

# Doc Data Science

Data science c'est quoi ?

La data science est un domaine multidisciplinaire qui utilise des techniques statistiques, de l'analyse de données, et des algorithmes informatiques pour extraire des connaissances et à partir de données structurées et non structurées.

Elle sert à plusieurs fins :

1. **Prise de décision basée sur les données :** En transformant de grandes quantités de données en insights (un insight est une révélation importante ou une compréhension profonde obtenue à travers l'analyse des données, qui peut guider les actions, les stratégies et les décisions, pour une entreprise par exemple.) compréhensibles, la data science aide les entreprises et les organisations à prendre des décisions éclairées.
2. **Prévision et modélisation :** Elle permet de construire des modèles prédictifs pour anticiper les tendances futures, comme la demande des consommateurs ou les mouvements du marché.
3. **Personnalisation et recommandation :** Les algorithmes de data science sont utilisés pour personnaliser l'expérience utilisateur, comme dans les systèmes de recommandation de Netflix ou Amazon entre autres.
4. **Automatisation et efficacité opérationnelle :** La data science peut optimiser les processus opérationnels, réduire les coûts et améliorer l'efficacité.
5. **Détection de fraudes et gestion des risques :** Elle est cruciale pour identifier les activités suspectes dans les secteurs comme la finance ou la cyber-sécurité.
6. **Recherche et développement :** La data science stimule l'innovation en analysant les tendances et en aidant à la conception de nouveaux produits ou services.
7. **Santé et médecine :** Elle joue un rôle clé dans le développement de traitements personnalisés, la gestion des soins de santé et la recherche médicale.

La data science est essentielle dans le monde moderne pour transformer les données brutes en informations utiles, guidant ainsi les décisions stratégiques et opérationnelles dans divers secteurs.

# Les différents langages de programmation

En data science, plusieurs langages de programmation sont utilisés, chacun ayant ses propres bibliothèques et outils spécialisés. Les plus populaires sont Python, R, SQL, et dans une moindre mesure, Java, Scala, et Julia.

## Python

1. **Pandas** : Pour la manipulation et l'analyse de données. Permet de travailler facilement avec des données structurées.
2. **NumPy** : Pour les opérations numériques, particulièrement utile avec les tableaux multidimensionnels.
3. **SciPy** : Utilisé pour les calculs scientifiques et techniques.
4. **Matplotlib et Seaborn** : Pour la visualisation de données.
5. **Scikit-learn** : Pour le machine learning, offrant des outils simples et efficaces pour l'analyse prédictive.
6. **TensorFlow et PyTorch** : Pour le deep learning et les réseaux neuronaux.
7. **Jupyter Notebook** : Environnement interactif pour la programmation et la visualisation.

Dans cette documentation seul Python sera expliqué car je l'utilise.

Chaque langage a ses points forts. Python est réputé pour sa simplicité et sa grande communauté, R est privilégié pour les statistiques avancées et la visualisation de données, tandis que SQL est incontournable pour la manipulation de bases de données. Java et Scala sont souvent choisis pour les applications à grande échelle, et Julia pour les calculs à haute performance.

# Exercice avec un dataset sur python et jupyter

Pré-requis important pour cet exo :

Il faudra d'abord installer python, ensuite il faut installer les différentes bibliothèques nécessaires.

Pour pandas dans le terminal : **pip install pandas**

Pour matplotlib dans le terminal : **pip install matplotlib**

Pour numpy dans le terminal : **pip install numpy**

Ensuite nous allons installer l'extensions **Jupyter Notebook** sur vscode ( ou autres IDE)

Pour finir il nous faudra un dataset pour la pratique, j'ai choisi un dataset qui répertorie les ventes de jeux vidéos jusqu'en 2020 , le dataset porte le nom de **vgsales.csv**

L'exercice consistera en plusieurs points :

1. **Lire le Dataset:** lire le fichier CSV dans un DataFrame Pandas.
2. Afficher les premières et dernières lignes du DataFrame.
3. Vérifier les types de données de chaque colonne.
4. Vérifier s'il y a des valeurs manquantes dans le dataset.
5. Calculer les statistiques de base (moyenne, médiane, écart-type, etc.) pour les colonnes numériques.
6. Compter le nombre de jeux par plateforme et par genre.
7. Trier les jeux par ventes globales en ordre décroissant.
8. Filtrer pour afficher uniquement les jeux d'un genre spécifique, par exemple, les jeux d'action ou autres.
9. Grouper les jeux par année et calculer les ventes totales par année.
10. Grouper les jeux par plateforme et calculer les ventes moyennes par plateforme.
11. Créer une nouvelle colonne qui représentera le ratio des ventes en Europe par rapport aux ventes globales.
12. Convertir les années en décennies et créer une nouvelle colonne pour cela.

13. Créer un histogramme des ventes globales.
14. Créer un diagramme en barres des nombres de jeux par plateforme.
15. Créer un diagramme en barres empilées montrant les ventes par région pour les 5 jeux les plus vendus.
16. Créer un graphique en ligne pour montrer l'évolution des ventes globales au fil des années.
17. Utiliser NumPy pour calculer des statistiques personnalisées qui ne sont pas directement disponibles dans Pandas.
18. Créer des filtres complexes en utilisant des opérations NumPy sur des colonnes.
19. Analyser la corrélation entre les ventes dans différentes régions.

