Intro à la visualisation de données en R

Florence Vallée-Dubois (cours POL 2809)

Table des matières

Installation de la bibliothèque nécessaire

Ces notes de cours sont une introduction à la visualisation de données en `R`. Elles utilisent les banques de données mises à la disposition des étudiant.e.s du cours POL 2809 (offert à l'automne 2019). Le code contenu dans ces notes est enregistré sous le nom "intro-visualisation.R" dans le projet "Intro à la visualisation" sur R Studio Cloud.

Dans ces notes, vous apprendrez les rudiments de `ggplot()`, une fonction très flexible de visualisation de données. `ggplot()` se trouve dans la bibliothèque `ggplot2`, qui ne vient pas par défaut dans `R`. Il faut donc l'installer de la manière suivante [1] :

```
> install.packages("ggplot2")
```

La bibliothèque n'a besoin d'être installée qu'une seule et unique fois (sauf si vous ouvrez un autre projet dans `R Studio Cloud`). Il est par contre nécessaire de "lire" la bibliothèque à chaque nouvelle ouverture du projet. On lit la bibliothèque de cette façon :

```
> library(ggplot2)
```

Ouverture des banques de données

Avant d'aller plus loin, il faut importer les banques de données dans notre environnement. Pour les démonstrations qui suivront, nous utiliserons les deux banques de données quqi vous sont accessibles pour le travail final, soit CSES et Quality of Governance.

```
> cses = read.csv("CSES.csv", sep = ",")
> quality = read.csv("quality_governance.csv", sep = ",")
```

La fonction `ggplot()`

```
> ?ggplot
```

1. Plusieurs autres fonctions nécessitent d'être installées dans `R`. Pour le cours, nous avons seulement utilisé des fonctions "de base", comme `summary()`, `lm()`, `plot()`, etc. Si vous continuez d'utiliser `R` dans l'avenir, vous vous renderez rapidement compte qu'il est essentiel d'installer des bibliothèques avant de pouvoir utiliser des fonctions plus complexes. Pour plus d'informations, consultez les notes "Ateliers R", disponibles sur mon site web.

La fonction `ggplot()` crée un objet de type ggplot, dans lequel on peut spécifier 1) les données utilisées sous l'argument `data =` et 2) les variables qui prendront place sur l'axe des X et des Y sous l'argument `mapping = aes(x = ..., y = ...)`. On peut aussi y spécifier la manière dont seront regroupées les données, dans le cas de données groupées.

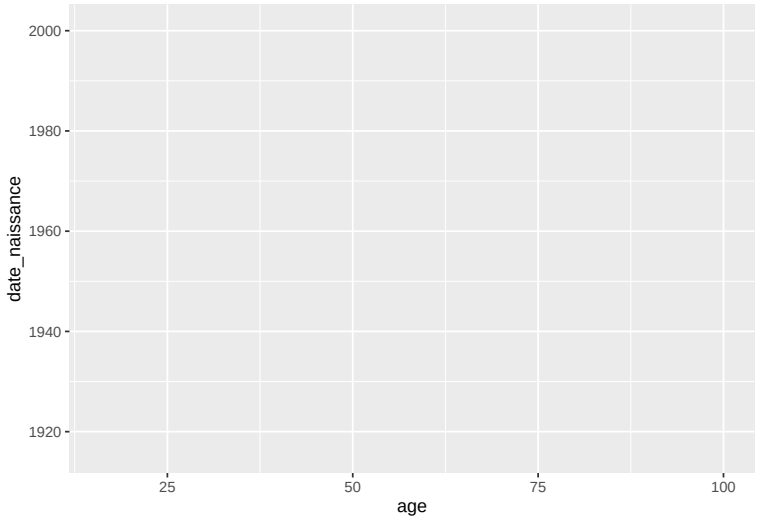
Étape 1

La première étape est donc de spécifier ces éléments dans la fonction `ggplot()`, et d'enregistrer le tout dans un objet. Par exemple :

```
> # Je crée un objet que j'appelle "graphique"
> graphique <- ggplot(data = cses, mapping = aes(x = age, y = date_naissance))
```

Si nous affichons l'objet `graphique`, `R` fera apparaître un graphique complètement vide sous l'onglet "Plots".

```
> graphique
```

