Introduction à l'utilisation du package ggplot2 pour représenter les données - avec corrigé en fin de document

Marie Laure Delignette-Muller et Karine Chalvet-Monfray

2024-02-22

Contents

Préambule

Les graphiques de base dans R font appel au package (ou librairie) `graphics` qui est installé d'office lorsque l'on installe R. La fonction générique `plot()` fournit un graphique différent selon la classe de l'objet qui lui est donné en premier argument. Nous ne l'utiliserons que ponctuellement car nous préférons vous initier à des outils plus modernes. En effet de plus en plus d'utilisateurs de R utilise aujourd'hui un autre **package appelé `ggplot2`**. Ce dernier offre la possibilité de produire facilement une grande diversité de graphes directement utilisables dans les rapports / articles ... Comme ce package n'est pas encore dans la distribution de base de R, si vous ne l'avez pas encore fait, il convient de l'installer (*dans Rstudio onglet Tools/Install Packages puis taper ggplot2 dans le champ Packages en vérifiant que Install Dependencies est bien coché en dessous*). Ensuite, il faudra mettre dans un chunk dans votre rapport d'analyse le code ci-dessous pour charger le package (on installe le package sur son ordinateur une fois pour toutes, mais on doit le charger avec cette commande à chaque nouvelle session R).

```
require(ggplot2)
```

Rappelons que la partie représentation des données est capitale dans l'analyse des données et ne doit pas être négligée. Les graphes apportent généralement beaucoup plus d'information que les résultats des tests statistiques notamment, et sont le plus souvent indispensables pour guider le choix des résumés statistiques (avec leurs intervalles de confiance) et tests à réaliser éventuellement en complément pour préciser / consolider les conclusions.

Importation et présentation du jeu de données utilisé

Importez le jeu de données que l'on utilisera pour la suite et jetez un oeil rapide aux variables qu'il comporte, à partir du résultat de la fonction `str()`et des indications ci-dessous.

```
# Importation du jeu de données
d <- read.table("ENQ2223.txt", header = TRUE, stringsAsFactors = TRUE)
str(d) # pour rappel de la structure du jeu de données
```

```
## 'data.frame':    149 obs. of  19 variables:
##  $ ident           : Factor w/ 149 levels "Anonyme1","Anonyme10",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ civilite        : Factor w/ 2 levels "Madame","Monsieur": 2 1 1 1 1 1 1 2 1 2 ...
##  $ cursus          : Factor w/ 3 levels "A1","Autre","BCPST": 1 1 3 3 3 3 3 3 3 3 ...
##  $ tps_travail     : int  2 10 0 5 8 8 15 35 7 0 ...
##  $ nb_CM           : int  0 10 0 10 20 20 10 17 8 0 ...
##  $ apprentissage   : Factor w/ 3 levels "intermediaire",..: 2 2 2 1 1 3 1 3 2 2 ...
##  $ tps_job         : num  0 0 5 0 0 0 0 0 0 3 ...
##  $ tps_loisirs     : int  25 10 14 6 13 7 6 6 5 20 ...
##  $ nb_rattrapages  : int  0 0 0 0 0 0 1 0 2 0 ...
##  $ niveau_global   : int  13 11 15 10 7 13 14 15 5 15 ...
##  $ memoire         : int  16 13 20 15 15 15 14 13 8 18 ...
##  $ orthographe     : int  18 9 19 17 18 16 18 13 19 19 ...
##  $ biostat_proba   : int  7 1 5 12 5 16 17 18 2 16 ...
##  $ biostat_info    : int  9 9 5 8 10 16 17 19 10 17 ...
##  $ reorientation   : Factor w/ 2 levels "non","oui": 1 1 1 1 1 1 1 2 2 1 ...
##  $ bien_etre       : int  17 15 20 10 19 10 10 15 1 19 ...
##  $ serenite_examens: int  15 11 19 8 15 13 8 8 3 18 ...
##  $ interet_cours   : int  13 10 18 12 15 14 12 14 15 15 ...
##  $ filiere         : Factor w/ 5 levels "AC","autre","equine",..: 1 5 4 4 4 1 4 1 5 1 ...
```