**LET'S GO !**

## 1) Lire le Dataset: lire le fichier CSV dans un DataFrame Pandas.

Nous allons avant toute chose importer nos bibliothèques

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

✓ 3.5s

Puis lire notre dataframe (le csv)

```
# Je lis le fichier csv
# df signifie data frame
# pd signifie pandas
# read_csv signifie lire un fichier csv
df = pd.read_csv('vgsales.csv')
```

✓ 0.0s

Voici le résultat :

index	Rank	Name	Platform	Year	Genre	Publis...	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mari...	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart ...	Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports ...	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33
4	5	Pokemon R...	GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1	31.37
5	6	Tetris	GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26
6	7	New Super ...	DS	2006	Platform	Nintendo	11.38	9.23	6.5	2.9	30.01
7	8	Wii Play	Wii	2006	Misc	Nintendo	14.03	9.2	2.93	2.85	29.02
8	9	New Super ...	Wii	2009	Platform	Nintendo	14.59	7.06	4.7	2.26	28.62
9	10	Duck Hunt	NES	1984	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31

## 2) Afficher les premières et dernières lignes du DataFrame.

La vue des 5 premiers

```
#Affiche les premières ligne du DataFrame.
# df.dropna() supprime les lignes avec des valeurs manquantes
df = df.dropna()
# .head() affiche les 5 premières lignes
df.head()
```

✓ 0.0s

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37

## La vue des 5 derniers

```
#Affiche les dernières ligne du DataFrame.
# df.dropna() supprime les lignes avec des valeurs manquantes
df = df.dropna()
# .tail() affiche les 5 dernières lignes
df.tail()
```

✓ 0.0s

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
16593	16596	Woody Woodpecker in Crazy Castle 5	GBA	2002.0	Platform	Kemco	0.01	0.00	0.0	0.0	0.01
16594	16597	Men in Black II: Alien Escape	GC	2003.0	Shooter	Infogrames	0.01	0.00	0.0	0.0	0.01
16595	16598	SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	Activision	0.00	0.00	0.0	0.0	0.01
16596	16599	Know How 2	DS	2010.0	Puzzle	7G//AMES	0.00	0.01	0.0	0.0	0.01
16597	16600	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00	0.0	0.0	0.01

### 3) Vérifier les types de données de chaque colonne.

```
# Vérifier les types de données de chaque colonne.
# .dtypes affiche les types de données de chaque colonne.
df.dtypes
```

✓ 0.0s

Rank	int64
Name	object
Platform	object
Year	float64
Genre	object
Publisher	object
NA_Sales	float64
EU_Sales	float64
JP_Sales	float64
Other_Sales	float64
Global_Sales	float64
dtype:	object

4) Vérifier s'il y a des valeurs manquantes dans le dataset.

```
# Vérifier s'il y a des valeurs manquantes dans le dataset.  
# .isnull() affiche les valeurs manquantes dans le dataset.  
# .sum() affiche le nombre de valeurs manquantes dans le dataset.  
df.isnull().sum()
```

✓ 0.0s

Rank	0
Name	0
Platform	0
Year	0
Genre	0
Publisher	0
NA_Sales	0
EU_Sales	0
JP_Sales	0
Other_Sales	0
Global_Sales	0
dtype:	int64

5) Calculer les statistiques de base (moyenne, médiane, écart-type, etc.) pour les colonnes numériques.

```
# Calculer les statistiques de base (moyenne, médiane, écart-type, etc.) pour les colonnes numériques.  
# .describe() affiche les statistiques de base (moyenne, médiane, écart-type, etc.) pour les colonnes numériques.  
df.describe()
```

✓ 0.0s

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
count	16291.000000	16291.000000	16291.000000	16291.000000	16291.000000	16291.000000	16291.000000
mean	8290.190228	2006.405561	0.265647	0.147731	0.078833	0.048426	0.540910
std	4792.654450	5.832412	0.822432	0.509303	0.311879	0.190083	1.567345
min	1.000000	1980.000000	0.000000	0.000000	0.000000	0.000000	0.010000
25%	4132.500000	2003.000000	0.000000	0.000000	0.000000	0.000000	0.060000
50%	8292.000000	2007.000000	0.080000	0.020000	0.000000	0.010000	0.170000
75%	12439.500000	2010.000000	0.240000	0.110000	0.040000	0.040000	0.480000
max	16600.000000	2020.000000	41.490000	29.020000	10.220000	10.570000	82.740000

Ce tableau est le résultat de la méthode `.describe()` de Pandas, qui fournit des statistiques descriptives qui résument la tendance centrale, la dispersion et la forme de la distribution d'un dataset, excluant les valeurs NaN.

## Explication des différentes données du tableau :

1. **count:** Le nombre de valeurs non-nulles pour chaque colonne. Ici, chaque colonne a 16291 valeurs non-nulles, ce qui indique qu'il n'y a pas de valeurs manquantes dans ces colonnes numériques.
2. **mean:** La moyenne des valeurs. Pour les ventes, cela représente les ventes moyennes par jeu dans chaque région (par exemple, en moyenne, un jeu a vendu environ 0.26 millions d'unités en Amérique du Nord).
3. **std:** L'écart-type, qui mesure la quantité de variation ou de dispersion des valeurs. Un faible écart-type indique que les valeurs sont proches de la moyenne du jeu, tandis qu'un écart-type élevé indique que les valeurs sont réparties sur une plus large gamme de valeurs.
4. **min:** La plus petite valeur. Pour les années, cela indique que le jeu le plus ancien de ce dataset est sorti en 1980. Pour les ventes, cela montre que le minimum enregistré est de 0 (ce qui suggère que certains jeux n'ont pas réalisé de ventes dans certaines régions).
5. **25%:** Le 1er quartile, ou la médiane des données inférieures, indique que 25% des valeurs sont inférieures à cette valeur.
6. **50%:** La médiane, ou le 2eme quartile, est la valeur qui sépare la moitié inférieure de la moitié supérieure des données. Pour les années, par exemple, cela signifie que la moitié des jeux sont sortis en 2007 ou avant.
7. **75%:** Le 3eme quartile, 75% des valeurs sont inférieures à cette valeur.
8. **max:** La plus grande valeur. Pour les années, cela montre que le jeu le plus récent dans le dataset est sorti en 2020. Pour les ventes, cela montre les ventes maximales enregistrées par région et globalement.

