PARTIE 2

Visualiser les données décisionnelles

Dans ce module, vous allez :

- Créer divers types de graphiques
- Configurer les graphiques pour une analyse profonde

40 heures

CHAPITRE 1

CRÉER DIVERS TYPES DE GRAPHIQUES

Ce que vous allez apprendre dans ce chapitre :

- Utiliser efficacement les scatter plots
- Utiliser efficacement les bar charts
- Utiliser efficacement les histograms
- Utiliser efficacement les box plots
- Utiliser efficacement les bubble charts
- Pratiquer la création de graphiques avec Python/Excel/R

30 heures

CHAPITRE 1

CRÉER DIVERS TYPES DE GRAPHIQUES

1. **Scatter plots**
2. Bar charts
3. Histograms
4. Box plots
5. Bubble charts
6. Création de graphiques avec Python/Excel/R

PARTIE 2

Introduction sur la visualisation

La science de la visualisation des données consiste à représenter graphiquement les données. Elle permet de rendre les données complexes plus accessibles et compréhensibles, facilitant ainsi l'interprétation des tendances, et la détection des anomalies.

Concepts fondamentaux : Axes X et Y

Variable Indépendante (X) : La variable que vous **manipulez** ou contrôlez pour voir son effet sur une autre variable. Elle est placée sur l'axe des abscisses horizontal (X)

Variable Dépendante (Y) : La variable que vous **mesurez**, qui est influencée par la variable indépendante. Elle est placée sur l'axe des ordonnées vertical (Y)

Tracer Y en fonction de X : En géométrie, une équation linéaire peut être représentée graphiquement à l'aide des axes X et Y sous forme de ligne droite. Prenons un exemple pour mieux comprendre :

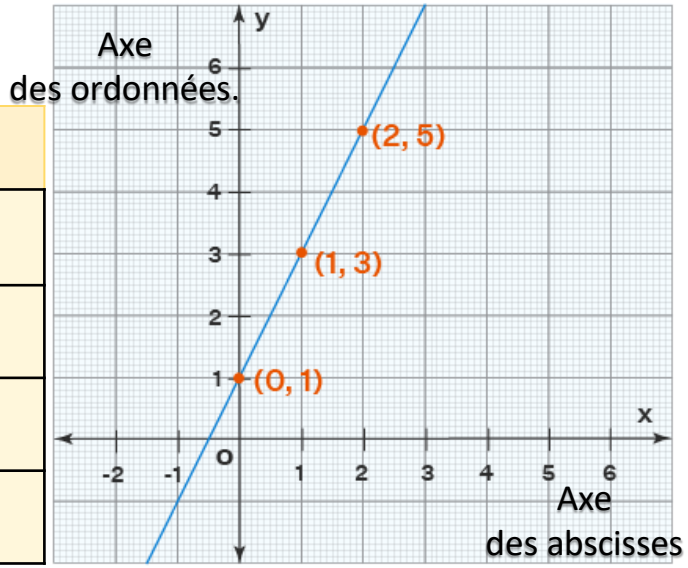
Considérons l'équation linéaire y = 2x + 1. Pour la tracer, construisons un tableau avec deux colonnes pour les valeurs de x et y.

Choisissons quelques valeurs pour la variable x et trouvons les valeurs correspondantes pour y.

Données :

x	y
0	1
1	3
2	5

Graph présent Y en fonction de X



Introduction

Exemple Pratique : Relation entre l'âge et le revenu

Formulation :

Nous étudions le **revenu (variable dépendante Y)** en fonction de **l'âge (variable indépendante X).**

Cela signifie que nous observons comment les variations de l'âge (variable indépendante) affectent le revenu (variable dépendante).

Revenu en fonction de l'âge

Interprétation :

Dans ce diagramme de dispersion (Scatter Plot), chaque point représente un individu avec son âge sur l'axe X et son revenu sur l'axe Y.
En observant ces points, on peut déterminer s'il existe une tendance ou une relation entre l'âge et le revenu.

PARTIE 2

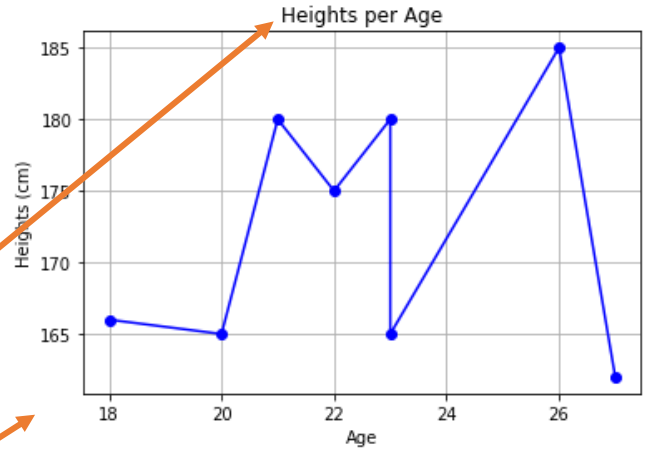
Scatter Plot (Diagramme de dispersion / Nuage de points) :

Un diagramme de dispersion est un type de graphique qui affiche des points de données individuels sur un axe bidimensionnel, montrant la relation entre deux variables.

Usage : diagramme de **dispersion** utilisé pour démontrer les **relations** entre deux variables (indépendante X, dépendante Y).

"**heights per age**" équivaut à dire "hauteurs **en fonction** d'âge".

Cela signifie que les hauteurs sont représentées en fonction de l'âge.

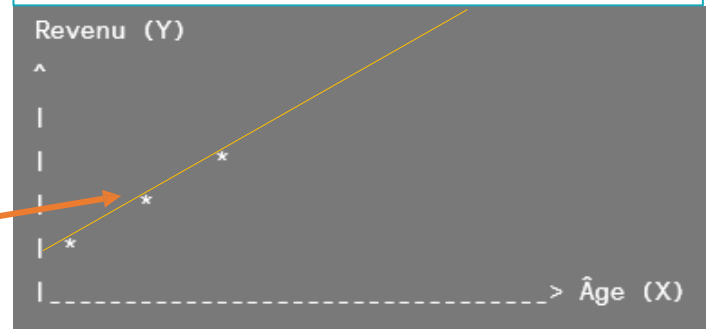


Caractéristiques :

- **Ordre important**.
- Les deux axes contiennent généralement des **valeurs numériques**.

Aucune corrélation

Revenu en fonction de l'âge



Interprétation :

Cherchez des tendances (trends and patterns), des clusters et des valeurs aberrantes.

Par exemple, un motif ascendant pourrait indiquer une corrélation positive.

Corrélation positive

Exemple :

Tracer l'âge par rapport au revenu pour voir s'il y a une corrélation.

PARTIE 2

Scatter Plot (Diagramme de dispersion / Nuage de points) :

Exemple sur Tips Data :

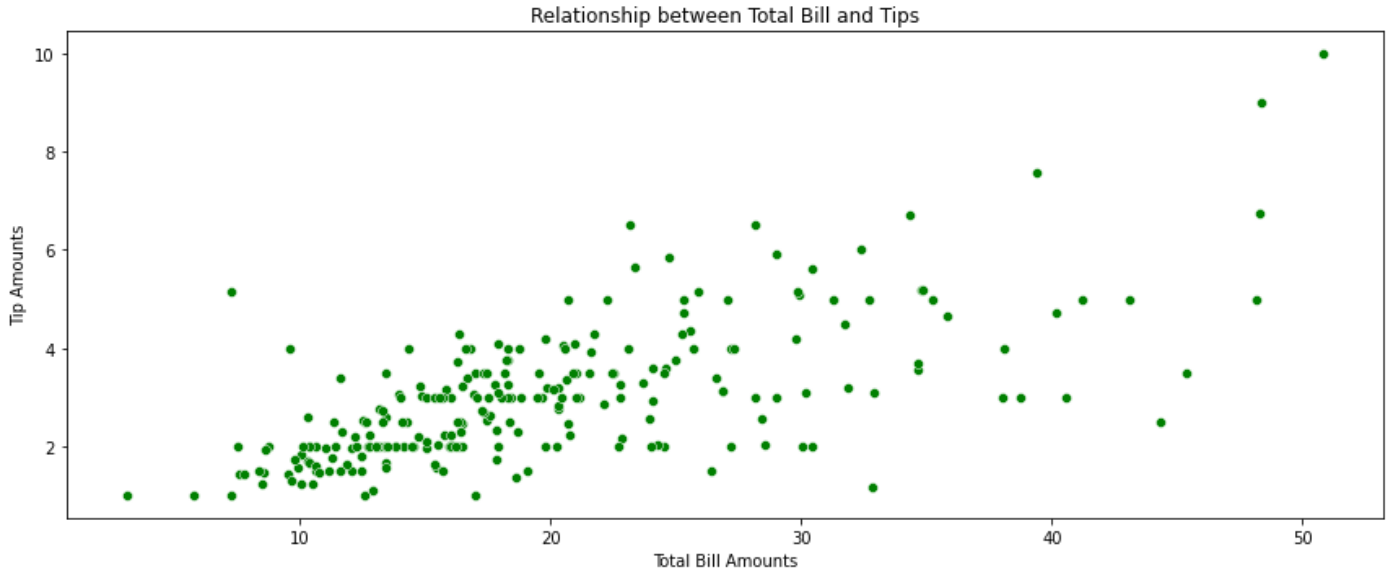
Tracer « Total bill» par rapport au « Tips » pour voir s'il y a une corrélation.

Indique que :

- Total bill sur l'axe X (axe horizontal).
- Tips sur l'axe Y (axe vertical).

Nous étudions les **Tips** en fonction de **Total bill.**

il est courant de dire "y en fonction de x" pour indiquer que la variable y dépend de la variable x



Dans ce diagramme de **dispersion**, chaque point représente une facture totale et le pourboire correspondant. En observant la disposition des points, vous pouvez déterminer s'il existe une relation entre le montant total de la facture et le montant des pourboires. Si les points suivent une tendance ascendante, cela indique une **corrélation positive** où des factures plus élevées sont associées à des pourboires plus élevés.

PARTIE 2

CHAPITRE 1

CRÉER DIVERS TYPES DE GRAPHIQUES

1. Scatter plots
2. **Bar plots**
3. Histograms
4. Box plots
5. Bubble charts
6. Création de graphiques avec Python/Excel/R

Bar Plot (Graphique à barres)

Un diagramme à barres est un graphique qui représente les catégories de données avec des barres rectangulaires dont les longueurs et hauteurs sont proportionnelles aux valeurs qu'elles représentent.

Usage: Utilisé pour **comparer** différentes catégories ou groupes, souvent pour visualiser des données catégorielles.

Caractéristiques :

- **Ordre pas important**.

•Un des axes contiennent habituellement des **valeurs catégoriques.**

Types: Barres verticales, barres horizontales.

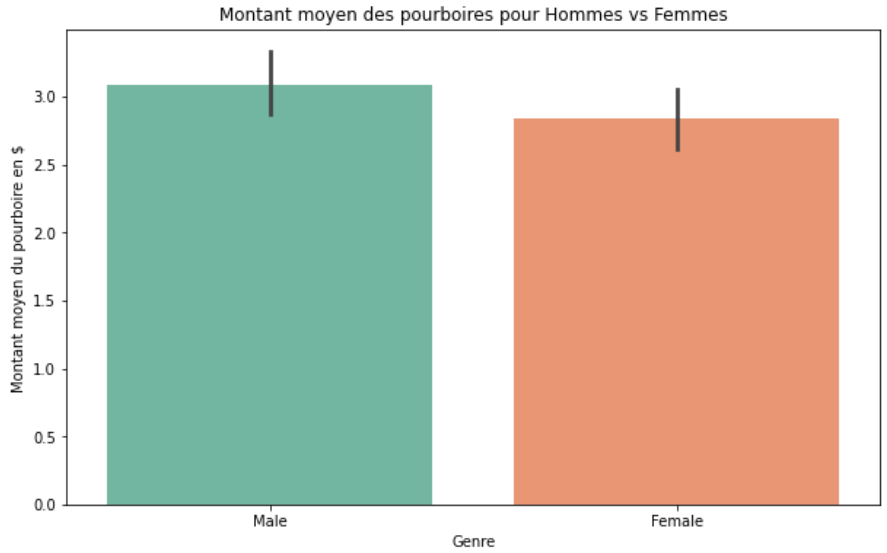
Interprétation:

Comparez les longueurs des barres pour comprendre les différences entre les catégories.

