

# Analyse exploratoire des données

De la théorie à la pratique (TP 1)

Jacques Zuber

This version: novembre 2, 2024

## Préambule

Le logiciel de statistique qui sera utilisé dans les travaux pratiques est **R**, logiciel libre distribué sous les termes de la *GNU, General Public Licence*, au site web du **CRAN** (*Comprehensive R Archive Network*). 

(a) **R**.

Ce logiciel est disponible pour les systèmes d'exploitation Linux, Windows et Mac OS X. Des exécutables précompilés de la version actuelle **R-4.4.1** (Race for Your Life) sont disponibles sur l'un des miroirs du CRAN. Les instructions à suivre pour les installer s'y trouvent.

Pour faciliter votre apprentissage du logiciel, Emmanuel Paradis et Julien Barnier ont écrit de bonnes documentations françaises pour **R**, “**R** pour les débutants” et “Introduction à **R**”, qui se trouvent dans la page [Moodle du cours](#).

Le logiciel de statistique **R** fonctionne principalement par commandes. L'attente de commandes, par défaut le symbole **>**, apparaît au démarrage du logiciel et indique que **R** est prêt à exécuter les commandes. Sous Windows, en utilisant l'interface **R-GUI** de **R**, certaines d'entre elles (par exemple accès à l'aide et ouverture de fichiers) peuvent être exécutées par les menus.

Le logiciel **R**, créé vers 1994 par Ross Ihaka et Robert Gentleman de l'Université d'Auckland, est davantage qu'un simple logiciel de statistique. Il s'agit non seulement d'un outil d'analyse statistique et graphique mais aussi d'un langage reposant sur le langage **S** créé par AT & T Bell Laboratories. John M. Chambers, l'un des créateurs de **S**, a reçu en 1998 le *Software System Award* de la prestigieuse ACM (“Association for Computing Machinery”).

WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH | SPORTS | OPINION

**Search Technology**  Go **Inside Technology**

Internet | Start-Ups | Business Computing | Companies Bits  
Blo

## Data Analysts Captivated by R's Power



Stuart Isett for The New York Times

R first appeared in 1996, when the statistics professors Robert Gentleman, left, and Ross Ihaka released the code as a free software package.

Figure 2: Tiré du “New York Times” du 6 janvier 2009.

Les possibilités offertes par **R** sont vastes et permettent à l'utilisateur d'effectuer des analyses de données très pointues. Le logiciel est reconnu pour sa flexibilité. En effet, les résultats d'une analyse sont stockés dans un "objet"; il est alors possible de n'afficher que la partie des résultats qui intéresse l'utilisateur. Cette facilité n'est pas offerte par tous les logiciels classiques. Notons que toutes les actions de **R** sont effectuées sur les objets présents dans la mémoire vive de l'ordinateur. Aucun fichier temporaire n'est utilisé. Pour mieux comprendre le fonctionnement de **R**, il est fortement recommandé de lire le chapitre 2 de l'aide d'Emmanuel Paradis (2005).

Le logiciel de statistique **R** nécessite un apprentissage qui peut paraître pénible et difficile en raison du recours aux commandes plutôt qu'aux menus déroulants.

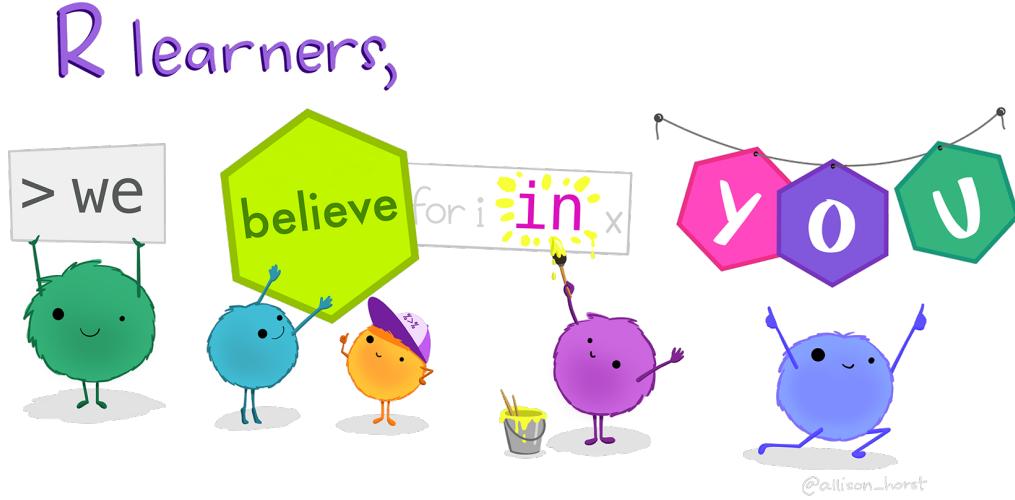


Figure 3: Dessin de Allison Horst.

Rassurez-vous, après s'être rapidement familiarisé avec quelques notions et concepts de base, l'utilisateur pourra employer efficacement le logiciel dont le fonctionnement reste finalement très intuitif. De plus, les commandes vous offrent un horizon de possibilités bien plus large que celui des menus déroulants. Un aide-mémoire des principales commandes de **R** figure dans le fichier “aide\_memoire.pdf” qui se trouve dans la page [Moodle du cours](#).

Une aide en ligne existe directement dans **R**. Elle est très utile pour connaître l'utilisation des fonctions du logiciel. Plusieurs méthodes existent pour y accéder : la première en utilisant la commande

```
help("mean")
```

Une deuxième possibilité consiste à utiliser l'alias de la commande `help()`, un point d'interrogation, `?mean`, et finalement, une dernière variante revient à utiliser simplement le menu de l'interface **R-GUI** de **R**.

Un moteur de recherche pratique pour obtenir une aide supplémentaire et complète sur **R**, ses fonctions, ses librairies complémentaires et la programmation dans **R** est [Rseek.org](http://Rseek.org). L'utilisation de **R** peut aussi être facilitée en utilisant le [Quick-R](#).

Pour les utilisateurs de Linux, Windows et Mac OS X, il existe un éditeur, **RStudio IDE** (*Integrated Development Environment*), encore plus convivial que celui que vous propose **R** par défaut. Il vous permet d'écrire et conserver des scripts dans lesquels figure une suite de commandes qui seront exécutées successivement. Vous pourrez lancer les scripts directement depuis l'éditeur sans avoir besoin de procéder à un “copier-coller”. Les scripts de commandes peuvent être archivés et accessibles à tout moment. Il est également possible d'afficher simultanément plusieurs fichiers contenant différents scripts et passer aisément de l'un à l'autre. La possibilité d'écrire des scripts, de les archiver, de les exécuter plusieurs fois en des temps différents est indéniablement un avantage par rapport à ce que vous proposent les logiciels à menus déroulants.

**RStudio**<sup>1</sup> vaut la peine d'être installé pour ses nombreux avantages. Cependant, si vous préférez, vous pouvez utiliser **Visual Studio Code** ou encore **Positron** le tout dernier qui est encore à sa version de développement. **Positron** qui se trouve à la croisée de **R** et de **Python** dans le vaste univers de la science des données vous permet de passer aisément de l'un à l'autre des deux langages.

Il est aussi possible d'écrire de manière simple ses propres fonctions. Sans entrer dans les détails, une fonction **R** est écrite dans un fichier sauvé au format ASCII  avec extension **.R**. Comme les autres langages, **R** possède des structures de contrôle qui ne sont pas sans rappeler celles du langage **C**.

