB= 237,121,83), création d’une palette sequentielle d’une longueur de 15, allant de “237,121,83*0” (proche blanc) à ”237,121,83*15”.

colorpalette "237 121 83", intensity(0(.1)1.5)

|  |
| --- |
| Palette séquentielle (a) |

***Création d’une palette séquentielle avec augmentation de l’intensité***

Création d’une palette séquentielle d’une longueur 15 allant de la couleur 1 à la couleur 3 de la palette plasma à 4 couleurs.

colorpalette "13 8 135" "237 121 83", ipolate(15)

|  |
| --- |
| Palette séquentielle (b) |

Si on ajoute la couleur 2 de la palette d’origine comme couleur de transition

colorpalette "13 8 135" "156 23 158" "237 121 83", ipolate(15)

|  |
| --- |
| Palette séquentielle (c) |

Si on utilise le blanc comme couleur de transition, on obtient une palette divergente.

colorpalette "13 8 135" white "237 121 83", ipolate(15)

|  |
| --- |
| Palette divergente |

**Créer, enregistrer et charger sa propre palette**

colorpalette permet de conserver en mémoire sa propre palette, en la générant dans un .ado (voir la fin du chapitre dédié aux macros) ou dans le programme principal. Par exemple avec la palette issue du premier exemple (palette plasma à 4 couleur).

\* Création de la palette mypal
capt program drop colorpalette\_mypal
program colorpalette\_mypal
c\_local P 13 8 135, 156 23 158, 237 121 83, 240 249 33
c\_local class qualitative
end

\* On charge la palette avec la commande colorpalette\_mypal
colorpalette\_mypal

\* On affiche la palette my\_pal
colorpalette my\_pal

### 2.1.1 **Utilisation de colorpalette pour générer un graphique**

On remarque, avec return list, que la commande génère une liste de macros qui enregistre les codes RGB des couleurs de la palette.

* la macro r(p) permet d’utiliser les couleurs de la palette dans un graphique à un seul bloc d’éléments.
* les macros r(p#) permettent d’utiliser les couleurs dans un graphique composé de plusieurs éléments.

La commande colorpalette n’est qu’un générateur de couleurs qui renvoie une liste de codes RGB sous forme de macros. Si pour des graphiques simples la technique du « copié/collé » de codes RGB reste possible, l’utilisation des macros va être une nouvelle fois fortement recommandée. On peut déjà néanmoins indiquer qu’avec le générateur de thème \*\*grstyle (voir la section suivante), également programmé par *B.Jann*, une sélection de couleurs pourra être directement intégrée à un graphique sans manipulation supplémentaire.

**Macros générées par colorpalette**

On va partir de la palette plasma précédente avec 4 couleurs et un % d’opacité de 80%. Avec return list, on affiche sous forme de macros la liste des codes RGB des couleurs

colorpalette plasma, n(4) opacity(80) nograph
return list

/\*
scalars:
 r(n) = 4

macros:
 r(ptype) : "color"
 r(pname) : "plasma"
 r(pnote) : "plasma colormap from matplotlib.org"
 r(psource) : "https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/\_cm\_listed.py"
 r(pclass) : "sequential"
 r(p) : ""13 8 135%80" "156 23 158%80" "237 121 83%80" "240 249 33%80""
 r(p4) : "240 249 33%80"
 r(p3) : "237 121 83%80"
 r(p2) : "156 23 158%80"
 r(p1) : "13 8 135%80"
\*/

La liste renvoie la macro r(p) qui liste l’ensemble des codes couleurs. Cette macro est utilisée pour les graphiques avec un seul objet qui liste pour l’axe des ordonnées une série de variables.

***Exemple***

sysuse uslifeexp
colorpalette plasma, n(4) opacity(80) nograph
tw line le\_wmale le\_wfemale le\_bmale le\_bfemale year, lc(`r(p)')

La macro r(p) a déjà deux doubles quotes, on ne doit pas enfermer cette liste par ““.

La liste de macros r(p#) renvoie le code couleur de chaque élément de la palette, ici r(p1) à r(p4). Ces macros sont utilisées pour les graphiques composés de plusieurs objets. Entrée directement dans la syntaxe d’un graphique, ces macros devront être enfermées dans des doubles quotes.

***Exemple***
On va afficher un graphique avec 4 densités sous forme d’aires générées aléatoirement. De nouveau on utilisera la palette plasma réduite à 4 couleurs et 80% d’opacité.

*Données*

clear
set obs 1000

gen y1= rnormal(0,1.5)
gen y2= rnormal(3, 2)
gen y3= rnormal(6, 3)
gen y4= rnormal(9, 4)

forv i=1/4 {
kdensity y`i', n(500) gen(x`i' d`i') nograph
}

*Graphique* (sans macros empilées pour faciliter la lisibilité du programme)

Colorpalette plasma , n(4) opacity(80) nograph

local ops lc(black) lw(\*.5)

#delimit ;
tw line d1 x1, recast(area) `ops' fc("`r(p1)'")
|| line d2 x2, recast(area) `ops' fc("`r(p2)'")
|| line d3 x3, recast(area) `ops' fc("`r(p3)'")
|| line d4 x4, recast(area) `ops' fc("`r(p4)'")

||, legend(order(1 "r(p1)" 2 "r(p2)" 3 "r(p3)" 4 "r(p4)") pos(11) row(1) region(color(%0)))
;

|  |
| --- |
| Test palette plasma |

Sans modifier la syntaxe du graphique, on peut alors simplement changer de palette, en modifiant son nom et/ou en modifiant une ou plusieurs options comme l’opacité.

