

Seaborn : choisir, tracer, interpréter

Une fiche de révision visuelle : fonctions, paramètres du cours et pièges des exercices.

Le bon réflexe : déterminer ce que représente une ligne du tableau, puis choisir ce que doit représenter un point ou une barre. Seaborn peut agréger les données : le graphique ne montre pas toujours les valeurs individuelles.

La question porte sur...	Choix du graphique
Une distribution numérique	histplot / kdeplot / ecdfplot
Deux variables numériques	scatterplot ; Implot pour une régression
Une évolution dans le temps	lineplot, après comptage si la question demande un nombre
Une valeur numérique par catégorie	strip / swarm : individus ; box / violin / boxen : distributions
Une moyenne par catégorie	barplot ou pointplot
Un nombre par catégorie	countplot sur données brutes ; barplot sur comptages préparés
Plusieurs variables numériques	pairplot ou heatmap de la matrice de corrélation

Une fonction directe ou une fonction avec grille ?

Fonction avec grille	kind → fonction directe
displot	"hist" → histplot ; "kde" → kdeplot ; "ecdf" → ecdfplot
relplot	"scatter" → scatterplot ; "line" → lineplot
catplot	"strip", "swarm", "box", "violin", "boxen", "count", "bar", "point"
Implot / pairplot	Grilles spécialisées ; aucune variante catplot

Fonctions directes : dessinent dans un axe Matplotlib et acceptent ax=. Taille : plt.subplots(figsize=(...)).

Fonctions avec grille : créent leur propre figure. Facettes : col / row ; taille : height et aspect. Ne pas leur passer ax.

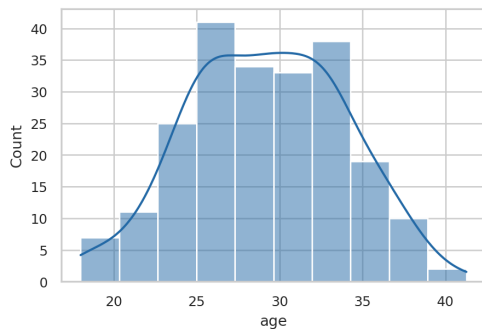
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="ticks", context="notebook", palette="deep")
```

Dans les codes : df désigne ton tableau ; les noms entre guillemets désignent ses colonnes. Les exemples illustrent des mécanismes, pas les résultats des datasets NBA, accidents ou Netflix.

Distribution : où sont les valeurs ?

Une variable numérique. Aucun de ces trois graphiques ne passe par catplot.

1. Histogramme



Quand ? Repérer les zones fréquentes, les creux et une asymétrie.

Lecture → Une barre compte les valeurs dans un intervalle. Une queue à droite indique quelques grandes valeurs.

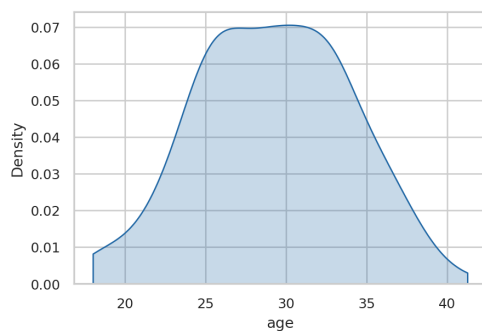
Paramètres → `bins=10` : découpage ; `kde=True` : courbe estimée ; `rug=True` (`displot`) : petites tiges.

Piège → Les conclusions visuelles dépendent des bins. Ce ne sont pas des catégories.

```
sns.displot(data=df, x="age", bins=10, kde=True, rug=True)
```

Équivalence → `displot(kind="hist")` ; directe : `histplot`.

2. KDE : densité estimée



Quand ? Voir la forme lissée d'une distribution.

Lecture → Les pics indiquent des zones denses ; plusieurs pics peuvent suggérer plusieurs populations.

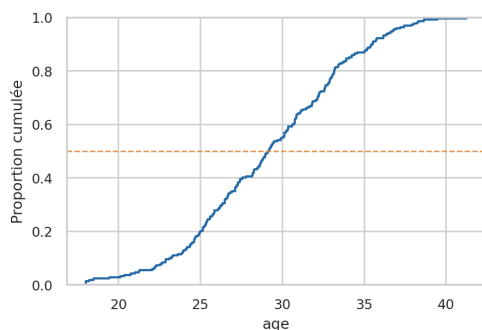
Paramètres → `cut=0` : arrêter aux valeurs observées ; `clip=(18, 50)` : borner le tracé ; `fill=True` : remplir.

Piège → Une hauteur de densité n'est pas une probabilité ponctuelle. Le lissage influence les pics.