Lorsque vous terminez votre session **R**, n'oubliez pas d'en sauver une image. Elle vous permettra de conserver les objets et de récupérer les dernières commandes utilisées. Dans l'interface **R-GUI** de **R**, d'autres options très pratiques vous sont offertes comme par exemple charger et sauver l'environnement de travail (utile si vous travaillez sur plusieurs projets distincts), charger et sauver l'historique des commandes.

L'étudiant *doit rendre un rapport* du travail pratique dans lequel figurent les réponses aux questions posées ainsi que les graphiques tracés. Il devra le rendre individuellement dans la page [Moodle du cours](#) avant la date butoir. Le rapport comptera le **15 %** du deuxième travail écrit. L'étudiant sera aussi interrogé sur les travaux pratiques aux travaux écrits.

Les rapports doivent être rédigés à l'aide de **quarto®**, système calligraphique libre conçu notamment pour écrire des documents scientifiques. Il se base sur **Pandoc**, logiciel libre permettant la conversion de documents en ligne de commande.

Le fonctionnement de **quarto®** pour créer un document est résumé dans la figure ci-dessous.

---

<sup>1</sup> **RStudio**; la société qui développe l'interface s'appelle **Posit PBC** depuis novembre 2022.

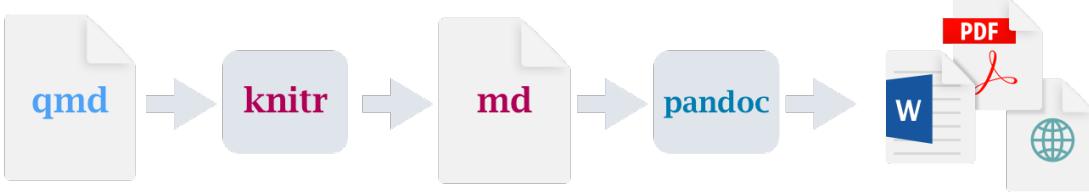


Figure 4: Source : quarto®.

Sans entrer dans les détails, quarto®<sup>2</sup> permet de générer des documents de manière dynamique en mélangeant texte et résultats obtenus à l'aide du code **R**. Les documents créés peuvent être notamment au format HTML, PDF, Word. Ainsi, quarto® est un outil très pratique pour exporter, communiquer et diffuser des résultats d'analyses statistiques. La librairie sous-jacente et fort élégante de **R** pour compiler, ou plutôt “tricoter” (*knit* en anglais), un document quarto® (.qmd) afin de visualiser le document généré est **knitr**. Elle crée entre autres un fichier .html dans lequel sont insérés les commandes, les sorties ainsi que les graphiques tracés. Vous avez entre vos mains ou à l'écran un document confectionné à l'aide de **knitr**<sup>3</sup>.

La librairie **knitr** exécute les morceaux de code de **R** contenus dans le document quarto® que vous avez écrit et crée un document **markdown** (.md) qui contient le code et sa sortie. Brièvement, **Markdown** est un langage de balisage très commode. Pandoc se charge ensuite de transformer le fichier *markdown* en un document de format désiré, comme par exemple en format .html, .pdf et Microsoft® Word.

L'objectif de ce travail pratique consiste à se familiariser avec les commandes de base du logiciel **R**, à utiliser les techniques d'analyse pour une variable statistique puis deux et finalement à apprendre à rédiger des rapports d'analyse de données.

En route !

## Exercice 1

L'une des forces de **R** est qu'il est capable de traiter de grands jeux de données de manière très rapide. Cet avantage a évidemment son prix : la lecture des données peut paraître ennuyeuse, particulièrement lorsqu'elles se trouvent sur support informatique. Il existe cependant plusieurs possibilités pour lire dans **R** des données contenues dans un fichier. Si elles se présentent sous la forme d'une liste de valeurs telles que chacune d'elles figure sur une ligne ou si elles sont séparées par un espace, on peut utiliser la commande **scan()** qui renvoie un vecteur. Lorsque les données se présentent sous la forme d'une table, i.e. une ligne par observation et une colonne

<sup>2</sup>Mickaël Canouil a réuni de nombreuses contributions, notamment des modèles, à l'adresse [quarto](#). Elles nous permettent de se familiariser avec le système calligraphique.

<sup>3</sup>Des tutoriaux, démonstrations et exemples se trouvent à l'adresse [knitr](#).

par variable, l'instruction à utiliser est `read.table()` si les données se trouvent dans un fichier texte (ASCII). Des variantes de cette fonction existent comme par exemple `read.csv2()`<sup>4</sup>. Dans cet exercice, nous allons enregistrer dans **R** les données qui seront utilisées dans le travail pratique. Vous pouvez organiser comme vous le souhaitez votre travail. Néanmoins, nous vous suggérons de créer deux répertoires : l'un contenant les données et l'autre votre travail (script, rapport, résultats). Pour être plus structuré, vous pouvez même ajouter un sous-répertoire à votre répertoire de travail pour y stocker vos graphiques; vous pouvez également créer un projet pour mieux gérer votre travail pratique.

a. Les données que nous allons traiter dans ce travail pratique se trouvent dans la page [Moodle du cours](#). Copiez-les dans votre répertoire de données.

b. Charger les données dans **R** en utilisant les fonctions `scan()` et `read.table()`.

Les utilisateurs se chargeront d'adapter les chemins à leur répertoire de travail et à leur système d'exploitation. Les fichiers `cpus.txt` et `examen.txt` sont ainsi accessibles dans **R** sous les noms `cpus` et `examen` respectivement.

c. Pour voir le contenu de l'objet `cpus`, taper l'instruction

```
cpus
```

```
[1] 62 130 11 12 144 16 133 26 32 7 24 19 24 66 915 92 10 45 12  
[20] 58 33 38 38 25 110 40 370 185 22 18 136 44 36 76 66 56 141 64  
[39] 36 510 30 100 14 60 46 65 36 21 66 24
```

Il en est de même pour `examen`. Les objets `cpus` et `examen` sont de nature toute différente. En effet, le premier est un *vecteur*, le second un tableau de données, *data.frame*<sup>5</sup> en anglais.

d. Pour accéder à la 19ème composante du vecteur `cpus`, utiliser la commande

```
cpus[19]
```

```
[1] 12
```

e. Pour obtenir une partie du vecteur `cpus` comme par exemple les éléments du vecteur compris entre la 5ème et la 21ème composante, taper l'instruction

```
cpus[5:21]
```

```
[1] 144 16 133 26 32 7 24 19 24 66 915 92 10 45 12 58 33
```

---

<sup>4</sup>Pour de plus amples informations, voir “**R** pour les débutants”, E. Paradis, 2005, pages 12–16.

<sup>5</sup>Pour de plus amples informations, voir “**R** pour les débutants”, E. Paradis, 2005, pages 12.

