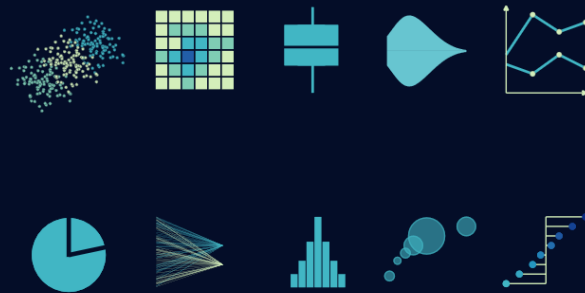


# Graphiques avec Stata

Marc Thevenin - Ined-Sms



## Table des matières

---

Introduction .....	5
Générer et éditer un graphique .....	6
Eléments de syntaxe.....	6
Types de graphique .....	6
Choix du délimiteur .....	6
Syntaxe .....	7
Options .....	14
Couleurs et épaisseurs.....	14
Options pour les axes .....	22
Options pour la légende .....	25
Options pour les titres, notes, texte libre .....	26
Autres .....	27
Combiner des graphiques .....	28
Facettes automatiques .....	28
Combinaison libre .....	28
Utilisation des macros .....	29
Rappels sur les macros .....	29
local - global .....	29
Objets return et ereturn .....	29
Autres : token,tempvar.....	29
Alléger la syntaxe .....	29
Automatiser la syntaxe.....	29
Macro empilée .....	29
Routine.....	29
Palettes de couleurs et styles.....	30
Palettes de couleur .....	30
Rappel sur les palettes de couleur.....	30
Les palettes Stata .....	30
Colorpalette de Ben Jann .....	30
Styles .....	30
Les styles Stata .....	30
Grstyle de Ben Jann.....	30
Visualisation des données .....	31
Histogramme et densité.....	31

Histogramme .....	31
Densité .....	31
Boxplot, violin, courbes de crête .....	31
Boxplot .....	31
Bean et violin .....	31
Densités de ridge .....	31
Application : probabilités assignées .....	31
Nuages, densités 2d et bubble plot .....	31
Nuages et overplotting .....	31
Densités 2d .....	31
Bubble plot .....	31
Courbes et associés .....	31
Line, lowess et connected .....	31
Application : effet spagetti et popularité des prénoms .....	31
Barres, lollipop, haltères et coordonnées parallèles .....	31
Barres .....	31
Lollipop et haltères .....	31
Coordonnées parallèles .....	32
Application : gender wage gap .....	32
Graphiques pour variables catégorielles .....	32
Barres (catplot) .....	32
Mosaique - Merimekko (spineplot) .....	32
Pie (ou pas pie) .....	32
Graphiques pour résultats d'une régression .....	32
Forest plot .....	32
Effets marginaux .....	32
Diagnostic .....	32
Compléments .....	33
Graphiques animés et interactifs .....	33
Graphiques animés .....	33
Graphiques interactifs .....	33
Python (à partir de Stata v16) .....	33
Les principales bibliothèques utilisables avec Stata .....	33
Plotnine : « tous les chemins mènent à ggplot » .....	33
Bibliographie .....	34
Sémiologie Graphique: sélection de Bénédicte Garnier .....	34
Ouvrages Stata .....	34

Articles Stata Journal .....	34
Sites (liens) .....	34
Data visualization (liens).....	34

- Applications: Stata v16 - Python v3.8
- Conversion vidéo [version html] : ffmpeg
- Relecture, conseils : Bénédicte Garnier - Ined Sms
- Impression : Philippe Olivier - Ined PLP

## Introduction

---

# Générer et éditer un graphique

---

## Eléments de syntaxe

### Types de graphique

On peut identifier deux types de graphiques avec Stata.

Les graphiques de type one-way : coordonnées sur un seul axe avec ou non un axe discret. Ils sont préfixés ou non par **gr** (**graph**) : **graph box**, **graph bar**, **histogram** .....

Les graphiques de type two-way : coordonnées sur deux axes, optionnellement selon le type de graphique un troisième axe peut être renseigné. Ils sont préfixés par **tw** (**twoway**) : **tw scatter**, **tw line**, ...

Des types de graphiques peuvent être de type *oneway* ou *twoway*. C'est le cas des histogrammes avec une commande **histogram** et une commande **tw histogram**

Les coordonnées sont généralement renseignées par des noms de variables. Pour certaines commandes graphiques on renseigne directement les valeurs : **tw scatteri** pour un nuage ou **tw pci** pour des segments [ici 4 coordonnées : (ymin, xmin) et (ymax, xmax)].

Un graphique peut être généré par une fonction : **tw function y = f(x)**

Liste et aides des commandes graphiques officielles : **help graph**

### Choix du délimiteur

Un graphique peut devenir assez gourmand en options. Par défaut l'exécution d'une est délimitée par un retour chariot **#delimit cr**, pour exécuter une commande sur plusieurs lignes, il est d'usage d'utiliser un triple slash **///**. Pour les graphiques cette option manque de souplesse et peut générer des messages d'erreur lorsque **///** est collé par inadvertance au dernier caractère d'une ligne.

Lorsque le nombre de lignes devient assez conséquent, il est préférable d'utiliser **;** comme délimiteur. Le changement se fait par **#delimit ;** et on indique la fin de l'exécution avec **;**. On retourne au délimiteur par défaut avec **#delimit cr**

```
#delimit ;
tw scatter y x,
[option1]
[option2]
[option3]
.
.
[option n]
;

#delimit cr
* Suite du programm
```

Ce n'était pas le cas pour la première version du document, mais j'utiliserai, sauf exception, cette approche dans la suite du document

## Syntaxe

Pour une raison de vocabulaire, je vais utiliser le terme de **géométrie** issue de la syntaxe de l'incontournable librairie **ggplot2** de **R** pour identifier un type de graphique comme **scatter**, **line**....Mais pas seulement, en fin de document je traite de l'utilisation de la librairie Python **plotnine**, qui est un wrapper relativement complet de **ggplot2**, directement exécutable dans un éditeur **.do** ou **.ado**. depuis la version 16 de Stata.

En première approche, la syntaxe d'un graphique avec Stata est particulièrement simple :

```
[tw/graph] type_géométrie coordonnées [if/in] [weight], [options géométrie]
[, [options du graphique]]
```

## Un bloc d'objets graphiques (même type)

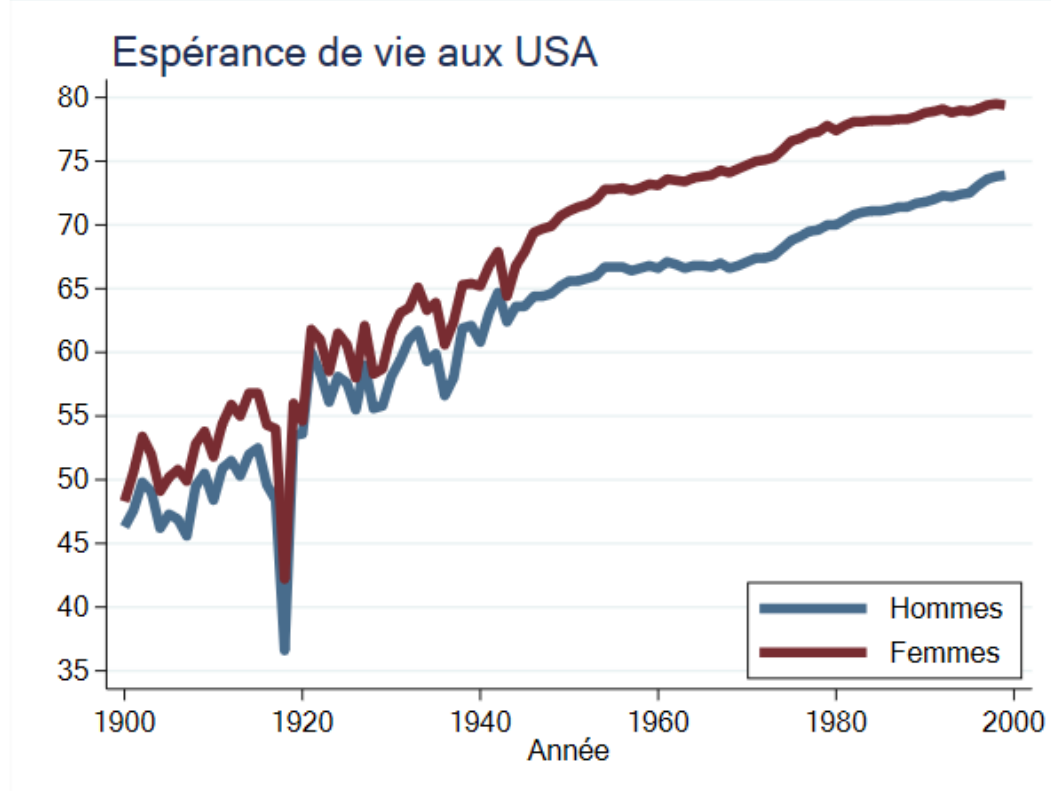
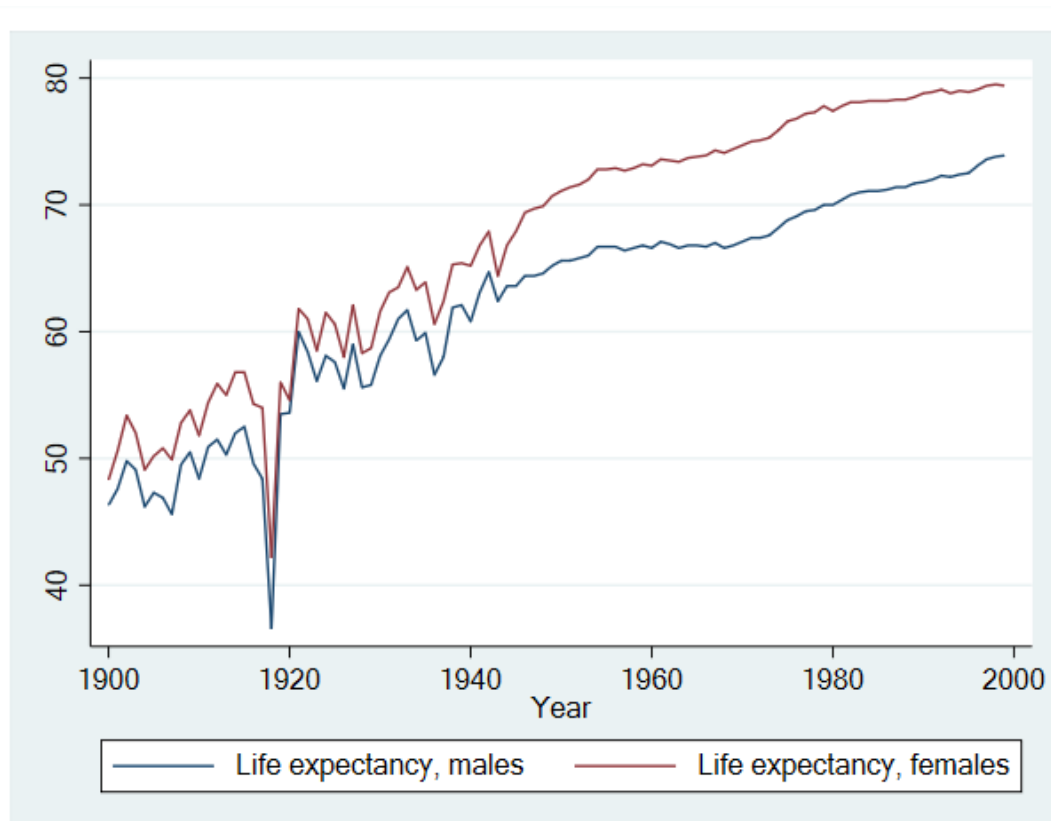
Un graphique comportant plusieurs géométries de même type peut être exécuté avec un seul bloc de coordonnées pour l'axe des ordonnées. Ces différentes coordonnées peuvent être des variables correspondant à diverses modalités d'une variable cible [voir la commande **separate** plus bas]

```
[tw/graph] type_géométrie Y1 Y2 Y3.... [X] [, option1(Y1 Y2 Y3...)..options_du
graphique]
```