```
sns.kdeplot(data=df, x="age", fill=True, cut=0)
```

Équivalence → `displot(kind="kde")` ; directe : `kdeplot`.

3. ECDF : proportion cumulée



Quand ? Répondre à « quelle part est inférieure ou égale à un seuil ? ».

Lecture → À un âge x , y donne la proportion d'âges $\leq x$. À $y=0,5$, on lit un seuil médian.

Paramètres → `kind="ecdf"` avec `displot` ; `hue` pour comparer des groupes ; axes x/y pour l'orientation.

Piège → Courbe croissante par construction : elle ne décrit pas une évolution dans le temps.

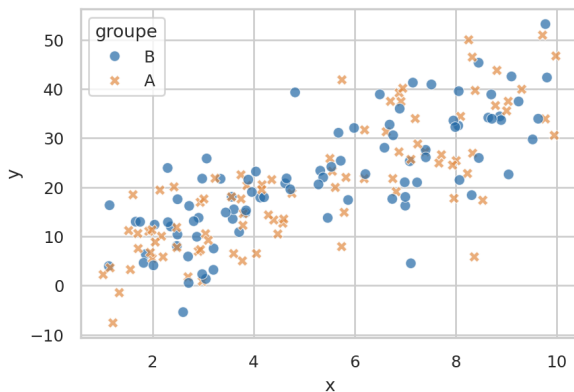
```
sns.displot(data=df, x="age", kind="ecdf")
```

Équivalence → `displot(kind="ecdf")` ; directe : `ecdfplot`.

Relations : observations ou moyennes ?

relplot(kind="scatter") et relplot(kind="line") ; pas de variante catplot.

4. Scatterplot : nuage de points



Quand ? Étudier l'association entre deux variables numériques.

Lire le visuel → Chaque point est une observation. Examiner direction, dispersion, forme, groupes et points isolés.

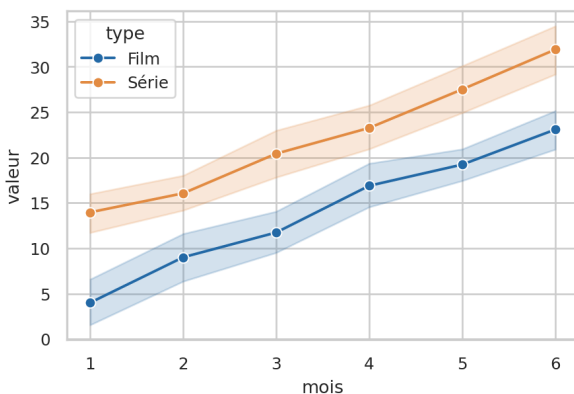
Paramètres → hue : couleur ; style : forme ; size : taille liée à une colonne ; alpha : transparence.

```
sns.scatterplot(data=df, x="x", y="y",  
                hue="groupe", style="groupe", size="volume")
```

Version avec grille → `sns.relplot(data=df, x="x", y="y", kind="scatter", col="groupe")`

À retenir → Une tendance croissante suggère une association positive ; elle ne prouve pas une cause.

5. Lineplot : courbe d'évolution



Quand ? Suivre une mesure sur un axe ordonné : temps, âge, etc.

Lire le visuel → Par défaut : moyenne de y pour chaque x et chaque groupe hue. Bande : incertitude de cette moyenne.

Paramètres → hue : courbes colorées ; style : tirets/marqueurs ; size : épaisseur ; marker="o" ; ci=None (cours).