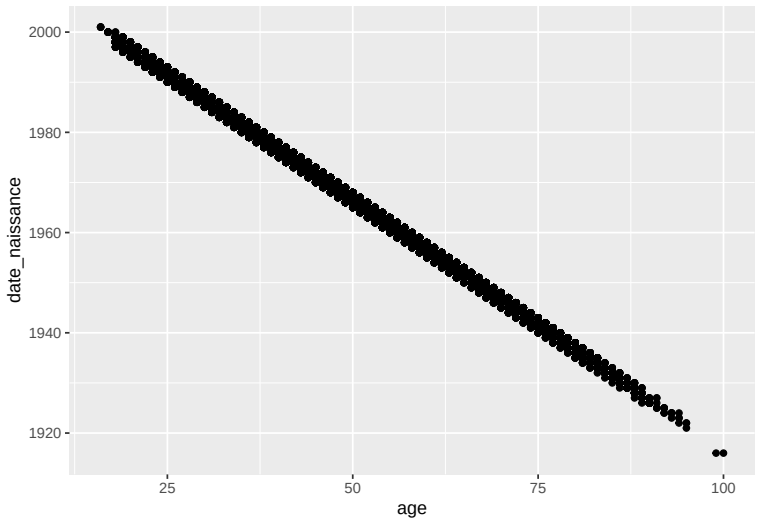


Cela s'explique par le fait que nous avons spécifié les données et les variables, mais nous n'avons toujours pas dit à `R` quel type de graphique (un nugae de points, un diagramme à bâtons, à ligne, etc.) nous voulions à l'intérieur.

Étape 2

C'est pourquoi la deuxième étape de `ggplot` est absolument essentielle. Nous ajouterons un type de graphique à notre objet original, à l'aide du symbole `+`.

```
> graphique <- graphique + geom_point()
>
> # Si on affiche ce nouvel objet "graphique", auquel on a ajouté un type, on obtient:
> graphique
```

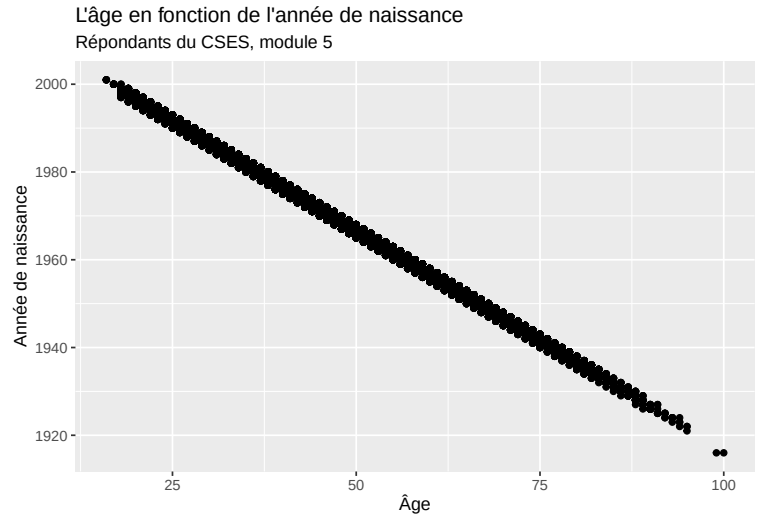


Nous avons donc un graphique avec l'âge du répondant sur l'axe des X, et l'année de naissance sur l'axe des Y. Les répondants du CSES ne sont pas tous sondés la même année, donc les points ne sont pas tous sur la même ligne. Mais en général, le nuage de point suit une tendance négative (l'année de naissance est négativement associée à l'âge).

Étape 3

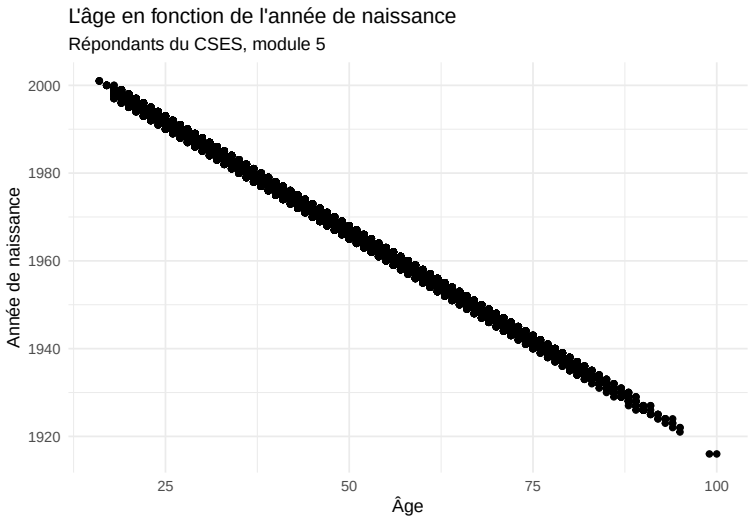
L'avantage de `ggplot()`, contrairement aux fonctions de base comme `plot()`, c'est qu'il est possible de rajouter des étages à nos graphiques. Chaque étage à sa propre utilité. Par exemple, nous pourrions ajouter un étage qui ajoute un titre et modifie les titres des axes.

```
> graphique <- graphique + labs(title = "L'âge en fonction de l'année de naissance",
+                   subtitle = "Répondants du CSES, module 5",
+                   x = "Âge",
+                   y = "Année de naissance")
>
> graphique
```