Exemples:

Comparer les chiffres de vente dans différentes régions.

Ou comparez les montants des pourboires entre hommes et femmes dans un restaurant.



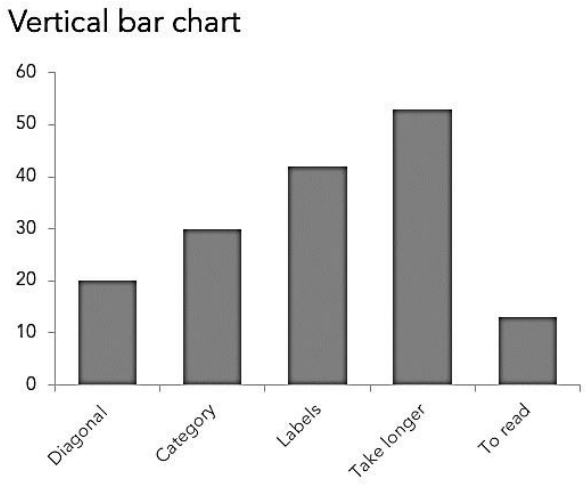
Source : Application au jeu de données Tips de la section précédente

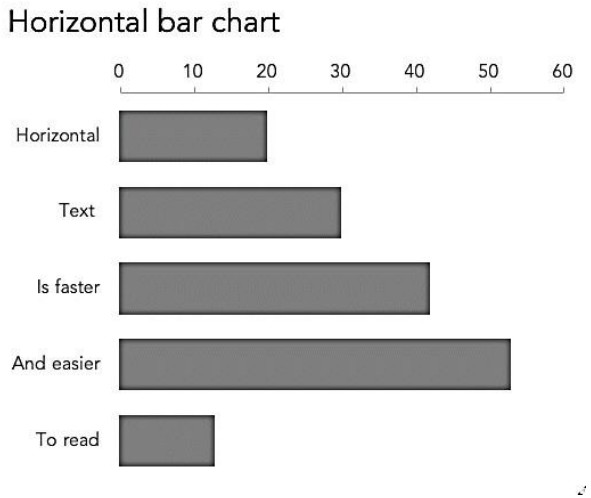
PARTIE 2

Bar Chart (Histogramme à barres)

Visualisation Catégorielle et Comparaison :

- Les diagrammes à barres sont couramment utilisés pour représenter graphiquement des données **catégorielles** et les **comparer** entre elles. Ils offrent une visualisation claire des relations et des distinctions entre différentes catégories, permettant une interprétation rapide et précise des données.
- Chaque barre représente une catégorie distincte et sa hauteur est proportionnelle à la valeur associée à cette catégorie.
- Les barres peuvent être agencées **horizontalement** ou **verticalement** en fonction de la préférence ou de la lisibilité du graphique.

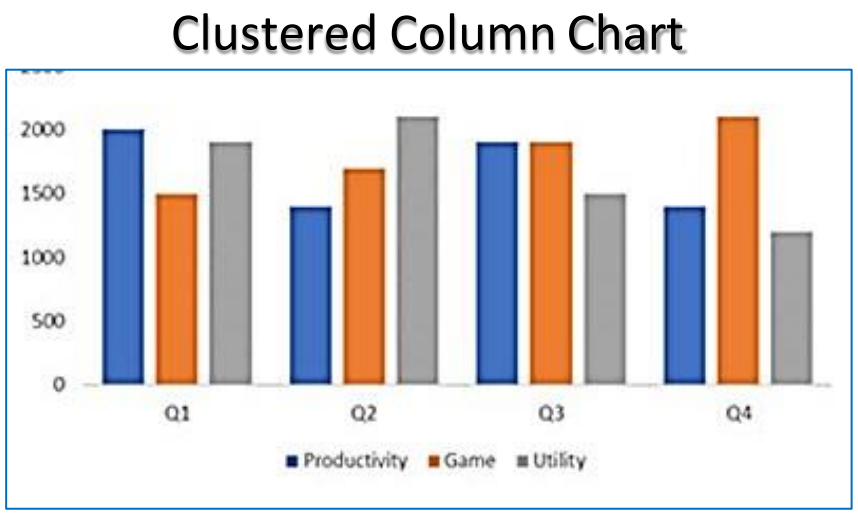


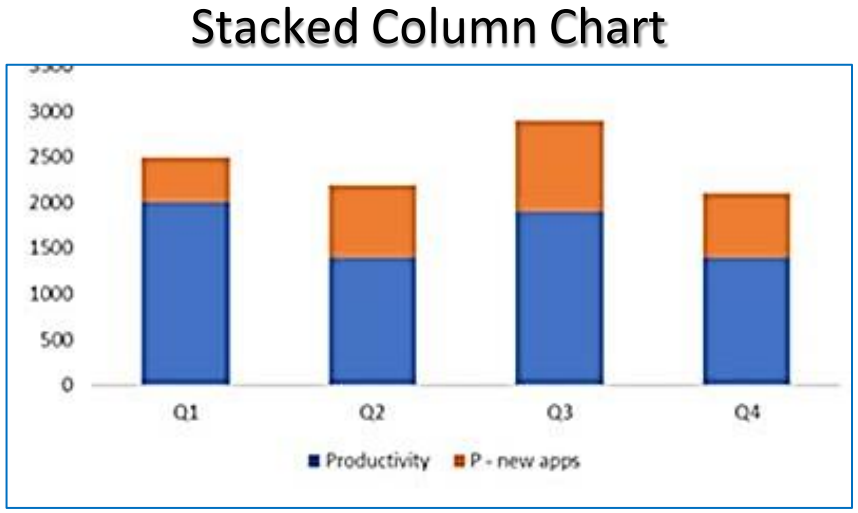


Bar Chart (Histogramme à barres)

Visualisation Catégorielle et Comparaison :

- Les colonnes empilées '**Stacked**' et groupées '**Clustered**' dans un graphique à barres permettent de comparer des catégories et des sous-catégories de données.
- Les colonnes empilées '**Stacked**' montrent la contribution de chaque sous-catégorie à l'ensemble, tandis que les colonnes groupées '**Clustered**' permettent de comparer directement les valeurs des sous-catégories entre elles.





Les deux graphiques montrent la productivité d'une entreprise au cours des trimestres de l'année. Le diagramme à barres groupées « Clustered » compare les éléments au sein de chaque groupe, tandis que le diagramme à barres empilées « Stacked » facilite les comparaisons entre les groupes les uns par rapport aux autres pour chaque trimestre.

PARTIE 2

CHAPITRE 1

CRÉER DIVERS TYPES DE GRAPHIQUES

1. Scatter plots
2. Bar charts
3. **Histograms**
4. Box plots
5. Bubble charts
6. Création de graphiques avec Python/Excel/R

Histograms

Histogram (Histogramme)

Un histogramme est un graphique qui représente la distribution d'un ensemble de données numériques à l'aide de barres continues dont la hauteur correspond à la fréquence des valeurs dans chaque intervalle.

Usage : L'Histogramme permet de visualiser la **distribution** d'un ensemble de données numériques. Il montre comment les valeurs sont réparties et permet d'identifier la forme de la distribution, la dispersion des données et leur relation par rapport à la moyenne.

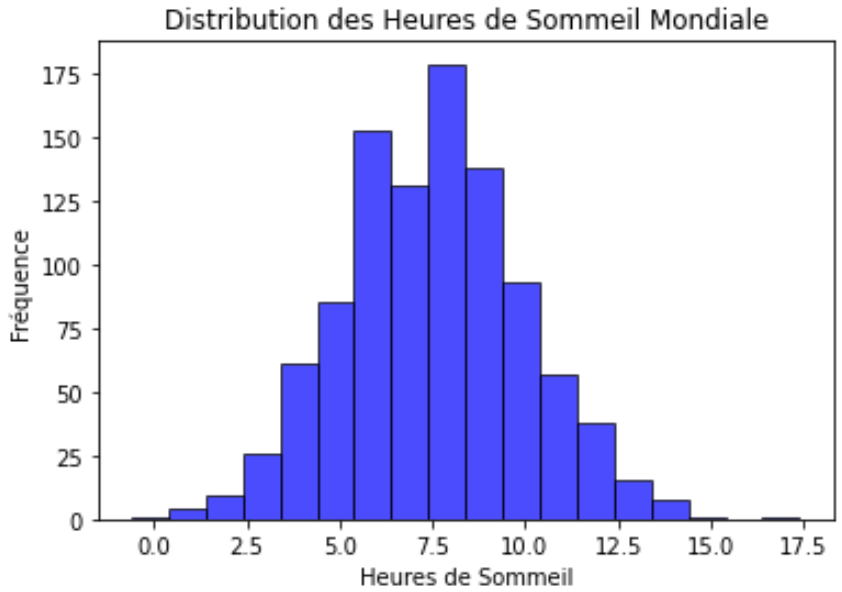
Caractéristiques :

- Barre continue et **l'ordre est important**
- Les deux axes d'un histogramme contiennent habituellement des **valeurs numériques .**

Interprétation : Observez la forme de l'histogramme pour comprendre la distribution

(par exemple, distribution normale, uniform, asymétrie..).

Exemple : Montrer la fréquence des heures de sommeil d'un échantillon .



Cela reflète bien que :
l'axe X représente les **intervalles** de valeurs des données,
et l'axe Y représente la **fréquence** des données dans ces intervalles.

PARTIE 2

CHAPITRE 1

CRÉER DIVERS TYPES DE GRAPHIQUES

1. Scatter plots
2. Bar charts
3. Histograms
4. **Box plots**
5. Bubble charts
6. Création de graphiques avec Python/Excel/R

Box Plot (Boîte à moustaches)

Le Boxplot est un graphique résumant la distribution de données.

Usage : Visualisez les distributions en affichant la médiane, les quartiles supérieurs et inférieurs où se concentre la majorité des données (50 %), ainsi que les moustaches supérieures et inférieures indiquant les valeurs min et max, permettant aussi d'identifier les valeurs aberrantes, comme expliqué dans la partie précédente du cours qui étudie les valeurs aberrantes.

Caractéristiques :

- "Moustaches" s'étendant aux valeurs minimale et maximale, à l'exclusion des valeurs aberrantes.
- Boîte montrant l'intervalle interquartile (Q1 25% à Q3 75% quartiles).
- Ligne à l'intérieur de la boîte représentant la médiane.

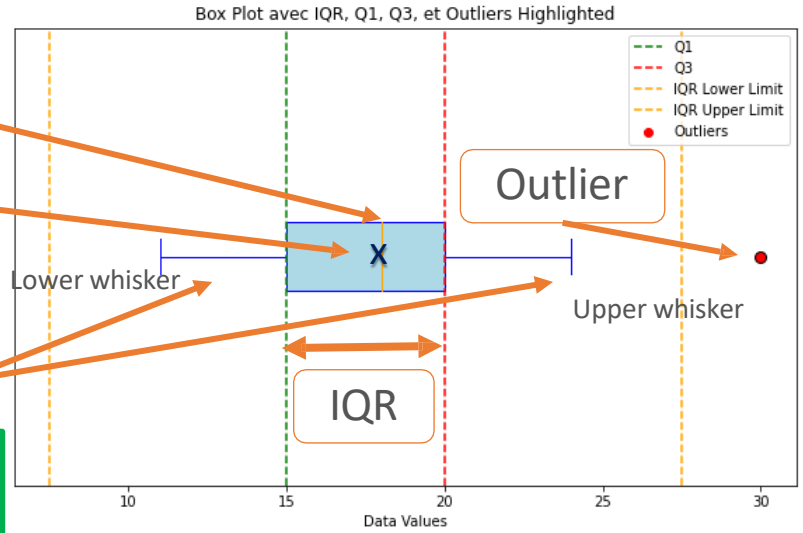
Exemple : Résumer la distribution des revenus dans une population.