Le thème principal du questionnaire rempli en mars 2023 par les étudiants de A2 dans le cadre des TD de biostatistique, était d'évaluer l'impact du cursus d'origine des étudiants (A1, BCPST ou autre) sur la façon dont ils vivent leur scolarité, leur façon de travailler, ... A visée pédagogique nous avions volontairement défini plusieurs questions sous forme de scores allant de 0 à 20. Ces scores correspondent aux questions posées suivantes:

- `niveau_global` : "Situez sur une échelle de 0 à 20 la perception de votre niveau global en A2 en prenant comme référence une note de 10 correspondant à un étudiant moyen de A2, tous cursus initiaux confondus (0 = niveau nettement plusfaible que la moyenne, 20 niveau nettement plus élevé la moyenne) ?"
- `memoire` : "Evaluez de la même façon sur une échelle de 0 à 20 vos capacités en apprentissage par coeur (20 =excellentes) notamment de mots nouveaux (nécessaire par ex. pour apprendre le nom des divers organes, des races, des espèces, ...)."
- `orthographe`: "Evaluez votre niveau en orthographe par une note de 0 à 20 (représentative des notes que vous aviez en moyenne en dictée au collège)."
- `biostat_proba` : "Evaluez, toujours sur une échelle de 0 à 20, vos facilités en biostatistique concernant les aspects conceptuels faisant appel à des notions de maths / probabilités (20 = très à l'aise)."
- `biostat_info` : "Evaluez, toujours sur une échelle de 0 à 20, vos facilités en biostatistique concernant les aspects informatique /programmation utiles pour la mise en pratique avec R (20 = très à l'aise)."
- `bien_etre` : "Evaluez état de bien être actuel (au sens absence de stress) sur une échelle de 0 à 20 (de 0 = stress maximum à 20 = aucun stress)."
- `serenite_examens` : "Evaluez votre état de sérénité lié aux examens à VetAgro Sup de 0 à 20 (de 0 = je suis tétanisée par peur d'échouer à 20 = j'y vais tranquille)."
- `interet_cours` : "Evaluez votre intérêt global pour les enseignements qui vont sont proposés à VetAgro Sup de 0 à 20 (de 0 =sans intérêt, tout me barbe, à 20 = tous m'intéressent)."

Parmi les variables autres variables issues de ce questionnaire on peut lister :

- `civilite` : la civilité (Monsieur ou Madame)

- `cursus` : le cursus (A1, Autre, BCPST)
- `nb_CM` : le nombre moyen de séquences de CM suivi par semaine
- `apprentissage` : quand ils se mettent à apprendre leurs cours (intermediaire, tôt, tard)
- `tps_loisirs` : le temps de loisirs moyen en heures par semaine
- `tps_job` : le temps de job étudiant moyen en heures par semaine
- `nb_rattrapages` : le nombre de rattrapage au 1er semestre-
- `reorientation` : la réponse à la question "Avez-vous déjà pensé à vous réorienter depuis le début de cette année ?" (oui/non)
- `filiere` : la filière envisagée (AC, equine, mixte, rurale, autre)

La grammaire des graphes proposée dans `ggplot2`

ggplot2 est un package qui a été créé en 2007 par Hadley Wickham sur une idée très originale de grammaire des graphes (gg pour "grammar of graphics' '), qui permet de définir un graphe comme un objet R.