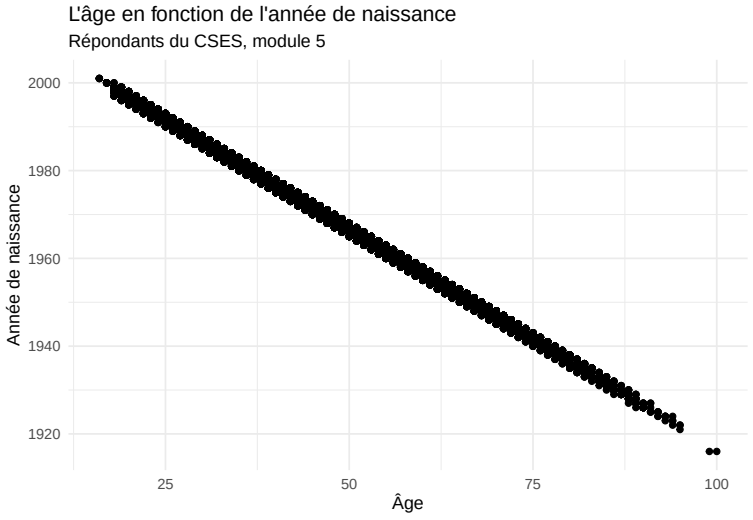
On peut aussi changer le thème. Par exemple, le thème `theme_minimal()` rend blanc le fond du graphique.

```
> graphique <- graphique + theme_minimal()
>
> graphique
```



En résumé, le code complet de notre graphique tient à quelques lignes de codes seulement. On pourrait le faire tout d'un coup, comme cela :

```
> graphique <- ggplot(data = cses, mapping = aes(x = age, y = date_naissance))
> graphique <- graphique + geom_point() +
+   labs(title = "L'âge en fonction de l'année de naissance",
+        subtitle = "Répondants du CSES, module 5",
+        x = "Âge",
+        y = "Année de naissance") +
+   theme_minimal()
>
> graphique
```

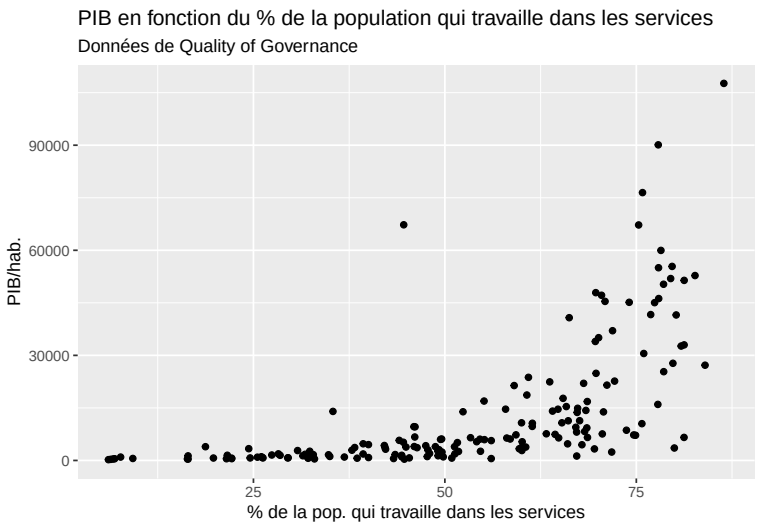


Autres fonctions et types de graphiques

Nuages de points

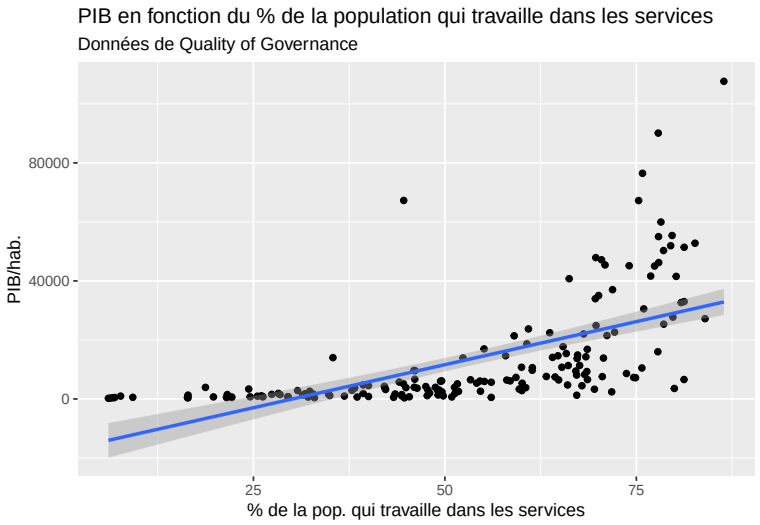
Le graphique donné en exemple vous a appris la mécanique de la fonction `ggplot`, mais n'était pas très informatif. Essayons-en un autre, qui met en relation le PIB d'un pays et le pourcentage de sa population qui oeuvre dans le secteur des services.

```
> graphique <- ggplot(data = quality, mapping = aes(x = emploi_serv, y = pib_habitant))
> graphique <- graphique + geom_point() +
+   labs(title = "PIB en fonction du % de la population qui travaille dans les services",
+        subtitle = "Données de Quality of Governance",
+        x = "% de la pop. qui travaille dans les services",
+        y = "PIB/hab.")
>
> graphique
```