Interprétation : Analysez la dispersion et identifiez les valeurs aberrantes.

médiane |

moyenne x

moustaches



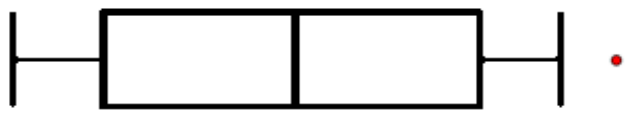
Normalement Les box plots affichent la médiane, mais dans certains cas (Excel), un petit 'x' peut faire référence à la moyenne.

PARTIE 2

Box plot

Box Plot (Boîte à moustaches) : Points clés

- La partie **box** d'un box-and-whisker plot couvre les 50% centraux des valeurs dans l'ensemble de données.
- Les whiskers couvrent chacun 25% des valeurs des données.
  - Le **Lower whisker (inférieur)** couvre toutes les valeurs des données depuis la valeur minimale jusqu'à Q1, c'est-à-dire les 25% les plus bas des valeurs des données.
  - L' **Upper whisker (supérieur)** couvre toutes les valeurs des données entre Q3 et la valeur maximale, c'est-à-dire les 25% les plus élevés des valeurs des données.
- La **médiane** se situe à l'intérieur de la box et représente le centre des données. 50% des valeurs des données se trouvent au-dessus de la médiane et 50% se trouvent en dessous de la médiane.
- Les **outliers**, ou valeurs extrêmes, dans un ensemble de données sont généralement indiqués sur un box-and-whisker plot par des points individuels en dehors des moustaches.



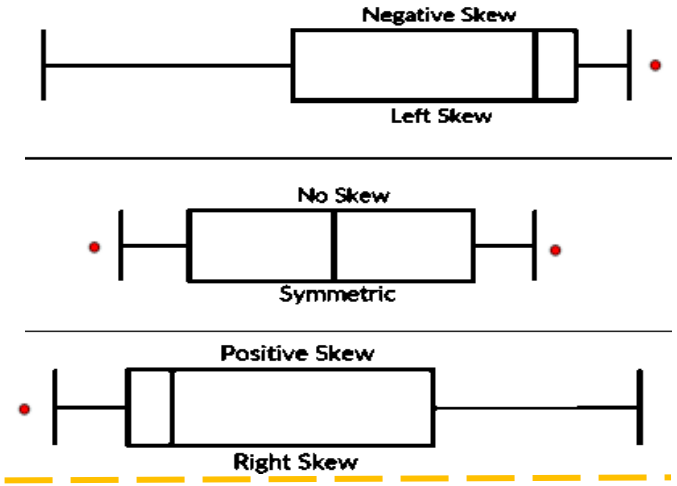
PARTIE 2

Box plot

Box Plot (Boîte à moustaches) : Evaluation de la symétrie de la distribution

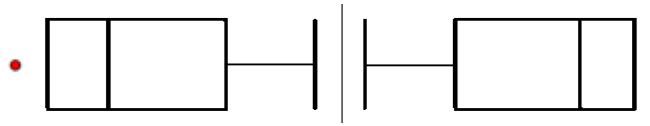
Skewness (Asymétrie) :

- **Asymétrie négative :** si la médiane proche de la fin et dernière moustache plus courte.
- **Symétrique :** si la ligne médiane est au centre.
- **Asymétrie positive** si la médiane proche du début et première moustache plus courte.



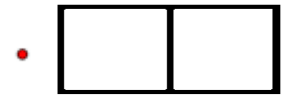
L'absence d'un élément dans une Box plot peut indiquer différentes situations :

- **Moustache unique au bord de la boîte :** Cela se produit lorsque **la valeur minimale est égale au premier quartile (Q1) ou si la valeur maximale est égale au troisième quartile (Q3)**. Dans ce cas, une moustache semble se chevaucher avec le bord de la boîte et n'est donc pas visible.



Moustache unique visible

- **Absence de moustaches :** Si **la valeur minimale est égale à Q1 et la valeur maximale est égale à Q3**, le box plot peut sembler ne pas avoir de moustaches. Cela donne l'apparence d'une simple boîte sans extensions, car les moustaches se chevauchent avec la boîte.



Absence de moustaches visible

- **Absence de ligne médiane visible :** Si la **ligne médiane est égale à Q1 ou Q3**, elle peut se chevaucher avec les bords de la boîte, rendant la ligne médiane invisible.

Absence de médiane visible

PARTIE 2

CHAPITRE 1

CRÉER DIVERS TYPES DE GRAPHIQUES

1. Scatter plots
2. Bar charts
3. Histograms
4. Box plots
5. **Bubble charts**
6. Création de graphiques avec Python/Excel/R

Bubble Chart (Graphique à bulles)

Les graphiques à bulles, une sous-catégorie des Scatter plots , permettent de visualiser des données multivariées. Dans ce type de visualisation, les nuages de points peuvent représenter la relation entre plusieurs variables en colorant les points ou en modifiant leur forme ou leur taille.

Usage : Utilisé pour démontrer les **relations** entre trois variables (X, Y, taille de la bulle).

Caractéristiques:

Combinaison de nuages de points avec une dimension supplémentaire représentée par la taille de la bulle.

Permet de visualiser les interactions complexes entre trois variables.

Interprétation: Comparez les bulles pour comprendre les relations et les tailles relatives.

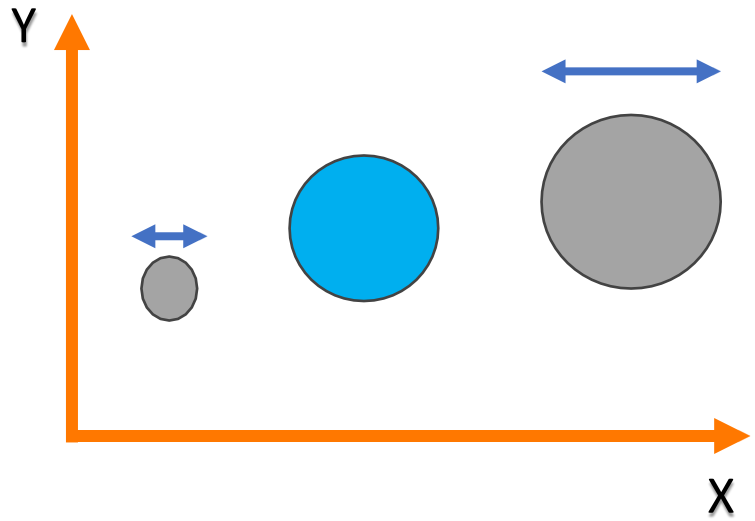
Exemples:

Tracer les ventes (X), les villes (Y) et la part de marché (taille de la bulle).

Tracer le GDP par habitant (X), l'espérance de vie (Y) et la population (taille de la bulle).

***GDP: Gross Domestic Product (En)**

***PIB : Produit Intérieur Brut (Fr)**



Pie charts

Un pie chart, ou diagramme circulaire, est une représentation graphique utilisée pour montrer la répartition ou la proportion des différentes catégories par rapport à un tout. Chaque catégorie est représentée par une part du cercle, dont l'angle correspond au pourcentage de cette catégorie sur l'ensemble des données.

Usage :

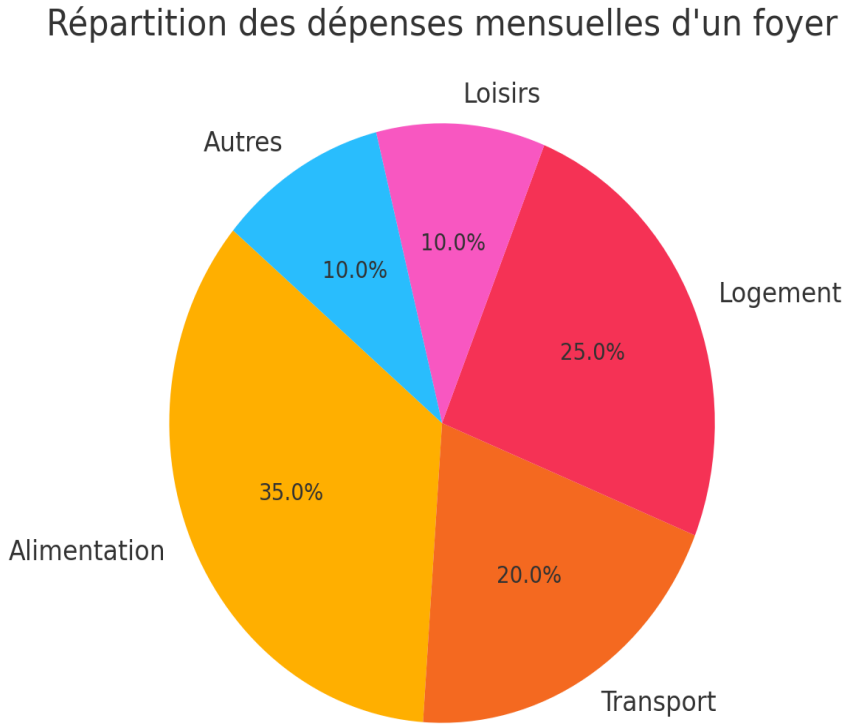
Comparaison des proportions entre différentes catégories dans un ensemble.
Idéal pour des données qualitatives ou catégorielles ayant des parts relativement simples à comparer.

Exemples courants :

Répartition des dépenses d'un foyer par catégorie.
Distribution des votes entre différents candidats lors d'une élection.

Remarques :

Les parts à montrer en pourcentages
La somme des parts est 100%
Attention au biais : beaucoup de categories ou pourcentages trop proches



PARTIE 2

CHAPITRE 1

CRÉER DIVERS TYPES DE GRAPHIQUES

1. Scatter plots
2. Bar charts
3. Histograms
4. Box plots
5. Bubble charts
6. **Création de graphiques avec Python/Excel/R**

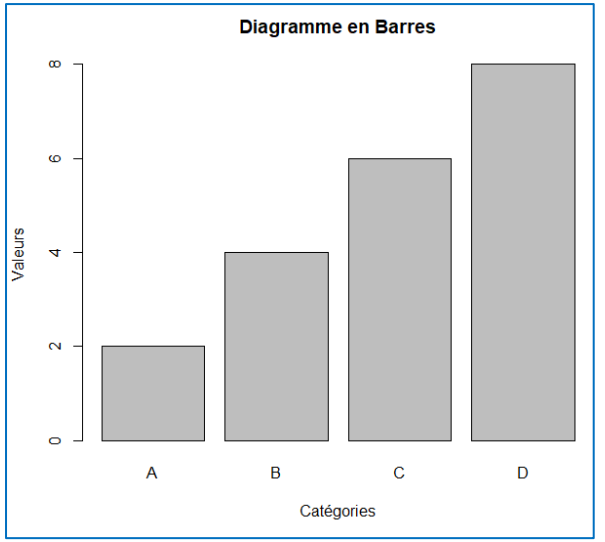
PARTIE 2

Création de graphiques avec R : Bar Chart

1. Pour créer un graphique à barres simple :

```
# valeurs de l'axe x
x <- c("A", "B", "C", "D")
# valeurs de l'axe y
y <- c(2, 4, 6, 8)
# création du diagramme en barres
barplot(y, names.arg = x, xlab = "Catégories", ylab = "Valeurs", main = "Diagramme en Barres")
```

xlab : définit l'étiquette pour l'axe des x ("Catégories").
ylab : définit l'étiquette pour l'axe des y ("Valeurs").
main : définit le titre principal du graphique ("Diagramme en Barres")

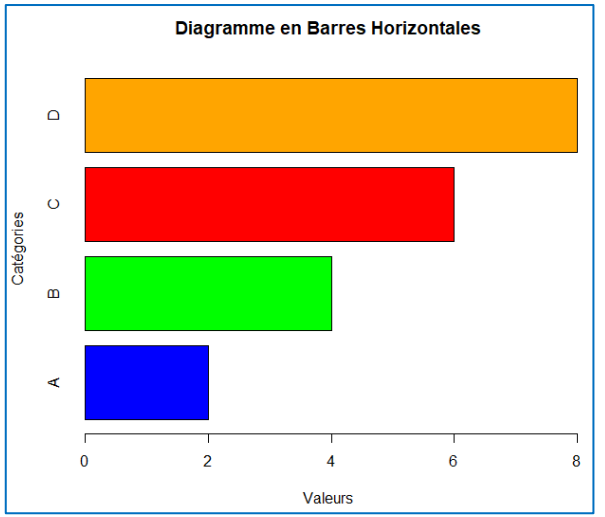


2. Pour personnaliser votre graphique à barres :

Sur RStudio :

```
# couleurs des barres
couleurs <- c("blue", "green", "red", "orange")

# création du diagramme en barres horizontales avec couleurs personnalisées
barplot(y, names.arg = x, horiz = TRUE, col = couleurs, xlab = "Valeurs", ylab = "Catégories", main = "Diagra
```

horiz = TRUE : spécifie que le diagramme en barres sera horizontal.
col = couleurs : utilise le vecteur couleurs pour définir les couleurs des barres.