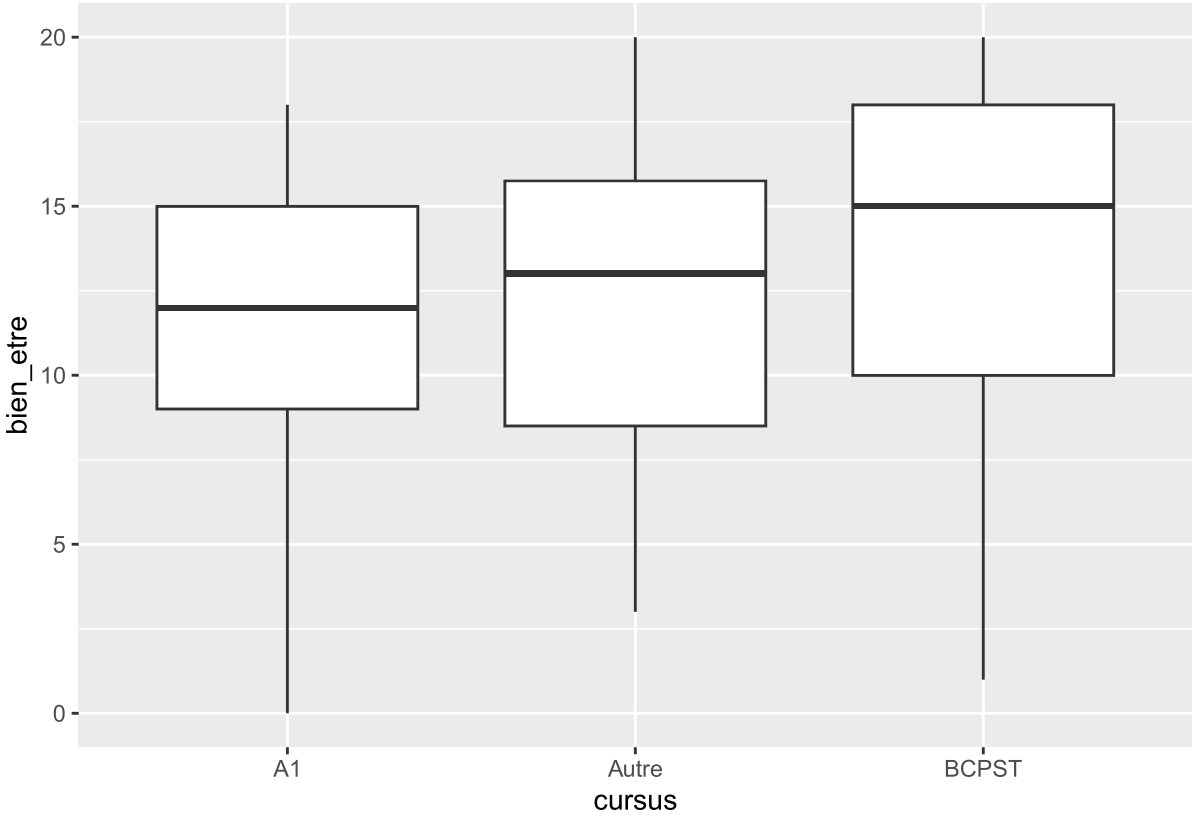
Un graphe créé à partir de la fonction `ggplot()` est un objet R défini tout d'abord par un **jeu de données** (argument `data`) et une **esthétique** (argument `mapping` définie avec la fonction `aes()`). Cette esthétique permet par exemple de définir la variable représentée sur l'axe des x, la variable représentée sur l'axe des y, la variable qui code pour la couleur, celle qui code pour la forme des points, celle qui permet de séparer des groupes... Un graphe est défini par superposition de couches (`layers`) : la fonction `ggplot()` crée un objet graphique en définissant le jeu de données et l'esthétique et il faut lui ajouter une ou des fonctions (cf. liste non exhaustive ci-dessous) avec le symbole + pour compléter sa définition.

- Pour spécifier le type de graphe on utilise les fonctions de géométrie (fonctions `geom_point()`, `geom_line()`, `geom_boxplot()`,...).