Les options de la géométrie (plusieurs coordonnées pour Y) comme les couleurs ou les tailles/épaisseurs sont indiquées à la suite dans l'option choisies. Si on veut changer les couleurs de plusieurs courbes associées aux variable **y1**, **y2** et **y3** : **color(couleur\_y1 couleur\_y2 couleur\_y3)**

## Exemple

Graphique de type **line**, reportant les espérances de vie aux USA de 1900 à 2000. Le premier graphique est sans option, le second en comporte.



Premier graphique

```
tw line le year, title("Espérance de vie")
```



## Second graphique

```
#delimit ;
tw line le_male le_female year,

    lc(*.8 *.12)
lw(*4 *4)

    title("Espérance de vie aux USA", pos(11))

    legend(order(1 "Hommes" 2 "Femmes") pos(4) col(1) ring(0))
ylabel(35(5)80, angle(0))
xtitle("Année")
graphr(color(white)) plotr(color(white))
;
```

Avec le changement de délimiteur, il est plus facile de distinguer le mapping du graphique, les options qui affectent directement les courbes (couleur, épaisseur) et les options générales du graphique (titre, légende, labels de y à l'horizontal, titre de x, couleur de fond).

### Options des courbes

- **lc()**: je baisse la saturation de la courbe des hommes de 20% (plus claire) et j'augmente celle des femmes de 20% (plus foncé). Les couleurs par défaut sont conservées (palette Stata *s2color*)
- **lw()**: j'augmente fortement l'épaisseur des courbes (\*4) par rapport à celle par défaut. Stata dispose de plusieurs unités pour les tailles et épaisseurs, celle utilisée par défaut est une unité relative (voir la section dédiées plus bas).

### Options du graphique

- J'ajoute un titre avec l'option **title(...)** que je positionne à 11 heures à l'extérieur avec l'argument **pos(11)**.
- Je change les labels de la légende dans l'option **legend()** avec l'argument **order()**. Je change la position de la légende en la mettant dans la zone du graphique avec **ring(0)** et à 4 heures avec **pos(4)**. Les labels sont reportés sur une colonne avec l'argument **col(1)**.
- Je modifie les labels des ordonnées avec l'option **ylabel()**, en changeant le delta des coordonnées reportées et en mettant les labels à l'horizontal avec l'argument **angle(0)**.
- Je modifie le titre des abscisses avec l'option **xtitle()**.
- Je modifie les couleurs de fonds du graphique qui est composée de deux zones : la zone où est réellement tracé le graphique avec l'option **plotr()** [**plotregion()**] et la zone externe avec **graphr()** [**graphregion**] où se trouve reporté par défaut les titres et les légendes. Pour changer la couleur de fond, on utilise l'argument **color()**.

## Plusieurs blocs d'objets graphiques

Dans l'exemple précédent on avait deux variables, mais il est plus standard que les bases donnent des observations à partir d'une variable qui peut être croisées avec d'autres informations regroupées dans des variables discrètes. Par rapport à d'autres langages graphiques, on pense bien évidemment à R avec ggplot, les choses se gâtent un peu du côté de Stata car plusieurs graphiques doivent être exécutés séparément dans la commande graphique. On donnera plus loin un moyen via les macros et les boucles d'automatiser cette exécution. Cela ne concerne que les éléments superposés, Stata possédant une option pour générer des sous graphiques de type *facettes* (voir section sur les graphiques combinés)

Chaque sous graphique a son propre jeu d'options et on doit bien indiquer leur séparation, soit avec des parenthèses soit avec des doubles barres horizontales

***(graph 1, options graph1) (graph2, options graph2)....(graph n, option graph n) , options graphiques***

ou

***graph 1, options graph1 || graph2, options graph2 || .... || graph n, option graph n || , options graphiques***

Ma préférence va clairement à la deuxième solution pour éviter une surabondance de parenthèses déjà très présentes, et souvent imbriquées dans les options. Par ailleurs on visualise mieux la séparation entre les différents objets graphiques.

Principe de la syntaxe avec ; comme délimiteur

*Graphiques séparés par ()*

```
#delimit ;
[tw/graph]
(type_géométrie1 Y1 [X1] [Y2] [X2] [Z] [in if] [weight]
[, options(1)])

(type_géométrie2 Y2 [X1] [Y2] [X2] [Z] [in if] [weight]
[, options(2)])
...)

[, options_graphiques]
;
```