Création de graphiques avec R : Scatter Plot

1. Pour créer un Nuage de points :

Utilisons jeu de données intégré à R pour créer un graphique de dispersion. Nous allons utiliser l'ensemble de données, **mtcars** qui contient des informations sur différentes voitures. Voici comment procéder :

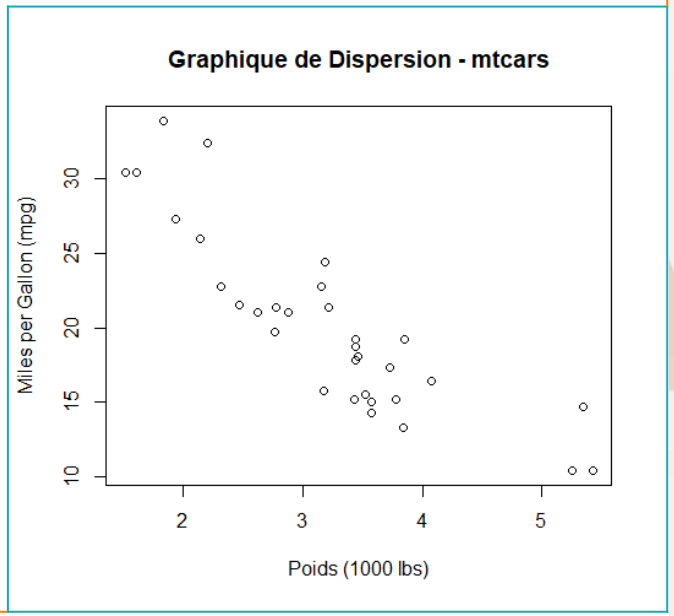
```
# Chargement de l'ensemble de données mtcars

data(mtcars)

# Affichage des premières lignes de l'ensemble de données mtcars pour comprendre sa structure

head(mtcars)
```

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
# Création du graphique de dispersion

plot(mtcars$wt, mtcars$mpg,

    xlab = "Poids (1000 lbs)", ylab = "Miles per Gallon (mpg)", main = "Graphique de Dispersion - mtcars")
```

Étiquettes les axes

Titre du graphique



PARTIE 2

Création de graphiques avec R : Scatter Plot

2. Explication des étapes suivies dans l'application précédente mtcars :

- **Chargement des données :** Nous avons chargé l'ensemble de données mtcars en utilisant data(mtcars).
- **Exploration initiale :** Pour mieux comprendre la structure de l'ensemble de données et voir ses premières observations, nous avons utilisé head(mtcars).
- **Création du graphique de dispersion :** Nous avons créé un graphique de dispersion avec plot(mtcars$wt, mtcars$mpg, ...), où **wt** représente le poids des voitures en (thousands of pounds) et **mpg** indique le nombre de miles par gallon.
- **Personnalisation du graphique :** Les options xlab, ylab et main ont été utilisées pour définir respectivement les étiquettes des axes x et y, ainsi que le titre du graphique, tous en français.

Vous pouvez explorer d'autres variables de l'ensemble de données mtcars en remplaçant wt et mpg par d'autres colonnes telles que hp (puissance), disp (déplacement du moteur), etc., selon ce qui vous intéresse.

Création de graphiques avec R : Histogram

Création d'un histogramme :

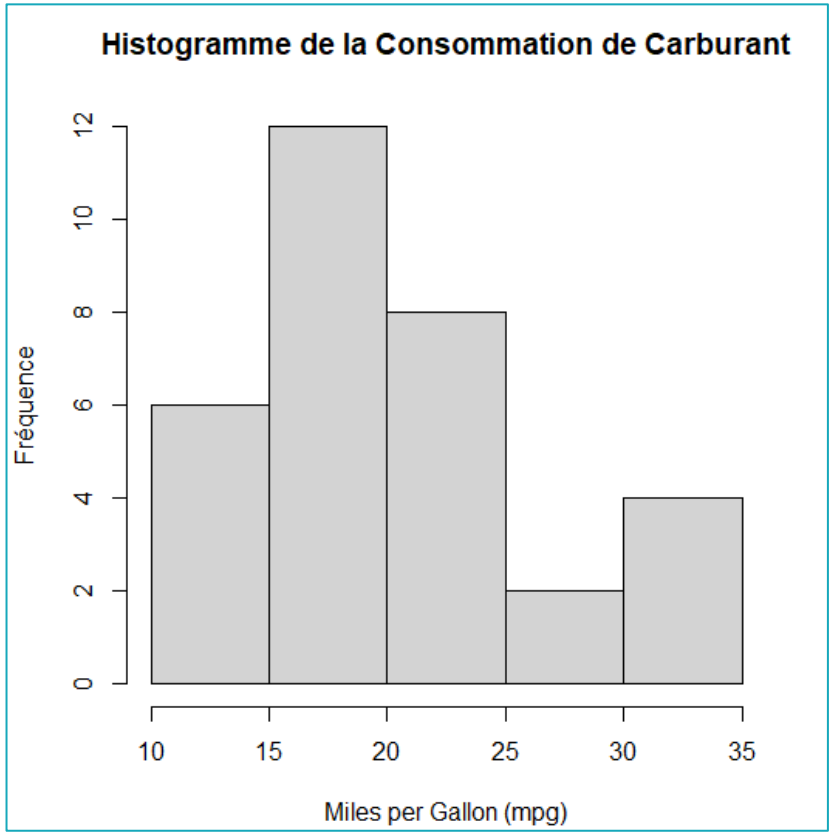
Voici comment créer un histogramme avec l'ensemble de données **mtcars** en R :

```
# Histogramme de la variable mpg (miles per gallon)
hist(mtcars$mpg,
    xlab = "Miles per Gallon (mpg)", ylab = "Fréquence",
    main = "Histogramme de la Consommation de Carburant")
```

Crée un histogramme des valeurs de la variable mpg.

Étiquettes respectivement les étiquettes des axes x et y

Définit le titre de l'histogramme




```
> dim(mtcars)
[1] 32 11
```

L'histogramme montre clairement que la barre représentant l'intervalle de 15 à 20 mpg atteint une fréquence de 12. Alors on peut dire que 12 voitures ont une consommation comprise entre 15 et 20 miles par gallon.

L'histogramme nous permet de visualiser la répartition des consommations en carburant (mpg) des voitures dans l'ensemble de données mtcars.

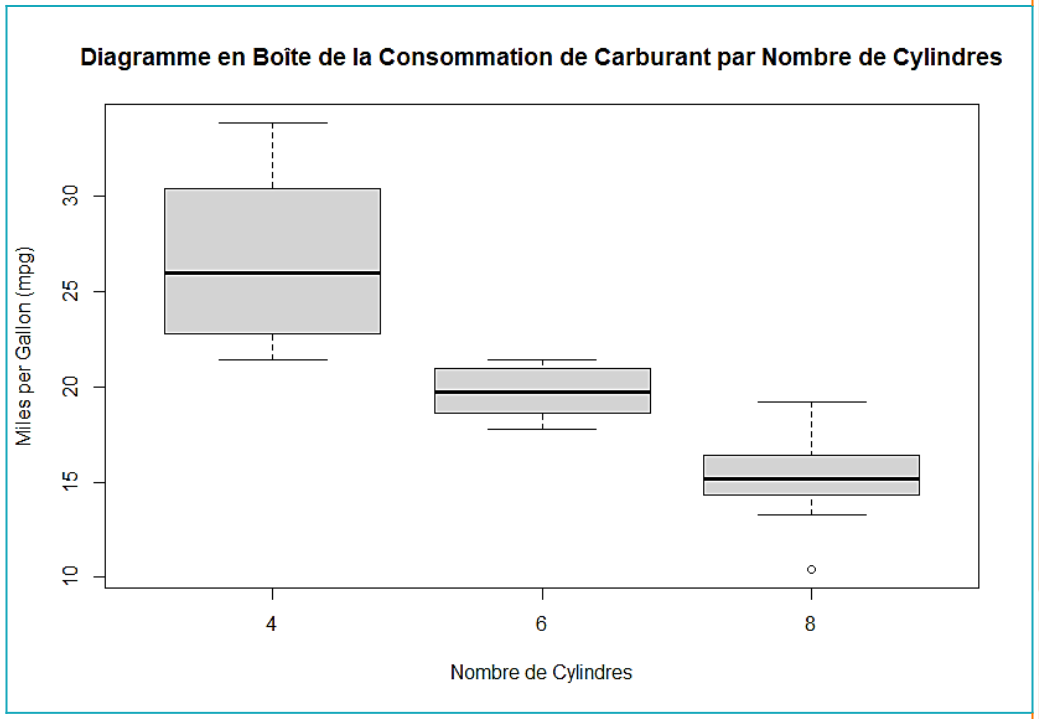
PARTIE 2

Création de graphiques avec R : Box Plot

Le diagramme en boîte (box plot) est utilisé pour représenter la distribution des données et pour identifier les valeurs aberrantes (outliers). Nous allons créer un diagramme en boîte pour visualiser la consommation de carburant (mpg) par nombre de cylindres (cyl) des voitures.

Création d'un Box Plot :

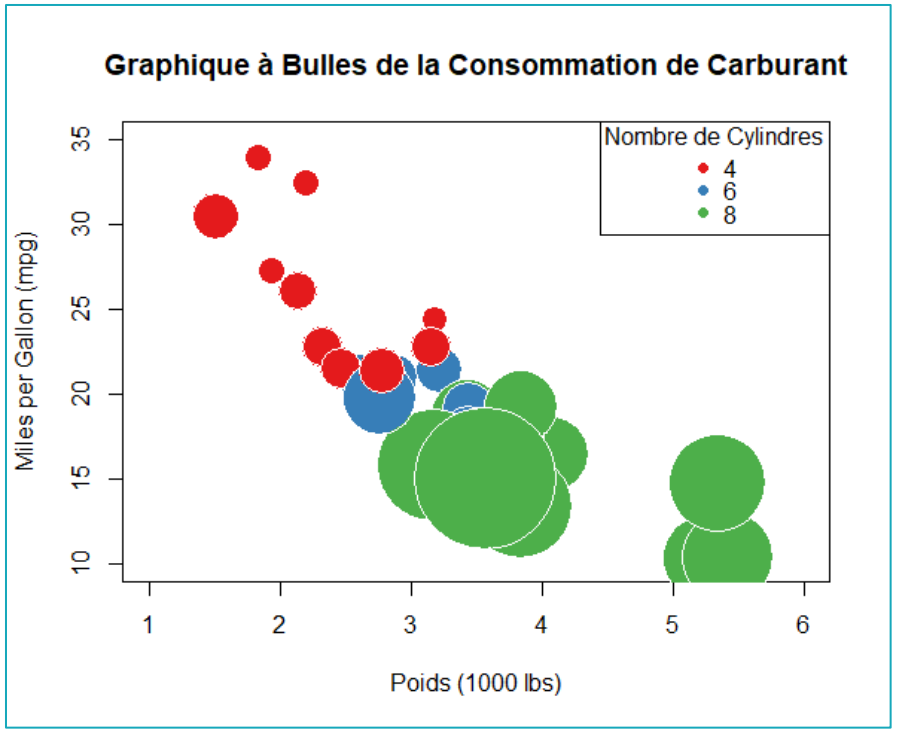
```
# Diagramme en boîte de la variable mpg par nombre de cylindres

boxplot(mpg ~ cyl, data = mtcars,

xlab = "Nombre de Cylindres",

ylab = "Miles per Gallon (mpg)",  # Étiquettes des axes

main = "Diagramme en Boîte de la Consommation de Carburant par Nombre de Cylindres")  # Définit le titre de l'histogramme
```



Création de graphiques avec R : Box Plot

Pour créer un graphique à bulles, nous pouvons utiliser la fonction symbols pour représenter les bulles. Voici comment procéder avec l'ensemble de données mtcars :