Les nuages de points peuvent être accompagnés de lignes (par ex. une ligne de régression). Je recommande de commencer par visualiser les points, puis d'ajouter une ligne de régression pour avoir une idée de la dispersion des données avant que la visualisation ne puisse être influencée par une ligne de régression. Ici, j'ai ajouté une ligne de régression linéaire en ajoutant `geom_smooth(method = "lm")` au graphique original. L'intervalle grise autour de la ligne correspond à l'intervalle de confiance de 95%. Si on faisant cette analyse à nouveau avec un autre échantillon, la ligne de régression de trouverait dans cette intervalle 95 fois sur 100. D'autres types de lignes de régression (autre que la régression linéaire) peuvent être ajoutées. Voir la documentation de la fonction (`?geom_smooth`) pour plus de détails.

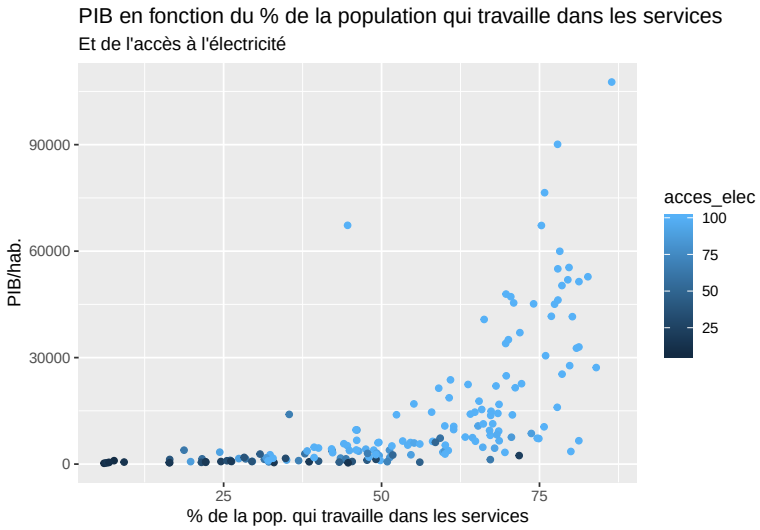
```
> graphique <- graphique +
+   geom_smooth(method = "lm")
>
> graphique
```



Points regroupés selon différentes valeurs

Nous pouvons mettre l'emphase sur une troisième variable à l'intérieur même d'un nuage de point (qui met déjà en relation 2 variables). Pour ce faire, il existe l'option `color =`, que l'on peut ajouter après avoir défini x et y. Ici, nous visualiserons encore le PIB en fonction du pourcentage de la population qui travaille dans les services, mais nous teinterons les points en fonction du pourcentage de la population qui a accès à l'électricité dans chaque pays.

```
> graphique <- ggplot(data = quality, mapping = aes(x = emploi_serv, y = pib_habitant,
+                                                    color = acces_elec))
> graphique <- graphique + geom_point() +
+   labs(title = "PIB en fonction du % de la population qui travaille dans les services",
+        subtitle = "Et de l'accès à l'électricité",
+        x = "% de la pop. qui travaille dans les services",
+        y = "PIB/hab.")
>
> graphique
```

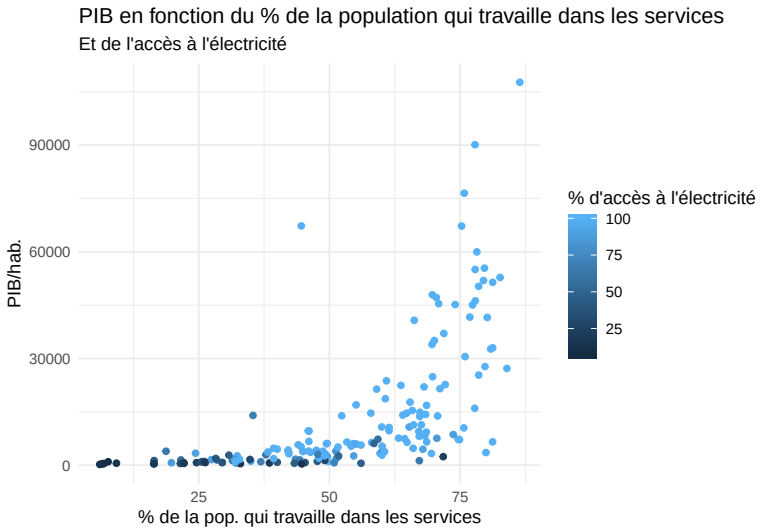


On observe que les pays qui ont un plus petit PIB et un plus petit pourcentage de la population qui travaille

dans les services ont aussi, généralement, un accès plus restreint à l'électricité. Il faut faire attention de ne pas interpréter notre graphique comme des résultats d'analyse causale. Or, si ces trois variables sont centrales à notre analyse, il peut être utile de les visualiser de cette manière dans la partie descriptive de notre recherche.