Ces statistiques sont utiles pour obtenir une vue d'ensemble des données numériques d'un dataset, en particulier pour comprendre la distribution et l'échelle des valeurs des différentes colonnes.



## 6) Compter le nombre de jeux par plateforme et par genre.

```
# Compter le nombre de jeux par plateforme et par genre.
# count_games permet de compter le nombre de jeux par plateforme et par genre.
# .groupby permet de grouper les données par plateforme et par genre.
# .count() permet de compter les données.
# .reset_index() permet de réinitialiser les index.
# name permet de renommer la colonne.
count_games = df.groupby(['Platform', 'Genre'])['Name'].count().reset_index(name='Count')

# sorted_count_games permet de trier count_games par plateforme et par nombre de jeux.
# .sort_values permet de trier les données.
# ['Platform', 'Count'] permet de trier par plateforme et par nombre de jeux.
# ascending=[True, False] permet de trier par ordre croissant et décroissant.
sorted_count_games = count_games.sort_values(['Platform', 'Count'], ascending=[True, False])

# print sert à afficher les données.
print(sorted_count_games)
```

Voici un échantillon du résultat :

index	Platfo...	Genre	Count
46	GB	Action	6
47	GB	Adventure	5
54	GB	Simulation	5
51	GB	Racing	2
53	GB	Shooter	1
57	GBA	Action	162
61	GBA	Platform	139
67	GBA	Sports	88
60	GBA	Misc	86
64	GBA	Role-Play...	73
63	GBA	Racing	64
65	GBA	Shooter	40
62	GBA	Puzzle	39
58	GBA	Adventure	36
59	GBA	Fighting	23
66	GBA	Simulation	18

7) Trier les jeux par ventes globales en ordre décroissant.

```
#Trier les jeux par ventes globales en ordre décroissant.

# Trier les jeux par ventes globales ('Global_Sales') en ordre décroissant
sorted_by_global_sales = df.sort_values(by='Global_Sales', ascending=False)

# Afficher les résultats triés
print(sorted_by_global_sales)
```

Voici son résultat :

index	Rank	Name	Platfo...	Year	Genre	Publis...	NA_Sa...	EU_Sal...	JP_Sales	Other...	Global_Sales	↓
0	1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74	
1	2	Super M...	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24	
2	3	Mario Ka...	Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82	
3	4	Wii Sport...	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33	
4	5	Pokemon...	GB	1996	Role-Play...	Nintendo	11.27	8.89	10.22	1	31.37	
5	6	Tetris	GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26	
6	7	New Sup...	DS	2006	Platform	Nintendo	11.38	9.23	6.5	2.9	30.01	

Ce qui nous intéresse ici c'est le Global\_Sales on voit bien que c'est décroissant.

8) Filtrer pour afficher uniquement les jeux d'un genre spécifique, par exemple, les jeux d'action ou autres.

```
# Filtrer pour afficher uniquement les jeux d'un genre spécifique, par exemple, les jeux d'action ou autres.
# .loc permet de filtrer les données.
# .loc permet de sélectionner les données.
# ['Genre'] permet de sélectionner les données par genre.
# ['Action'] permet de sélectionner les données par genre action.
# .head() permet d'afficher les 5 premières lignes.
df.loc[df['Genre'] == 'Action'].head()
```

✓ 0.0s

Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
16	Grand Theft Auto V	PS3	2013.0	Action	Take-Two Interactive	7.01	9.27	0.97	4.14	21.40
17	Grand Theft Auto: San Andreas	PS2	2004.0	Action	Take-Two Interactive	9.43	0.40	0.41	10.57	20.81
23	Grand Theft Auto V	X360	2013.0	Action	Take-Two Interactive	9.63	5.31	0.06	1.38	16.38
24	Grand Theft Auto: Vice City	PS2	2002.0	Action	Take-Two Interactive	8.41	5.49	0.47	1.78	16.15
38	Grand Theft Auto III	PS2	2001.0	Action	Take-Two Interactive	6.99	4.51	0.30	1.30	13.10

9) Grouper les jeux par année et calculer les ventes totales par année.

```
# Grouper les jeux par année et calculer les ventes totales par année.  
# .groupby permet de grouper les données.  
# ['Year'] permet de grouper les données par année.  
# ['Global_Sales'] permet de grouper les données par ventes globales.  
# .sum() permet de calculer les ventes totales par année.  
# .reset_index() permet de réinitialiser les index.  
# name permet de renommer la colonne.  
df.groupby(['Year'])['Global_Sales'].sum().reset_index(name='Global_Sales')
```

✓ 0.0s

	Year	Global_Sales
0	1980.0	11.38
1	1981.0	35.77
2	1982.0	28.86
3	1983.0	16.79
4	1984.0	50.36
5	1985.0	53.94
6	1986.0	37.07
7	1987.0	21.74
8	1988.0	47.22

Ici les Global\_Sales sont en millions donc pour l'année 1980 c'est écrit 11.38millions d'unités vendu.