*Graphiques séparés par ||*

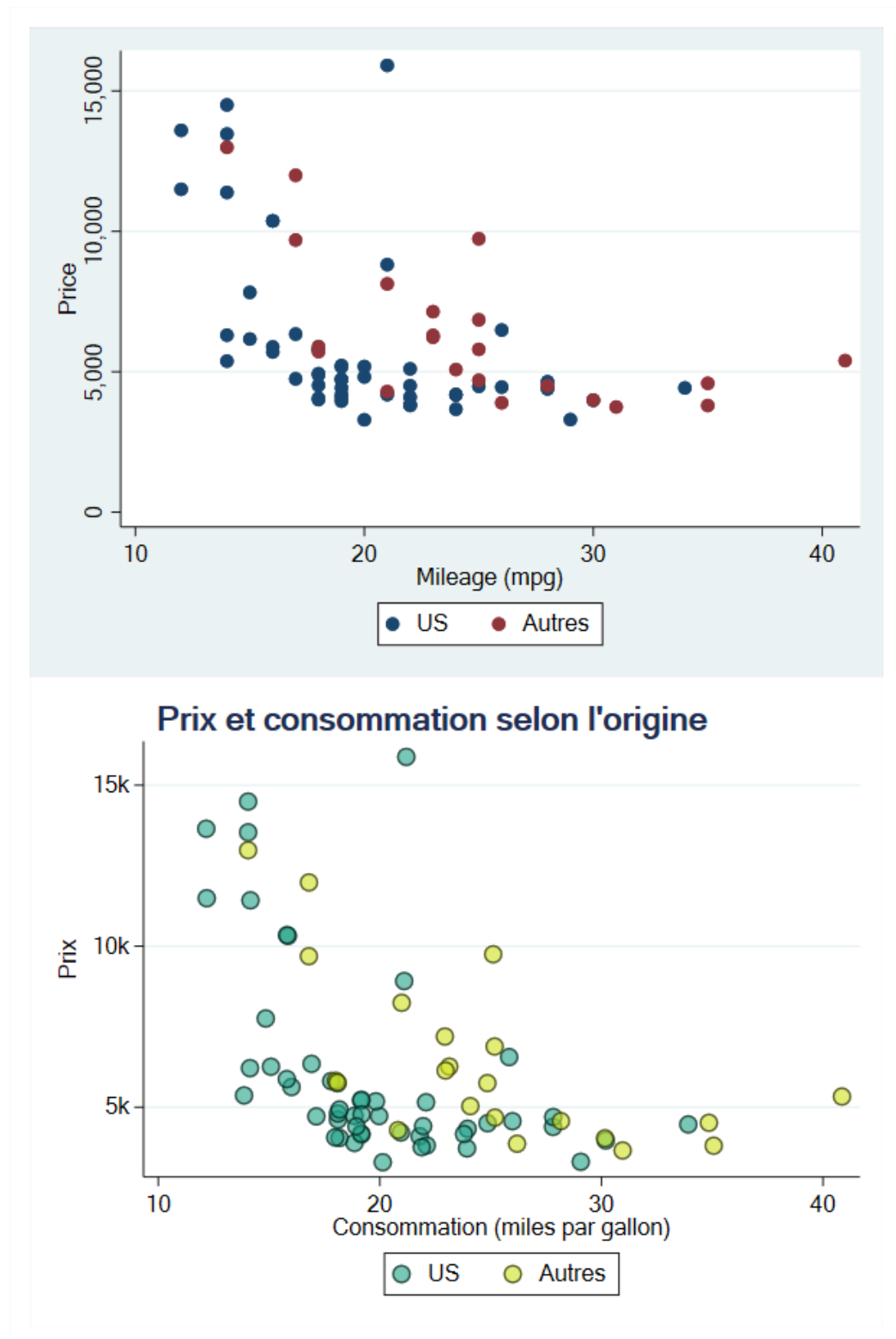
```
#delimit ;
[tw/graph]
type_géométrie1 Y1 [X1] [Y2] [X2] [Z] [in if] [weight]
[, options(1)]

|| type_géométrie2 Y2 [X1] [Y2] [X2] [Z] [in if] [weight]
[, options(2)]

...
|| [, options_graphiques]
;
```

### Exemple

Comme dans le premier graphique, celui du haut est sans option et celui du bas avec. On va afficher un nuage de point avec la base auto, entre les variables price et mpg (consommation d'essence en gallon) selon l'origine de la voiture (US ou autres pays).



## Premier graphique

```
sysuse auto, clear

#delimit
tw
    scatter price mpg if !foreign
||  scatter price mpg if  foreign

||, legend(order(1 "US" 2 "Autres"))
;
```

Lorsqu'on superpose les sous graphiques de cette manière, Stata ne reconnaît pas par défaut les labels à affecter à la légende, et affiche deux fois le label *price* pour chaque origine. On doit renseigner manuellement cette information.

Petite astuce pas forcément connue de toutes et tous. Lorsqu'une variable est type indicatrice (0,1) il n'est pas nécessaire de préciser la valeur avec un opérateur de condition.

`!foreign` est identique à `foreign==0`

`foreign` est identique à `foreign==1`

## Second graphique

```
#delimit ;
tw
    scatter price mpg if !foreign,
    mc("31 161 135%60") msize(*1.5) mlc(black) mlw(.3) jitter(1)
|| scatter price mpg if foreign,
    mc("207 225 28%60") msize(*1.5) mlc(black) mlw(.3) jitter(1)

||, title("{bf: Prix et consommation selon l'origine}", pos(11))
    legend(order(1 "US" 2 "Autres"))
    ylabel(, angle(0))
    xtitle("Consommation (miles par gallon)")
    ytitle("Prix")
    graphr(color(white)) plotr(color(white))
;
```

### Options des nuages

- `mc()` modifie la couleur des bulles. On verra par la suite plus en détail les alterations des couleurs, mais ici on a manuellement changé leur couleur avec un code RGB sur lequel on a réduit l'opacité (augmenté la transparence) à 60%.
- `mlc()` modifie la couleur du contour des bulles, ici en noir. Par défaut la couleur du contour est identique à la couleur de remplissage de la bulle.
- `msize()` modifie la taille de la bulle.
- `mlw()` modifie l'épaisseur du contour de la bulle.
- `jitter()` : très à la mode, les jitters sont des nuages dont les coordonnées ont été perturbé aléatoirement pour réduire les effets de superposition, ici liés à la variable mpg. A utiliser avec prudence, et éviter de trop modifier lorsque les coordonnées sont des valeurs continues.

### Options du graphique

- Le titre a été mis en gras avec une balise **SMCL** (Stata Markup Control Language) : `{bf :texte}`.
- Les valeurs des prix reportés sur l'axe des ordonnées à été modifié manuellement avec des unités k (1k = 10000).
- Le reste des options sont de même nature que celles relative au graphique sur les espérances de vie.

### Préférer un graphique avec un seul bloc d'objet ??

Plutôt que de multiplier le nombre de sous graphique et d'options, on peut créer une variable pour chaque modalité avec la commande `separate` et générer plus facilement un graphique.

Ici j'utilise `preserve` `restore` pour ne pas conserver les deux variables créées .

```
preserve
separate price, by(foreign)

#delimit ;
tw scatter price0 price1 mpg,
    mc("31 161 135%60" "207 225 28%60") mlc(black black)
    msize(*1.5 *1.5) mlw(.3)
    jitter(1)

title("{bf: Prix et consommation selon l'origine}", pos(11))
legend(order(1 "US" 2 "Autres"))
ylabel(5000 "5k" 10000 "10k" 15000 "15k", angle(0))
xlabel("Consommation (miles par gallon)")
ytlabel("Prix")
graphr(color(white)) plotr(color(white))
;
restore
```

## Encadré : boîte de dialogue et fenêtre d'édition

### Générer un graphique :

- Ouverture d'une boîte dans la fenêtre *command* avec la commande *db*: `db command`
- Préférer *submit* à *OK* pour laisser la fenêtre ouverte

`db tw`

`db histogram`

### Editer un graphique:

- On peut éditer manuellement le graphique après sa création dans la boîte d'édition du graphique, ou en le chargeant après sauvegarde
- On peut enregistrer les modifications faites dans l'éditeur avec *record* et les appliquer *play* pour une édition ultérieure. Un fichier au format *.grec* est enregistrée et peut être édité.
- Quelques améliorations de l'interface d'édition sont fournies avec la version 16 de 16

`sysuse auto, clear`

*\* sauvegarde en mémoire temporaire*

`tw scatter price mpg, name(g1, replace)`

`graph display g1 [, scheme() play()....]`

*\* sauvegarde en dur*

`tw scatter price mpg, save(g1, replace)`

`graph display g1 [, scheme() play()....]`

## Options

Il s'agit ici d'un tour d'horizon forcément incomplet des options qui vont permettre d'habiller les graphiques, afin d'en améliorer la visibilité. On va regarder principalement les options liées aux couleurs, tailles/épaisseurs ; et quelques paramètres de bases pour les axes, légendes, titres.

Un très bon *cheat sheet*:

[https://geocenter.github.io/StataTraining/pdf/StataCheatSheet\\_visualization15\\_Syntax\\_2016\\_June-REV.pdf](https://geocenter.github.io/StataTraining/pdf/StataCheatSheet_visualization15_Syntax_2016_June-REV.pdf)

[https://geocenter.github.io/StataTraining/pdf/StataCheatSheet\\_visualization15\\_Plots\\_2016\\_June-REV.pdf](https://geocenter.github.io/StataTraining/pdf/StataCheatSheet_visualization15_Plots_2016_June-REV.pdf)

## Couleurs et épaisseurs

### Couleurs

3 Eléments entrent dans les options relatives aux couleurs

*Nom - code couleur*

Un nom de couleur prédéfini ou un code couleur numérique: la couleur *navy* (première couleur de la palette Stata *s2color*) a pour code RGB [Red-Green-Blue] "26 71 11".

Pour un élément de type ligne, courbe... : `lc(navy) = lc("26 71 11")`

### Saturation/intensité

On peut modifier la modification. Une couleur par défaut ou renseignée manuellement a une intensité de 1. On tire vers le blanc en réduisant cette saturation, vers le noir en l'augmentant. Après le nom ou le code couleur, l'intensité est modifiée par `*#` avec # supérieur ou égal à 0 (0=blanc).

A la suite du nom de la couleur prédéfinie ou collé à la dernière valeur du code RGB, on modifie la saturation par un coefficient multiplicateur #.

Pour un élément de type barre, l'option pour modifier une couleur est donnée par `fc()` (f pour fill) : `fc(*.5)` réduit la saturation de 50% de la couleur par défaut, `fc("26 71 11*1.2")` augmente de 20% la saturation de la couleur navy, ici renseignée par son code RGB.

