Diagrammes multivariés avec Seaborn

Maîtriser Seaborn : Démystifier les diagrammes complexes · https://moncoachdata.com/tutos/diagrammes-multivaries/

Les **diagrammes multivariés avec Seaborn** sont parmi les plus flexibles : ils permettent d'explorer les relations et les motifs entre trois variables ou plus à la fois. Autrement dit, ils étendent l'analyse à plus de deux variables, ce qui est très souvent nécessaire en analyse de données. Les exemples utilisent les jeux de données `tips` et `iris` :

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

tips = sns.load_dataset("tips")
iris = sns.load_dataset("iris")
```

Python

Ajouter des dimensions avec des paramètres

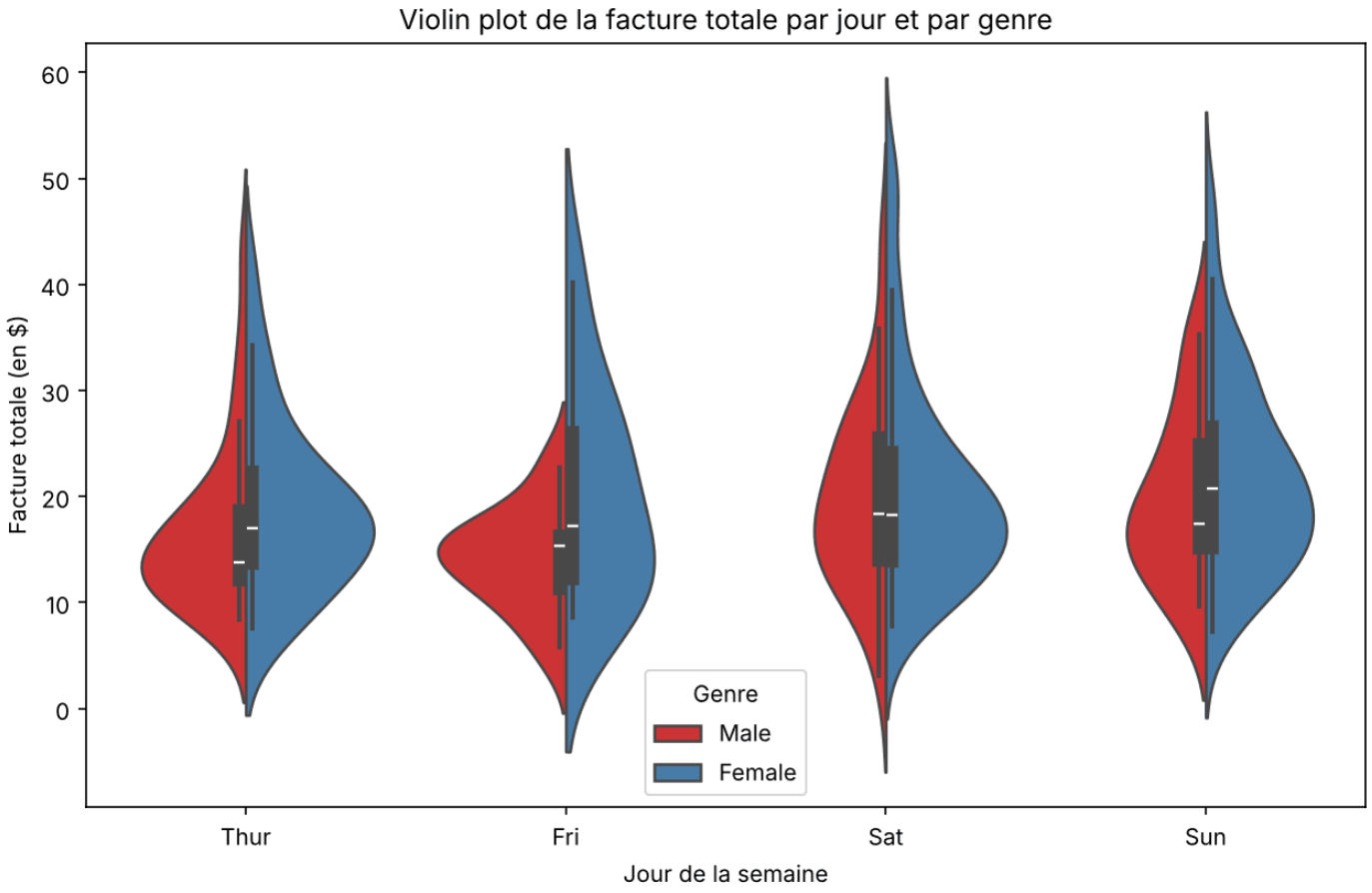
Le paramètre hue

Le paramètre `hue` colore le graphique selon une variable catégorielle, avec une couleur unique par catégorie. Il fonctionne avec presque tous les tracés : `scatterplot()`, `boxplot()`, `violinplot()`, `lineplot()`, etc.