10) Grouper les jeux par plateforme et calculer les ventes moyennes par plateforme.

```
# Grouper les jeux par plateforme et calculer les ventes moyennes par plateforme.  
# .groupby permet de grouper les données.  
# ['Platform'] permet de grouper les données par plateforme.  
# ['Global_Sales'] permet de grouper les données par ventes globales.  
# .mean() permet de calculer les ventes moyennes par plateforme.  
# .reset_index() permet de réinitialiser les index.  
# name permet de renommer la colonne.  
df.groupby(['Platform'])['Global_Sales'].mean().reset_index(name='Global_Sales')
```

✓ 0.0s

	Platform	Global_Sales
0	2600	0.746293
1	3DO	0.033333
2	3DS	0.493527
3	DC	0.307115
4	DS	0.384284
5	GB	2.622887
6	GBA	0.388830
7	GC	0.363727
8	GEN	1.050370
9	GG	0.040000
10	N64	0.690538
11	NES	2.561939
12	NG	0.120000
13	PC	0.271535
14	PCFX	0.030000
15	PS	0.611766

Dans ce tableau, la colonne **Global\_Sales** représente les ventes moyennes mondiales des jeux par plateforme, exprimées en millions d'unités. Le calcul de la moyenne se fait en prenant la somme totale des ventes mondiales de tous les jeux pour une plateforme donnée et en divisant cette somme par le nombre total de jeux disponibles sur cette plateforme.

Par exemple, pour la plateforme 2600 (c'est l'atari 2600), la vente moyenne globale par jeu est d'environ 0.746293 millions d'unités. Cela signifie que, en moyenne, un jeu sur Atari 2600 a vendu environ 746,293 exemplaires à travers le monde.

11) Créer une nouvelle colonne qui représentera le ratio des ventes en Europe par rapport aux ventes globales.

```
#Créer une nouvelle colonne qui représentera le ratio des ventes en Europe par rapport aux ventes globales.
# .assign permet de créer une nouvelle colonne.
# ['EU_Sales'] permet de sélectionner les données par ventes en Europe.
# ['Global_Sales'] permet de sélectionner les données par ventes globales.
# lambda permet de créer une fonction anonyme.
# x permet de sélectionner les données.
# x[0] permet de sélectionner les données de la première colonne.
# x[1] permet de sélectionner les données de la deuxième colonne.
# *100 permet de multiplier les données par 100.
# .round permet d'arrondir les données.
# 2 permet d'arrondir les données à 2 chiffres après la virgule.
df = df.assign(EU_Sales_Ratio=lambda x: x['EU_Sales'] / x['Global_Sales'] * 100).round(2)
df.head()
```

✓ 0.0s

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	EU_Sales_Ratio
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	35.07	3.77	8.46	82.74	42.39
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	8.90	6.81	0.77	40.24	22.12
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	35.96	3.79	3.31	35.82	100.39
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	33.36	3.28	2.96	33.00	101.09
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	28.34	10.22	1.00	31.37	90.34

La nouvelle colonne est calculée à partir des ventes en Europe (EU\_Sales) divisées par les ventes mondiales (Global\_Sales), et le résultat est multiplié par 100 pour obtenir un pourcentage.

**EU\_Sales\_Ratio:** Ce ratio exprime la part des ventes européennes dans les ventes mondiales d'un jeu donné, en pourcentage. Par exemple, un ratio de 42.39 pour "Wii Sports" signifie que 42.39% des ventes mondiales de ce jeu proviennent d'Europe.

Si le ratio est supérieur à 100, comme pour "Wii Sports Resort" avec un ratio de 101.09, cela signifie que les chiffres des ventes en Europe sont plus élevés que les ventes mondiales enregistrées, ce qui pourrait indiquer **une erreur dans les données** ou un problème avec la méthode de calcul. Habituellement, un ratio de vente devrait être inférieur à 100, car les ventes régionales sont une partie des ventes mondiales.

Un ratio élevé indique une popularité relative du jeu en Europe par rapport aux autres marchés. Par exemple, "Pokemon Red/Pokemon Blue" a un ratio de 90.34, ce qui suggère que ce jeu était particulièrement populaire en Europe en comparaison avec le reste du monde.

## 12) Convertir les années en décennies et créer une nouvelle colonne pour cela.

```
# Convertir les années en décennies et créer une nouvelle colonne pour cela.
# .astype permet de convertir les données.
# int permet de convertir les données en entier.
# /10 permet de diviser les données par 10.
# *10 permet de multiplier les données par 10.
df['Decade'] = (df['Year'].astype(int) / 10).astype(int) * 10
df.head()
```

✓ 0.0s

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	EU_Sales_Ratio	Decade
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	35.07	3.77	8.46	82.74	42.39	2000
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	8.90	6.81	0.77	40.24	22.12	1980
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	35.96	3.79	3.31	35.82	100.39	2000
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	33.36	3.28	2.96	33.00	101.09	2000
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	28.34	10.22	1.00	31.37	90.34	1990