```
sns.lineplot(data=df, x="mois", y="valeur",  
            hue="type", marker="o", ci=None)
```

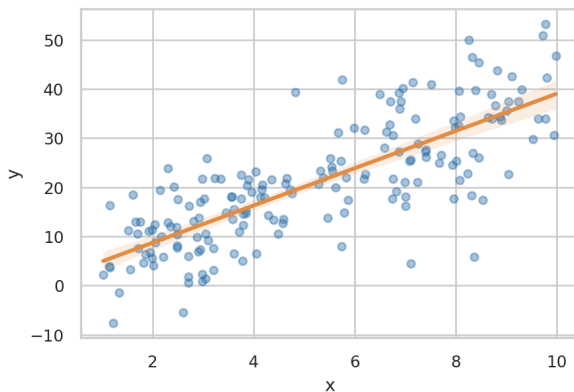
Version avec grille → `sns.relplot(data=df, x="mois", y="valeur", kind="line", hue="type")`

À retenir → Tri croissant de x par défaut. Une courbe n'est pas une régression. Compter d'abord si y doit être un nombre de contenus.

Régression et exploration multivariée

Implot ajuste une tendance ; pairplot compare plusieurs paires de variables.

6. Lmplot : tendance ajustée



Quand ? Visualiser une relation et sa régression, linéaire par défaut.

Lire le visuel → Points : observations ; ligne : ajustement ; bande : incertitude sur la tendance moyenne, pas sur chaque individu.

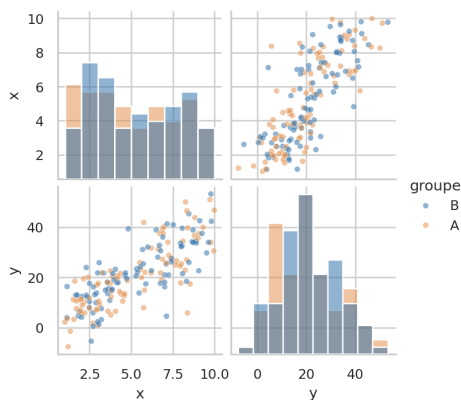
Paramètres → hue : un ajustement par groupe ; lowess=True : local ; order=2 : quadratique ; line_kws : ligne.

```
sns.lmplot(data=df, x="x", y="y", hue="groupe")  
# Variante : order=2, line_kws={"color": "red"}
```

Version avec grille → Implot accepte col / row / height. Directe dans un axe : regplot.

À retenir → Utiliser lowess OU order=2 selon le besoin. Une régression visuelle ne démontre ni causalité ni significativité.

7. Pairplot : matrice de graphiques



Quand ? Explorer les relations entre plusieurs variables numériques.

Lire le visuel → Diagonale : distribution d'une variable. Hors diagonale : relation entre la variable de la colonne et celle de la ligne.

Paramètres → hue : groupes colorés ; diag_kind="hist" ou "kde" ; sélection préalable des colonnes ; height.

```
sns.pairplot(data=df[["x", "y", "groupe"]],  
             hue="groupe", diag_kind="hist")
```

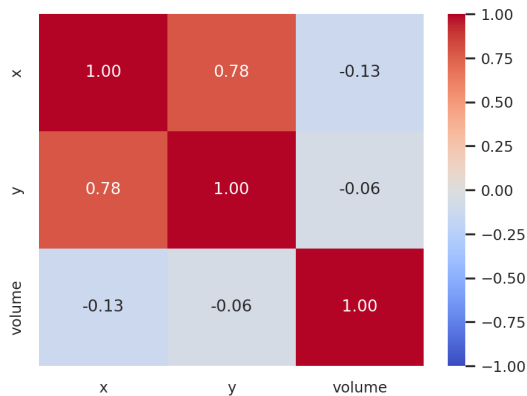
Version avec grille → pairplot crée sa propre grille. Pas de kind dans catplot pour ce graphique.

À retenir → Limiter les variables pour garder une figure lisible. Les identifiants numériques ne sont pas forcément des mesures.

Corrélations et graphiques séparés

Une matrice pour explorer les liens ; des facettes pour comparer les sous-groupes.

8. Heatmap : matrice de corrélation



Quand ? Comparer la direction et la force de relations linéaires.

Lire le visuel → r proche de +1 : positive ; de -1 : négative ; de 0 : faible relation linéaire. Lire la valeur, pas seulement la couleur.

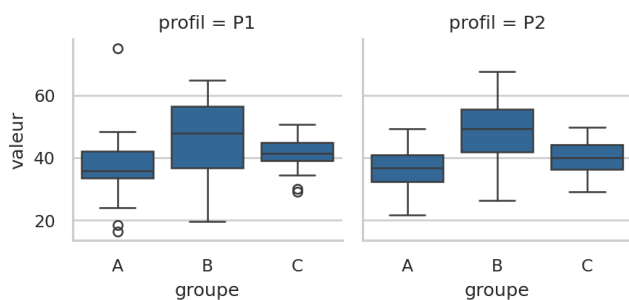
Paramètres → `annot=True` : valeurs ; `cmap` : palette ; `ax` : axe. Compléments : `fmt=".2f"`, `vmin=-1`, `vmax=1`.