**f.** Pour extraire du vecteur `cpus` ses éléments supérieurs à 200, utiliser la commande

```
cpus[cpus>200]
```

```
[1] 915 370 510
```

**g.** Il est possible d'accéder directement aux composantes d'une table par le nom. Par exemple, si on veut afficher la composante `note` de l'objet `examen`, on peut utiliser la commande

```
examen$note
```

```
[1] 3.5 5.4 5.0 3.2 3.7 2.6 4.6 4.2 4.9 5.2 4.2 4.0 5.5 5.8 5.3 5.2 4.3 4.4 5.1  
[20] 4.5 5.0 4.0 3.8 4.9 4.6 4.0 4.6 3.0 4.5 5.0 4.2 4.0 3.7 5.3 3.3 NA 4.0 4.9  
[39] 4.9 3.7 4.9 5.4 5.1 1.2 1.6 5.1 4.8 5.5 3.6 4.1 5.6 5.3 3.2 5.3 4.8 3.4 5.7  
[58] 5.6 4.3 4.9 4.5 3.3 4.0 4.0 5.6 3.6 5.2 4.6 4.6 2.6 3.9 4.1 4.0 5.2 5.0 4.7  
[77] 3.7
```

**h.** On peut aussi accéder en profondeur aux composantes comme par exemple par la commande

```
examen$note[9]
```

```
[1] 4.9
```

**i.** La méthode la plus simple pour créer un vecteur consiste à énumérer ses éléments à l'aide de la fonction `c()` :

```
mesdonnees<-c(2.9, 3.4, 3.4, 3.7, 3.7, 2.8, 2.1, 2.5, 2.6, 1.5)  
mesdonnees
```

```
[1] 2.9 3.4 3.4 3.7 3.7 2.8 2.1 2.5 2.6 1.5
```

```
couleurs<-c("bleu", "vert", "blanc", "noir", "jaune")  
couleurs
```

```
[1] "bleu"  "vert"  "blanc" "noir"  "jaune"
```

**j.** On peut ôter des composantes d'un vecteur en indiquant entre crochets les indices précédés du signe négatif comme par exemple

```
mesdonnees[-c(5:9)]
```

```
[1] 2.9 3.4 3.4 3.7 1.5
```

**k.** Finalement, le contenu de votre environnement de travail est affiché à l'aide de la fonction `ls()`.

```
ls()
```

```
[1] "couleurs"    "cpus"        "data_files"   "dataDir"      "examen"  
[6] "mesdonnees"
```

## Exercice 2

La performance relative au processeur *IBM 370/158-3* de 50 processeurs d'ordinateurs a été relevée.

62	130	11	12	144	16	133	26	32	7
24	19	24	66	915	92	10	45	12	58
33	38	38	25	110	40	370	185	22	18
136	44	36	76	66	56	141	64	36	510
30	100	14	60	46	65	36	21	66	24

L'objet `cpus` contient les valeurs observées.

**a.** Construire un diagramme branche-et-feuilles, une boîte à moustaches et un histogramme des données observées à l'aide des commandes ci-dessous.

```
stem(cpus)
```

```
par(mfrow=c(1,2), pty="s")  
boxplot(cpus, xlab="performance relative", col="darkslategray4", horizontal=T) (1)  
rug(cpus) (2)  
hist(cpus, xlab="performance relative", ylab="fréquence", main="",  
     col="darkslategray4")  
par(mfrow=c(1,1))
```

**(1)** Quels sont les effets de cette commande ?

**(2)** Quel est l'effet de la fonction `rug()` ?

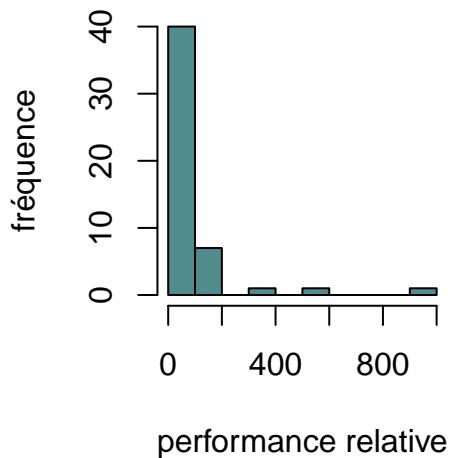
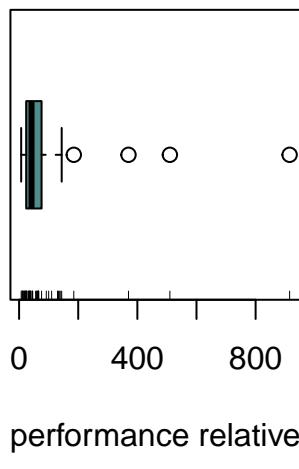


Figure 5: Boîte à moustaches et histogramme

Observer les résultats obtenus par chaque commande.

- b.** Commenter la distribution des valeurs observées en se basant sur les graphiques de la Figure 5 : valeur(s) atypique(s), asymétrie.
- c.** Calculer la performance relative médiane et la performance relative moyenne des valeurs observées en utilisant les fonctions de **R** adéquates.

Est-il plus approprié d'utiliser la médiane ou la moyenne ?

- d.** Déterminer le(s) mode(s) des valeurs observées à l'aide des commandes suivantes :

```
n.cpus<-table(cpus)
as.numeric(names(n.cpus)[n.cpus==max(n.cpus)])
```

- e.** Que fait la commande suivante ?

```
summary(cpus)
```

- f.** Décrire l'effet sur la moyenne et sur la médiane des trois interventions suivantes :

1. ajouter un processeur de performance relative 43;
2. soustraire 9 à chaque valeur observée;

3. diviser chaque observation par 3.

g. Calculer l'écart-type des performances relatives une fois avec les valeurs atypiques et une fois sans en utilisant la fonction `sd()`. Les valeurs atypiques peuvent être déterminées à l'aide de la fonction `boxplot()` avec `plot=FALSE` comme argument.

Que constate-t-on ? L'écart-type est-il un indicateur robuste ?

## Exercice 3

Les étudiants suivant un cours de Probabilités et Statistique dans une école d'ingénierie ont passé l'examen de fin d'unité. Le cours était donné par le même professeur à 77 étudiants répartis en deux groupes notés *A* et *B*. Les résultats obtenus figurent dans la table ci-dessous et sont contenus dans l'objet `examen`.

On se demande si une différence significative existe entre les deux groupes à l'examen.

a. Tracer les boîtes à moustaches en parallèle en utilisant les commandes suivantes :

```
lblue<-"#528B8B"  
par(pty="s")  
boxplot(note~groupe, data=examen, ylim=c(1,6), xlab="groupe",  
       varwidth=T, col=lblue, main="examen")  
abline(h=4, lty=2)
```

Table 1: Notes obtenues par les étudiants à l'examen de fin d'unité du cours de Probabilités et Statistique.