```python
# Violin plot avec une teinte (hue)
plt.figure(figsize=(10, 6))
sns.violinplot(
    x="day",           # axe x : jour de la semaine (catégoriel)
    y="total_bill",    # axe y : montant de la facture (numérique)
    data=tips,
    hue="sex",         # couleur selon le genre (catégoriel)
    palette="Set1",    # palette de couleurs
    split=True         # un demi-violon par catégorie de hue
)

plt.title("Violin plot de la facture totale par jour et par genre")
plt.xlabel("Jour de la semaine")
plt.ylabel("Facture totale (en $)")
plt.legend(title="Genre")
plt.show()
```

Python



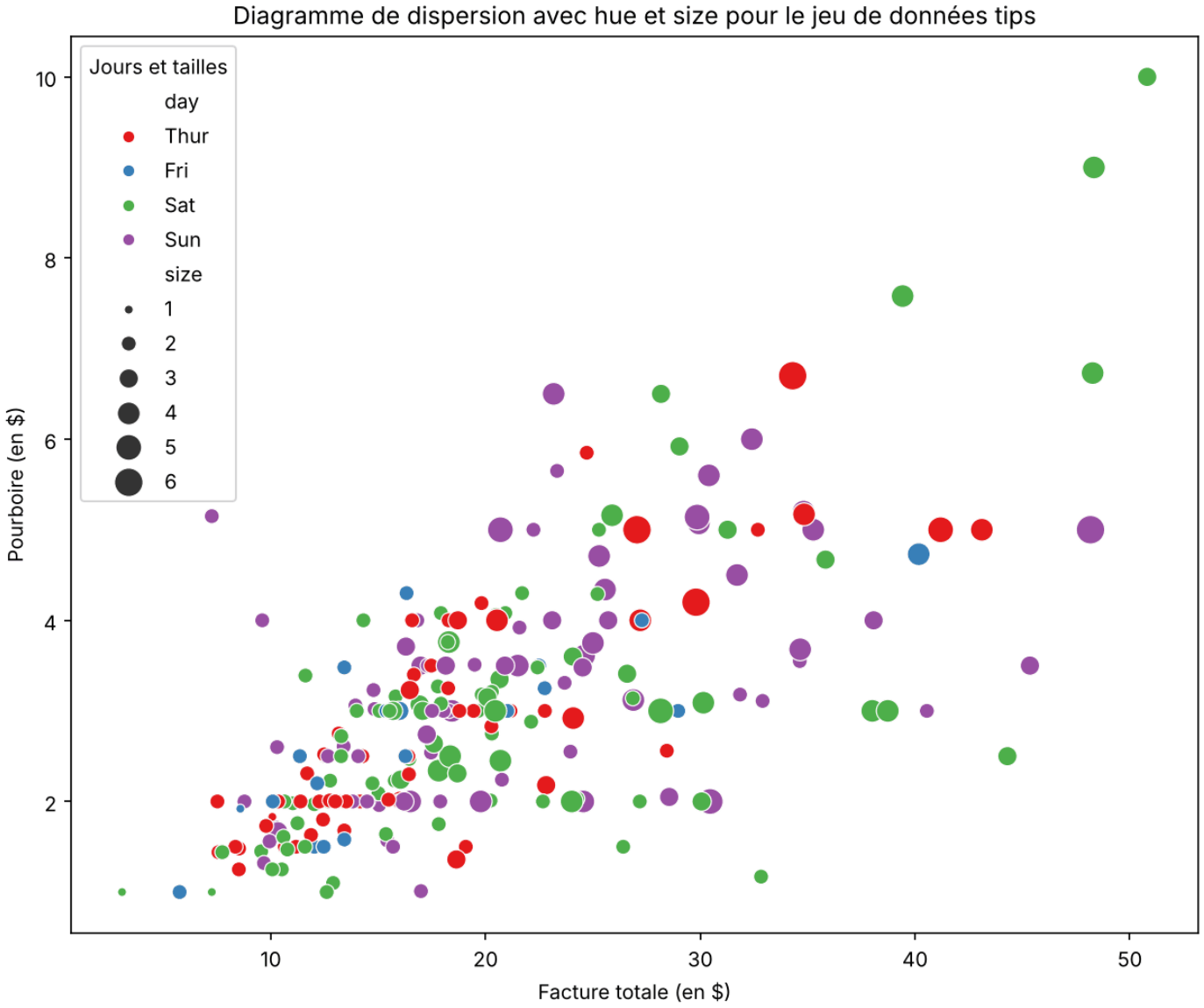
La colonne `sex` colore les violons selon le genre, ce qui ajoute une dimension au graphique.

Les paramètres hue et size

Le paramètre `size` ajoute une dimension numérique, en faisant varier la taille des marqueurs :