### Transparence

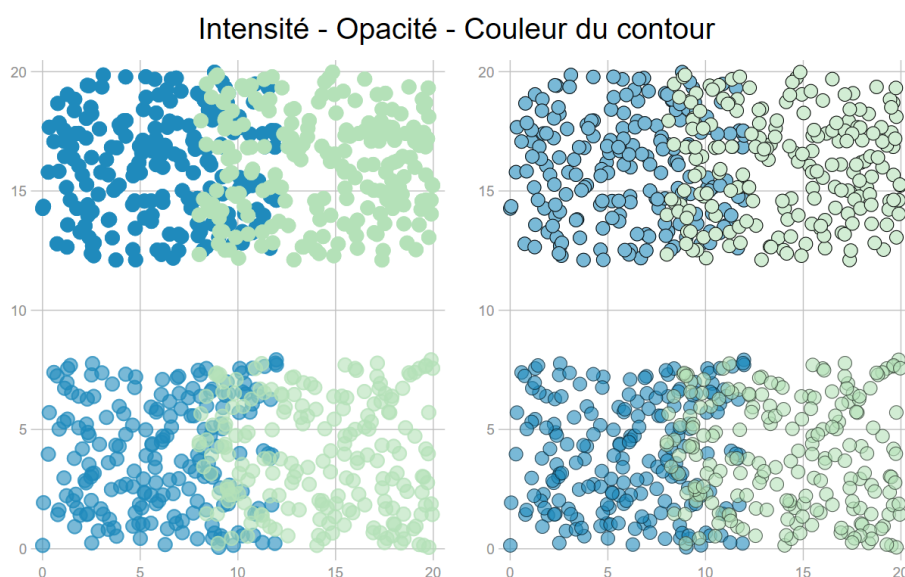
En retard sur ce point jusqu'à la version 15, on peut maintenant modifier l'opacité (transparence) des couleurs.

La transparence permet de gérer les effets de superposition, mais on doit se méfier des effets de flou qui peuvent se révéler assez désagréable, en particulier pour les nuages, en l'absence de contour. L'argument de l'option exprime un pourcentage d'opacité avec valeur minimale 0 et une valeur maximale 100 (100 = valeur par défaut). Après le nom ou le code de la couleur, l'opacité est réduite par `%#`. On peut utiliser une transparence totale (`%0`) pour cacher des éléments d'un graphique.

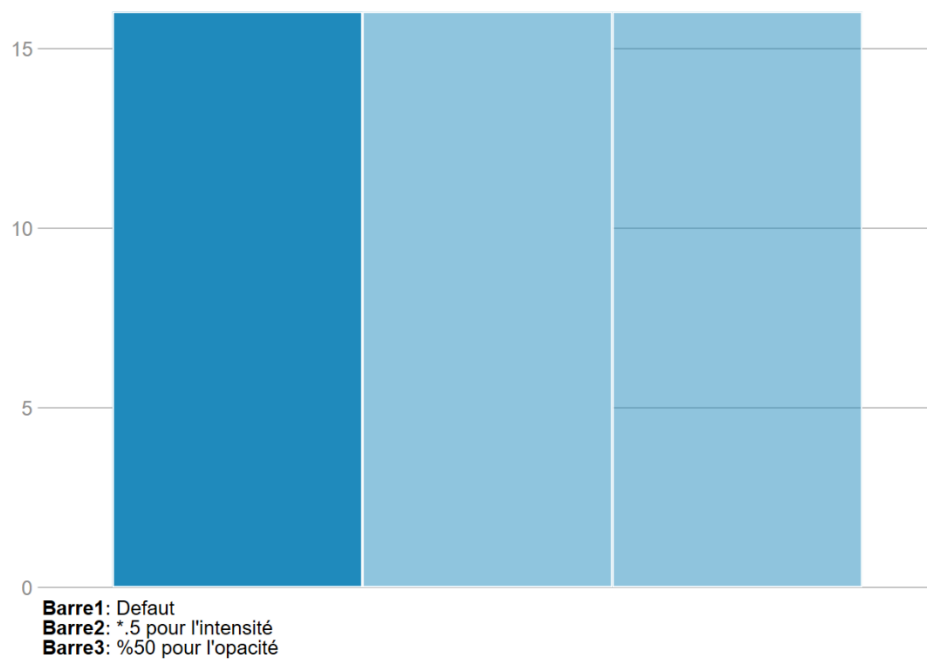
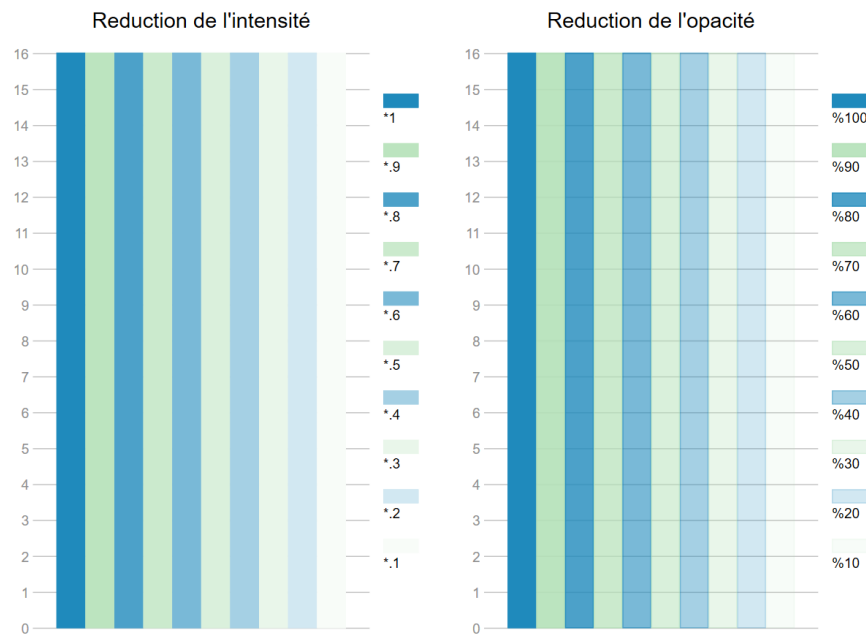
Pour un élément de type bulle : `mc(%50)` réduit de moitié l'opacité de la couleur par défaut, `mc("26 71 11%70")` applique 70% de transparence à la couleur navy.

Comme on le voit dans le second graphique ci-dessous, il n'y a pas trop de sens à baisser simultanément la saturation et l'opacité, une couleur plus transparente étant plus claire. On peut à l'inverse augmenter la saturation et baisser l'opacité.

### Exemples



**Nord-Ouest:** par défaut  
**Nord-Est:** contour noir et réduction de l'intensité \*.5  
**Sud-Ouest:** contour par défaut et réduction de l'opacité %50  
**Sud-Est:** contour noir et réduction de l'opacité %50





Le tableau qui suit donne le nom des options relatives aux couleurs, avec leur expression tronquée (color = c). Comme nombre important de types de géométrie peuvent être associés à tw line dont ils partagent les options, ils ne sont pas recensés (help twoway)

#### Graphiques twoway (non exhaustif)

scatter -scatteri	<b>mc</b>	Remplissage et contour.
	<b>mfc</b>	Remplissage.
	<b>mlc</b>	Contour.
line, pci et associés	<b>lc</b>	Couleur de la courbe.
connected		Options de scatter et line.
area - area - bar	<b>col</b>	Remplissage et contour.
	<b>fc</b>	Remplissage.
	<b>lc</b>	Contour.
	<b>fi</b>	Intensité du remplissage.
dropline		Options de scatter et line.
dot	<b>dc</b>	Remplissage et contour.
	<b>dfc</b>	Remplissage.
	<b>dlc</b>	Contour.
contour - contourline	<b>sc</b>	Couleur de départ.
	<b>ec</b>	Couleur d'arrivée.
	<b>cc</b>	Liste de couleurs pour chaque niveau.
	<b>crule</b>	Règle de transition entre couleurs.
	<b>color1</b>	Pour contourline seulement - une couleur par niveau.
lfitci	<b>clc</b>	Couleur de la droite de régression.
	<b>acol</b>	Remplissage et contour de l'intervalle.
	<b>fc</b>	Remplissage de l'intervalle
	<b>acl</b>	Contour de l'intervalle

#### Graphiques oneway

graph matrix		Options de tw scatter
graph bar		Options de tw area...
graph dot		Options de tw dot
graph box		Options de tw bar - line - scatter
graph pie	<b>color</b>	Couleur de la part

## Axes

xlabel - ylabel	<b>labc</b>	Couleur des labels
	<b>tlc</b>	Couleur des coches
	<b>glc</b>	Couleur du grid
Xscale - yscale	<b>lc</b>	Couleur de la ligne de l'axe

## Légende

	<b>color</b>	Couleur du texte
Dans la sous option		Couleur de remplissage et du contour de la
<code>region()</code>	<b>color</b>	zône
	<b>fc</b>	Remplissage de la zone
	<b>lc</b>	Contour de la zone
Exemple : <code>legend(order( 1 "A" 2 "B") title("Légende", color("red"))</code> <code>region(fc(yellow) lc(black)))</code>		