Table des notes selon les groupes

groupe	formation	mode_de_formation	somme	note
A	EE	PT	24.5	3.5
A	EE	PT	43.5	5.4
A	EE	PT	39.5	5.0
A	EE	PT	21.5	3.2
A	EE	PT	26.5	3.7
A	EE	PT	15.5	2.6
A	EE	PT	36.0	4.6
A	EE	PT	32.0	4.2
A	EE	PT	39.0	4.9
A	EE	PT	41.5	5.2
A	EE	PT	32.0	4.2
A	EE	PT	29.5	4.0
A	EE	PT	45.0	5.5
A	EE	PT	47.5	5.8
A	EE	PT	42.5	5.3
A	EE	PT	42.0	5.2
A	EE	PT	33.0	4.3
A	EE	PT	34.0	4.4
A	EE	PT	41.0	5.1
A	EE	PT	35.0	4.5
A	EE	PT	39.5	5.0
A	EE	PT	30.0	4.0
A	EE	PT	27.5	3.8
A	EE	PT	38.5	4.9
A	EE	PT	36.0	4.6
A	EE	PT	30.0	4.0
A	EE	PT	35.5	4.6
A	EE	PT	19.5	3.0
A	EE	PT	35.0	4.5
A	EE	PT	40.0	5.0
A	EE	PT	31.5	4.2
A	EE	PT	30.0	4.0
A	EE	PT	26.5	3.7
B	EE	PT	42.5	5.3
B	EE	EE	23.0	3.3
B	EE	PT	NA	NA
B	EE	PT	29.5	4.0
B	EE	PT	39.0	4.9
B	EE	PT	39.0	4.9
B	EE	PT	26.5	3.7
B	EE	PT	39.0	4.9
B	EE	PT	43.5	5.4
B	EE	PT	40.5	5.1
B	EE	PT	2.0	1.2
B	EE	PT	6.0	1.6

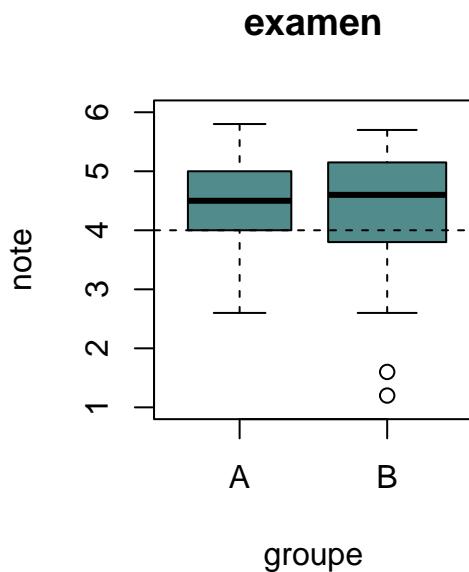


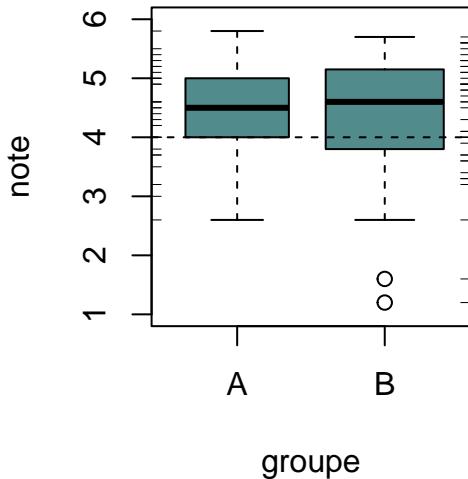
Figure 6: Boîtes à moustaches en parallèle.

- b.** Rajouter les bâtonnets des notes des étudiants des deux classes, sur le côté gauche des boîtes à moustaches pour la classe *A* (`side=2` comme argument de la fonction `rug()`) et sur le côté droit pour la classe *B* (`side=4` comme argument de la fonction `rug()`).

*Indication* : pour séparer puis enregistrer les notes des étudiants selon les groupes, on peut utiliser les commandes

```
note.A<-split(examen$note, examen$groupe)$A
note.B<-split(examen$note, examen$groupe)$B
```

## examen



c. En se basant sur la Figure 6, existe-t-il une différence significative entre les deux groupes à l'examen de fin d'unité ?

d. Observe-t-on sur les boîtes à moustaches une différence entre les dispersions des deux groupes ?

e. Calculer les écarts-types des deux groupes à l'aide des fonctions `by()` et `sd()`.

En se basant sur les écarts-types, existe-t-il une différence en dispersion entre les deux groupes à l'examen de fin d'unité ?

f. Que peut-on déduire en comparant les conclusions établies en c., d. et e. ?

g. Un autre graphique pour étudier les éventuelles différences entre les deux groupes à l'examen de fin d'unité se trouve dans la Figure 7.

À votre avis, entre les boîtes à moustaches en parallèle et le graphique tracé ci-dessus, lequel est le plus approprié ?

## Exercice 4

Une partie de la base de données du recensement américain<sup>6</sup> de 1994 a été extraite. Elle concerne 48'842 personnes adultes dont on s'intéresse notamment à l'influence que peut avoir

---

<sup>6</sup>Par intérêt, un coup d'œil à l'adresse du [Gouvernement Américain](#).

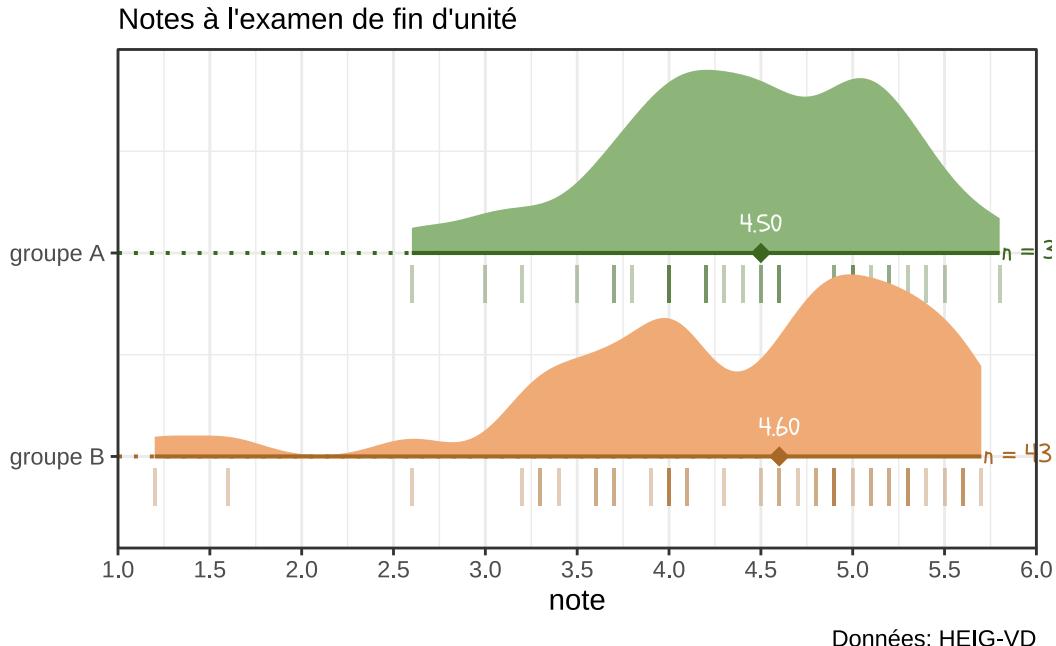


Figure 7: Diagramme de densité en parallèle.

le type de scolarité, formation acquise par l'individu, sur le nombre d'heures de travail par semaine. Par simplicité et pour préserver l'authenticité du système éducatif américain, le nom des variables n'est pas traduit en français.

a. Nous nous proposons de tracer les boîtes à moustaches en parallèle du temps consacré au travail par les individus recensés. Pour y parvenir, nous utilisons la librairie `ggplot2`<sup>7</sup> qu'il faut d'abord installer puis activer dans votre session.

La librairie `ggplot2` explicite les liens conceptuels entre graphiques et analyses statistiques. Sa syntaxe est particulière mais ingénieuse. Elle se base sur un ensemble de composants indépendants qui peuvent être combinés de différentes manières<sup>8</sup>.

Les données du recensement se trouvent dans la librairie `arules` de **R** qui doit être installée puis activée.