```
cor = df.select_dtypes(include="number").corr()
sns.heatmap(cor, annot=True, cmap="coolwarm",
            vmin=-1, vmax=1, fmt=".2f")
```

Version avec grille → `heatmap` est une fonction directe ; aucun équivalent `catplot`.

À retenir → $r \approx 0$ n'exclut pas une relation non linéaire. Exclure les IDs ; corrélation ne signifie pas causalité.

9. Hue ou col / row ?



Quand ? Séparer visuellement les groupes quand un seul axe devient chargé.

Lire le visuel → Ici : un panneau par profil. `hue` mettrait les profils dans le même panneau avec des couleurs différentes.

Paramètres → `col` : panneaux en colonnes ; `row` : en lignes ; `col_wrap=3` : retour à la ligne ; `height=3` ; `aspect=1.2`.

```
sns.catplot(data=df, x="groupe", y="valeur",
            kind="box", col="profil", height=3)
```

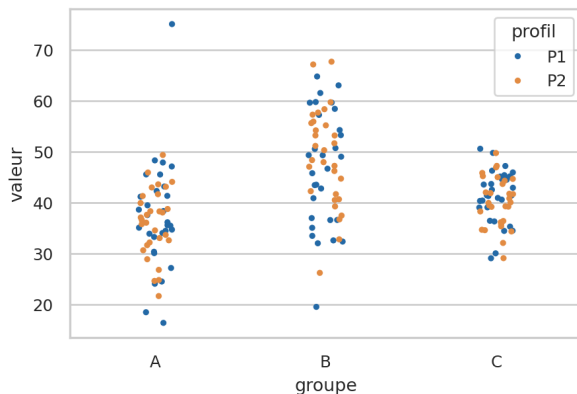
Version avec grille → Ces paramètres servent aussi avec `relplot` et `displot`. `Insetplot` gère également les facettes.

À retenir → `col_wrap` n'est pas compatible avec `row`. `height` mesure la hauteur de chaque panneau en pouces ; `largeur` = `height` × `aspect`.

Catplot : voir les observations

Variable catégorielle x variable numérique. Les deux graphiques montrent les individus.

10. Stripplot : points en bandes



Quand ? Voir les valeurs individuelles de chaque catégorie.

Lire le visuel → Chaque point est une observation. Les amas indiquent une concentration, mais des points peuvent se superposer.

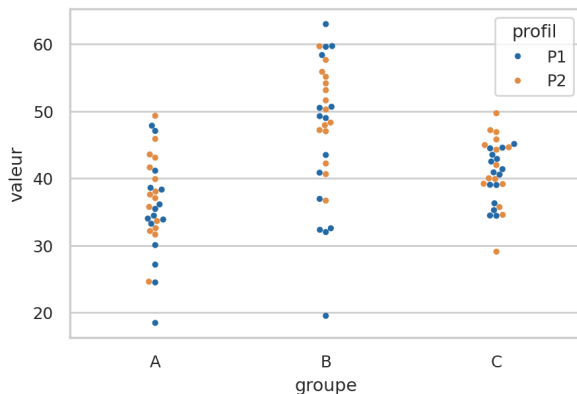
Paramètres → hue : couleur ; order : ordre ; jitter : décalage latéral ; size : taille fixe des points ; ax.

```
sns.stripplot(data=df, x="groupe", y="valeur",
              hue="profil", jitter=True, size=4)
```

Version avec grille → `sns.catplot(data=df, x="groupe", y="valeur", kind="strip", hue="profil")`

À retenir → Le décalage horizontal ne représente pas une mesure. Ici `size=4` est une taille fixe, pas une colonne comme dans `scatterplot`.

11. Swarmplot : essaim de points



Quand ? Voir les observations en limitant leur chevauchement.

Lire le visuel → Les points sont déplacés latéralement pour tenir dans la catégorie. Un essaim large révèle beaucoup de valeurs proches.

Paramètres → hue : couleur ; order : catégories ; size : taille fixe ; `dodge=True` : séparer les niveaux hue ; ax.