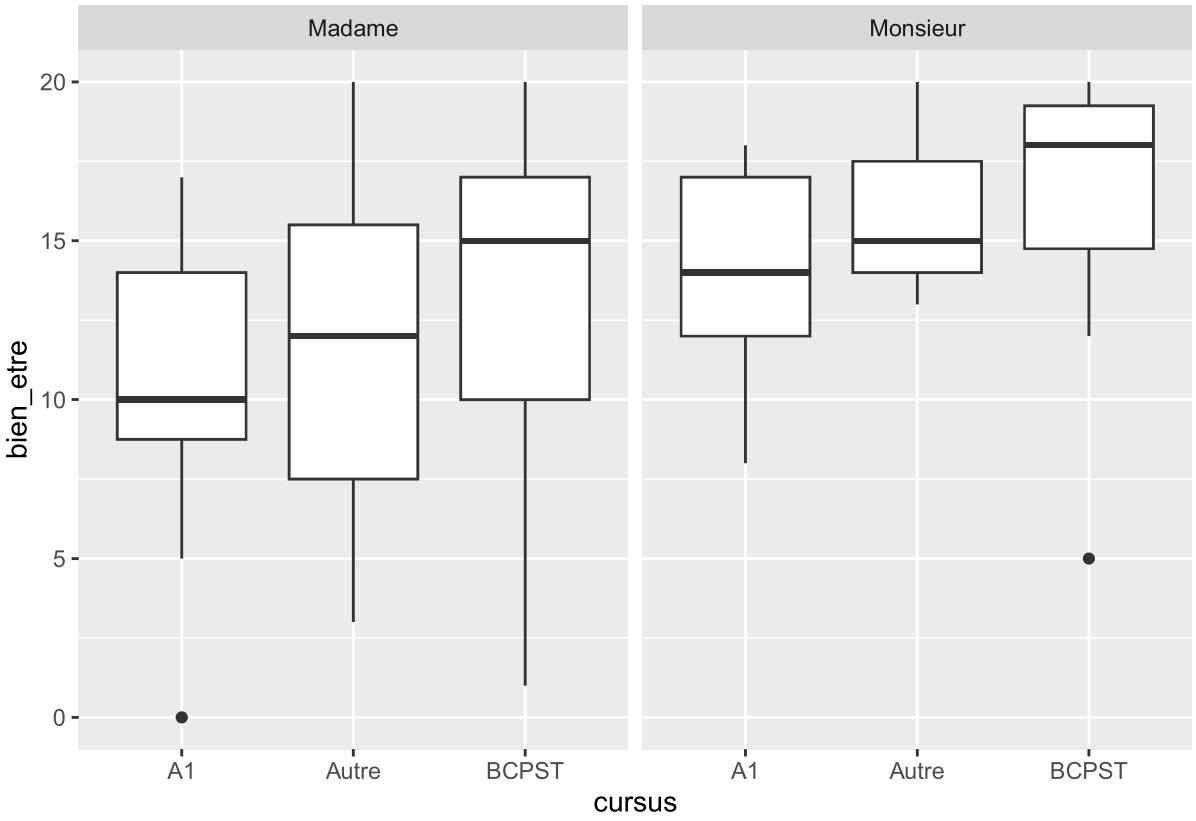
- Pour découper le graphe par groupe on utilise la fonction `facet_wrap()` pour un découpage basé sur une variable et `facet_grid()` pour un découpage basé sur deux variables.
- Pour modifier le thème de base (fond, grille) on utilise une fonction de type `theme_..()`, par ex. `theme_bw()` pour avoir un fond blanc.
- Pour ajouter / modifier titre / sous-titres, labels des axes, ..., on peut utiliser la fonction `labs()` (ex. `labs(title = "Montitre", x = "Mon label des x")`).
- Pour changer l'échelle des axes, on peut utiliser les fonctions `scale_x_log10()` et `scale_y_log10()`.

Pour bien comprendre comment cela fonctionne, faites tourner ligne à ligne le code suivant, en regardant bien la ligne de code et le résultat.