## Titres (title, xtitle, ytitle...), notes (note, caption), texte libre (text)

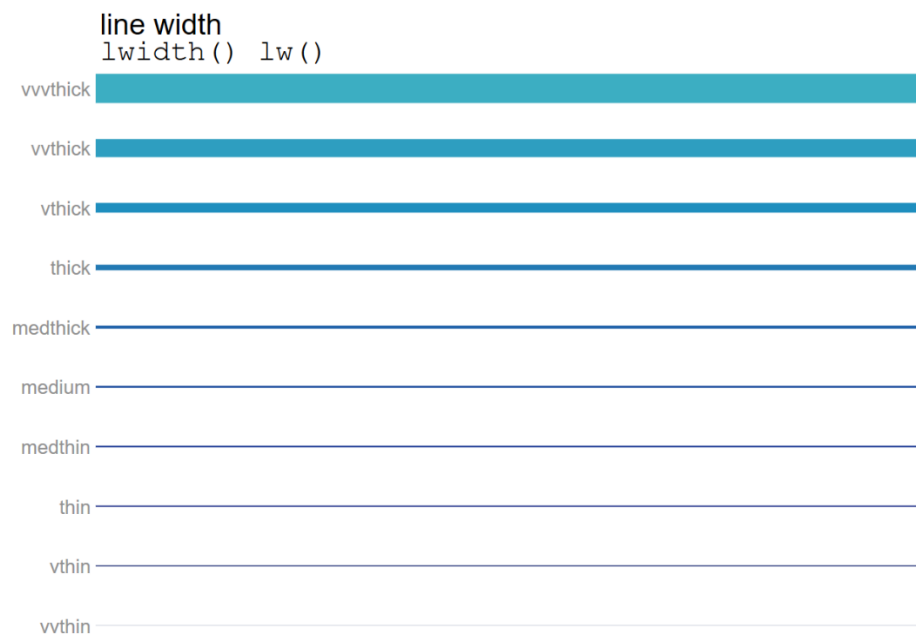
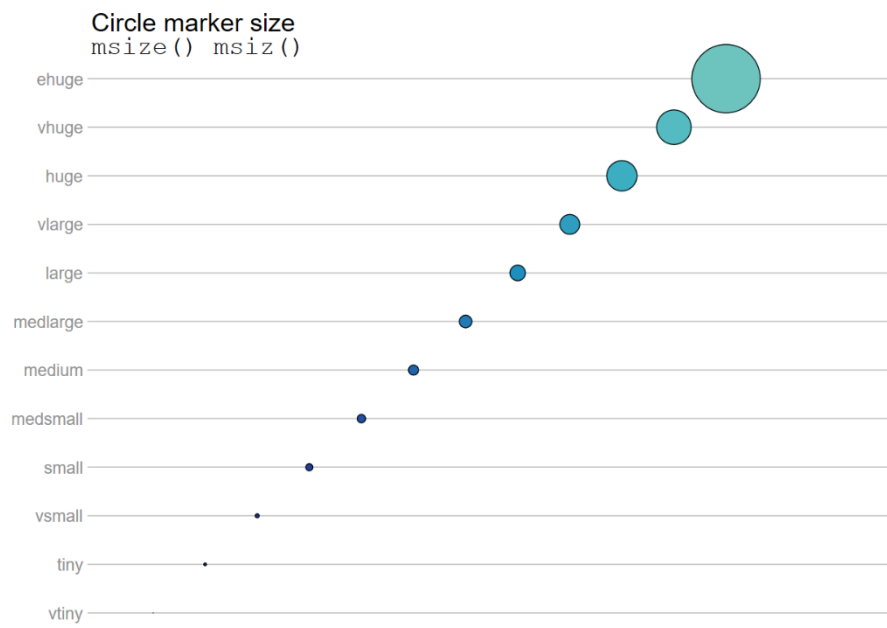
	<b>color</b>	Couleur du titre
avec l'option box	<b>bc</b>	Remplissage et contour de la zone
	<b>fc</b>	Remplissage de la zone
	<b>lc</b>	Contour de la zone
Exemple: <code>title("title", box bc(red) color(white))</code>		

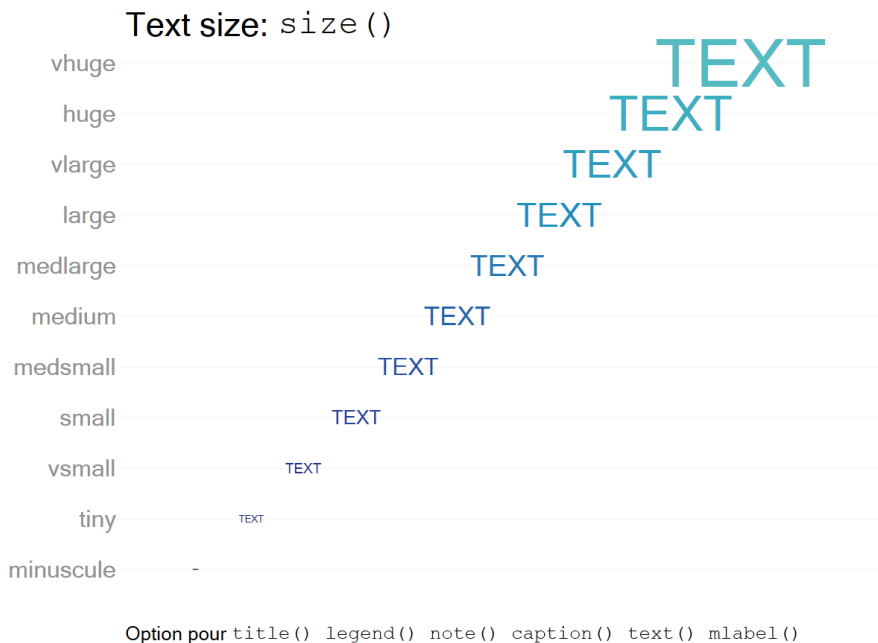
## Tailles/Épaisseurs

On peut altérer les tailles et épaisseurs d'éléments composants le graphiques en s'appuyant soit sur des arguments prédéfinis, soit en jouant sur les valeurs de plusieurs types d'unités, absolues (cm, point, pouce) ou relatives. Certaines ont été implémentées à la version 16 de Stata.

La mémorisation des épaisseurs et tailles prédéfinies pouvant s'avérer un peu laborieuse, avec un jeu d'expressions selon le type d'objet, on peut privilégier la modification directement par les valeurs.

## Tailles prédéfinies





### *Valeurs paramétrables*

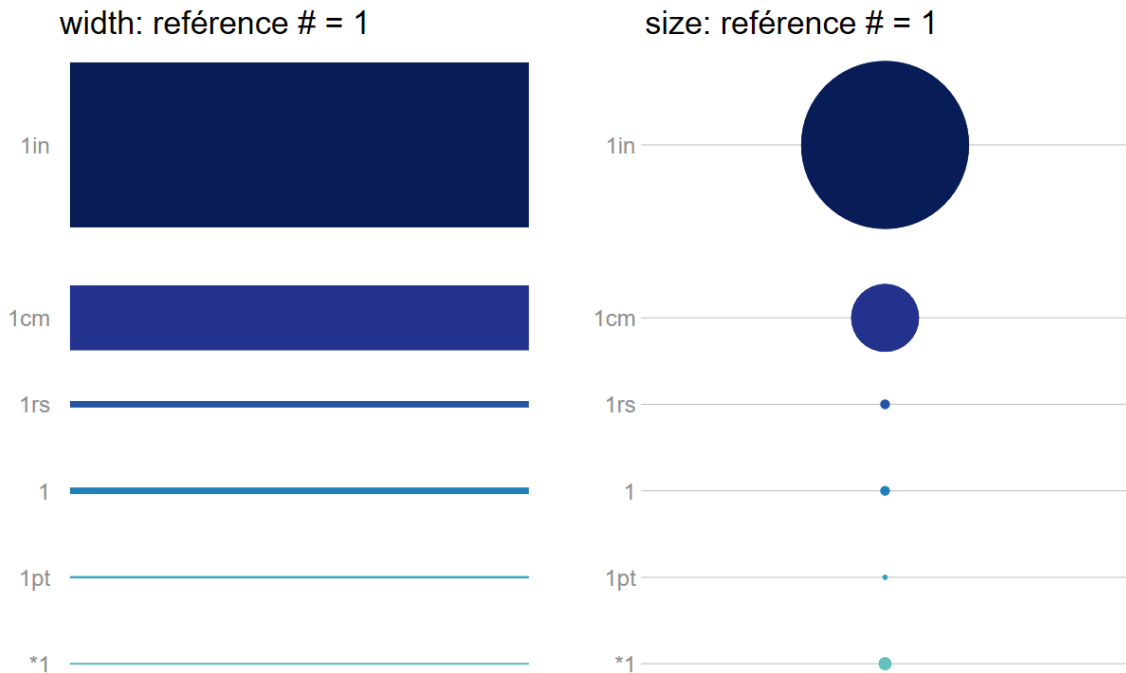
Pour modifier directement les valeurs des tailles et épaisseurs Stata propose 6 unités : 3 absolues et 3 relatives. Pour les paramètres relatifs, deux dépendent de la taille du graphique et une dépend des valeurs définies par le thème qui est appliqué au graphique.

**#** donne la valeur de la modification

Absolues (par ordre décroissant) : le pouce (**#in**) , le centimètre (**#cm**), et le point (**#pt**)

Relatives :

- **option(#rs)** : valeur relative par rapport au minimum de la longueur et de la largeur du graphique (=100). Si `option(.5rs)` et que le graphique fait 10 de largeur et 20 de longueur, la référence est la largeur avec une référence de 100. La taille l'épaisseur de l'élément fera alors .5% de cette référence. L'aide de Stata ne s'attarde pas sur ce type de modification, il est vrai un peu obscure.
- **option(#)** : également une valeur par rapport au minimum de la largeur et de la longueur, mais on garde cette valeur comme référence. Si `option(.5)` et que le minimum est toujours égal à 10 (largeur), la taille ou l'épaisseur de l'élément est égale à .5% de 10. Si le minimum était de 20 on aurait 5% de 20.
- **option(\*#)** : coefficient multiplicateur d'une taille ou épaisseur par défaut définie par la style (scheme) du graphique.