Pour modifier le titre de la légende, on ajoute `scale_color_continuous(name = ...)` au graphique original. J'ai aussi ajouté `theme_minimal()` puisque je préfère cette apparence.

```
> graphique <- graphique +
+   scale_color_continuous(name = "% d'accès à l'électricité") +
+   theme_minimal()
>
> graphique
```



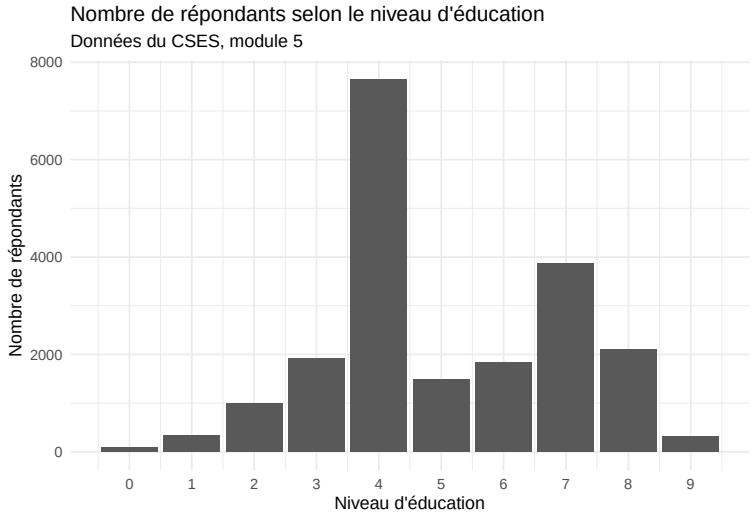
Graphiques à bâtons

À l'aide du diagramme à bâtons, nous pourrions visualiser le nombre d'observations dans différentes catégories d'une variable nominale ou ordinale. Ce type de graphique peut être créé grâce à `geom_bar()`. Par exemple, je visualise ici le nombre de répondant.e.s dans chaque catégorie de la variable `education` des données du `cses`. Pour ce type de graphique, il est seulement nécessaire de définir x, puisque l'axe des y correspond au nombre de répondants et non à une deuxième variable, comme c'était le cas avec le nuage de points.

J'ai renommé les axes et modifié les chiffres sur l'axe des X à l'aide de la fonction `scale_x_continuous()`, qui permet de changer la gradation d'un axe des x dans que cas d'une variable continue.

```
> # Graph de base
> graphique <- ggplot(data = cses, mapping = aes(x = education))
> graphique <- graphique + geom_bar()
>
> # Renommer les axes
> graphique <- graphique +
+   labs(title = "Nombre de répondants selon le niveau d'éducation",
+        subtitle = "Données du CSES, module 5",
+        x = "Niveau d'éducation",
+        y = "Nombre de répondants")
>
> # Changer les chiffres sur l'axe des x et changer l'apparence
> graphique <- graphique +
+   scale_x_continuous(breaks = c(0,1,2,3,4,5,6,7,8,9)) +
```

```
+   theme_minimal()
>
> # Afficher le graphique
> graphique
```



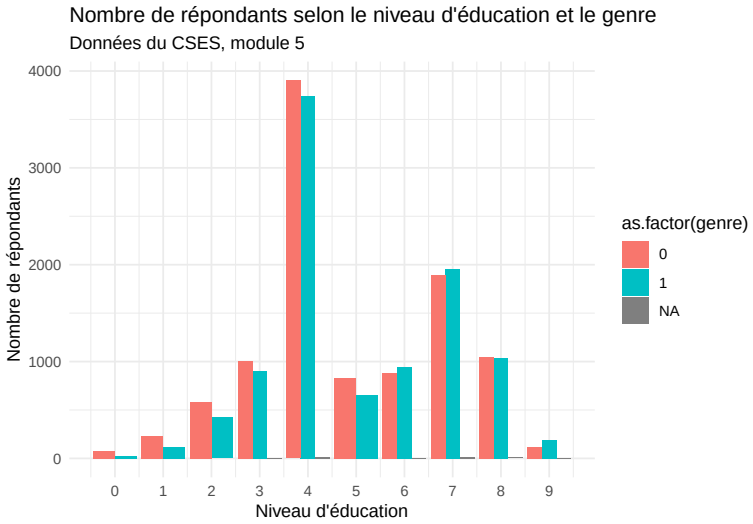
Bâtons regroupés selon différentes catégories

Tout comme nous avons fait pour le nuage de points, il est possible d'ajouter une autre variable à un diagramme à bâtons. Imaginez que nous aimerions visualiser le niveau d'éducation des répondants, mais cette fois-ci en fonction de leur genre (féminin ou masculin dans le sondage).

Pour ce faire, nous pourrions spécifier le regroupement dans l'argument `fill =`, qui suit `x =`. Étant donné que la variable `genre` est codée 0 ou 1, `R` considère qu'il s'agit d'une variable numérique qui peut prendre n'importe quelles valeurs. Or, on sait que notre variable *ne prend que les valeurs 0 et 1*. Pour forcer `R` à considérer cette variable comme une variable qui comporte seulement deux valeurs (comme s'ils s'agissaient des mots "femmes" et "hommes"), nous utilisons la fonction `as.factor()` autour de `genre`. Finalement, nous ajoutons `position = "dodge"` à l'intérieur de `geom_bar()` pour que les barres des femmes et des hommes soient positionnées côté-à-côte, et non l'une par-dessus l'autre.