Les observations sont lues dans le logiciel à l'aide de la commande

```
data("AdultUCI")
```

et les variables qui nous intéressent sont sélectionnées et stockées dans l'objet `dframe` par les commandes

<sup>7</sup>Une librairie correspondante, `plotnine`, a été mise à disposition des utilisateurs de *Python*.

<sup>8</sup>De plus amples informations se trouvent dans [Tidyverse](#).

```
dframe<-AdultUCI[, c("education", "hours-per-week")]
colnames(dframe)<-c("education", "hours_per_week")
str(dframe)
```

①

① Pourquoi ce changement de nom de variable ?

```
'data.frame': 48842 obs. of 2 variables:
$ education      : Ord.factor w/ 16 levels "Preschool"<"1st-4th"<...: 14 14 9 7 14 15 5 9 15 ...
$ hours_per_week: int 40 13 40 40 40 40 16 45 50 40 ...
```

Tracer les boîtes à moustaches en parallèle de la Figure 8 dans lesquelles est représenté le temps hebdomadaire consacré au travail par les Américains recensés selon leur formation.

```
n=dim(na.omit(dframe))[1]
today<-format(Sys.time(), "%a %b %d %X %Y")

ggplot(dframe, aes(x=hours_per_week, y=education)) +
  geom_point(colour="lightblue", alpha=0.1, position="jitter") +
  geom_boxplot(outlier.size=0, alpha=0.2) +
  theme_bw(base_size=12) +
  theme(
    legend.position="none",
    plot.title=element_text(hjust=0.5, face="bold")) +
  labs(
    color="",
    fill="",
    x="hours per week",
    y="",
    title=paste("Census Income Database (n = ",n,")", sep=""),
    caption = today
  )
```

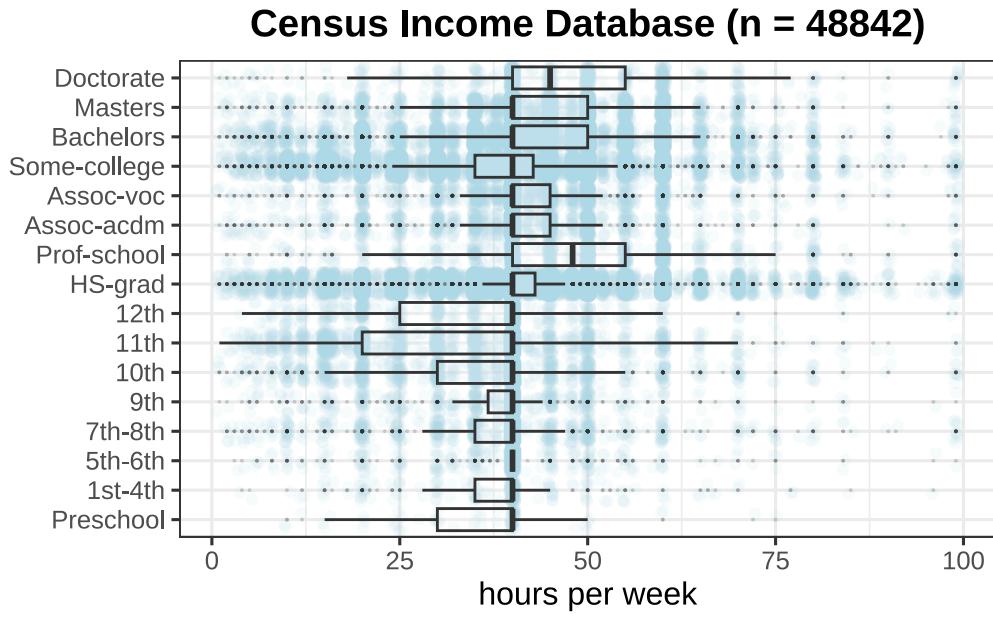


Figure 8: Boîtes à moustaches du temps de travail selon la formation.

Commenter le graphique obtenu.

- b.** Calculer la proportion d'observations contenant des valeurs manquantes en utilisant les commandes ci-dessous.

```
dim(AdultUCI)
nrows<-nrow(AdultUCI)
n.missing<-rowSums(is.na(AdultUCI))
sum(n.missing>0)/nrows
```

- c.** En se basant sur les boîtes à moustaches en parallèle de la Figure 8, pour quel type de formation observe-t-on la plus grande dispersion du temps de travail ? Existe-t-il une différence entre les médianes des types de formation ? En donner brièvement la raison.

- d.** Pour chaque type de formation, on peut déterminer puis afficher à l'écran le temps maximal de travail hebdomadaire à l'aide des commandes

```
nx<-by(dframe$hours_per_week, dframe$education, max, na.rm=T)
nx
```

La formation pour laquelle un temps maximal a été observé se détermine par les commandes

```
max(nx)  
names(nx) [nx==max(nx)]
```

Est-ce surprenant ?

- e. En s'inspirant des commandes utilisées ci-dessus, déterminer la formation pour laquelle la distribution des temps de travail se caractérise par le plus petit écart-type.
- f. Observe-t-on un résultat similaire en utilisant l'étendue interquartiles à l'aide de la fonction IQR()?

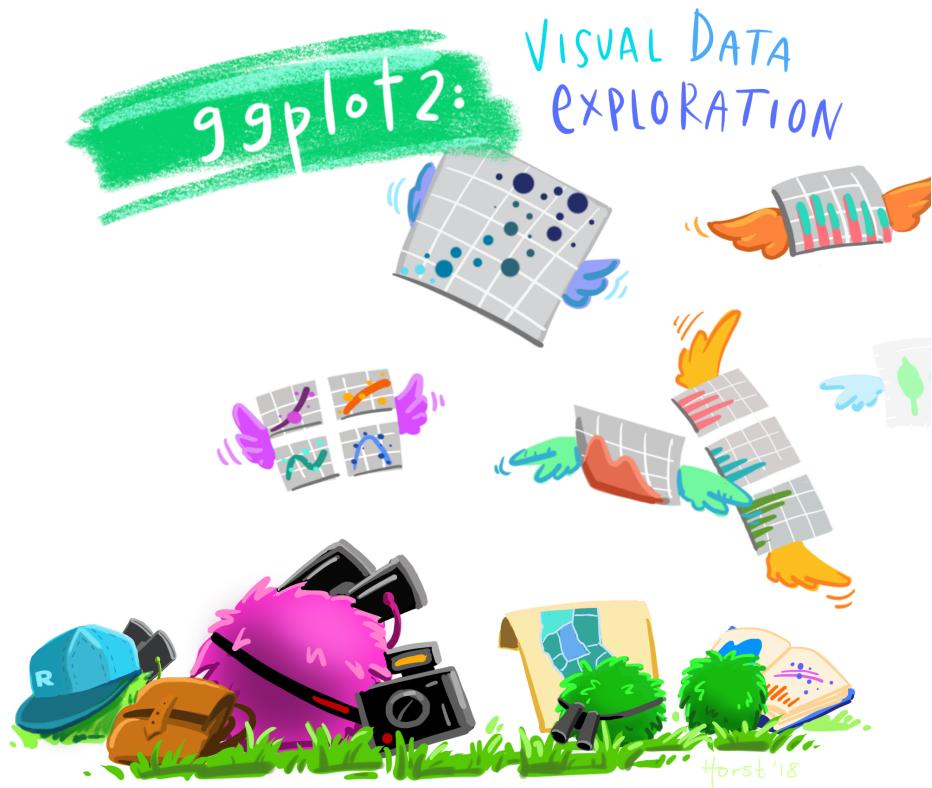


Figure 9: Les délices de la librairie ggplot2

## Exercice 5

Estimer et justifier les valeurs des coefficients de corrélation des séries de données à l'aide de leurs graphiques de nuage de points tracés dans la Figure 10, la Figure 11, la Figure 12 et la Figure 13.