```
sns.swarmplot(data=df, x="groupe", y="valeur",
              hue="profil", size=4)
```

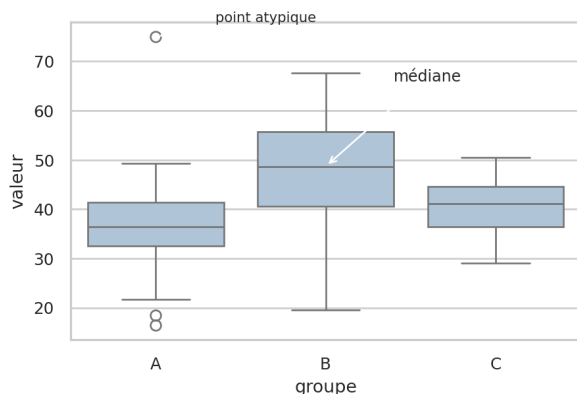
Version avec grille → `sns.catplot(data=df, x="groupe", y="valeur", kind="swarm", hue="profil")`

À retenir → Peut devenir lent et ne plus placer tous les points si la densité est forte. Réduire `size`, échantillonner ou préférer `boxplot`.

Catplot : comparer les distributions

Médiane et dispersion avec boxplot ; forme estimée avec violinplot.

12. Boxplot : boîte à moustaches



Quand ? Comparer médianes, dispersion centrale et valeurs atypiques.

Lire le visuel → Trait : médiane ; boîte : Q1 à Q3, soit 50 % centraux ; points isolés : valeurs au-delà des moustaches.

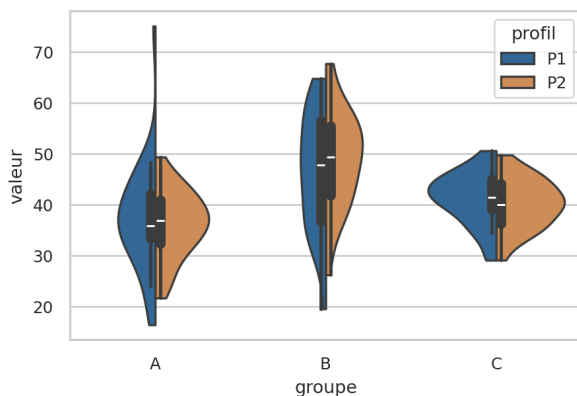
Paramètres → hue ; order ; palette / color ; ax. Complément : whis=1.5, valeur par défaut de la règle des moustaches.

```
sns.boxplot(data=df, x="groupe", y="valeur", hue="profil")
```

Version avec grille → `sns.catplot(data=df, x="groupe", y="valeur", kind="box", col="profil")`

À retenir → $IQR = Q3 - Q1$. Moustaches aux observations dans $[Q1 - 1,5 \times IQR ; Q3 + 1,5 \times IQR]$. Un point atypique n'est pas forcément une erreur.

13. Violinplot : densité par groupe



Quand ? Comparer la forme des distributions et leurs éventuels pics.

Lire le visuel → Partie large : zone de forte densité estimée. Le dessin dépend du lissage ; le centre affiche par défaut un résumé en boîte.

Paramètres → hue ; split=True : violons partagés ; cut=0 : limite aux valeurs observées ; order ; ax.