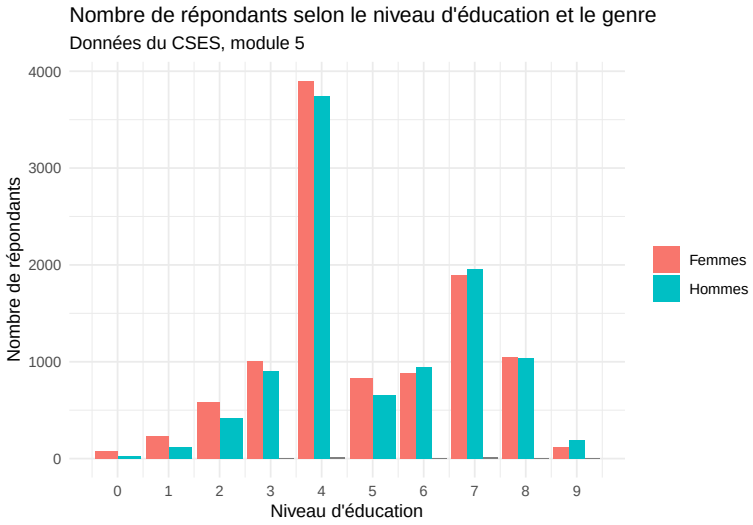
```
> # Graph de base
> graphique <- ggplot(data = cses, mapping = aes(x = education,
+                                                fill = as.factor(genre)))
> graphique <- graphique + geom_bar(position = "dodge")
>
> # Renommer les axes et l'apparence du graphique
> graphique <- graphique +
+   labs(title = "Nombre de répondants selon le niveau d'éducation et le genre",
+        subtitle = "Données du CSES, module 5",
+        x = "Niveau d'éducation",
+        y = "Nombre de répondants") +
+    theme_minimal()
>
> # Changer les chiffres sur l'axe des x
> graphique <- graphique +
+   scale_x_continuous(breaks = c(0,1,2,3,4,5,6,7,8,9))
>
```

```
> # Afficher le graphique
> graphique
```



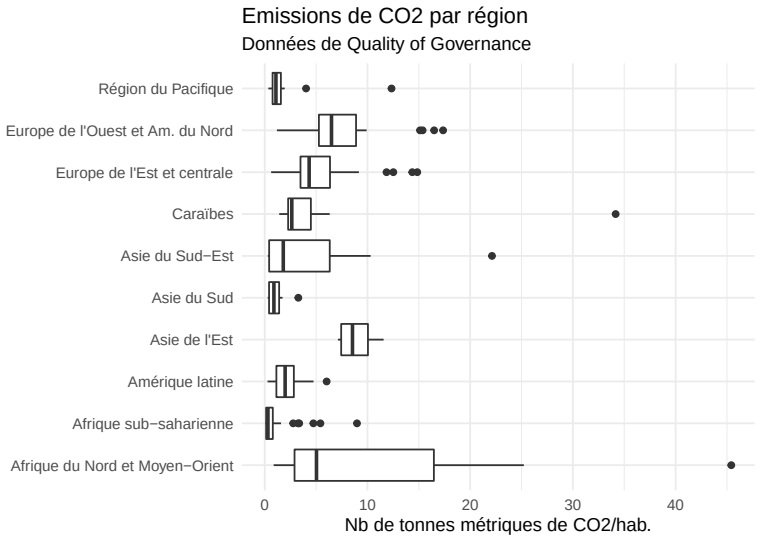
La légende n'est pas très informative. Étant donné que nous connaissons nos données, nous savons que la variable **genre** est codée 1 pour les hommes et 0 pour les femmes. Or, une personne qui lit notre analyse ne pourrait pas le savoir au seul coup d'oeil de notre graphique. Nous allons donc modifier la légende à l'intérieur de la fonction **scale_fill_discrete()**, qui permet de modifier les éléments qui ont été précisés sous **fill =**. Nous pouvons supprimer le titre de la légende, qui n'est pas essentiel, en ajoutant **theme(legend.title = element_blank())** au graphique.

```
> graphique <- graphique +
+   scale_fill_discrete(breaks = c("0", "1"),
+                       labels = c("Femmes","Hommes")) +
+   theme(legend.title = element_blank())
>
> graphique
```



Les boîtes à moustache

Un des graphiques populaires en science politique est le "boxplot", ou graphiques de quantiles normaux (ou encore : boîtes à moustaches!) Ce type de graphique est bien utile, puisqu'il permet de visualiser plusieurs statistiques descriptives d'une même variable en un seul coup. En voici un exemple :