```
# Chargement du package nécessaire pour des couleurs

library(RColorBrewer)

# Définition des couleurs pour les différentes catégories de cylindres

colors <- brewer.pal(3, "Set1")

# Création du graphique à bulles

symbols(mtcars$wt, mtcars$mpg, circles = mtcars$hp, inches = 0.5, fg = "white",

    bg = colors[as.factor(mtcars$cyl)], xlab = "Poids (1000 lbs)",

    ylab = "Miles per Gallon (mpg)", main = "Graphique à Bulles de la Consommation de Carburant",

    xlim = c(1, 6), ylim = c(10, 35))

# Ajout d'une légende

legend("topright", legend = levels(as.factor(mtcars$cyl)),

    col = colors, pch = 16, title = "Nombre de Cylindres")
```



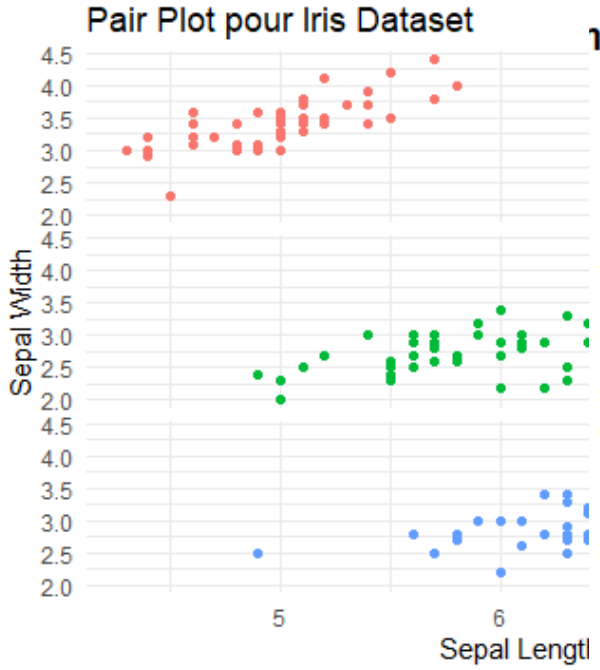
wt: Poids en milliers de livres
mpg: Miles per gallon (consommation de carburant)
hp: Puissance en chevaux-vapeur (horsepower)
cyl: Nombre de cylindres

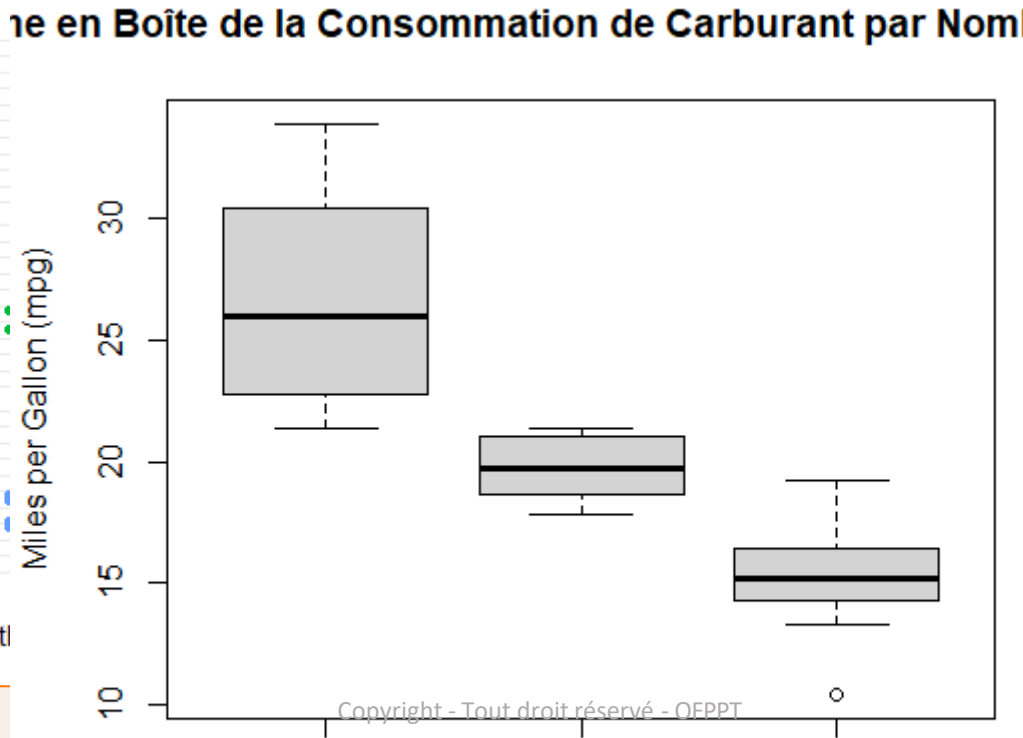
PARTIE 2

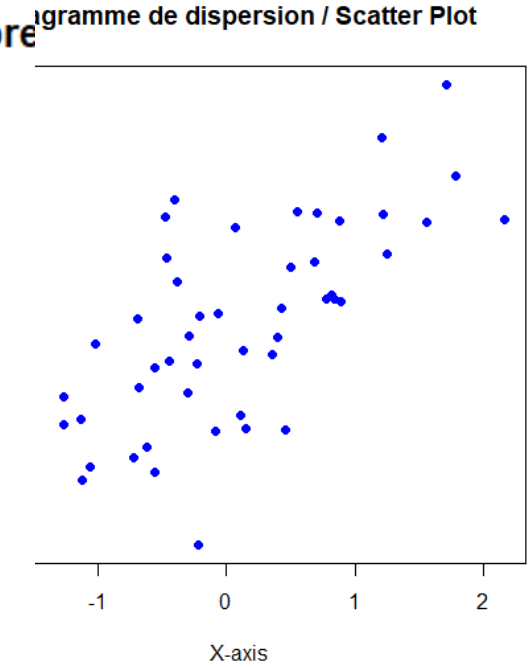
Création de graphiques avec R : Box Plot

Les graphiques statiques constituent un autre point fort de R. Il peut produire des graphiques de qualité professionnelle incluant des symboles mathématiques.

Exemples de graphiques : application sur Iris dataset







PARTIE 2

Création de graphiques avec Python : Scatter Plot (Diagramme de dispersion / Nuage de points)

Exemple sur Tips Data :

Pour visualiser la relation entre le montant total de la facture et les pourboires, nous allons utiliser les bibliothèques Seaborn et Matplotlib en Python. **Seaborn** simplifie la création de graphiques esthétiques, tandis que **Matplotlib** permet une personnalisation fine des visualisations.

Voici comment procéder :

```
# Importer les bibliothèques pour la visualisation et
le chargement des données :

import seaborn as sns
import matplotlib.pyplot as plt
# Charger le jeu de données Tips depuis seaborn
tips = sns.load_dataset("tips")
```

Seaborn est principalement utilisé pour la visualisation des données, même s'il contient quelques fonctions pour charger des jeux de données d'exemple.

Création de graphiques avec Python : Scatter Plot (Diagramme de dispersion / Nuage de points)

Exemple sur Tips Data :

```
# Créer une figure
plt.figure(figsize=(10, 4))

# Tracer le diagramme de dispersion
sns.scatterplot(data=tips, x='total_bill',
y='tip', color='green')
# Ajouter le titre
plt.title('Relation entre Total Bill et Tips')