## 13) Créer un histogramme des ventes globales.

```
# bins permet de créer des intervalles.
# np.arange permet de créer des intervalles.
# 0 permet de commencer à 0.
# df['Global_Sales'].max() permet de terminer à la valeur maximale de la colonne 'Global_Sales'.
# +0.1 permet d'ajouter 0.1 à la valeur maximale de la colonne 'Global_Sales'.
bins = np.arange(0, df['Global_Sales'].max() + 0.1, 0.1)

# plt.hist permet de créer un histogramme.
# df['Global_Sales'] permet de sélectionner les données par ventes globales.
# bins permet de créer des intervalles.
# color permet de choisir une couleur.
# alpha permet de choisir la transparence de la couleur.
plt.hist(df['Global_Sales'], bins=80, color='blue', alpha=0.7)

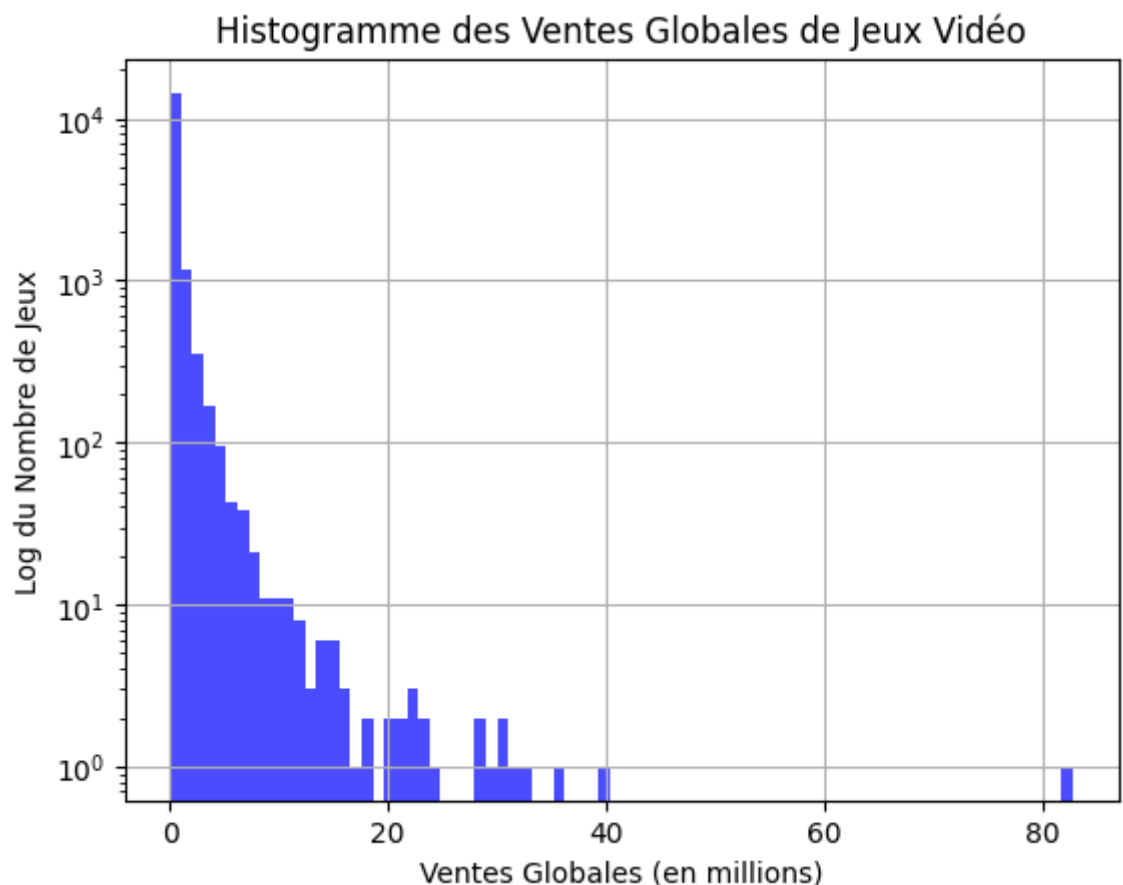
# plt.yscale permet de changer l'échelle de l'axe des ordonnées.
# 'log' permet de changer l'échelle de l'axe des ordonnées en logarithmique.
# c'est à dire que l'échelle de l'axe des ordonnées est en puissance de 10.
# ce qui permet de mieux visualiser les données.
plt.yscale('log')

# Définir les titres et étiquettes
plt.title('Histogramme des Ventes Globales de Jeux Vidéo')
plt.xlabel('Ventes Globales (en millions)')
plt.ylabel('Log du Nombre de Jeux')

# Afficher des lignes de grille pour une meilleure lisibilité
plt.grid(True)

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

Voici le résultat :



L'histogramme présenté montre la distribution des ventes globales de jeux vidéo, avec les ventes globales en millions sur l'axe des x et le log du nombre de jeux sur l'axe des y.

L'axe des x, représentant les ventes globales, est linéaire et montre des segments allant de 0 à plus de 80 millions d'unités. Chaque barre de l'histogramme correspond à une plage de ventes et la hauteur de la barre représente le nombre de jeux qui tombent dans cette plage de ventes.

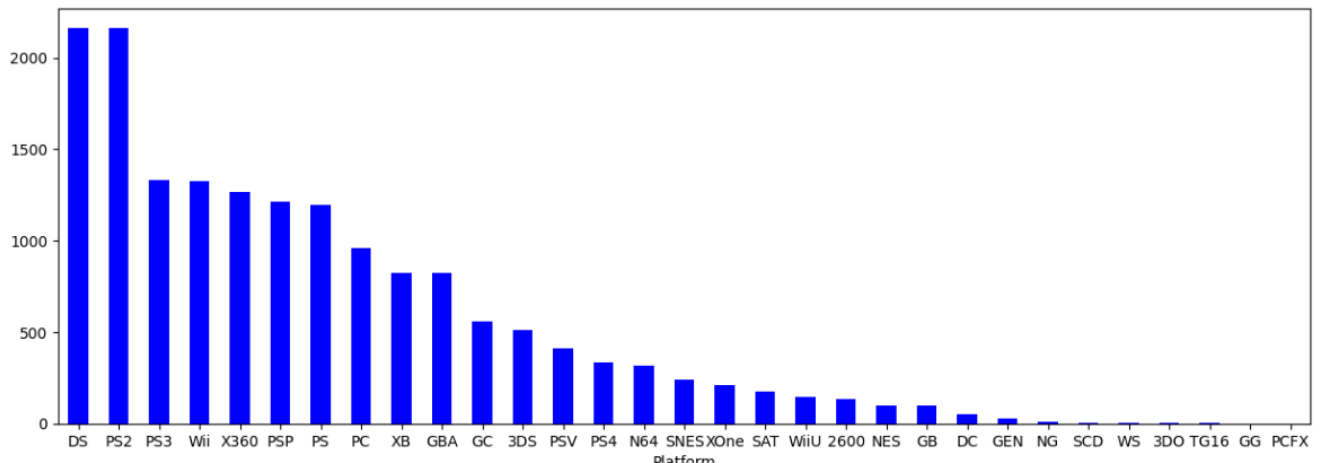
L'axe des y est en échelle logarithmique, ce qui signifie que chaque unité d'augmentation sur l'axe des y représente une multiplication par 10 du nombre de jeux. Par exemple, une valeur de 10 puissance 1 (ou 10 sur l'axe des y) signifie 10 jeux, 10 puissance 2 signifie 100 jeux, 10 puissance 3 signifie 1 000 jeux, et ainsi de suite. Cela est utilisé pour mieux visualiser les données lorsque avec une large gamme de valeurs, comme c'est le cas ici où quelques jeux se vendent exceptionnellement bien, tandis que la plupart se vendent en quantités beaucoup plus modestes.

L'histogramme montre clairement que la grande majorité des jeux vidéo se vendent à moins de 10 millions d'unités, avec un pic très prononcé dans la première barre, représentant les jeux avec des ventes très faibles. La distribution décroît rapidement, ce qui indique que très peu de jeux atteignent des niveaux de vente élevés. Les jeux qui se vendent à plus de 10 millions d'unités sont beaucoup moins fréquents, comme illustré par les barres plus basses sur la droite de l'histogramme.

14) Créer un diagramme en barres des nombres de jeux par plateforme.