Faire le même tableau que pour les couleurs

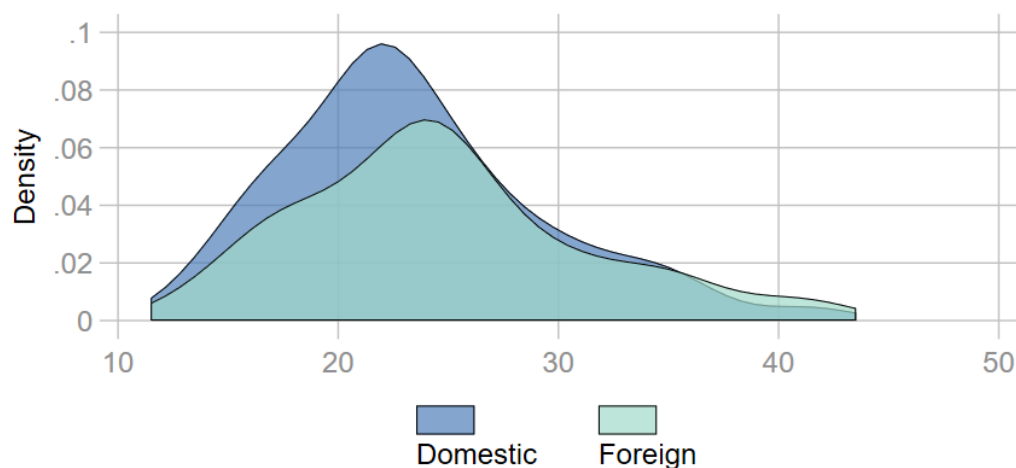
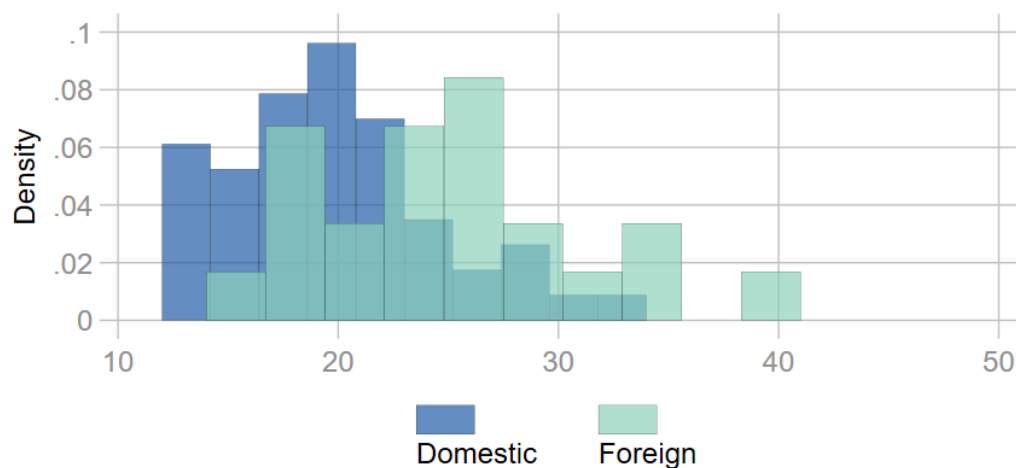
### Exercice

Reproduire les graphiques ci-dessous avec la base *auto* [`sysuse auto.dta`]

Programmes : [https://github.com/mthevenin/stata\\_fr/blob/master/exercices/exo1](https://github.com/mthevenin/stata_fr/blob/master/exercices/exo1)

### Graphique(s) I

- Reproduire l'un et/ou l'autre de ces graphiques avec les variable *mpg* et *foreign*
- Les histogrammes sont générés avec la fonction **tw histogram** (à préférer à la commande `type oneway histogram`)
- Les densités sont estimées avec la fonction **kdensity** [`help kdensity`] et le graphique est généré avec **tw area** [`help twoway area`]
- Les couleurs sont issues de la palette de type séquentielle **YlGnBu** de la collection Brewer (voir le chapitre sur les palettes de couleur), les codes des deux couleurs sont "34 94 168" et "139 209 187".



## Graphique II

Reproduire le graphique suivant:

- Nuage de points des variables **price** et **mpg** pour les deux modalités de la variable *foreign*, avec report des OLS.
- La droite de l'OLS est générée avec le type de graphique **tw lfitci**:
  - Couleur de la droite: option **clc()**
  - Couleur de remplissage l'intervalle de confiance: option **fc()**
  - Couleur du contour de l'intervalle de confiance: option **alc()**

## Options pour les axes

Il y a 3 titres options pour modifier les axes x/y x ou y) : `x/ylabel()`, `x/ytitle()`, `x/yyscale()`.

Une option permet de définir le choix de l'axe si un graphique comporte plusieurs définitions d'axes pour les abscisses et/ou pour les ordonnées : `x/yaxis(#)` avec # le numéro de l'axe, qui sera par la suite reporté dans les autres options relatives aux axes.

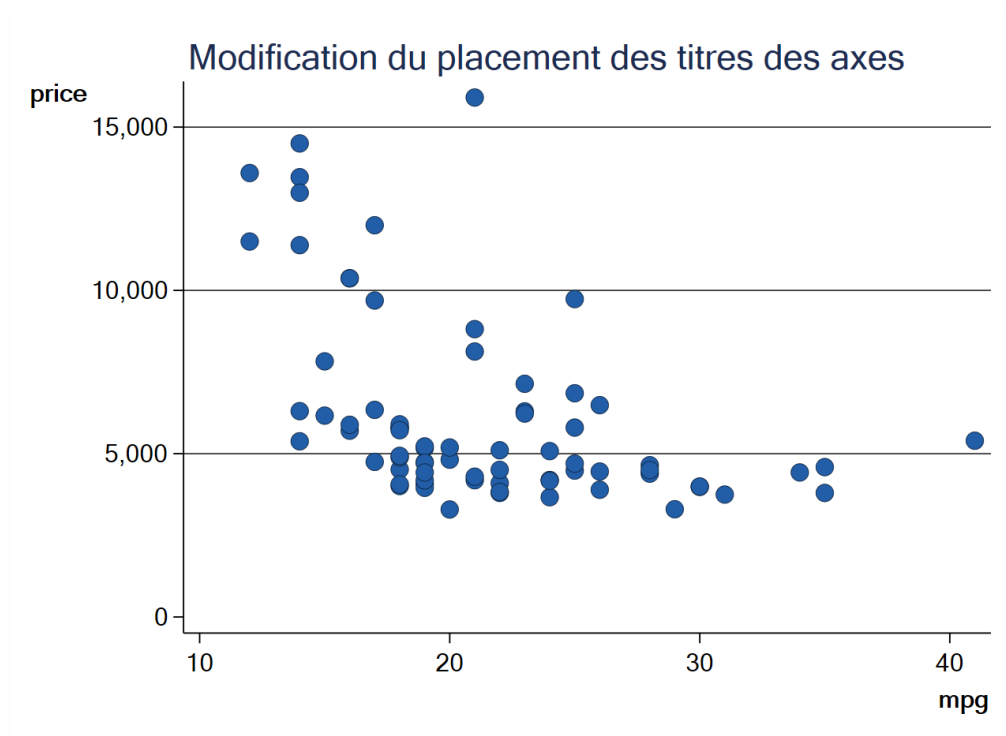
## Titres

`xtitle()` `ytitle()` `ttitle()` `ztitle()`: `xtitle(["titres"] [, options])`

Principales options:

- Taille `size()`, couleur `color()`. Si on ajoute l'option `box`, on peut modifier l'apparence de la zone qui l'entoure.
- Titres sur plusieurs lignes: `xtitle("ligne1" ligne2"....[, options])`
- On peut modifier la position des titres des axes avec `place(top/right/bottom/left)` pour leur position le long de l'axe (par exemple `right` pour `x` et `top` pour `y`) et contrôler leur éloignement avec `width(#)` pour `y` et `height(#)` pour `x`. Le graphique suivant modifie les position par défaut avec les options :

```
xtitle(", place(right) height(5))  
ytitle(", orient(horizontal) place(top) width(5))
```



## Coordonnées, labels, grid

`xlabel()` `ylabel()` `tscale()` `zscale()`: `xlabel([coordonnées] [,options])`

Modification des valeurs reportées sur les axes:

- Avec `min (delta) max`: exemple `xlabel(0(5)10)`.
- En indiquant le nombre de valeurs à reporter sur l'axe : `xlabel(#)`.
- Manuellement: `xlabel( #1 #2 "label" "label2"...)` ou `xlabel( #1 "label1" #2 "label2"...)`.  
Exemple: `xlabel(0 5 10)` ou `xlabel(0 "Zero" 5 "Cinq" 10 "Dix")`.
- On peut mixer les deux approches: `xlabel(0 .5 1(1)10 15 20)`.