***Exemple***: *palette winter* de la collection *matplotlib*, avec 50% d’opacité et en inversant l’ordre des couleurs.

colorpalette winter , matplotlib n(4) opacity(80) nograph

local ops lc(black) lw(\*.5)

#delimit ;
tw line d1 x1, recast(area) `ops' fc("`r(p1)'")
|| line d2 x2, recast(area) `ops' fc("`r(p2)'")
|| line d3 x3, recast(area) `ops' fc("`r(p3)'")
|| line d4 x4, recast(area) `ops' fc("`r(p4)'")

||, legend(order(1 "`r(p1)'" 2 "`r(p2)'" 3 "`r(p3)'" 4 "`r(p4)'")
 pos(11) row(1) region(color(%0)))
;

*Remarque : en utilisant les macros dans la légende, on a affiché les codes couleurs*

|  |
| --- |
| Palette winter (matplotli) |

# 3. **Exemples de palettes**

Sur le même principe que les deux graphiques précédents mais avec 6 éléments, on a mis à disposition une petite commande, **testpal**, qui permet de tester le rendu d’un graphique. La commande reporte 6 objets graphiques sous la forme de densités représentées par des aires et par des groupes. Les deux graphiques de la première colonne sont sur un fond blanc, ceux de la deuxième colonne sont sur un fond paramétrable (palette gs) par défaut très sombre.

***Installation***

 net install testpal, from("https://raw.githubusercontent.com/mthevenin/stata\_fr/master/ado/testpal/") replace

***Syntaxe***

testpal nom\_palette , [rev] [op(#)] [bf(#)]

* rev: inverse l’ordre des couleurs de la palette.
* op(#): modifie le pourcentage d’opacité des couleurs. Par défaut 100% (op(100)). # est compris entre 0+ et 100.
* bf(#): permet de modifier la clarté des graphiques de la seconde colonne (blanc pour la première). # est une valeur comprise entre 1 (noir) et 14 (presque blanc).

***Exemple avec la palette qualitative burd (François Briatte)***

testpal burd, rev op(100) bf(5)

|  |
| --- |
| testpal - Tableau (a) |

Pour les quelques palettes populaires qui vont être rapidement présentées, les options de la commande testpal n’ont pas été modifiées. Le nom de la palette est donné par le titre du graphique.

## 3.1 **Palettes qualitatives**

**Tableau**

**https://www.tableau.com/about/blog/2016/7/colors-upgrade-tableau-10-56782**

|  |
| --- |
| testpal - tableau (b) |

**HUE**

* Il s’agit de la palette par défaut de ggplot2 (R)
* De plus en plus remplacée par la palette Viridis (voir plus loin)

|  |
| --- |
| testpal - hue |

## 3.2 **Palettes séquentielles**

* Utilisées massivement en cartographie et plus généralement pour représenter des fréquences (graphique type barre) ou des valeurs ordonnées
* Couleurs allant du plus clair au plus foncé (ou inversement). – Modification de l’intensité d’une couleur. – Une ou plusieurs gammes de couleurs : par exemples du jaune au rouge, du jaune au bleu…

En réduisant la taille de certaines palettes séquentielles ou divergentes, on peut obtenir une palette plutôt qualitative. C’est le cas de la collection, très en vogue, *Viridis*.

**Collection viridis**

* La palette **Viridis** et les ses palettes associées (plasma, magma, cividis…) est la palette “star” du moment.
* Développée pour Python pour donner une palette alternative à *Matplotlib*, elle est devenue la palette par défaut de la librairie graphique
* Avantages:
	+ Même rendu, ou très proche, des couleurs sur toutes les parties d’un écran (uniformité).
	+ Différences de couleurs maintenue à l’impression n&b.
	+ Gère la plupart des formes de daltonisme.
* Limites : pour les courbes, un fond blanc ou très clair ou un fond noir ou très foncé, les couleurs aux extrémités passent difficilement. Prévoir un fond gris moyen ou ne pas sélectionner les couleurs aux extrémités de la palette.
* Les autres palettes : plasma est également très utilisée, en particulier pour le remplissage de surface (aire).
* Plus d’infos : **https://rtask.thinkr.fr/fr/ggplot2-welcome-viridis/**

|  |
| --- |
| testpal - viridis |

|  |
| --- |
| testpal - plasma |

**Collection colorbrewer**

Très populaire en cartographie: **https://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3**

|  |
| --- |
| testpal YlOrBr |

On présentera enfin une seule palette de type divergente, issue de cette collection, la palette *spectral*.

