TP2 - Statistique avec R - Graphiques avec ggplot2

OBJECTIFS DU TP :

Le but de ce TP est de s'initier au **package** `ggplot2` pour la réalisation de graphiques sous R.

Dans ce TP, nous aborderons également la **manipulation de dates avec le package** `lubridate`. Le 1er exercice montre un cas de "point levier" en regression. L'exercice 2 est particulièrement important. Il montre quelques exemples de générations de variables aléatoires et illustre un problème important en statistiques : l'erreur écologique.

> Astuce
>
> Des commentaires et des indications sont présentes en fin de TP pour vous aider dans certaines questions !

Travaillez soit sur le cluster R soit sur le Datalab/Onyxia. Vous pouvez aussi librement choisir de coder sur un document Quarto `.qmd` (au sein de *chunks* donc) ou sur un scrpit `.R` classique.

> Pour les tuteurs :
>
> Les étudiants ont créé un Rprojet `Stats_avec_R` au TP1, associé à leur repo Git personnel. Incitez-les à charger ce repo (via **File > New Project... > Version Control > Git** ou bien en suivant la procédure du TP1 : on **colle le lien https du repo à l'ouverture du service RStudio**, le token étant déjà renseigné).
> Il est impératif que les étudiants sachent créer un RProjet "classique" (via File > New Project > New Directory > New Project) pour le TP noté. Ainsi, nous leur imposerons d'envoyer sur Moodle un .zip contenant les données + le script (bien plus simple pour vos corrections).

Commencer par charger votre Rprojet `Stats_avec_R` créé au TP1, puis affecter un nouveau dossier "TP2 ggplot2". Pour importer vos données, cliquez sur le bouton `Upload` situé dans l'onglet `Files`, en dessous de votre environnement.

```
# Chargement des packages nécessaires

library(dplyr)
library(ggplot2)
library(lubridate)
library(plotly)
```

Présentation du package `ggplot2`

`ggplot2` est un package de visualisation de données très populaire sur `R`. Des livres entiers y sont dédiés (par exemple), des vidéos YouTube, notamment en raison de son caractère évolutif. Il fait partie de la suite *tidyverse* (tout comme *dplyr*), qui regroupe un ensemble d'outils puissants pour la manipulation et l'analyse de données.

Le principe fondamental de **ggplot2** repose sur la **grammaire des graphiques**, une approche qui définit un graphique comme une combinaison d'éléments distincts, permettant une grande flexibilité dans la création de visualisations, et rendant possible une infinité de représentations.

> Grammaire des graphiques
>
> L'idée de base de `ggplot2` est de décomposer un graphique en plusieurs composants :
> - **Données** : informations que vous souhaitez visualiser.
> - **Mapping esthétique (`aes`)** : l'assignation de variables aux axes et autres éléments visuels comme la couleur ou la taille.
> - **Géométries (`geom_...`)** : formes qui seront utilisées pour représenter les données, telles que des points (`geom_point`), des barres (`geom_bar` ou `geom_histogram`), des lignes (`geom_line`), une boîte à moustaches (`geom_boxplot`), etc. Souvent nommées couches (*layer*), il est possible d'en superposer plusieurs.
> - **Facettes** : la possibilité de diviser les graphiques en sous-graphes selon une ou plusieurs variables.
> - **Thème** : contrôle de l'apparence du graphique, comme les couleurs de fond, les axes, etc.
>
> La fonction principale de **ggplot2** est `ggplot()`, c'est le 1er composant. Elle est utilisée pour créer/initialiser un graphique en partant des données. On enchaine ensuite avec au moins un 2nd composant (`geom_...`) où l'on précise si l'on veut une représentation de type nuage de points, diagramme en bâton, etc.
> Chaque composant, mais aussi chaque couche, s'enchaine avec l'opérateur '+' (le + est une sorte de pipe pour 'ggplot2').

```
library(ggplot2)
# Exemple 1 : nuage de points

# 1er composant :
ggplot(data = mpg, # mpg (Miles Per Gallon) : données  sur la consommation d'essence des véhicules
       # aes() : on déclare quelle variable est sur l'axe des x/des y, tailles, couleurs, etc.
       mapping = aes(x = displ, y = hwy)) +

  # 2nd composant
  #  geom_...() : couche (layer) décrivant comment représenter les données.
  geom_point() +

  # labs() : ajout d'un titre et des étiquettes aux axes.
  labs(title = "Consommation de carburant vs Volume total des cylindres moteur",
       x = "Volume total des cylindres (L)",
       y = "Autonomie autoroute (mpg)")
```

```
# Exemple 2 : nuage de points avec des facettes (un graphique pour chaque type de voiture),
#   en coloriant les points selon la variable `class`.
ggplot(mpg,
       # On peut ajuster les couleurs des points, lignes, etc., en utilisant
       #  les arguments `color`, `fill`, ou `alpha` (transparence).
       aes(x = displ, y = hwy, color = class)) +
  geom_point() +
  # `facet_wrap()` ou `facet_grid()` : diviser un graphique en sous-graphes par catégories
  facet_wrap(~ class) +
  theme_minimal() +
  labs(title = "Comparaison de la consommation de carburant selon les types de voiture")
```