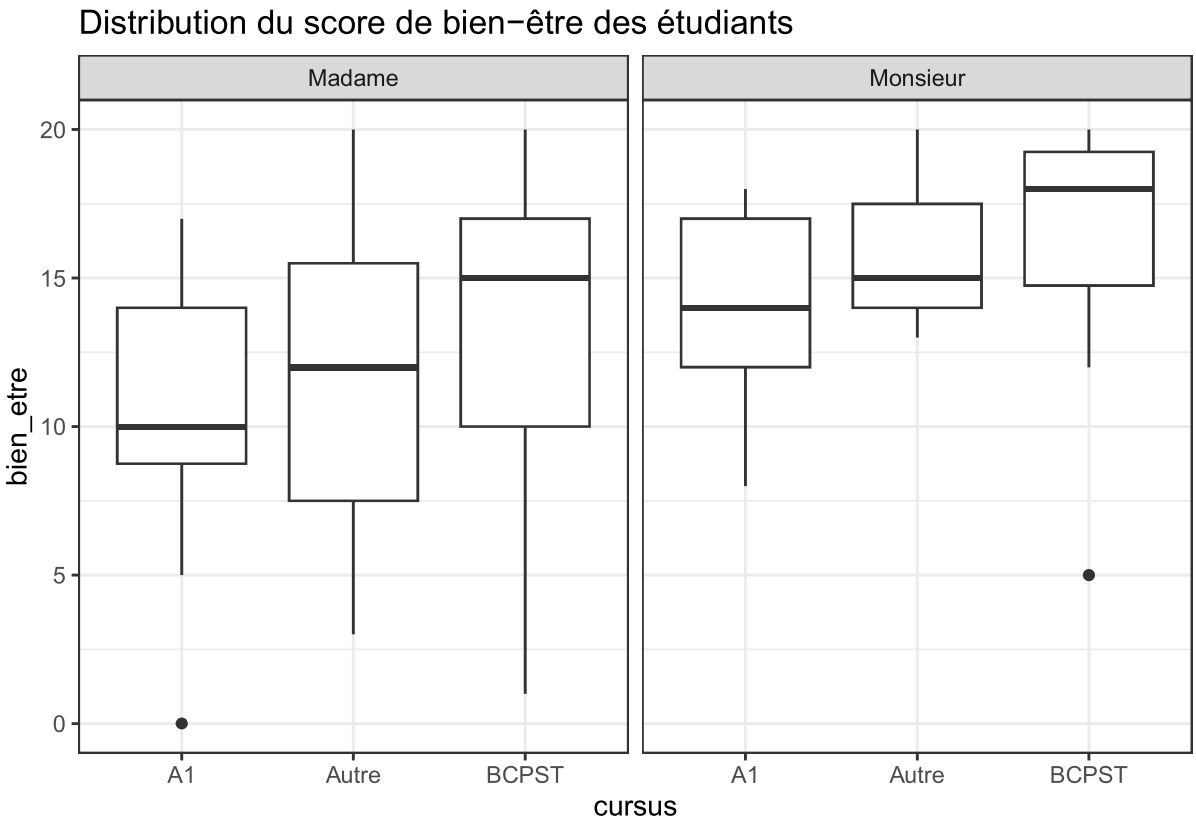
```
# Chargement du package (à mettre une fois en début de rapport d'analyse)
require(ggplot2)

# Définition d'un graphe comme objet R
gg1 <- ggplot(data = d, mapping = aes(x = cursus, y = bien_etre)) +
        geom_boxplot()

# Tracé du graphe
gg1
```