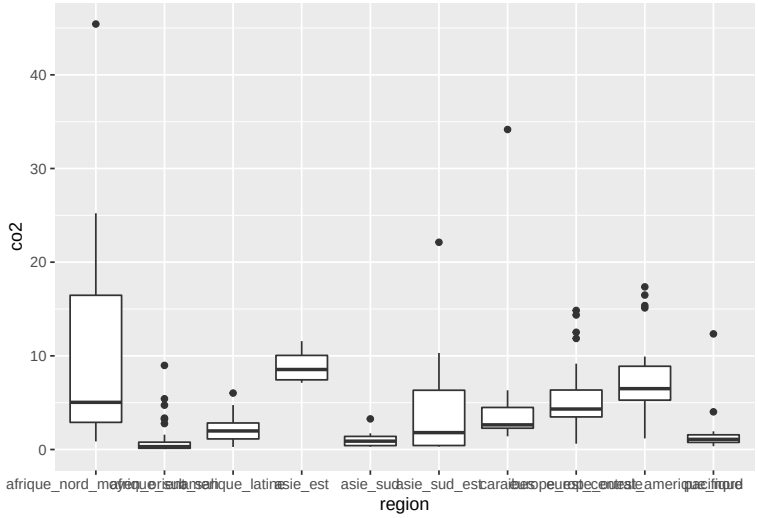


La ligne qui se situe au centre de chaque boîte correspond à la médiane de la variable 'CO2' (donc la valeur qui sépare la distribution en deux parties égales). Les deux extrémités de chaque boîte correspondent au 1er et 3e quartiles (donc les valeurs situées à 25% et et 75% de la distribution). Les 'moustaches', ou les lignes qui sortent de chaque boîte, s'arrêtent aux valeurs les plus extrêmes de la distribution, en excluant les observations aberrantes ("outliers"). Les observations aberrantes représentent les points. Ce sont les valeurs situées à une distance d'au moins 1,5*l'écart inter-quartile de la distribution.

En une seule image, nous avons une idée de la distribution d'une variable (ici, la variable CO2) selon différentes catégories de notre banque de données (ici, les régions). Les boîtes à moustache remplacent en quelque sorte les tableaux de statistiques descriptives!

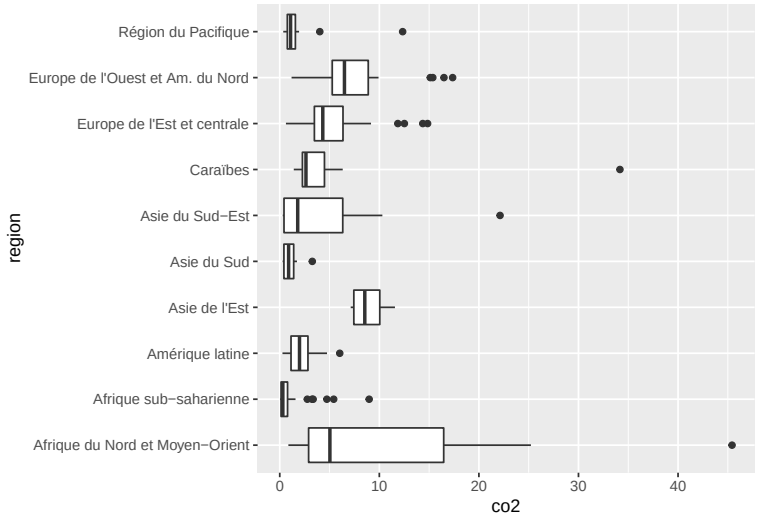
Pour obtenir ce graphique, on définit d'abord un graphique de base à l'aide de la fonction `ggplot()`. X représente la région, tandis que Y est la variable CO2. Puis, on ajoute `geom_boxplot()`.

```
> box <- ggplot(data = quality, mapping = aes(x = region, y = co2))
> box <- box + geom_boxplot()
>
> box
```