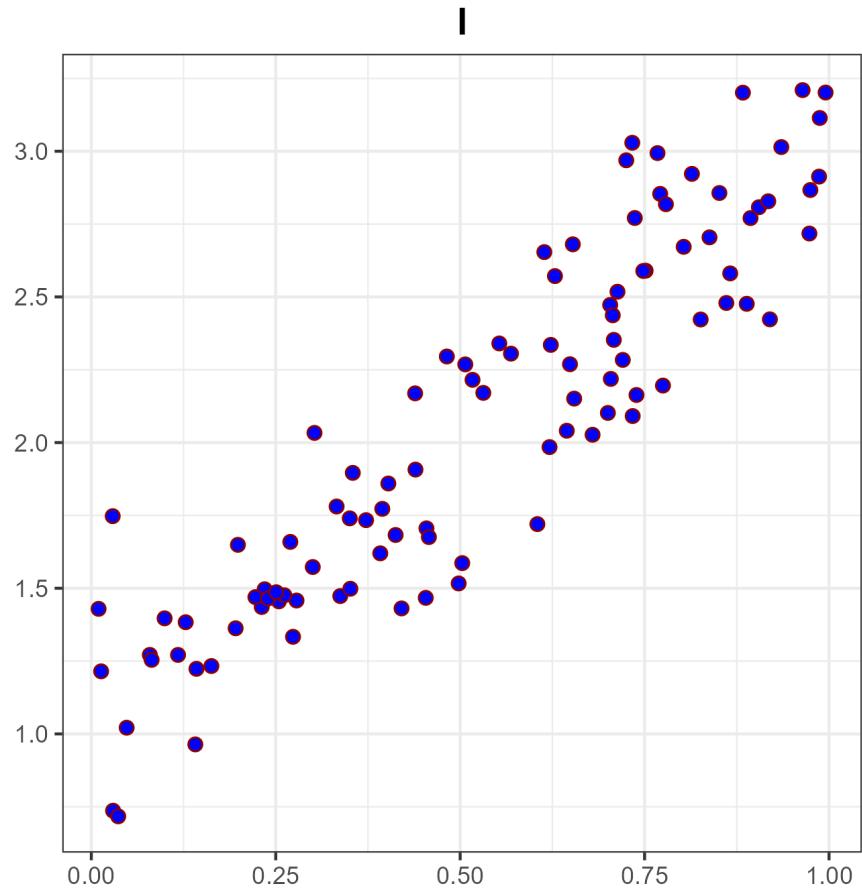


Figure 10: Graphiques de nuage de points I.

II

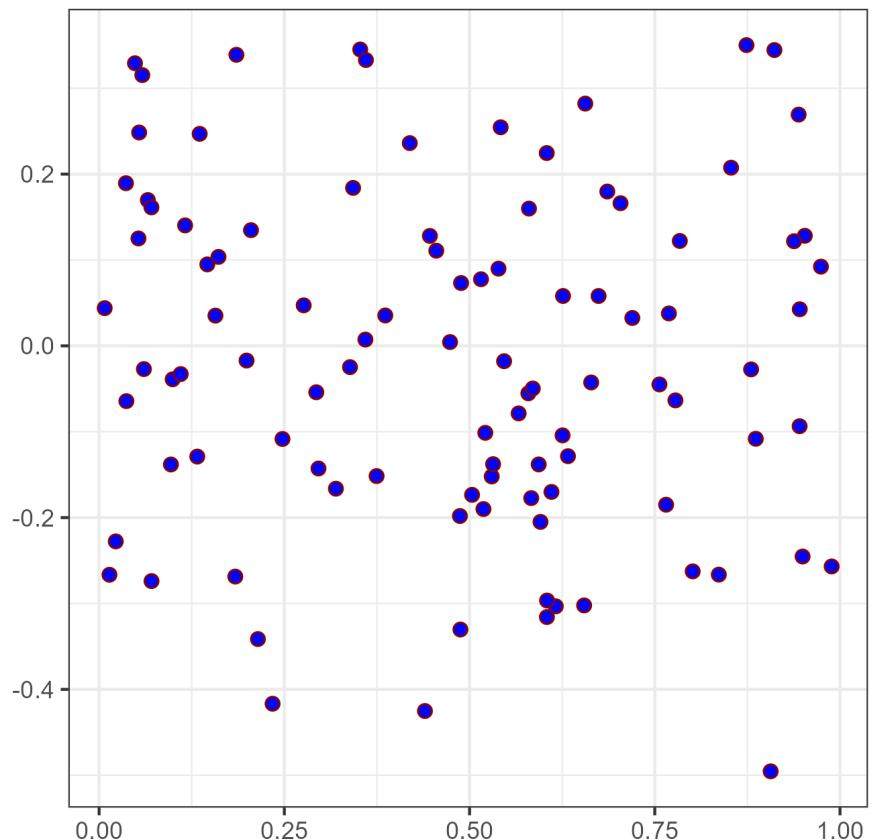


Figure 11: Graphiques de nuage de points II.

III

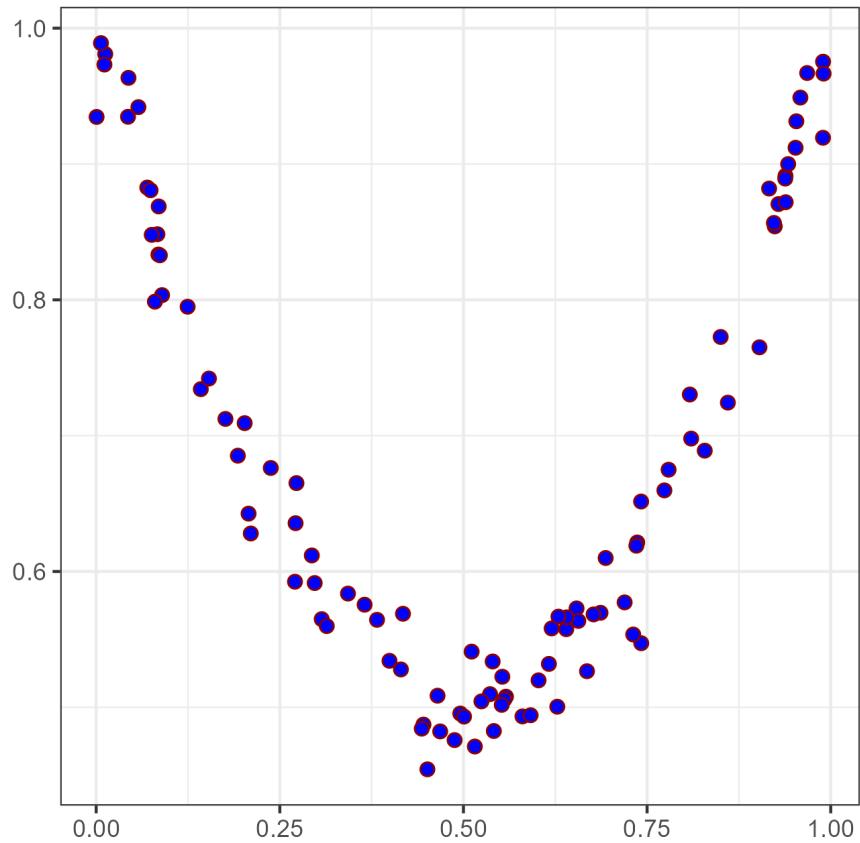


Figure 12: Graphiques de nuage de points III.