# Étiqueter l'axe X
plt.xlabel('Montants Total Bill')
# Étiqueter l'axe Y
plt.ylabel('Montants Tips’)

# Ajouter une grille
plt.grid(True)

# Afficher le graphique
plt.show()
```

- Initialisez une figure avec des dimensions adaptées, avec figsize=(width ,height).
- Utilisez Seaborn « sns,scatterplot() » pour tracer vos données : spécifiez la source des données, les axes X et Y, et les couleurs appropriées.
- Utilisez Matplotlib pour définir un titre descriptif et des étiquettes clairs pour le graphique.
- Ajouter une grille pour améliore la lisibilité du graphique en fournissant des repères visuels pour les valeurs des axes X et Y.

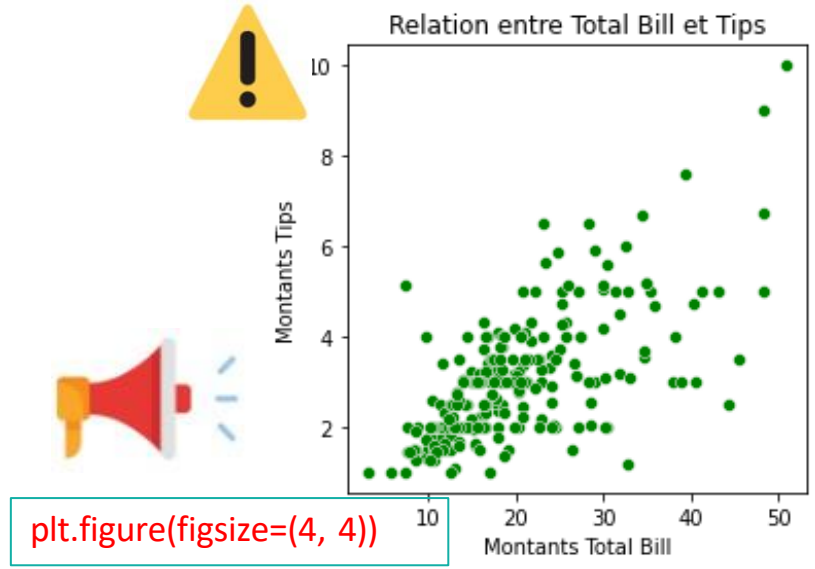
PARTIE 2

Création de graphiques avec Python : Scatter Plot (Diagramme de dispersion / Nuage de points)

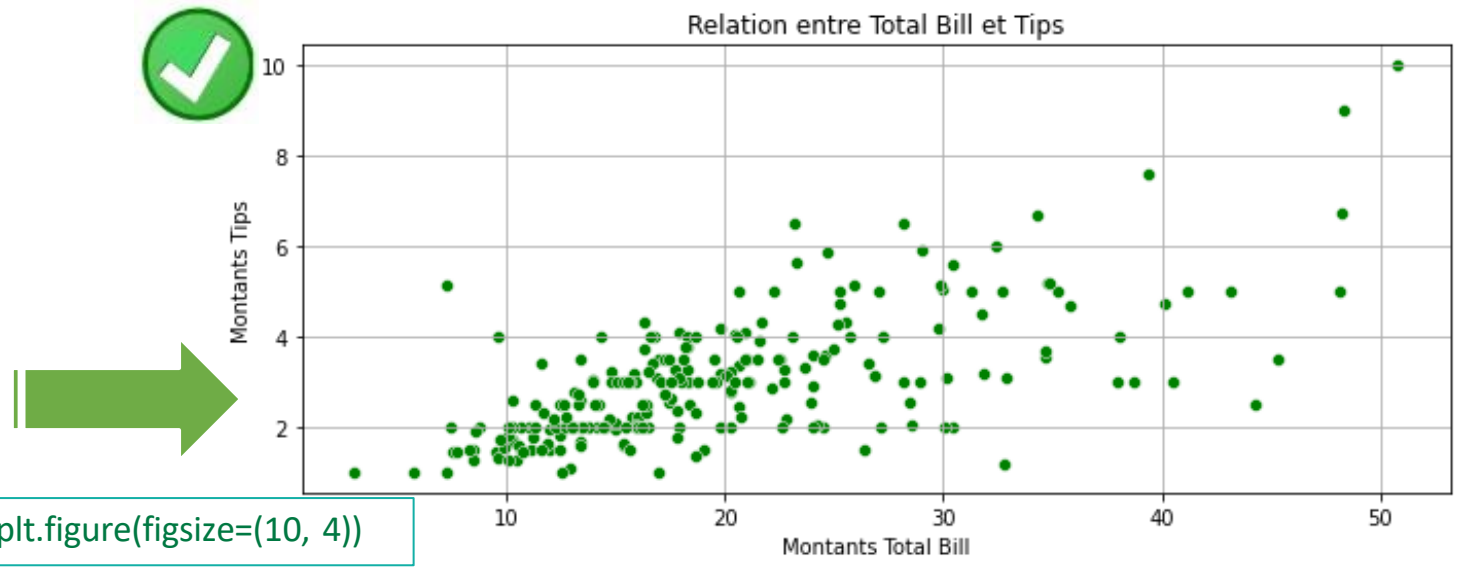
Recommandation :

La spécification de la taille du graphique est importante car elle peut intentionnellement affecter l'interprétation des données.

La taille du graphique doit être **proportionnelle** aux données pour une interprétation précise. Par exemple, si le pourboire va jusqu'à **10$** et le total de la facture jusqu'à **50$** (**même unité '$'**), nous avons choisi une hauteur de figure plus petite pour les pourboires afin de maintenir cette proportionnalité, assurant une représentation équilibrée des données et une analyse précise de leur relation.



plt.figure(figsize=(4, 4))



plt.figure(figsize=(10, 4))

Création de graphiques avec Python : Bar Chart (Histogramme à barres)

À ce stade, vous devez réaliser que chaque graphique comporte un titre, une source de données et peut avoir des axes. L'initialisation de ces éléments en Python est similaire au graphique précédent. Maintenant, pour la création d'un diagramme à barres, la tâche devient très simple : Nous utiliserons **la fonction barplot() de seaborn** pour la création d'un Bar Chart.

Voici les étapes à suivre :

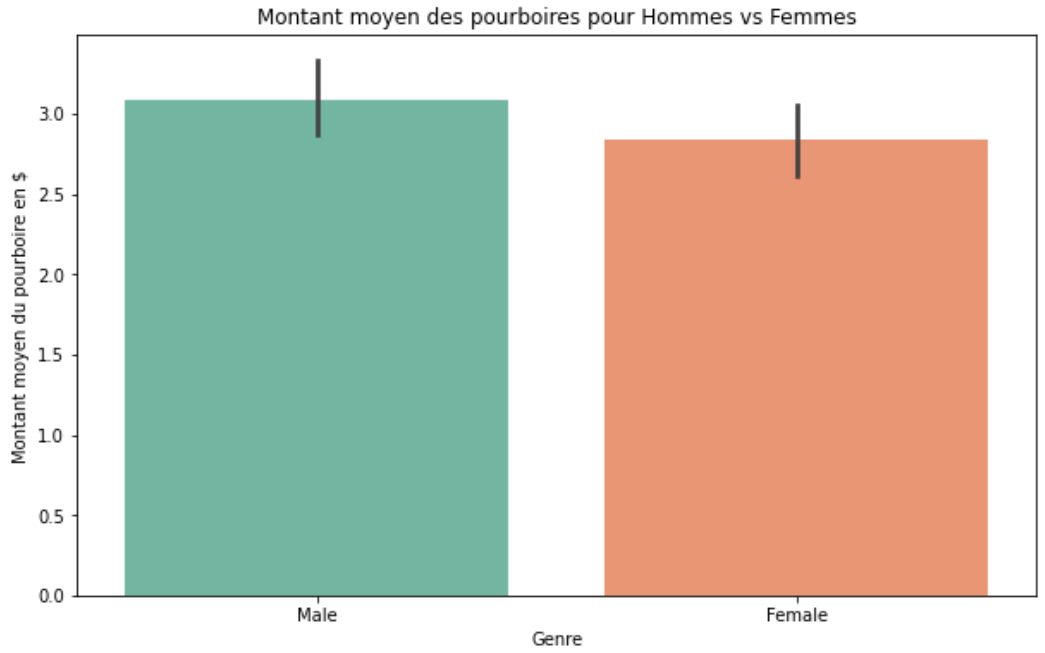
- Importer les bibliothèques de visualisation et de chargement des données et charger le jeu de données Tips depuis Seaborn.
- Créer un graphique en barres pour comparer les pourboires entre Hommes et Femmes et initialiser une figure avec une taille de 10x6 pouces.
- Utiliser Seaborn pour tracer un graphique en barres de tips avec le Genre sur l'axe X et pourboire sur l'axe Y, et choisir une palette de couleurs 'Set2'
- Définir le titre, les étiquettes et Afficher le graphique

```
import matplotlib.pyplot as plt
import seaborn as sns

# Charger de données
tips = sns.load_dataset("tips")

# Créer un graphique en barres
plt.figure(figsize=(10, 6))
sns.barplot(x='sex', y='tip', data=tips, palette='Set2')
# Définir le titre et les étiquettes
plt.title('Montant moyen des pourboires pour Hommes vs Femmes')
plt.xlabel('Genre')
plt.ylabel('Montant moyen du pourboire en $')
# Afficher le graphique
plt.show()
```

Création de graphiques avec Python : Bar Chart (Diagramme à barres)

Pour lire le graphique de l'application précédente :

Ce graphique montre le montant moyen des pourboires (axe Y) **en fonction** du genre (axe X). Les barres montrent la moyenne des pourboires donnés par les hommes et les femmes, respectivement.



Création de graphiques avec Python : Histogram

La génération de la distribution des Heures de Sommeil Mondiale :

En tant qu'application de création d'histogramme, nous reviendrons pour créer l'histogramme précédent qui montre la distribution des heures de sommeil mondiale, qui a été créé pour mettre en valeur l'effet de différentes variances sur les données.

Voici les étapes à suivre :

1- Génération de données :

Générer un ensemble de données simulant les heures de sommeil dans le monde, en supposant une moyenne de 7,5 heures et un écart type de 2,5 heures.

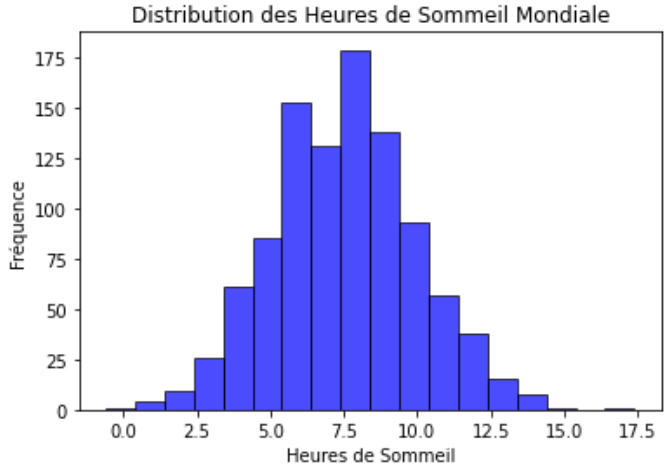
```
import numpy as np
import matplotlib.pyplot as plt

# Générer une distribution plus réaliste des heures de sommeil
dans le monde
np.random.seed(42)  # Fixer la graine pour la reproductibilité
mean_sleep_hours = 7.5  # Supposons une moyenne de 7,5
std_deviation = 2.5  # Supposons un écart type de 2,5
heures
# Générer un ensemble de données
sleep_hours = np.random.normal(mean_sleep_hours,
std_deviation, 1000)
```

Création de graphiques avec Python : Histogram

La génération de la distribution des Heures de Sommeil Mondiale :

2- Création de l'histogramme :

```
# Visualisation graphique
plt.hist(x=sleep_hours,
         bins=np.arange(min(sleep_hours),
         max(sleep_hours) + 1, 1),
         alpha=0.7, color='blue', edgecolor='black’)
# Créer l'histogramme avec des barres de largeur 1
# Ajouter un titre au graphique
plt.title('Distribution des Heures de Sommeil Mondiale')
# Étiqueter l'axe des
plt.xlabel('Heures de Sommeil’)
# Étiqueter l'axe des Y
plt.ylabel('Fréquence’)
# Afficher le graphique
plt.show()
```



bins : Définir les intervalles des barres de l'histogramme. Et crée des bacs d'une largeur de 1 heure, couvrant toutes les valeurs de sommeil observées.
il est possible de créer un histogramme sans spécifier explicitement le paramètre bins. Si vous ne fournissez pas ce paramètre, Matplotlib choisira automatiquement un nombre approprié de bacs en fonction des données.

alpha : pour ajuster la transparence des barres.

PARTIE 2

Création de graphiques avec Python : Histogram

Paramètres principaux de plt.hist() :

x : Les données que vous souhaitez visualiser sous forme d'histogramme.

bins : Le nombre de bacs (ou barres) à utiliser dans l'histogramme.

Peut-être un entier ou une séquence définissant les bornes des bacs.

range : La plage de valeurs à inclure dans l'histogramme (tuple de deux valeurs).

density : Si True, l'aire sous l'histogramme sera normalisée à 1 (produit une estimation de la densité).

weights : Une séquence de poids, de la même longueur que x. Utilisé pour pondérer les valeurs dans x.

cumulative : Si True, produit un histogramme cumulatif.

Autrement dit, les hauteurs des barres de l'histogramme ne représentent plus le nombre de données dans chaque intervalle (ou bac), mais plutôt la densité des données. C'est utile pour comparer des distributions avec des échantillons de tailles différentes ou pour visualiser la fonction de densité de probabilité des données.