Vous trouverez facilement beaucoup de documentation sur ce package. Vous avez par exemple ce lien. Il existe également une fiche synthétique cheatsheet présentant les principales fonctions

Exercice 1

0. Il est fréquent que les packages contiennent des bases de données. Faire apparaître la liste des tables disponibles dans le package `dplyr` en lançant la commande suivante :

```
data(package="dplyr")
```

1. Nous allons travailler (pas longtemps) sur la table starwars. Pour cela charger la table avec l'instruction suivante et afficher son contenu. Que remarquez-vous dans le format des variables ?

```
data("starwars")
str(starwars)
```

2. Nous voulons d'abord étudier la relation entre le poids (mass) et la taille (height). Commencer par enlever toutes les valeurs manquantes de la table pour ces 2 variables. Vous ferez cela avec la syntaxe `dplyr`.
3. Représenter la variable mass (ordonnées) en fonction de la variable height (abscisses), sous forme d'un nuage de point. **Faire cela en R base.**
4. Faire le nuage de points en utilisant `ggplot` (fonction principale de `ggplot2`). Pour cela, adapter le code suivant en complétant les …

```
ggplot(data=...,
       aes(x = ..., y= ... ))+
  geom_...
```

6. Faire la regression linéaire simple de mass (variable à expliquer) en fonction de height (variable explicative). Vous utiliserez la fonction `lm()`. Vous pouvez vous reporter aux indications en dernière page du TP. Stocker cela dans un objet `reg_Q6`.
7. Afficher ensuite un résumé de la regression précédente en utilisant la fonction `summary()` sur votre objet. Combien vaut le R^2 ? Le modèle semble-t-il bon ?
8. Retirer l'observation qui semble aberrante (au vu des graphiques) puis refaire le nuage de points entre `mass` et `height.`
9. Maintenant que l'observation aberrante a été retirée, refaire la regression linéaire simple entre `mass` et `height.` Le modèle semble t-il meilleur ?
10. Il est possible qu'au sein de notre population, la relation entre `mass` et `height` dépendent d'autres facteurs. Compléter le nuage de points précédent (fait avec `ggplot`) en coloriant les points selon la variable `species.`
11. Il y a beaucoup d'espèces, et ne sachant pas comment les regrouper (pas fan de starwars), on essaye avec une autre variable. Essayer en coloriant avec la variable `birth_year`. Vous rajouterez également un paramètre de forme qui dépendra de la variable `sex`.
12. Finalement, vous vous arrêtez sur un graphique avec un coloriage en fonction de la variable `sex` et c'est tout (pas de 4e variable). Au passage rajouter un titre à votre graphique, une source et un thème différent : fond blanc plutôt que gris. Vous pouvez à nouveau utiliser `plotly` pour avoir un rendu final dynamique.