**IV**

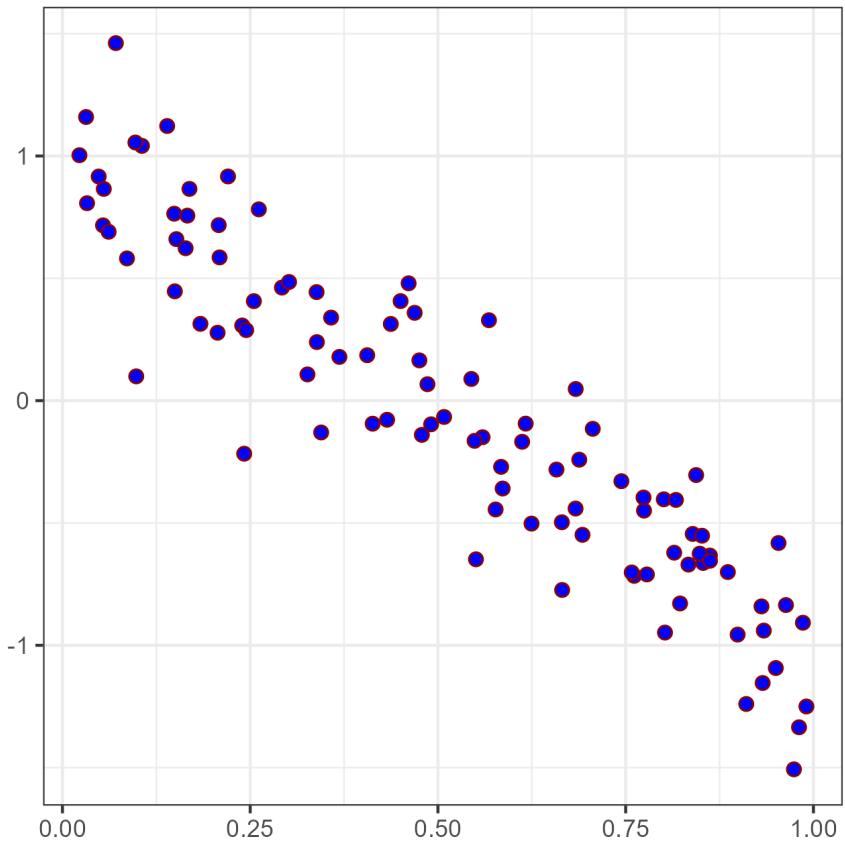


Figure 13: Graphiques de nuage de points IV.

## Exercice 6

Une étude a été réalisée en botanique sur 150 iris. Cinq variables ont été relevées : la longueur (`Sepal.Length`) et la largeur (`Sepal.Width`) des sépales, la longueur (`Petal.Length`) et la largeur (`Petal.Width`) des pétales, l'unité utilisée étant le centimètre, ainsi que l'espèce (`Species`) de la fleur (Setosa, Versicolor et Virginica).



Iris versicolor



Iris virginica



Iris setosa

Figure 14: Les trois espèces d'iris.

Ces données, qui avaient été récoltées par Edgar Anderson, se trouvent déjà dans **R**. Pour les utiliser dans votre session actuelle, il suffit de taper dans la console le nom de l'objet, `iris`, qui les contient.

- Utiliser les librairies `skimr` et `summarytools` pour afficher les sorties qui permettent

d'effectuer une analyse exploratoire des données observées. Pour la librairie `skimr` appliquer la fonction `tbl_summary()` à l'objet `iris`; appliquer la fonction `skim()` à l'objet `iris` pour la librairie `skimr`.

- b. La distribution de la largeur du sépale (`Sepal.Width`) est-elle plutôt symétrique ?
- c. La distribution de la largeur du pétale (`Petal.Width`) est-elle unimodale ou bimodale ?
- d. Tracer le nuage de points de la largeur (`Petal.Width`) versus la longueur (`Petal.Length`) des pétales des iris en utilisant les librairies `ggplot2` et `ggforce`.

```
pCol <- c('#057076', '#ff8301', '#bf5ccb')

plot.iris<-ggplot(iris, aes(x=Petal.Length, y=Petal.Width, col=Species)) +
  scale_color_manual(values=pCol) +
  scale_x_continuous(breaks=seq(0.5, 7.5, by=1), limits=c(0.5, 7.5)) +
  scale_y_continuous(breaks=seq(-0.5, 3, by=0.5), limits=c(-0.5, 3)) +
  labs(title="Edgar Anderson's Iris Data",
       x="Petal Length",
       y="Petal Width") +
  theme(plot.title=element_text(size=12, hjust=.5),
        axis.title=element_text(size=10, vjust=-2),
        axis.text=element_text(size=10, vjust=-2)) +
  geom_point(aes(color=Species), alpha=.6, size=3) +
  theme_minimal()

plot.iris +
  ggforce::geom_mark_ellipse(
    aes(fill=Species, label=Species),
    alpha=.15, show.legend=FALSE
  )
```

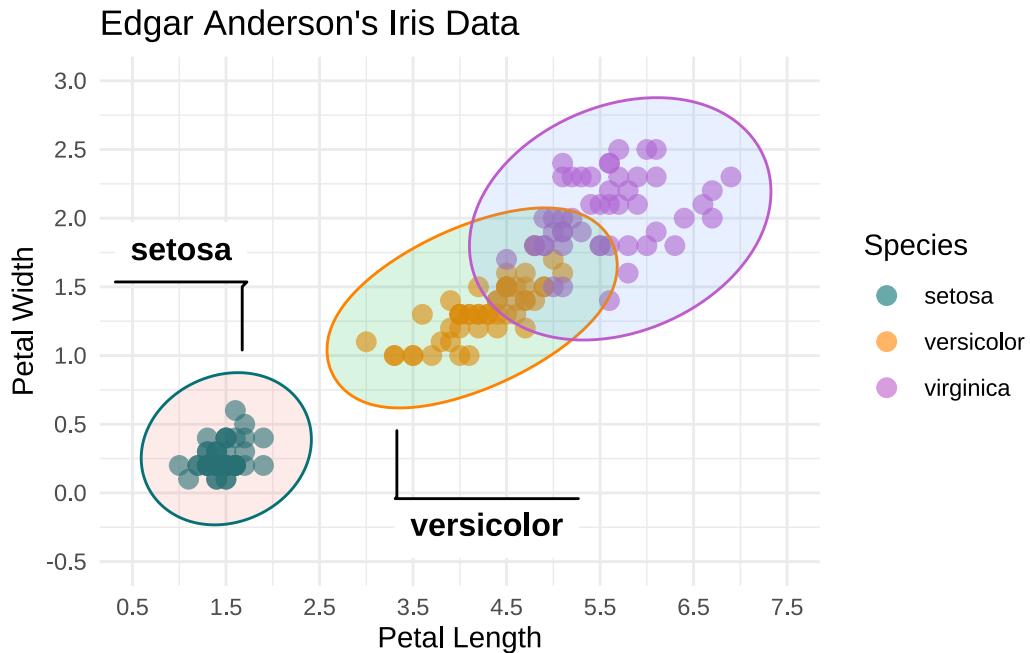


Figure 15: Graphique de nuage de points des iris

- e. En se basant sur le graphique de nuage de points, existe-t-il une relation entre la largeur et la longueur des pétales des iris ? Dans l'affirmative, de quelle nature est-elle ?
- f. Remarque-t-on des observations inhabituelles dans le graphique de nuage de points ?
- g. Déterminer la corrélation entre la largeur et la longueur des pétales des iris en utilisant la fonction `cor()`.
- h. Quelle valeur attribueriez-vous à la longueur des pétales des iris pour distinguer les iris Setosa des deux autres espèces ?
- i. Des animations peuvent être créées dans **R** en utilisant la librairie `ganimate`. Un exemple peut être conçu en utilisant le code ci-dessous.

```
library(ganimate)

anim<-plot.iris+
  transition_states(Species,
    transition_length = 2,
    state_length = 1)
```

```
anim
```

```

anim +
  enter_fade() +
  exit_shrink() +
  ggtitle('Now showing {closest_state}',  

          subtitle = 'Frame {frame} of {nframes}')