```
# Exemple :
plt.hist(x=sleep_hours,
         bins=my_bins,
         range=(0, 15),
         density=False,
         weights=None,
         cumulative=False,
         histtype='bar',
         align='mid',
         orientation='vertical',
         rwidth=0.9,
         log=False,
         color='blue',
         label='Heures de sommeil',
         stacked=False)
```

PARTIE 2

Création de graphiques avec Python : Histogram

Paramètres principaux de plt.hist() : Suit

histtype : Le type d'histogramme à tracer ('bar', 'barstacked', 'step', 'stepfilled').

align : L'alignement des barres ('left', 'mid', 'right').

orientation : L'orientation des barres ('vertical', 'horizontal').

rwidth : La largeur relative des barres en pourcentage de la largeur des bacs.

log : Si True, l'axe Y sera en échelle logarithmique.

color : La couleur des barres de l'histogramme.

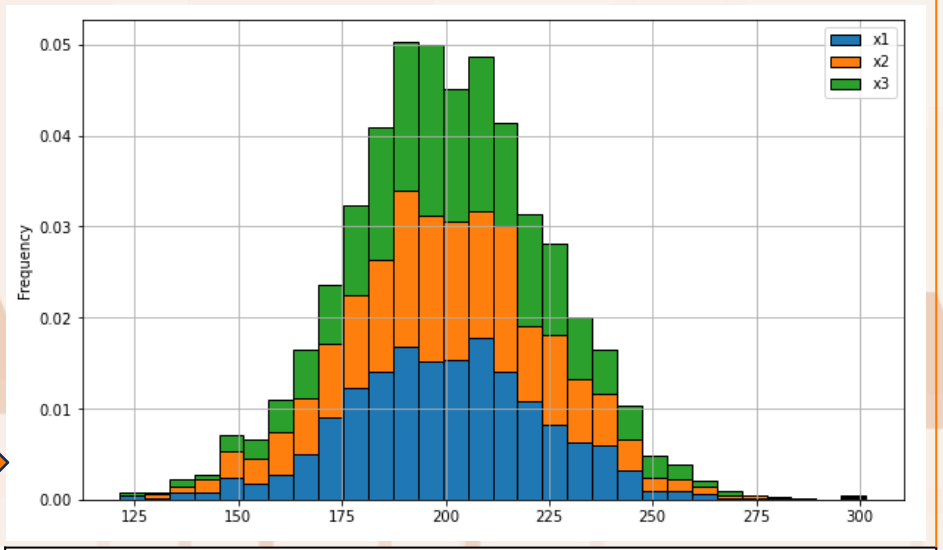
label : L'étiquette de la légende pour cet histogramme.

stacked : Si True et plusieurs données sont fournies, les histogrammes seront empilés.

Un histogramme empilé « **stacked Histogram** » est un graphique montrant la distribution de données en catégories, où les segments colorés empilés représentent différentes sous-catégories, permettant de visualiser à la fois la distribution totale et les contributions individuelles.

Pour créer un histogramme avec Matplotlib en utilisant la fonction plt.hist(), le seul paramètre vraiment nécessaire est le **x** « Les données », qui représente les données à visualiser. Les autres paramètres sont facultatifs.

Stacked Histogram



condition : les variables doivent avoir la même unité

Création de graphiques avec Python : Box plot

Application sur CarsMPG data set :

Pour créer un box plot utilisant le dataset des voitures (mpg) et visualiser la consommation de carburant (miles per gallon) en fonction du nombre de cylindres, nous allons suivre les mêmes étapes que précédemment. Nous utiliserons la bibliothèque Seaborn pour créer le graphique et Matplotlib pour le personnaliser et l'afficher.

Description du Jeu de Données mpg: Le jeu de données mpg (miles per gallon) contient des informations sur les performances de consommation de carburant de diverses voitures. Le jeu de données mpg de Seaborn comprend les données suivantes :

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86.0	2790	15.6	82	usa	ford mustang gl

PARTIE 2

Création de graphiques avec Python : Box plot

Application sur CarsMPG data set : suit

Signification de chaque colonne :

- **Miles per gallon (mpg) :** Mesure de la consommation de carburant d'une voiture, indiquant combien de miles une voiture peut parcourir par gallon de carburant.
- **Cylinders :** Le nombre de cylindres dans le moteur de la voiture.
- **Displacement :** La cylindrée totale du moteur, mesurée en pouces cubes.
- **Horsepower :** La puissance du moteur, mesurée en chevaux-vapeur.
- **Weight :** Le poids de la voiture, mesuré en livres.
- **Acceleration :** Le temps nécessaire pour passer de 0 à 60 mph, mesuré en secondes.
- **Model year :** L'année modèle de la voiture.
- **Origin :** Le pays d'origine de la voiture (États-Unis, Europe, Japon).
- **Name :** Le nom du modèle de la voiture.

Création de graphiques avec Python : Box plot

Voici les étapes à suivre :

1. Importer les bibliothèques pour la visualisation et le chargement des données :

```
import seaborn as sns  # Pour la visualisation des données
import matplotlib.pyplot as plt  # Pour personnalisation et affichage des graphiques

# Charger le jeu de données mpg de Seaborn
cars = sns.load_dataset("mpg")
```

2. Initialiser la figure et Créer un box plot pour visualiser la consommation de carburant par nombre de cylindres :

```
plt.figure(figsize=(10, 6))  # Initialiser une figure avec une taille définie
# Créer un box plot
sns.boxplot(x='cylinders', y='mpg', data=cars, palette='Set2')
```

Nous observons la consommation de carburant mpg (Y) en fonction du nombre de cylindres (X).

Comme vous pouvez le constater, la création de ce graphique est très simple et similaire à celle d'autres types de graphiques. Il suffit d'initialiser les **axes X et Y**, de spécifier la **source des données** et de choisir une **palette de couleurs**.

Création de graphiques avec Python : Box plot

3. Ajouter les étiquettes, le titre et Afficher le graphique

```
# Ajouter les étiquettes et le
plt.xlabel('Nombre de Cylindres’)
plt.ylabel('Consommation de Carburant (mpg)')
plt.title('Consommation de Carburant par Nombre de Cylindres')
# Afficher le graphique
plt.show()
```

Maintenant, l'output attendu de ce graphique sera la création d'un box plot pour chaque nombre de cylindres avec la variation de la consommation de carburant mpg (miles per gallon).

Avec :

1 mile (US) = 1.6093472187 km

1 gal (US) = 3.785411784 L

Création de graphiques avec Python : Box plot

Interprétation des résultats :

3 Cylindres : La médiane est relativement basse, indiquant que les voitures avec 3 cylindres ne sont pas très efficaces en termes de consommation de carburant par rapport aux autres.

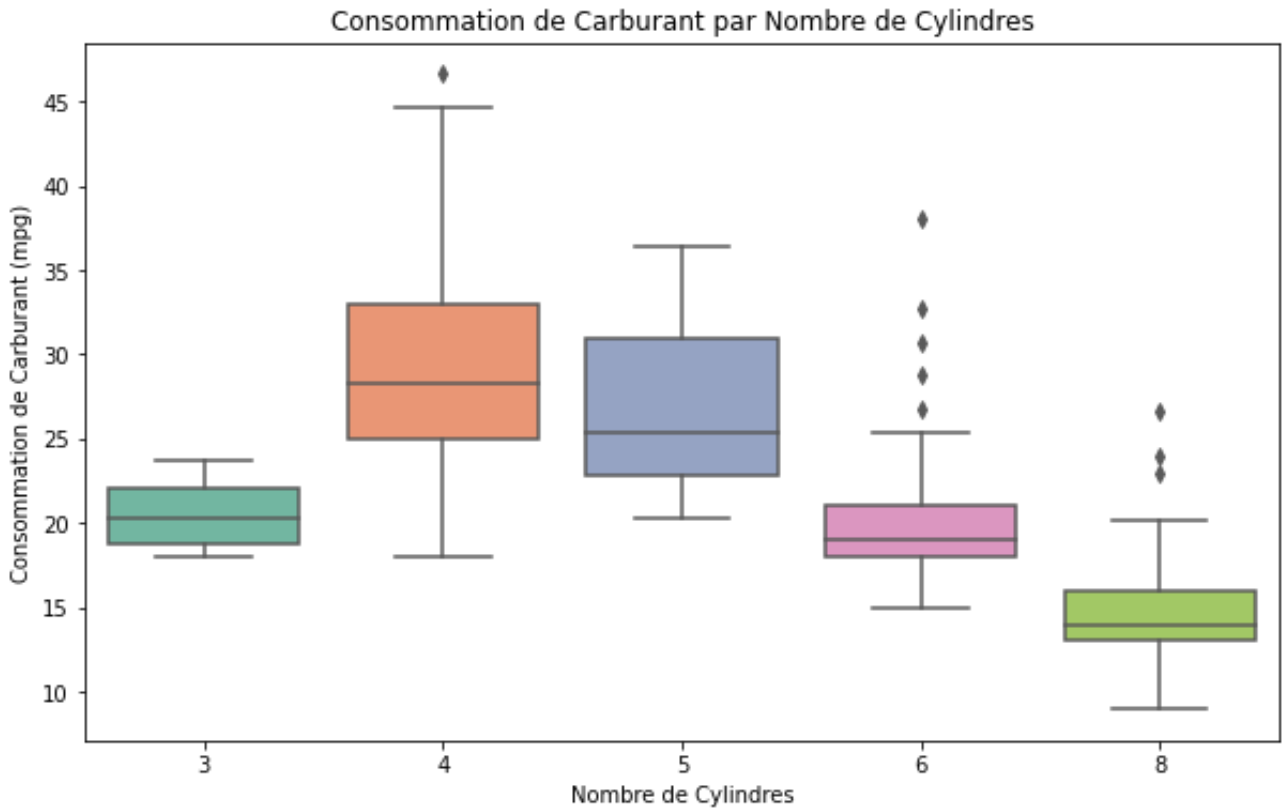
4 Cylindres : Les voitures avec 4 cylindres montrent la plus grande variabilité et la médiane la plus élevée (28 mpg), suggérant que ces voitures sont généralement plus efficaces et peuvent parcourir plus de miles par gallon de carburant.

5 Cylindres : La médiane est intermédiaire, Cela suggère une efficacité énergétique modérée.

6 Cylindres : La médiane est inférieure à celle des 4 cylindres, et il y a quelques valeurs aberrantes au-dessus de la boîte. indiquant une efficacité énergétique moindre.

8 Cylindres : Ces voitures ont l'efficacité de consommation la plus basse, avec une médiane basse (14 mpg) et plusieurs valeurs aberrantes. Cela signifie qu'elles sont les moins efficaces en termes de consommation de carburant.

Graphique produit :



Création de graphiques avec Python : Box plot

Conclusion de l'analyse de l'application CarsMPG :

Le box plot montre que les voitures avec 4 cylindres sont les plus efficaces en termes de consommation de carburant, pouvant parcourir plus de miles par gallon. Les voitures avec 8 cylindres, en revanche, sont les moins efficaces, parcourant moins de miles par gallon de carburant.

Les valeurs plus élevées de miles per gallon (mpg) indiquent une meilleure performance énergétique, ce qui signifie que les voitures qui ont des valeurs élevées sur l'axe Y sont meilleures en termes de consommation de carburant.

PARTIE 2

Création de graphiques avec Python : Bubble Chart (Graphique à bulles)

La Relation entre le GDP , l'espérance de vie et la population :

Pour analyser la relation entre le PIB par habitant, l'espérance de vie et la population, nous devons créer un graphique à bulles interactif.

Le graphique à bulles vous de visualiser facilement au moins trois dimensions essentielles :

- PIB par habitant **(X)** : Représentant la richesse économique par personne.
- Espérance de vie **(Y)** : Indiquant la santé et le développement des pays.
- Population **(taille de la bulle)** : Montrant la taille relative de la population de chaque pays.

•Note : **Les couleurs** des bulles peut être utiliser pour représentent les continents, facilitant ainsi la distinction géographique des données.

Voici les étapes et le script Python correspondant pour réaliser cette visualisation avec Plotly Express :

```
# Importer la bibliothèque Plotly Express :
import plotly.express as px

# Charger les données Gapminder
df = px.data.gapminder()
```

Plotly Express est un package de visualisation de données qui vous permet de créer des plots interactifs avec moins de code.

Objectif : Utiliser les données Gapminder, qui contiennent des informations sur le développement des pays, pour notre analyse.

Création de graphiques avec Python : Bubble Chart (Graphique à bulles)

La Relation entre le GDP , l'espérance de vie et la population :

```
# Filtrer les données pour l'année 2007 et créer le graphique à bulles
fig = px.scatter(
    df.query("year==2007"),
    x="gdpPercap",
    y="lifeExp",
    size="pop",
    color="continent",
    hover_name="country",
    log_x=True,
    size_max=60
)