Principales options:

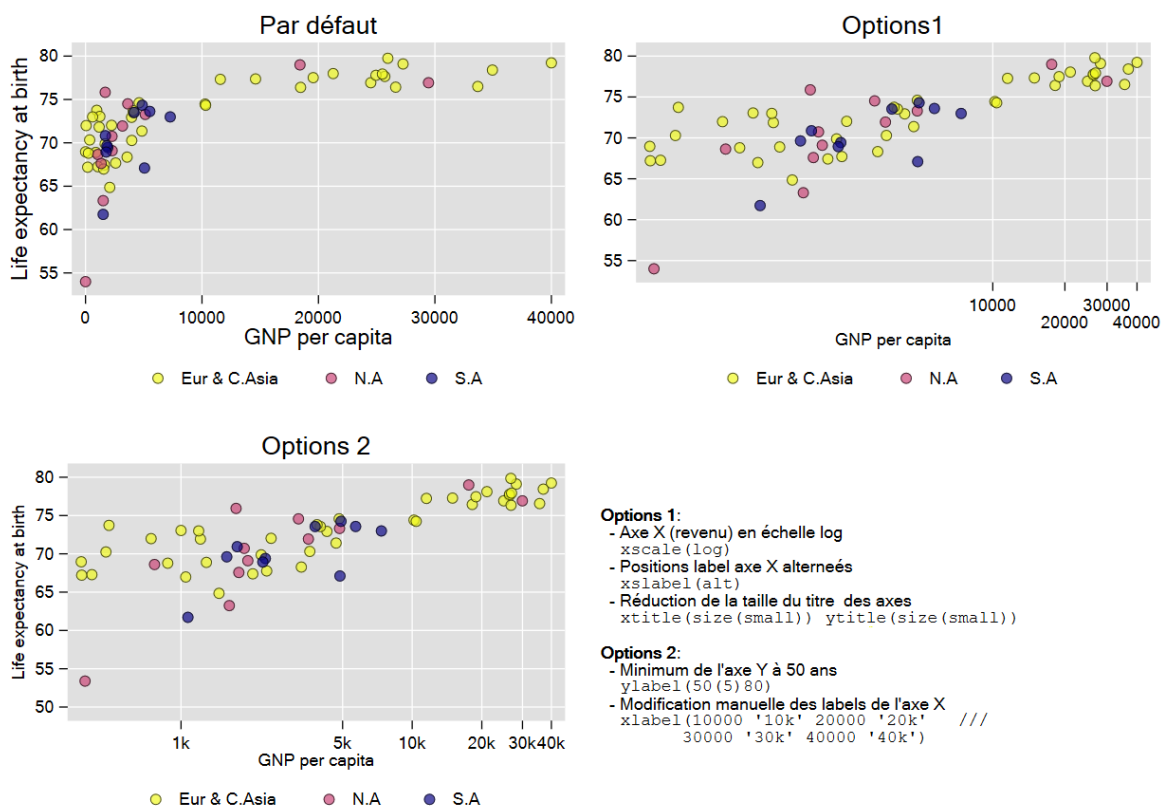
- Taille et couleur du texte avec `labs()` et `labc()`.
- Alternier les positions pour éviter les chevauchements avec `alt`.
- Modifier l'angle de report des labels: `angle(#)` avec # valeur de l'angle [0 (horizontal), 45, 90 (vertical) ...].
- Modification du grid et des coches (tick): la couleur avec `glc()` et `tlc()` et l'épaisseur avec `glw()` et `tlw()`.

## Echelles

`xscale()` `yscale()` `tscale()` `zscale()`

- Utilisation d'une échelle logarithmique: **log**. Seule l'échelle est modifiée, les valeurs d'origines sont reportées sur l'axe, ce qui est plutôt bien vu.
- Echelle en ordre décroissant: **reverse** (pas conseillé).
- Supprimer l'affichage des axes : **off**.
- Modifier les limites de l'axe : `range(min max)`.

Les graphiques qui suivent donnent quelques exemples de modification des options des axes. Les données sont issues de la base d'exemple *lifeexp* et les graphiques reportent sous forme de nuage les espérances de vie à la naissance (y) et le revenu par habitant (x).



**Warning : Attention** pour la dernière option reportée dans le graphique combiné, le label doit être indiqué par des doubles quotes "label " et non par 'label': `xlabel(1000 "1k" 5000 "5k" 10000 "10k" 20000 "20k" 30000 "30k" 40000 "40k")` et non `xlabel(1000 '1k'.....)`. J'ai du composer avec la syntaxe permise pour les éléments de type texte.

Ajouter un point sur les axes multiples avec exemple



## Options pour la légende

**legend(), clegend(), plegend: legend(contenu [options])**

Les options **clegend()** et **plegend()** sont réservées aux graphiques de type courbes de niveau avec une troisième dimension (hauteur, densité...): **clegend()** pour **tw contour** et **plengend()** pour **tw contourline**. Ces options ont des arguments spécifiques que je ne décrirais pas ici.

Stata utilise par défaut le label de la variable y pour alimenter le contenu de la légende.

- Ne pas afficher une légende: **legend(off)**
- Position par défaut: 6 heures (Sud) - à l'extérieur (*graphregion*). Modifiable avec **pos(#)** et **ring(0 ou 1)**. Pour modifier le nombre de lignes et de colonnes: **row(#)** et **col(#)**
- Pour modifier les labels de la légende:  
**order()** : `order( 1 "label1" 2 "label2" .....)`  
ou  
**lab()** : `lab(1 "label1") lab(2 "label2") ...)`  
J'utilise de préférence **order** plus parcimonieux en parenthèses. L'apparence (gras, italique...) des labels est modifiable avec des balises **smcl** ; les tailles et couleurs sont également modifiables.
- Les labels peuvent être reportés sous les symboles avec l'option **stack**.
- On peut modifier l'aspect de la zone (couleur du fond, contour...) avec la sous option **region()** ou **r()**.

### Encadré : le smcl

Le Smcl « Stata Markup Control Langage » est le langage d'édition de Stata. Pour les graphiques, il permet de modifier les textes des titres, labels des axes, contenu des légendes : gras, italique... changement de police, ajout de formules.

Balise smcl : "{type\_option: texte}"

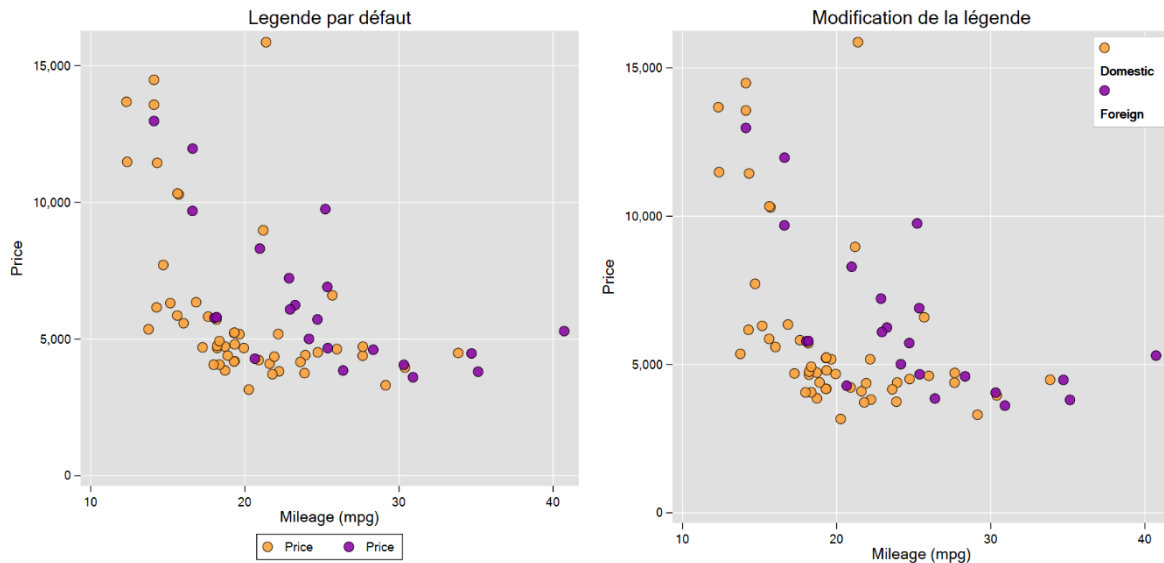
**Gras** : "{bf: texte}"

*Italique* : "{it: texte}"

**Gras et italique** : "{bf:{it :texte}}"

Le choix des polices est limité et dépend du système d'exploitation, les polices sont sélectionnées par leur type : "{stSans : texte}" pour type sans serif, "{stSerif : texte}" pour type serif, "{stMono: texte}" pour type mono, et "{stSymbol : texte}" pour les lettres grecques ou les symboles mathématiques.

Exemple police avec windows : **arial** (sans serif), **times new roman** (serif), **courrier new** (mono) et **symbol** (symbol)



Options de la légende :

```
legend(order(1 "{bf:Domestic}" 2 "{bf:Foreign}") pos(1) ring(0) col(1)
r(lw(*.1)) stack )
```

- Le texte de labels a été mis en gras avec une balise `Smcl`.
- La légende a été positionnée à l'intérieur du graphique avec `ring(0)` et à 1 heure avec `pos(1)`.
- La légende est reportée sous forme de colonne avec `col(1)`.
- Le contour de la zone a été aminci avec `r(lw(*.1))`.
- Le texte est positionné sous le symbole avec l'option `stack`.

## Options pour les titres, notes, texte libre

La grande partie des options ont déjà été précisées pour les titres des axes plus haut. Les titres et les notes sont comme les légendes positionnés par rapport à des valeurs de l'horloge (de 1 à 12), pour le texte libre on doit renseigner directement les coordonnées.