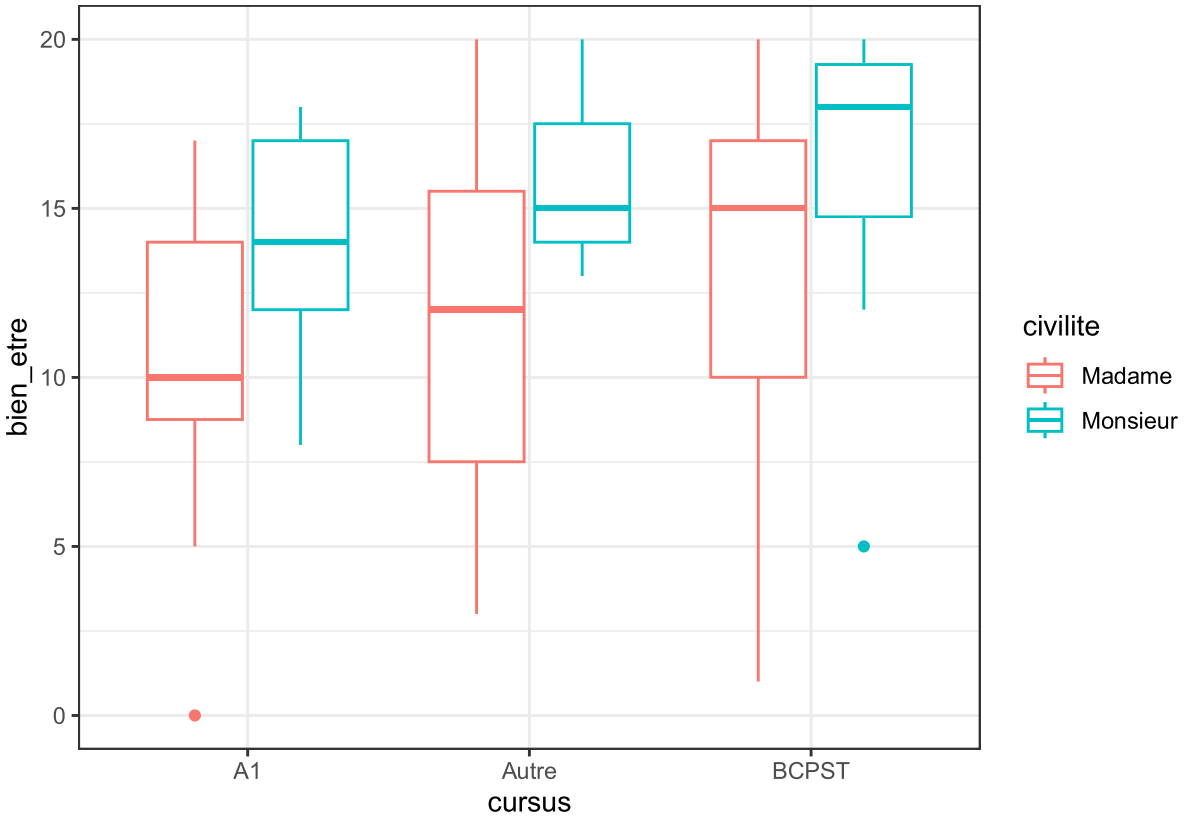

```
# Ajout d'une couche supplémentaire qui divise le graphe en
# 2 par civilité
gg1 <- gg1 + facet_wrap(~ civilite)
gg1
```




```
# Ajout d'une couche pour changer le thème et mettre un titre
gg1 <- gg1 +
      theme_bw() + labs(title = "Distribution du score de bien-être des étudiants")
gg1
```




```
# Autre graphe des mêmes données en utilisant la couleur pour
# différencier femmes et hommes
# si on entoure l'instruction de parenthèse voit l'objet R en même temps
# (cela évite de réécrire une ligne de code en dessous pour voir l'objet créé)
(gg2 <- ggplot(data = d,
        mapping = aes(x = cursus, y = bien_etre, col = civilite)) +
        geom_boxplot() + theme_bw())
```



Les graphes les plus classiques

Pour chacune des sous-parties ci-dessous, faites tout d'abord tourner le code donné et regardez chaque graphe obtenu en ajoutant dans le chunk, pour chaque graphe, une ligne de code avec le nom du graphe (pour que R le montre), ou en entourant la ligne de code de parenthèses. Prenez le temps d'identifiez de quel type de graphe il s'agit (ex. histogramme de fréquences, diagramme quantile-quantile, diagramme en bâtons, nuage de points, ...).

Ensuite faites les exercices demandés en écrivant le code dans un chunk que vous insérez à la place indiquée. Prenez le temps de comprendre le code éventuellement en le modifiant légèrement (ex. changement de variables, de valeurs, d'argument). Essayez d'imaginer le résultat à la lecture du code et vérifier que c'est ce que l'on obtient.

Représentation d'une variable quantitative

> **UTILISATION des fonctions** `geom_dotplot()`, `geom_histogram()`, `geom_boxplot()`, `stat_ecdf()`, `geom_qq()`, `labs()`, `theme_bw()`.

LORSQUE VOUS NE SAVEZ PAS QUELS SONT LES ARGUMENTS D'UNE FONCTION, VOUS POUVEZ TOUJOURS ALLER VOIR SA PAGE D'AIDE (ex. `?geom_dotplot`)

Exemples de représentation d'une série d'observations

```
gg3 <- ggplot(d, aes(x = tps_loisirs)) + geom_dotplot()
# on peut ne pas mettre les noms des premiers arguments
# si on les met bien dans le bon ordre (cf. ?ggplot)

gg4 <- ggplot(d, aes(x = tps_loisirs)) + geom_histogram()
gg4bis <- ggplot(d, aes(x = tps_loisirs)) +
            geom_histogram(breaks = seq(from = 0, to = 40, by = 5))

gg5 <- ggplot(d, aes(y = tps_loisirs)) + geom_boxplot()

gg6 <- ggplot(d, aes(sample = tps_loisirs)) + geom_qq()
# notez qu'ici on ne peut pas utiliser l'argument x dans aes()
# mais il faut utiliser l'argument sample

gg7 <- ggplot(d, aes(x = tps_loisirs)) + stat_ecdf(geom = "step") +
  theme_bw() + labs(title = "Diagramme des fréquences cumulées")
```