```
# Créer un diagramme en barres des nombres de jeux par plateforme.  
# .value_counts permet de compter les données.  
# .plot.bar permet de créer un diagramme en barres.  
# color permet de choisir une couleur.  
# figsize permet de choisir la taille du graphique.  
# rot permet de choisir l'angle des étiquettes.  
df['Platform'].value_counts().plot.bar(color='blue', figsize=(15, 5), rot=0)
```

Voici le résultat :

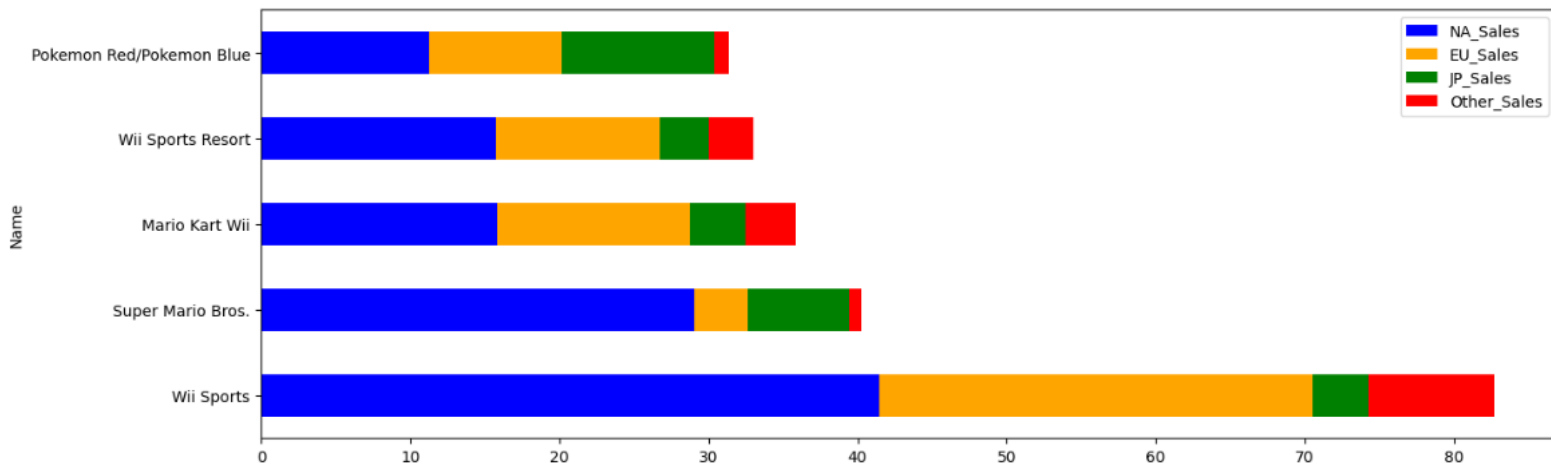


15) Créer un diagramme en barres empilées montrant les ventes par région pour les 5 jeux les plus vendus.