Exercice 2

Le but de cet exercice est de sensibiliser à un problème courant en statistiques : **l'erreur écologique**.

Nous allons travailler sur une simulation de 1000 observations (générées aléatoirement grâce aux fonctions `r...()` où `...` représente le nom de la loi). Pour le contexte : nous étudions l'influence de l'éducation (`Education`) et du niveau de revenu (`Salary`) sur le neuroticisme [1] (`Neuroticism`).

0. Commencer par lancer le code suivant pour fixer une "graine". Générer un 1er vecteur `Education` composé de 1000 variables aléatoires $X_i \sim B(2, 0.5)$

```
set.seed(2026)
```

1. Créer deux vecteurs (gaussien) Y1,Y2 composés de 1000 réalisations $Y_i \sim N(0, 1)$.
2. Créer un vecteur `Neuroticism` tel que $Neuroticism = Y1 + Education$. Créer également un vecteur "Salary" tel que $Salary = 2 * Education + Y2 - Neuroticism * 0.1$. Convertir ensuite le vecteur `Education` en factor avec les labels suivants ("Low", "Medium", "High"). Ces labels seront associés aux levels ("0","1","2") respectivement.
3. Créer un `tibble` (le `data.frame` de `dplyr`) à partir des 3 vecteurs précédemment calculés.
4. À l'aide de ggplot, représenter la variable `Salary` en fonction de `Neuroticism` sous la forme d'un nuage de points. À partir des options de `ggplot`, ajouter également la droite de regression liant ces 2 variables. Quel est le coefficient de correlation entre les 2 variables ?
5. Refaire le graphique précédent en colorant les observations selon la variable `Education`. Que constatez-vous ?

Exercice 3

1. On va cette fois utiliser des tables issues du package `ggplot2`. Afficher les tables de ce package.
2. Charger les tables `economics` et `presidential`, puis afficher leur contenu. Vous trouverez le dictionnaire des variables de `economics` ici
3. Trier la table `economics` selon la variable de date (en `dplyr`) dans l'ordre décroissant
4. Créer une variable `annee`.
5. Les années sont-elles toutes bien renseignées ? Afficher les années ne comportant pas 12 mois. Faites cela avec la syntaxe `dplyr`.
6. Dans la question précédente, nous avons compté le nombre de mois par année en considérant que s'il y en avait 12, l'année était bien renseignée. S'assurer qu'au sein de chaque année, il n'y a pas 2 fois le même mois.
7. Combien de jours séparent la date la plus ancienne de la plus récente ? La date la plus ancienne à juillet 2007 ?

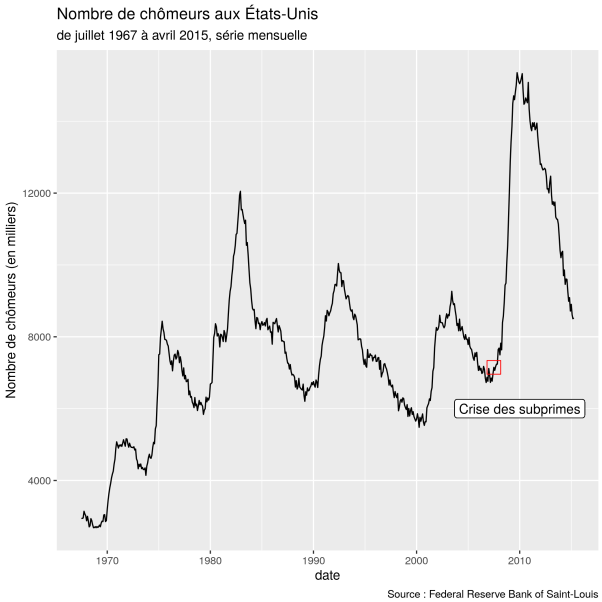
```
max(economics$date)-min(economics$date)
ymd("2007-07-01")-min(economics$date)
```