```
sns.violinplot(data=df, x="groupe", y="valeur",  
               hue="profil", split=True, cut=0)
```

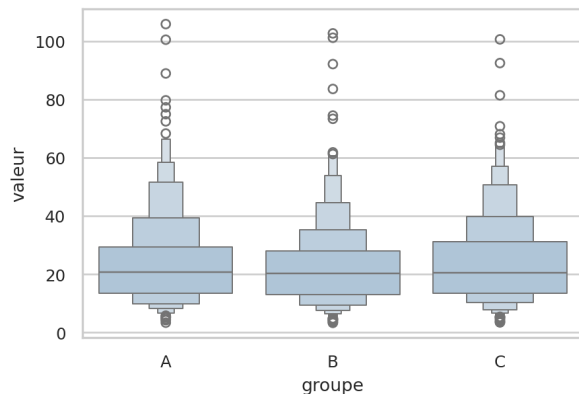
Version avec grille → `sns.catplot(data=df, x="groupe", y="valeur", kind="violin", hue="profil", split=True)`

À retenir → Pour le cours : utiliser split avec deux niveaux hue. Depuis Seaborn 0.13, la restriction à deux niveaux a été assouplie.

Catplot : quantiles ou effectifs ?

Boxenplot décrit des distributions ; countplot compte les observations.

14. Boxenplot : quantiles détaillés



Quand ? Explorer de grands échantillons, surtout les queues de distribution.

Lire le visuel → Les boîtes représentent plusieurs intervalles de quantiles. Plus on va vers les extrêmes, moins il y a d'observations.

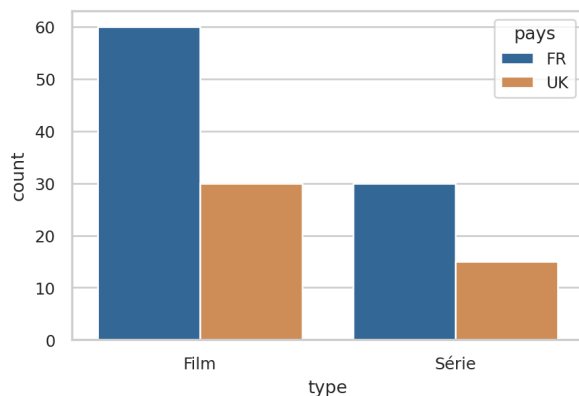
Paramètres → x / y ; hue ; order ; palette / color ; ax. Le cours utilise surtout la fonction et kind="boxen".

```
sns.boxenplot(data=df, x="groupe", y="valeur")
```

Version avec grille → `sns.catplot(data=df, x="groupe", y="valeur", kind="boxen")`

À retenir → Les boîtes ne représentent pas des sous-groupes différents. Lire leur étendue sur l'axe numérique, pas comme un histogramme.

15. Countplot : combien par catégorie ?



Quand ? Compter des lignes selon une catégorie, éventuellement par sous-groupe.

Lire le visuel → Hauteur ou longueur = effectif. Avec hue, chaque catégorie est subdivisée. Ce graphique ne calcule pas une moyenne.

Paramètres → x OU y : catégorie ; hue ; order ; hue_order ; palette ; ax. Avec y, les barres sont horizontales.

```
sns.countplot(data=df, x="type", hue="pays")
```

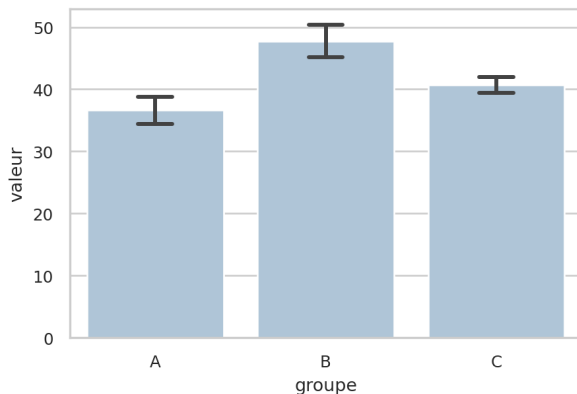
Version avec grille → `sns.catplot(data=df, x="type", hue="pays", kind="count", col="profil")`

À retenir → Sur une table déjà agrégée, countplot compte les lignes de résumé ! Utiliser barplot pour tracer les effectifs calculés.

Catplot : estimer une valeur par groupe

Barplot et pointplot montrent une moyenne par défaut, pas la distribution complète.

16. Barplot : moyenne ou résumé préparé



Quand ? Comparer une mesure numérique entre catégories.

Lire le visuel → Données brutes : hauteur = moyenne.
Trait noir : IC de cette moyenne par défaut. Tableau agrégé : valeur préparée par catégorie.

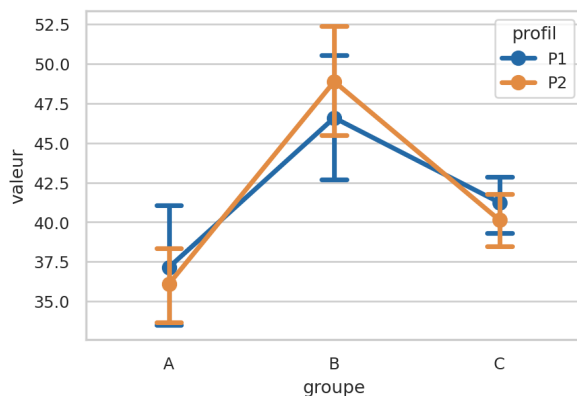
Paramètres → hue ; order ; ci=None (cours) ; ax. Compléments : estimator="median" ou "sum" ; errorbar=None (récent).