```
# Créer un diagramme en barres empilées montrant les ventes par région pour les 5 jeux les plus vendus.  
# .head() permet de sélectionner les 5 premières lignes.  
# .set_index permet de sélectionner les données.  
# ['Name'] permet de sélectionner les données par nom.  
# ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales'] permet de sélectionner les données par ventes par région.  
# .plot.barh permet de créer un diagramme en barres empilées.  
# stacked permet de créer un diagramme en barres empilées.  
# color permet de choisir une couleur.  
# figsize permet de choisir la taille du graphique.  
# rot permet de choisir l'angle des étiquettes.  
df.head().set_index('Name')[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']].plot.barh(stacked=True, color=['blue', 'orange', 'green', 'red'])
```



Voici le résultat :



Explication : La longueur de chaque couleur dans la barre représente la quantité de ventes dans cette région spécifique pour le jeu donné. Lorsque les couleurs sont empilées les unes sur les autres, la longueur totale de la barre représente les ventes globales du jeu.

Par exemple, la barre la plus longue, qui correspond à "Wii Sports", montre que ce jeu a les ventes les plus élevées parmi les jeux listés, avec une répartition notable des ventes entre l'Amérique du Nord et l'Europe, et une portion plus petite pour le Japon et les autres régions.

16) Utiliser NumPy pour calculer des statistiques personnalisées qui ne sont pas directement disponibles dans Pandas.

```
# global_sales_variance permet de calculer la variance des ventes globales.
# np.var permet de calculer la variance.
global_sales_variance = np.var(df['Global_Sales'])

# global_sales_median permet de calculer l'écart type médian des ventes globales.
# np.median permet de calculer la médiane.
global_sales_median = np.median(df['Global_Sales'])

# global_sales_mad permet de calculer l'écart type médian des ventes globales.
# np.mean permet de calculer la moyenne.
# np.abs permet de calculer la valeur absolue.
global_sales_mad = np.mean(np.abs(df['Global_Sales'] - global_sales_median))

# Afficher les résultats
print(f"Variance des ventes globales: {global_sales_variance}")
print(f"Écart type médian des ventes globales: {global_sales_mad}")
```

Voici le résultat :

```
Variance des ventes globales: 2.417966193459794
Écart type médian des ventes globales: 0.46708037112905165
```

**Variance des ventes globales (2.417966193459794):** La variance est une mesure de la dispersion des données par rapport à la moyenne. Une variance élevée signifie que les données sont très éparpillées autour de la moyenne ; une variance faible signifie qu'elles sont plus resserrées. Dans ce cas là , la variance des ventes globales est d'environ 2.42 millions. Cela indique la variabilité des ventes globales entre les différents jeux vidéo. Plus précisément, en moyenne, les ventes de chaque jeu diffèrent de la moyenne des ventes globales de 2.42 millions d'unités au carré.

**Écart type médian des ventes globales (0.46708037112905165):** L'écart type médian, souvent appelé déviation absolue médiane (MAD), est une autre mesure de la dispersion des données. Contrairement à la variance et à l'écart type, le MAD utilise la médiane comme mesure centrale plutôt que la moyenne, ce qui le rend plus robuste aux valeurs aberrantes. Un MAD de 0.467 signifie qu' en moyenne les ventes des jeux s'écartent de la médiane des ventes de 0.467 million d'unités. C'est une mesure de dispersion qui est moins sensible aux valeurs extrêmes que la variance ou l'écart type.

Ces deux nombres nous disent que même si les ventes de jeux varient (certains jeux se vendent beaucoup plus que d'autres), la majorité des jeux ont des ventes qui ne sont pas extrêmement éloignées de la vente "moyenne".

17) Créer des filtres complexes en utilisant des opérations NumPy sur des colonnes.

```
# Créer des filtres complexes en utilisant des opérations NumPy sur des colonnes.
# filter_mask permet de créer un masque.
# df['NA_Sales'] > 1 permet de sélectionner les données par ventes en Amérique du Nord supérieures à 1.
# df['Year'] > 2000 permet de sélectionner les données par année supérieure à 2000.
# df['Global_Sales'] > df['EU_Sales'] permet de sélectionner les données par ventes globales supérieures aux ventes en Europe.
filter_mask = (df['NA_Sales'] > 1) & (df['Year'] > 2000) & (df['Global_Sales'] > df['EU_Sales'])

# Appliquez ce masque à votre DataFrame pour filtrer les données
filtered_games = df[filter_mask]

# Affichez les résultats
print(filtered_games)
```

Explication plus en détails :

**df['NA\_Sales'] > 1** crée un tableau de booléens où chaque entrée est True si les ventes en Amérique du Nord dépassent 1 million d'unités.

**df['Year'] > 2000** crée un tableau de booléens où chaque entrée est True si l'année de sortie du jeu est postérieure à 2000.

**df['Global\_Sales'] > df['EU\_Sales']** crée un tableau de booléens où chaque entrée est True si les ventes globales du jeu sont supérieures aux ventes européennes.

Le masque **filter\_mask** est ensuite la combinaison de ces trois conditions, utilisant l'opérateur **&** pour les combiner logiquement (toutes les conditions doivent être vraies pour qu'une ligne soit sélectionnée).

Voici le résultat :

	Rank	Name	Platform	Year	\	
0	1	Wii Sports	Wii	2006.0		
2	3	Mario Kart Wii	Wii	2008.0		
3	4	Wii Sports Resort	Wii	2009.0		
6	7	New Super Mario Bros.	DS	2006.0		
7	8	Wii Play	Wii	2006.0		
...	...	...	...	...		
1825	1827	SOCOM: U.S. Navy SEALs Fireteam Bravo	PSP	2005.0		
1881	1883	Madden NFL 2004	XB	2003.0		
1901	1903	NCAA Football 13	X360	2012.0		
1917	1919	NCAA Football 14	X360	2013.0		
1957	1959	Diablo II: Lord of Destruction	PC	2001.0		
	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	\
0	Sports	Nintendo	41.49	29.02	3.77	
2	Racing	Nintendo	15.85	12.88	3.79	
3	Sports	Nintendo	15.75	11.01	3.28	
6	Platform	Nintendo	11.38	9.23	6.50	
7	Misc	Nintendo	14.03	9.20	2.93	
...	...	...	...	...	...	
1825	Shooter	Sony Computer Entertainment	1.03	0.00	0.01	
1881	Sports	Electronic Arts	1.02	0.02	0.00	
1901	Action	Electronic Arts	1.02	0.00	0.00	
1917	Sports	Electronic Arts	1.01	0.00	0.00	
1957	Role-Playing	Vivendi Games	1.03	0.02	0.00	
...						
1917	0.06	1.07				
1957	0.00	1.06				

18) Analyser la corrélation entre les ventes dans différentes régions.