8. Représenter le nombre de chômeurs (série `unemploy`) au cours du temps. Vous utiliserez la fonction ggplot(). Privilégiez des lignes plutôt que des points. Donner lui un titre et une source ("Federal Reserve Bank of Saint-Louis")
9. Compléter votre code de la question précédente avec le code suivant :

```
geom_point(data = economics %>%
           filter(date=="2007-07-01"))
```

10. Adapter votre code pour entourer en rouge la date de juillet 2007. Vous mettrez également une étiquette précisant "Crise des subprimes". Vous devez ainsi retrouver le graphique suivant :

1. Le neuroticisme (ou névrosisme, névrotisme, neurotisme) est un trait de personnalité normal et universel qui se caractérise par une sensibilité accrue aux émotions négatives.



11. On souhaite savoir qui était le président et quel était le parti au pouvoir à chaque date. Faire la jointure avec la table `presidential` pour récupérer l'information. Pour cela, vous aurez besoin du package `fuzzyjoin`.
12. Représenter le nombre de chomeurs tel que fait précedemment en colorant selon le parti au pouvoir.
13. Constituer une table annuelle qui ne conservera que les variables numériques et l'année. Chaque variable numérique aura pour valeur sa moyenne sur l'année. Faites cela en `dplyr` (3 lignes de code suffisent).
14. Rajouter une variable de taux de croissance annuel des dépenses de consommation (`gr_pce`).

COMMENTAIRES/INDICATIONS

Exercice 1

2. Vous pouvez utiliser les fonctions `filter()` et `is.na()`
3. La fonction `lm()` s'utilise de la manière suivante : `lm(data=ma_table, var_expliquee~var_explicative)`
4. Utiliser l'argument `color =` au sein de la fonction `aes()`
5. Pour rajouter un paramètre de forme selon une variable, il existe l'argument `shape =`.
6. Vous pouvez utiliser la fonction `labs()` en couche supplémentaire (à rajouter à votre code avec un `+`). Dans cette fonction, les paramètres à utliser sont : `title =`, `x =`, `y =`, `caption =`. Pour la modification du thème (fond blanc), vous pouvez utiliser `theme_bw()`.

Exercice 2

Cet exercice illustre le problème d'erreur écologique (ecological fallacy). Un autre problème est le paradoxe de Simpson, problème assez similaire à celui de l'erreur écologique : tirer des conclusions fallacieuses à partir de données agrégées. Ayez toujours en tête ce risque, notamment pour vos projets statistiques.

Exercice 3

Le but de cet exercice est aussi de manipuler des dates. Cet article présente quelques subtilités liées à leur manipulation. Vous pouvez aussi lire ce blog. Enfin, vous trouverez le cheatsheet du package lubridate ici

3. Vous pouvez utiliser la fonction `year()` et `month()` du package `lubridate.`
4. Cela revient à compter le nombre de mois par année, puis à afficher les années n'ayant pas 12 mois.
5. Cela revient à compter les couples (année, mois). Chaque couple doit être unique.
6. Pour calculer une période en nombre de jours, on fait une simple soustraction.
7. `geom_point()` c'est pour faire des points, pour faire des lignes c'est…
8. Pour faire un carré, on utilise l'option `pch=0`. Chaque forme (carré, triangle, cercle, etc…) est désignée par un numéro de `pch` particulier. Vous pouvez régler la taille avec l'option `size`.

Enfin les étiquettes peuvent se faire avec la fonction `geom_label()`. Pour la position de l'étiquette, vous pouvez utiliser les coordonnées x et y. Attention pour la valeur du x : cette valeur est un nombre de jours.

11. Comme pour (quasiment) tous les problèmes que vous rencontrerez en programmation, il existe une solution : taper "join on period r" sur google. Le package `fuzzyjoin` est un package permettant de faire des jointures malgré des correspondances approximatives.
12. Rajouter l'option `group=1` pour ne pas avoir 2 courbes.