```
sns.barplot(data=df, x="groupe", y="valeur", hue="profil")
```

Version avec grille → sns.catplot(data=df, x="groupe", y="valeur", kind="bar", hue="profil")

À retenir → Les IC ne décrivent pas les extrêmes individuels. Leur chevauchement ne suffit pas à décider si une différence est significative.

17. Pointplot : moyenne et comparaison



Quand ? Comparer les moyennes entre catégories, notamment selon hue.

Lire le visuel → Chaque point est une moyenne ; trait vertical : IC. Les segments facilitent la comparaison entre catégories ou profils.

Paramètres → hue ; order ; ci=None (cours) ; ax. Compléments : estimator, errorbar, markers et linestyle.

```
sns.pointplot(data=df, x="groupe", y="valeur", hue="profil")
```

Version avec grille → sns.catplot(data=df, x="groupe", y="valeur", kind="point", hue="profil")

À retenir → Ne pas confondre avec scatterplot : un point résume un groupe. Une pente entre catégories ne représente pas forcément le temps.

Mémo pratique : paramètres et préparation

Les mêmes mots n'ont pas toujours le même sens d'une fonction à l'autre.

Paramètre	Sens et fonctions à retenir
hue	Couleur selon une colonne ; fréquent dans les familles relationnelles, catégorielles et de distribution.
style / size	scatter : forme / taille selon une colonne ; line : style / épaisseur. Pas de style universel dans catplot.
order / hue_order	Ordre des catégories / sous-groupes. order attend les noms, pas les effectifs.
col / row / col_wrap	Facettes via catplot, relplot, displot ; pas via boxplot ou scatterplot directs.
height / aspect / ax	Grilles : height et aspect. Fonctions directes : ax et figsize sur la figure Matplotlib.

Compter avant une courbe : exemple Netflix

```
all_content["year"] = pd.to_datetime(all_content["date_added"]).dt.year
evo = all_content.groupby(["year", "type"]).size()
evo = evo.reset_index(name="nombre")
sns.lineplot(data=evo, x="year", y="nombre", hue="type")
```

size() compte les lignes ; count() compte les valeurs non nulles d'une colonne ; nunique() compte les IDs distincts. reset_index renvoie un nouveau tableau : réaffecter le résultat. Ce graphique représente les ajouts annuels, pas le total cumulé du catalogue.

Top catégories ou deux graphiques côte à côte

```
top = df["dep"].value_counts().head(10)
sns.barplot(x=top.index, y=top.values, order=top.index)
fig, axes = plt.subplots(1, 2, figsize=(12, 4))
sns.barplot(data=uk_movies.head(5), x="numVotes", y="title", ax=axes[0])
sns.barplot(data=uk_series.head(5), x="numVotes", y="title", ax=axes[1])
```

Trier les contenus par numVotes décroissant avant head(5). Inverser l'ordre des catégories : top.index[::-1]. Avec hue, laisser Seaborn créer la légende ; un label commun peut ajouter des entrées inutiles.

Compatibilité du cours et corrections importantes

- ci=None du cours → errorbar=None pour lineplot, barplot et pointplot récents. Implot / regplot gardent ci.
- shade=True → fill=True pour kdeplot. Orientation de displot : utiliser x ou y, plutôt que vertical.
- Dans displot, les options du graphique choisi passent généralement directement ; kde_kws sert à la KDE ajoutée à l'histogramme. hist_kws / ecdf_kws ne sont pas une syntaxe générale actuelle.
- Une différence visuelle n'est pas un test. Un IC ne contient pas « 95 % des observations ». Une association ne prouve pas une cause.
- Une catégorie absente d'un groupby n'est pas automatiquement comptée à zéro. Le dénominateur d'une moyenne ou d'un pourcentage doit correspondre à la question.

Sources : tes 5 notebooks PDF (univarié, relplot, régression, catplot, accidents). Compléments vérifiés dans la documentation officielle Seaborn 0.13.2 :

Fonctions et grilles → [documentation officielle](#)

Estimation et barres d'erreur → [documentation officielle](#)

Compatibilité des violons → [documentation officielle](#)

Syntaxe de displot → [documentation officielle](#)

Astuces des exercices : préparer les données

Des transformations courtes qui changent le sens du graphique.

Filtrer avant de tracer