```
# Analyser la corrélation entre les ventes dans différentes régions.  
# .corr permet de calculer la corrélation.  
# .round permet d'arrondir les données.  
# 2 permet d'arrondir les données à 2 chiffres après la virgule.  
df[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']].corr().round(2)
```

Voici le résultat :

	NA_Sales	EU_Sales	JP_Sales	Other_Sales
NA_Sales	1.00	0.77	0.45	0.63
EU_Sales	0.77	1.00	0.44	0.73
JP_Sales	0.45	0.44	1.00	0.29
Other_Sales	0.63	0.73	0.29	1.00

Explication :

**Valeur de 1.00 :** Sur la diagonale, chaque région a une corrélation parfaite avec elle-même, ce qui est attendu.

**Valeurs entre 0 et 1 :** Représentent le degré de corrélation linéaire entre les ventes dans deux régions. Une valeur proche de 1 indique une forte corrélation positive, signifiant que si les ventes augmentent dans une région, elles ont tendance à augmenter aussi dans l'autre région.

**NA\_Sales et EU\_Sales (0.77) :** Montrent une corrélation assez forte, ce qui suggère que les jeux qui se vendent bien en Amérique du Nord ont également tendance à se vendre bien en Europe.

**JP\_Sales :** Montrent généralement des valeurs de corrélation plus basses avec les autres régions, particulièrement avec les autres régions (0.29), ce qui peut indiquer que les goûts du marché japonais pour les jeux vidéo sont assez différents de ceux des autres régions.

**EU\_Sales et Other\_Sales (0.73) :** Indiquent également une corrélation relativement élevée, ce qui suggère que les jeux qui sont populaires en Europe ont tendance à l'être aussi dans les autres régions du monde.

Cette matrice de corrélation aide à comprendre comment les ventes dans une région peuvent être liées aux ventes dans une autre région. Des valeurs de corrélation élevées peuvent indiquer des marchés avec des comportements d'achat similaires ou des influences partagées, tandis que des valeurs plus basses peuvent indiquer des marchés indépendants ou ayant des préférences distinctes.

# Synthèse

**Lecture et exploration de données:** J'ai commencé par lire le fichier CSV dans un DataFrame et effectué une exploration initiale pour comprendre les types de données, les valeurs manquantes et les statistiques de base.

**Statistiques descriptives:** J'ai calculé des statistiques descriptives comme la moyenne, la médiane et l'écart-type pour les colonnes numériques avec `.describe()`.

**Tri et filtrage:** J'ai trié les jeux par ventes globales et filtré les données pour des critères spécifiques.

**Groupelements:** J'ai groupé les jeux par année et par plateforme, calculant des sommes et des moyennes pour comprendre les tendances des ventes au fil du temps et entre les plateformes.

**Création et manipulation de colonnes:** J'ai ajouté des colonnes calculées, comme le ratio des ventes en Europe par rapport aux ventes globales.

**Visualisation de données:** J'ai créé un histogramme des ventes globales, ajustant les bins et les échelles pour améliorer la lisibilité, et un diagramme à barres empilées pour comparer les ventes régionales de jeux populaires.

**Calculs statistiques avec NumPy:** J'ai utilisé NumPy pour des calculs statistiques avancés, comme la variance des ventes globales et l'écart type médian, qui ne sont pas directement disponibles dans Pandas.


**Corrélation entre régions:** J'ai analysé la corrélation entre les ventes dans différentes régions, découvrant les relations entre les marchés.

## Conclusion :

Tout au long de ces exercices, j'ai amélioré ma compréhension des opérations de manipulation de données et de visualisation dans le contexte de l'analyse des données. J'ai exploré les relations entre les variables, découvert des tendances et des anomalies, et appris à interpréter des distributions complexes et des mesures statistiques. Cela m'a permis d'avoir une meilleure compréhension des dynamiques du marché du jeu vidéo et de la manière dont les données peuvent être utilisées pour tirer des insights significatifs.

## Ressources

Pour la réalisation de cette doc et les exercices j'ai utilisé :

- La chaîne Youtube de Machine Learning avec une suite de 30 vidéos qui apprend les bases pour l'analyse de données.  
 FORMATION PYTHON MACHINE LEARNING (2020) (1/30)
- Doc de numpy : [https://numpy.org/doc/1.26/user/absolute\\_beginners.html](https://numpy.org/doc/1.26/user/absolute_beginners.html)
- Doc de pandas :  
[https://pandas.pydata.org/docs/getting\\_started/index.html#getting-started](https://pandas.pydata.org/docs/getting_started/index.html#getting-started)
- Doc de matplotlib : [https://matplotlib.org/stable/users/explain/quick\\_start.html](https://matplotlib.org/stable/users/explain/quick_start.html)
- Doc jupyter notebook :  
<https://code.visualstudio.com/docs/datascience/jupyter-notebooks>
- Chat GPT : pour corriger les fautes d'orthographe et pour mieux structurer mes explication