Exemples de représentation de plusieurs séries d'observations

EXERCICE A

Si vous avez bien compris le fonctionnement de ggplot2, insérez ici un chunk pour tracer :

1. sur un même graphe, sur fond blanc (fonction `theme_bw()`), la courbe de fréquences cumulées du temps de loisirs pour les étudiants et les étudiantes avec deux couleurs différentes (argument `colde aes()`).
2. sur un même graphe, sur fond blanc, les diagrammes en boîte du temps de loisirs pour les étudiants et les étudiantes (avec deux couleurs différentes, en utilisant l'argument `fill` de `aes()` plutôt que `col`).
3. Sur 2 graphes côte à côte, le graphe de type dotplot du temps de loisirs pour les étudiants d'un côté et les étudiantes de l'autre, (fonction `facet_wrap(~ nom_de_la_variable_codant_pour_le_groupe)`).
4. Sur le même graphe, le diagramme en boîte du temps de loisirs par cursus d'origine (une couleur par cursus). Vous trouverez ci-dessous un code, que vous pouvez faire tourner, pour faire deux variantes de ce dernier graphe, avec le temps de loisirs en y:

```
ggEx4b <- ggplot(d, aes(x = civilite, y = tps_loisirs)) +
  geom_dotplot(binaxis = "y", binwidth = 1) + theme_bw()

ggEx4c <- ggplot(d, aes(x = civilite, y = tps_loisirs)) +
  geom_dotplot(binaxis = "y", stackdir = "center",
               binwidth = 1) + theme_bw()
```

5. *OPTIONNEL pour les plus rapides, tentez de faire deux-trois autres représentations intéressantes sur des variables quantitatives du jeu de données.*

Représentation d'une variable qualitative

UTILISATION de la fonction `geom_bar()`.

Exemples de représentation d'une série d'observations

```
gg8 <- ggplot(d, aes(x = cursus)) + geom_bar()

# si on veut mettre l'échelle de y en proportions
# ce qui sera utile notamment pour comparer deux ou plusieurs diagrammes
# en bâtons, le code devient un peu compliqué
gg9 <- ggplot(d, aes(x  = cursus)) +
        geom_bar(aes(y = after_stat(prop), group = 1))
```

Exemples de représentation de plusieurs séries d'observations

```
gg10 <- ggplot(d, aes(x = cursus, fill = civilite)) +
  geom_bar(col = "white")

gg11 <- ggplot(d, aes(x = cursus, fill = civilite)) +
  geom_bar(position = "fill", col = "white")

gg11inverse <- ggplot(d, aes(x = civilite, fill = cursus)) +
  geom_bar(position = "fill", col = "white")

gg12 <- ggplot(d, aes(x = cursus, fill = civilite)) +
  geom_bar(position = "dodge2")

gg13 <- ggplot(d, aes(x = cursus, fill = civilite)) +
  geom_bar(aes(y = after_stat(prop), group = civilite), position = "dodge2")
```

EXERCICE B

Insérez ici un chunk avec des lignes de code pour tracer deux-trois graphes différents représentant la distribution de la filière envisagée pour les étudiants et les étudiantes et sélectionnez ceux qui vous paraissent les plus parlants.

OPTIONNEL si vous allez très vite, explorez graphiquement un ou deux autres couples de variables qualitatives.

Représentation de nuages de points (deux variables quantitatives)