```
films = all_content.loc[all_content["type"] == "Movie"].copy()
selection = films.loc[(films["duree"] < 160)
    & (films["country"] == "United States")
    & films["listed_in"].str.contains("Action", na=False)]
```

Parenthèses autour de chaque condition ; & = ET, | = OU. == teste une égalité exacte ; str.contains cherche dans un texte. Ajouter >=0 après une égalité crée une comparaison en chaîne incorrecte.

Nettoyer une durée et extraire les dates

```
films["duree"] = films["duration"].str.replace(" min", "", regex=False)
films["duree"] = films["duree"].astype(int)
df["date"] = pd.to_datetime(df["date"])
df["jour_semaine"] = df["date"].dt.weekday
carac["heure"] = carac["hrmn"] // 100
```

La conversion en int suppose des durées valides et sans NA. str[:-4] enlève un suffixe uniforme ; str[:-3] laisse un espace. weekday : lundi=0, dimanche=6. Avec HHMM : //100 donne l'heure, %100 les minutes.

Créer des catégories numériques

```
df["age_cat"] = pd.cut(df["age"],
    bins=[19, 24, 28, 32, 36, 40],
    labels=["19-24", "25-28", "29-32", "33-36", "37-40"],
    include_lowest=True)
```

Par défaut, intervalles fermés à droite : (24,28]. include_lowest inclut la première borne. Ces libellés supposent des âges entiers ; vérifier les bornes et les valeurs hors intervalle.

Fusionner avec des noms de clés différents

```
all_content = df_netflix.merge(df_imdb,
    left_on=["title", "release_year"],
    right_on=["primaryTitle", "startYear"], how="inner")
```

Les clés correspondent dans le même ordre. inner garde les correspondances ; left conserve toutes les lignes de gauche. Des clés répétées peuvent multiplier les lignes : contrôler avant de compter.

Interpréter sans surinterpréter

Des formulations utilisables en examen et les bons dénominateurs.

Graphique	Ce qu'on peut écrire
Histogramme / KDE	« Les valeurs se concentrent entre... ; la distribution présente une queue vers... »
ECDF	« Environ ... % des observations ont une valeur inférieure ou égale à... »
Scatter / régression	« Une association positive/négative apparaît ; la dispersion est... ; certains points sont atypiques. »
Lineplot	« La mesure augmente/diminue entre... ; on observe un pic en... » Préciser moyenne ou nombre.
Box / violon / boxen	« La médiane de A est... ; les valeurs centrales sont plus dispersées dans... » Pour un violon : densité estimée.
Countplot / comptages	« A est la catégorie la plus représentée en nombre. » Ne pas confondre volume et risque.
Bar / point sur données brutes	« La moyenne de A est supérieure à celle de B dans cet échantillon. » Le graphique seul n'est pas un test.

Nombre, proportion, taux : trois questions différentes

- Un nombre d'accidents dépend aussi de la population et de l'exposition à la circulation : il ne mesure pas directement un risque individuel.
- Une part de clients Premium+ chez les femmes = femmes Premium+ / total des femmes. Un countplot par sexe ne calcule pas cette part.
- Plusieurs clients peuvent être impliqués dans un même accident : compter les Num_Acc distincts, pas seulement les clients.

```
part = carac["Num_Acc"].isin(clients["Num_Acc"]).mean() * 100
taux = df.groupby("sexe")["premium"].mean() * 100
# La seconde formule suppose premium codé 0/1.
```

Une moyenne par département doit-elle inclure les zéros ?

```
comptes = carac.groupby(["mois", "dep"]).size()
grille = pd.MultiIndex.from_product(
    [range(1, 13), carac["dep"].unique()], names=["mois", "dep"])
moyenne = comptes.reindex(grille, fill_value=0).groupby("mois").mean()
```

La grille inclut les mois sans accident pour les départements présents dans carac. Si l'on veut tous les départements administratifs, il faut une liste externe complète. Filtrer la zone avant ce calcul si nécessaire.

Avant de conclure → Ai-je compté les bons objets ? Les absences sont-elles des zéros ou des données manquantes ? Les groupes ont-ils le même dénominateur ? La dernière année est-elle complète ? Une baisse des ajouts n'est pas une baisse du catalogue total ; une note IMDb décrit les titres observés, pas toute la qualité d'une plateforme.