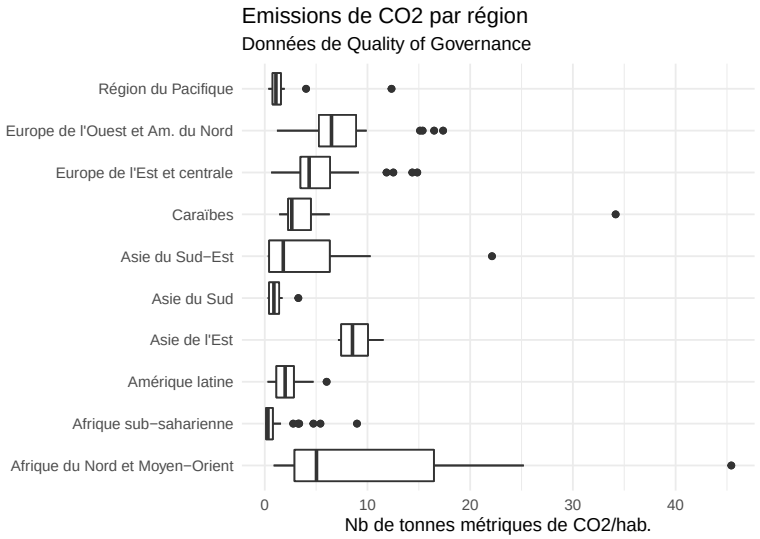
Il y a plusieurs problèmes avec ce graphique. Le problème le plus important concerne les étiquettes de l'axe des x : le nom des régions est très long, et n'est pas recodé. Nous allons d'abord recoder chaque catégorie à l'aide de la fonction `scale_x_discrete(labels = ...)`. Cette étape prend un peu de temps. Puis, pour que les noms des catégories aparaissent dans leur ensemble, nous allons échanger les axes des x et y. Nous pourrons maintenant lire les étiquettes à l'horizontal, sans qu'elles ne s'entre-croisent. Cela est accompli grâce à `coord_flip()`.

```
> box <- box +
+   scale_x_discrete(labels = c("Afrique du Nord et Moyen-Orient",
+                               "Afrique sub-saharienne",
+                               "Amérique latine",
+                               "Asie de l'Est",
+                               "Asie du Sud",
+                               "Asie du Sud-Est",
+                               "Caraïbes",
+                               "Europe de l'Est et centrale",
+                               "Europe de l'Ouest et Am. du Nord",
+                               "Région du Pacifique")) +
+   coord_flip()
>
> box
```



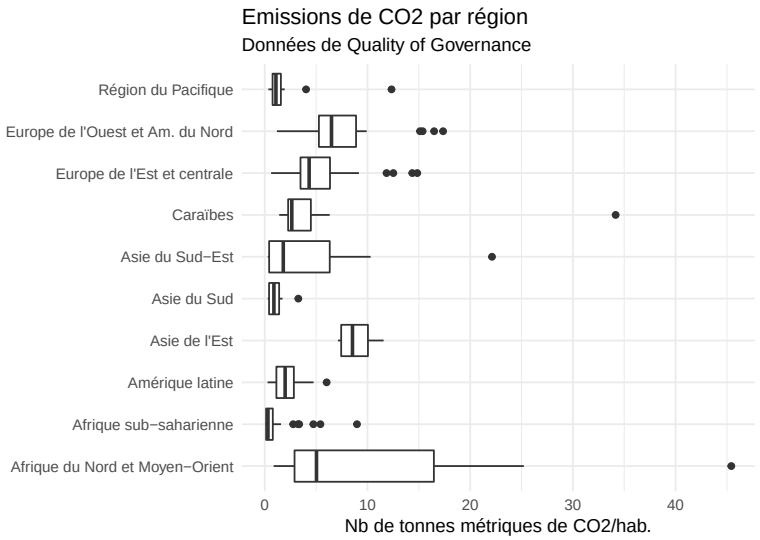
Nous y sommes presque. Il reste à modifier l'apparence générale du graphique, à renommer les axes et à donner un titre au graphique. J'ai pris la décision d'enlever le nom de l'axe vertical puisque les étiquettes des régions sont assez explicites à mon avis.

```
> box <- box + geom_boxplot() +
+   labs(title = "Emissions de CO2 par région",
+        subtitle = "Données de Quality of Governance",
+        x = "",
+        y = "Nb de tonnes métriques de CO2/hab.") +
+   theme_minimal()
>
> box
```



En bref :

```
> box <- ggplot(data = quality, mapping = aes(x = region, y = co2))
> box <- box + geom_boxplot() +
+   scale_x_discrete(labels = c("Afrique du Nord et Moyen-Orient",
+                               "Afrique sub-saharienne",
+                               "Amérique latine",
+                               "Asie de l'Est",
+                               "Asie du Sud",
+                               "Asie du Sud-Est",
+                               "Caraïbes",
+                               "Europe de l'Est et centrale",
+                               "Europe de l'Ouest et Am. du Nord",
+                               "Région du Pacifique")) +
+   coord_flip() +
+   labs(title = "Emissions de CO2 par région",
+        subtitle = "Données de Quality of Governance",
+        x = "",
+        y = "Nb de tonnes métriques de CO2/hab.") +
+   theme_minimal()
>
> box
```



Recommandations

Ces notes sont un survol *très* sommaire de ce qu'il est possible de faire avec la fonction `ggplot()`. Le temps nous manque pour que je vous explique toutes les possibilités. Si vous voulez jouer avec cette fonction, internet sera votre meilleur ami. Si vous avez des questions sur un type de graphique ou une modification particulière, il faudra chercher par vous-même à partir de maintenant. On peut passer des heures à modifier un simple élément d'un graphique, mais avec `ggplot()`, le résultat vaut amplement le temps investi!

Sur son site, Kieran Healy vous prend par la main pour vous apprendre `ggplot()`, de ses fonctions les plus simples aux plus avancées (en anglais). Cette ressource pourrait être utile/intéressante pour celles et ceux qui souhaitent approfondir leurs compétences en visualisation de données, ou qui aimeraient consulter des bons (et moins bons) exemples de visualisation. Amusez-vous!