# Afficher le graphique
fig.show()
```

Paramètres du graphique à bulles :

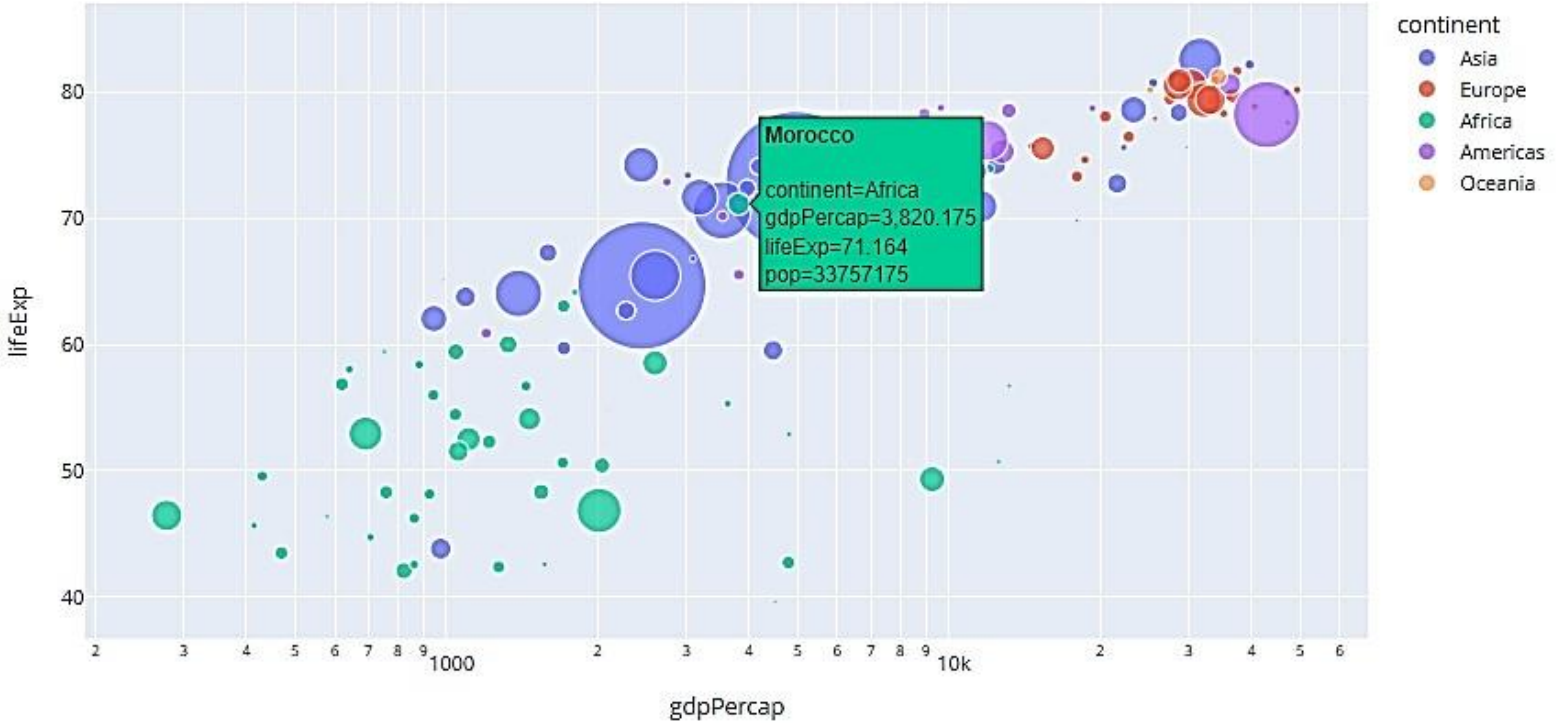
- df.query("year==2007") : Sélectionner uniquement les données de l'année 2007 pour une analyse précise.
- x="gdpPercap" : L'axe des X représente le PIB par habitant.
- y="lifeExp" : L'axe des Y représente l'espérance de vie.
- size="pop" : La taille des bulles représente la population des pays.
- color="continent" : Les bulles sont colorées en fonction des continents pour une distinction géographique.
- hover_name="country" : Le nom du pays s'affiche lors du survol de la bulle pour des informations supplémentaires.
- log_x=True : L'échelle logarithmique de l'axe des X permet de mieux visualiser les différences de PIB par habitant.
- size_max=60 : Limite la taille maximale des bulles pour une meilleure lisibilité.

Création de graphiques avec Python : Bubble Chart (Graphique à bulles)

La Relation entre le GDP , l'espérance de vie et la population :

Pour tracer le PIB par habitant (X), l'espérance de vie (Y) et la population (taille de la bulle) avec les couleurs représentant les continents, utilisez des données de Kaggle ou une bibliothèque. Pour visualiser les données récentes, téléchargez « World Banc Data » depuis Kaggle.

Lors du survol «hovering» de chaque point, vous pouvez maintenant voir le nom du pays, ce qui représente la cinquième dimension. Cela permet d'identifier facilement les pays en plus des quatre autres dimensions (PIB par habitant, espérance de vie, population, et continent). Cette fonctionnalité améliore la compréhension et l'interactivité du graphique.



PARTIE 2

Création de graphiques avec Python : Bubble Chart (Graphique à bulles)

Recommandation : Résolution des Erreurs d'importation (Troubleshooting Import Errors)

Dans le cadre de l'utilisation de la bibliothèque Plotly pour des visualisations interactives en Python, il est possible de rencontrer des erreurs d'importation. la section actuelle décrit le problème rencontré, les tentatives de résolution, et les mesures correctives prises pour garantir le bon fonctionnement de la bibliothèque.

Problème : Lors de l'importation de Plotly, l'erreur suivante a été rencontrée :

```
In [5]: import plotly
        plotly.__version__

---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_9000/632329486.py in <module>
----> 1 import plotly
      2 plotly.__version__

ModuleNotFoundError: No module named 'plotly'
```

PARTIE 2

Création de graphiques avec Python : Bubble Chart (Graphique à bulles)

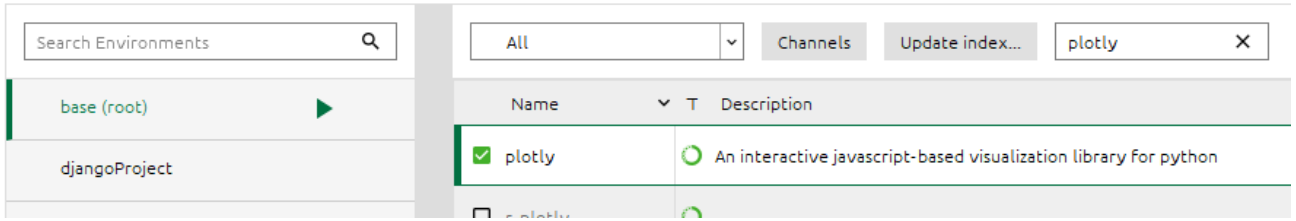
Recommandation : Résolution des Erreurs d'importation (Troubleshooting Import Errors)

Tentatives de résolution :

1. Installation via Conda :

Une tentative d'installation de Plotly en utilisant Conda a été effectuée :

```
In [3]: ! conda install plotly

        Collecting package metadata (current_repodata.json): ...working... done
        Solving environment: ...working... done

        # All requested packages already installed.


        ==> WARNING: A newer version of conda exists. <==
```



Par l'utilisation de la commande « conda install »
Ou par l'interface « Anaconda Nnavigator / environment »

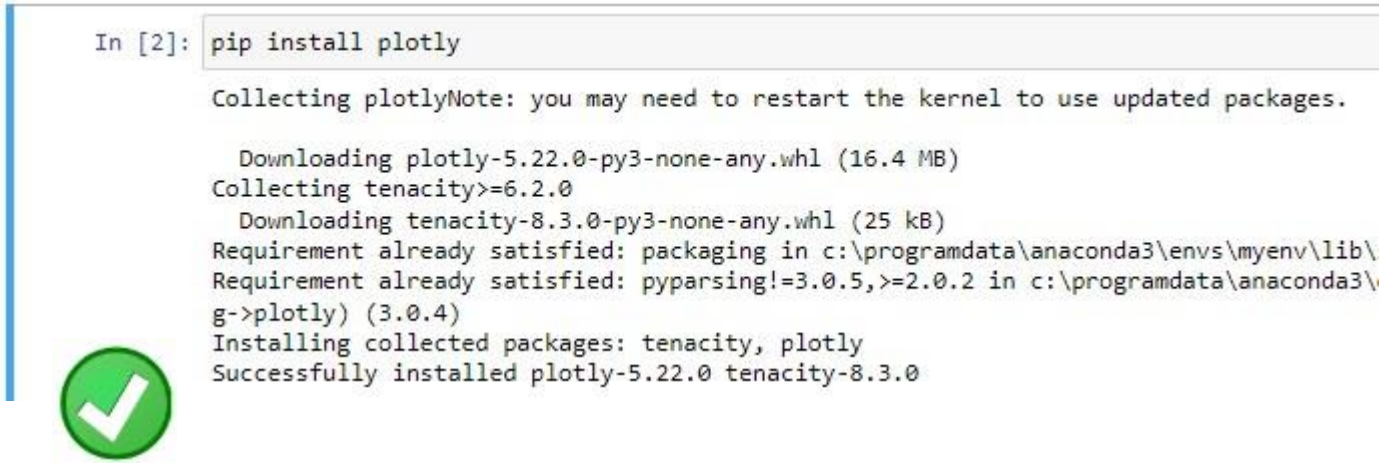
Si le package ne fonctionne toujours pas, nous pouvons mettre à jour l'ensemble de l'environnement conda, ou simplement utiliser la solution suivante.

Création de graphiques avec Python : Bubble Chart (Graphique à bulles)

Recommandation : Résolution des Erreurs d'importation (Troubleshooting Import Errors)

2. Installation via Pip :

Une seconde tentative d'installation de Plotly a été effectuée en utilisant Pip :



PARTIE 2

Création de graphiques avec Python : Bubble Chart (Graphique à bulles)

Recommandation : Résolution des Erreurs d'importation (Troubleshooting Import Errors)

Vérification :

Après l'installation via Pip, il est recommandé de redémarrer le noyau (kernel) de l'environnement Jupyter pour appliquer les modifications.

Voici les étapes à suivre :

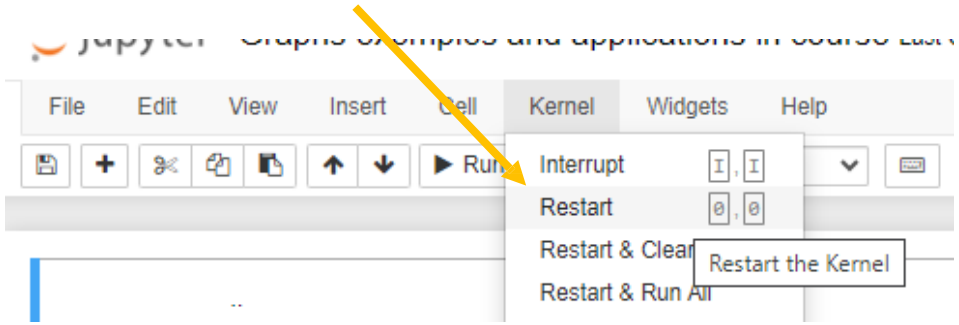
1. Redémarrage du kernel :

Après l'installation de Plotly, redémarrez le kernel en allant dans le menu Kernel et en sélectionnant Restart Kernel.

2. Vérification de l'importation :

Après le redémarrage du kernel, réessayez d'importer Plotly :

```
import plotly
print(plotly.__version__)
```



Si l'installation et le redémarrage ont été effectués correctement, l'importation de Plotly devrait fonctionner sans erreur et la version de Plotly devrait s'afficher.

Création de graphiques avec Excel :

Remarque :

Le prochain chapitre de la formation "CONFIGURER DES GRAPHIQUES POUR UNE ANALYSE APPROFONDIE" abordera la création de graphiques dans Excel. Dans la section "Choisir les graphiques en fonction de l'objectif et des données", nous ne nous concentrerons pas uniquement sur la création des graphiques. De plus nous verrons pourquoi un type de graphique est préférable à un autre, en fonction de nos objectifs et des données disponibles. Ainsi, vous apprendrez à sélectionner le graphique le plus approprié pour représenter vos données de manière claire et efficace, afin de répondre aux besoins spécifiques de votre projet d'analyse.

CHAPITRE 2

CONFIGURER LES GRAPHIQUES POUR UNE ANALYSE PROFONDE

Ce que vous allez apprendre dans ce chapitre :

- Choisir les graphiques en fonction d'objectif
- Choisir les graphiques en fonction des données
- Explorer la visualisation multivariée
- Utiliser stratégiquement les couleurs et les annotations
- Sélectionner les graphiques avancés avec Python/Excel

30 heures