UTILISATION des fonctions `geom_point()` et `geom_jitter()`.

```
gg14 <- ggplot(d, aes(x = biostat_proba, y = biostat_info)) + geom_point()

gg15 <- ggplot(d, aes(x = biostat_proba, y = biostat_info)) + geom_jitter()
# on utilise souvent geom_jitter qui bruite les valeurs de x et y
# pour visualiser les ex-aequos

gg16 <- ggplot(d, aes(x = biostat_proba, y = biostat_info)) +
        geom_jitter() + facet_wrap(~ civilite)

gg17 <- ggplot(d, aes(x = biostat_proba, y = biostat_info,
                      col = cursus)) + geom_jitter()
```

```r
gg18 <- ggplot(d, aes(x = biostat_proba, y = biostat_info, shape = cursus)) +
        geom_jitter(alpha = 0.4, size = 3) + theme_bw()
# l'argument alpha permet d'ajouter de la transparence
# (alpha = intensité de la couleur, 1 opaque, 0 transparent)
# et size ici de changer de la même façon la taille de tous les points
# vous pouvez changer les valeurs de ces deux arguments pour voir l'effet

gg19 <- ggplot(d, aes(x = biostat_proba, y = biostat_info,
                      col = cursus)) + facet_wrap(~ civilite) +
                      geom_jitter() + theme_bw()

gg20 <- ggplot(d, aes(x = biostat_proba, y = biostat_info,
            col = cursus, size = nb_rattrapages)) + theme_bw() +
            geom_jitter(alpha = 0.6) + facet_wrap(~ civilite) +
            labs(title = "Encore un graphe !",
            subtitle = "avec la taille des points et transparence")
```

EXERCICE C

Insérez ici un chunk avec des lignes de code pour tracer diverses variantes d'un nuage de points avec en x le score de niveau global et en y le score de bien être, et une ou plusieurs autres variables additionnelles sous forme de couleur, facettes, type de points, ...

Divers compléments

Comment sauver son graphe au format souhaité avec `ggsave()`

Rmarkdown permet de faire des rapports d'analyse en format **html**, **Word** (et pdf, mais ce dernier format nécessite l'installation au préalable de tinytex) qui contiennent les graphes. Néanmoins il peut être utile dans certains cas d'exporter un graphe dans un format donné (ex. jpeg, pdf, png, tiff). On peut utiliser pour cela la fonction `ggsave()`. Faites tourner le code suivant et tentez d'insérer chacun des deux dans un fichier Word, pour voir la différence entre les deux. Jouer ainsi sur la taille de l'image permet notamment de passer facilement d'un graphe insérable dans un(e) rapport / article / thèse (vu de près) au même graphe montré dans un diaporama (vu de loin).

```r
ggsave(gg21, filename = "graphe_pour_un_diaporama.jpg",
       device = "jpeg", width = 6, height = 4)
ggsave(gg21, filename = "graphe_pour_un_rapport.jpg",
       device = "jpeg", width = 12, height = 8)
```

OPTIONNEL si vous allez très vite, vous pouvez faire un graphe de votre choix, tentez de le personnaliser en utilisant les différents arguments des fonction utilisées, et le sauver dans le format de votre choix.

Comment faire un graphe interactif avec la fonction `ggplotly()` du package `plotly`

OPTIONNEL si vous allez très vite, vous pouvez installer le package `plotly` pour tenter de faire un graphe interactif à l'aide du code suivant et l'explorer.

```
require(plotly)
gg21 <- ggplot(d, aes(x = biostat_proba, y = biostat_info,
            col = cursus, size = nb_rattrapages, shape = civilite)) +
            theme_bw() + geom_jitter(alpha = 0.6)
ggplotly(gg21)
```

Un graphe interactif de ce type peut être manipulé sous Rstudio ou inséré dans un rapport en format html.

Comment utiliser la fonction `esquisser()` du package `esquisse` pour obtenir une aide pour réaliser un graphe avec `ggplot2` et récupérer le code R correspondant

Le package `esquisse` peut être utile notamment à ceux qui préfère les interfaces interactives à la programmation. Il permet, en utilisant le code ci-dessous, d'ouvrir une fenêtre interactive et de créer un graphe à l'aide des boutons proposés. Le code R correspodnant au graphe ainsi généré peut ensuite être récupéré (bouton en bas à droite de la fenêtre), pour l'insérer dans son rapport d'analyse. La **récupération de ce code est importante** notamment pour une question de **tracabilité** (pour savoir exactement comment le graphe a été réalisé) et aussi de **reproductibilité**, pour pouvoir le reproduire très facilement, notamment sur un jeu de données complété ou un autre jeu de données similaire.

```
require(esquisse)
esquisser(d) # l'argument est le jeu de données sur lequel on travaille
```

Liens utiles

- le "cookbook" de ggplot2 : https://r-graphics.org/
- la "cheat sheet" de ggplot2 : https://rstudio.github.io/cheatsheets/data-visualization.pdf