**title()** - **subtitle()**

Il y a également des options pour des titres qui sont positionnés par défaut à gauche, à droite ou en bas du graphique : `l1title()` et `l2title()` pour un titre à gauche, `r1title()` et `r2title()` à droite.

Par défaut le titre principal est positionné à 12 heures. Le sous titre est positionné en dessous du titre principal.

**note()** - **caption()**

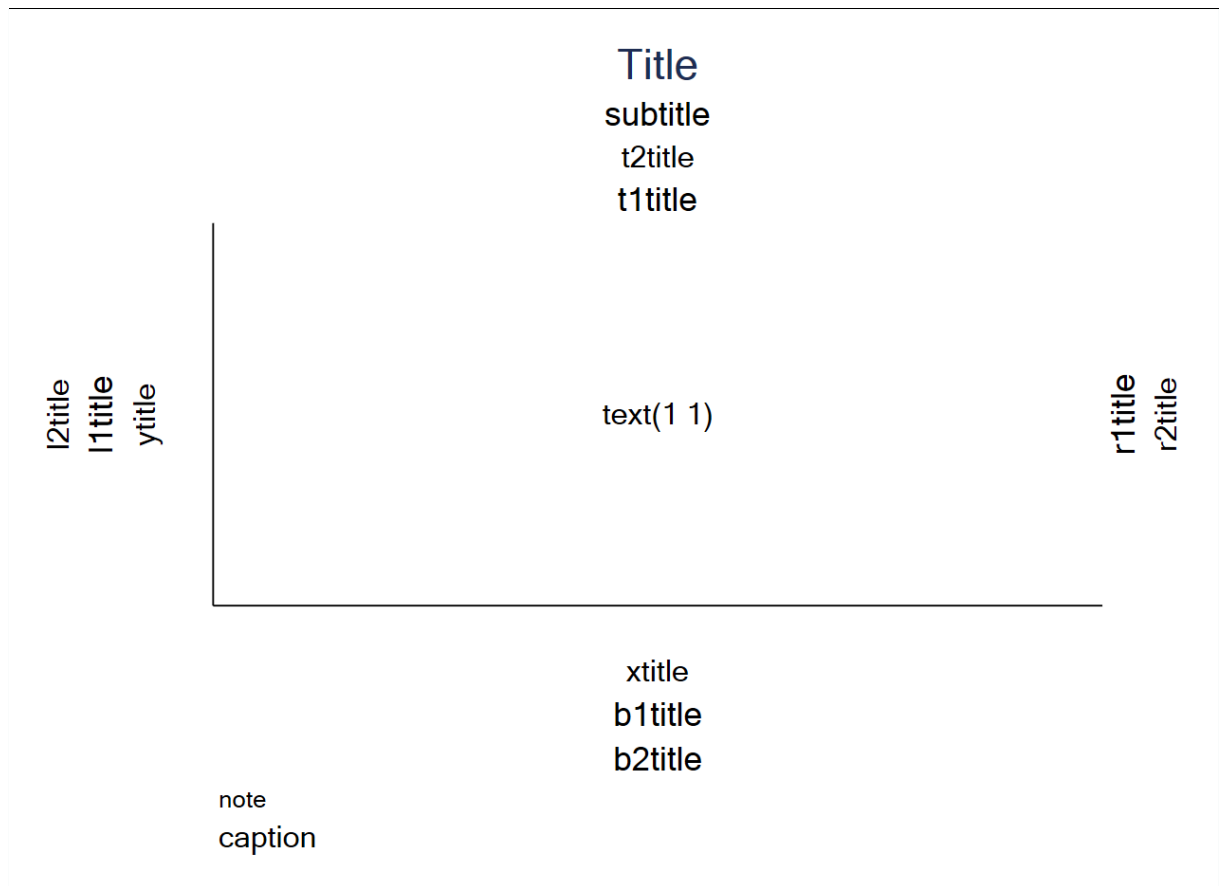
`note()` est positionné par défaut à 6 heures, `caption()` en dessous de `note()`. Ces options permettent de faire des notes de lecture ou de préciser une ou plusieurs sources au graphique.

**text(coordonnées "texte" [,options])**

Cette option permet d'afficher un texte dans la zone du graphique en entrant directement les coordonnées (y,x). Les sous options relatives à l'alignement peuvent être assez utiles si on ne

veut pas tâtonner avec les coordonnées (on peut passer par les macros pour générer les coordonnées, voir le chapitre dédié).

Le graphique suivant présente les options par défaut (position, taille, couleur) des titres, dont le titre des axes, ainsi qu'un texte libre positionné aux coordonnées (1,1).



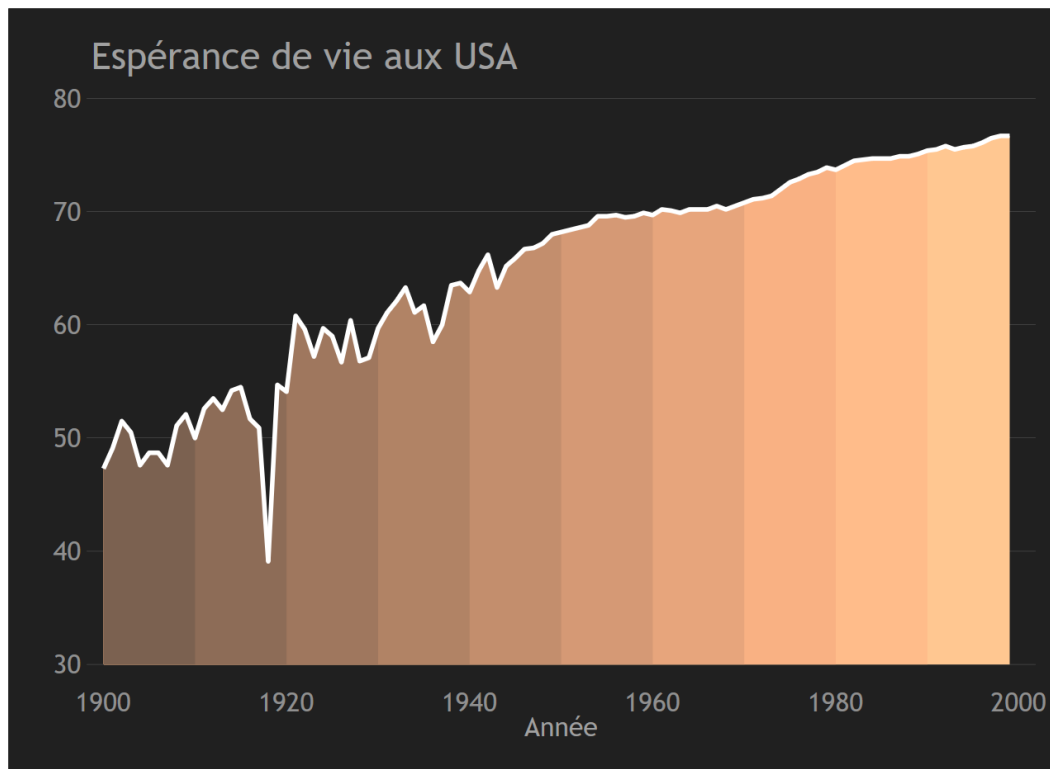
## Autres

### Changer la police

Windows: `graph set window fontface "nom_police"`

### Modifier le type de géométrie

`recast(type_geometrie)`



Taille du graphique

```
xsize() - ysize()
xfigsize() - fysize()
scale()
```

Options et commandes de Commande de gestion

```
save() - name() - export()
use - display
graph drop
```

## Combiner des graphiques

Facettes automatiques

⇒ Small multiples

Combinaison libre

## Utilisation des macros

---

### Rappels sur les macros

local - global

Objets return et ereturn

Autres : token,tempvar

### Alléger la syntaxe

### Automatiser la syntaxe

Macro empilée

Routine

## Palettes de couleurs et styles

---

### Palettes de couleur

Rappel sur les palettes de couleur

Les palettes Stata

Colorpalette de Ben Jann

### Styles

Les styles Stata

Grstyle de Ben Jann

# Visualisation des données

---

## Histogramme et densité

Histogramme

Densité

## Boxplot, violin, courbes de crête

Boxplot

Bean et violin

Densités de ridge

Application : probabilités assignées

## Nuages, densités 2d et bubble plot

Nuages et overplotting

Densités 2d

Bubble plot

## Courbes et associés

Line, lowess et connected

Application : effet spaghetthi et popularité des prénoms

## Barres, lollipop, haltères et coordonnées parallèles

Barres

Lollipop et haltères

Coordonnées parallèles

Application : gender wage gap

## Graphiques pour variables catégorielles

Barres (catplot)

Mosaïque - Merimekko (spineplot)

Pie (ou pas pie)

## Graphiques pour résultats d'une régression

Forest plot

Effets marginaux

Diagnostic



## Compléments

---

### Graphiques animés et interactifs

Graphiques animés

Graphiques interactifs

### Python (à partir de Stata v16)

Les principales librairies utilisables avec Stata

Plotnine : « tous les chemins mènent à ggplot »

## Bibliographie

---

Sémiologie Graphique: sélection de Bénédicte Garnier

Ouvrages Stata

Articles Stata Journal

Sites (liens)

Data visualization (liens)