```

- j. (*En option bonus*) Installer la librairie **reticulate** qui permet de faire du *Python* à partir de **RStudio IDE**. Fixer ensuite l'interpréteur *Python* dans la rubrique *Python* de la boîte de dialogue *Options*. Cette boîte de dialogue s'affiche à l'écran en utilisant le menu *Tools* puis *Global Options...* de **RStudio IDE**.

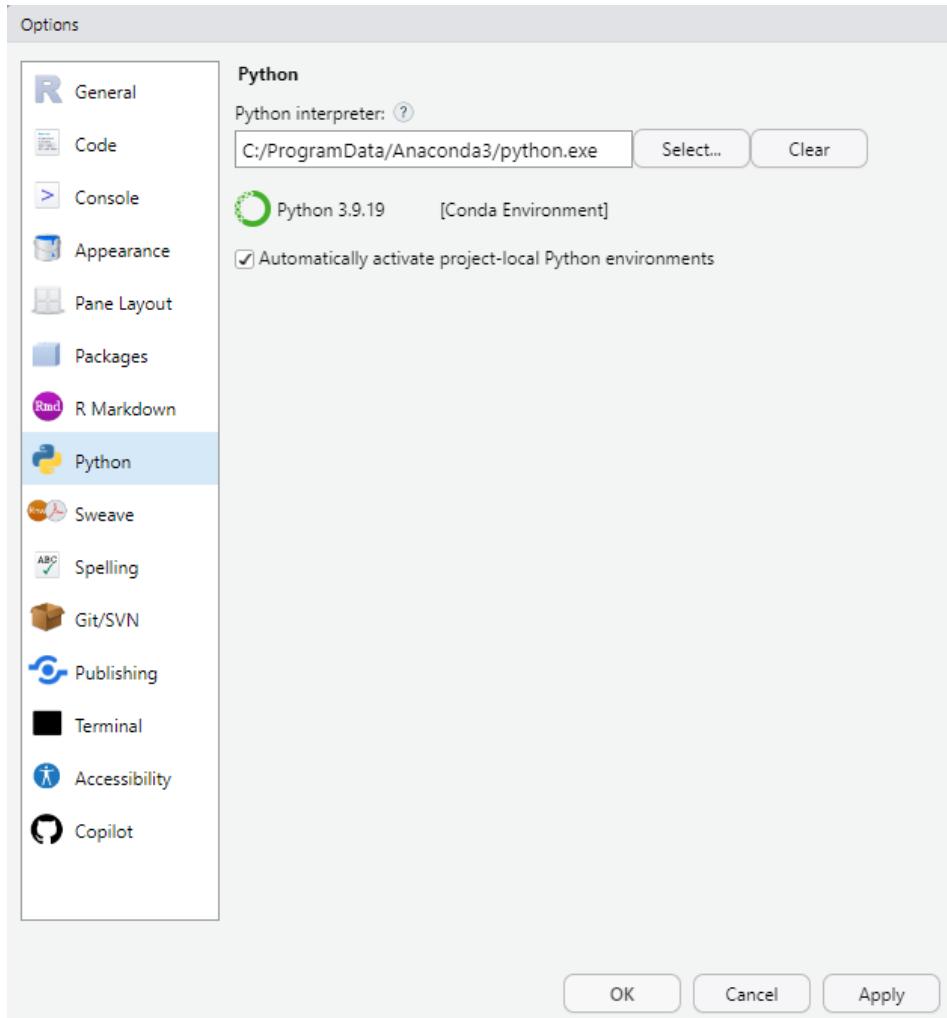
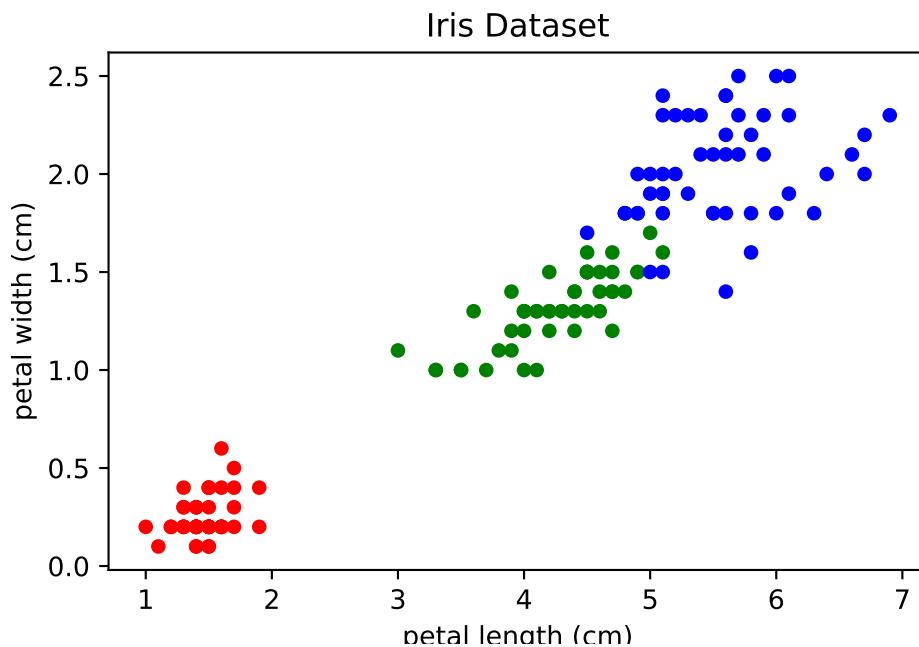


Figure 16: L'interpréteur Python de **RStudio IDE**.

La lecture de l'objet **iris** de **R** en *Python* s'effectue en utilisant la commande

```
iris_py = r.iris
```

Reconstituer le graphique ci-dessous en utilisant en particulier les librairies `pandas`, `numpy` et le module `matplotlib.pyplot` de la bibliothèque `matplotlib` de *Python*.



#### Le rapport doit

- être rédigé avec soin en utilisant *quarto®*;
- contenir une introduction dans laquelle se trouve les objectifs de l'analyse de données ainsi qu'une conclusion pour synthétiser le travail pratique;
- contenir les réponses aux questions posées;
- contenir les commandes de **R** utilisées, les résultats et graphiques obtenus;
- être rendu sur la page Moodle du cours en format .html ou .pdf avant la date butoir.