```python
# Diagramme de dispersion avec hue et size
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x="total_bill",
    y="tip",
    data=tips,
    hue="day",          # couleur selon le jour (catégoriel)
    size="size",        # taille du marqueur selon la taille du groupe (numérique)
    sizes=(20, 200),    # étendue des tailles de marqueurs
    palette="Set1"
)
plt.title("Diagramme de dispersion avec hue et size pour le jeu de données tips")
plt.xlabel("Facture totale (en $)")
plt.ylabel("Pourboire (en $)")
plt.legend(title="Jours et tailles")
plt.show()
```

Python



Chaque jour a une couleur différente (une troisième dimension avec `hue`), et la taille du marqueur dépend de la colonne `size` (une quatrième dimension).

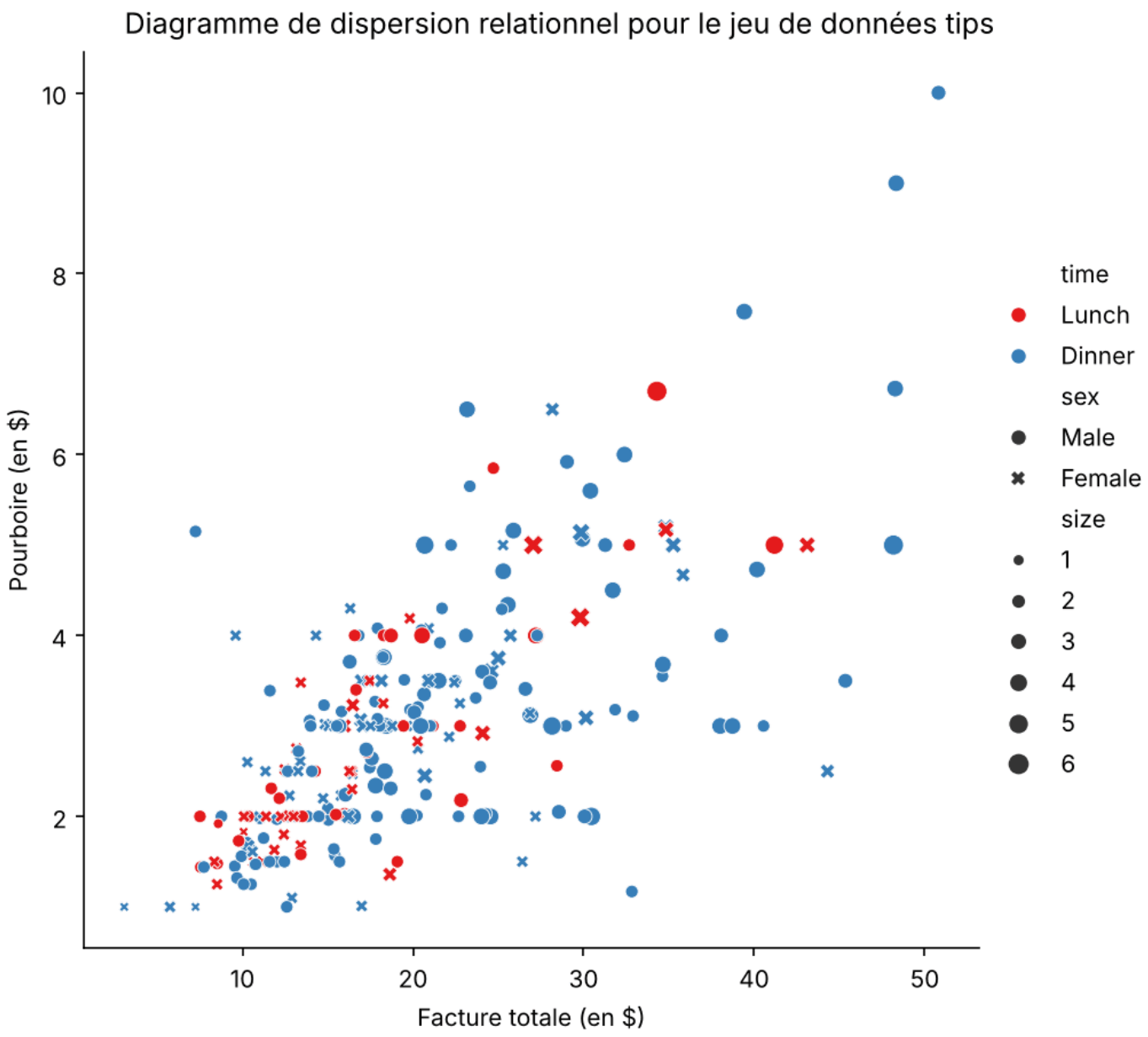
Diagramme relationnel (relplot)

Un diagramme relationnel visualise la relation entre deux variables numériques, avec des dimensions catégorielles ou numériques supplémentaires. Utilise `sns.relplot(x, y, data, hue, size, style)` :

- `hue` colore les points selon une variable catégorielle ;
- `size` fait varier la taille des marqueurs selon une variable numérique ;
- `style` différencie les marqueurs ou les lignes selon une variable catégorielle ;
- `kind` choisit le type de graphe relationnel (`'scatter'` par défaut, ou `'line'`).

```python
# Nuage de points avec relplot
g = sns.relplot(x="total_bill", y="tip", data=tips,
                hue="time", style="sex", size="size",
                palette="Set1", height=6)
g.set_axis_labels("Facture totale (en $)", "Pourboire (en $)")
g.figure.suptitle("Diagramme de dispersion relationnel pour le jeu de données tips", y=1.02)
plt.show()
```

Python



La colonne `time` colore les points (`hue`), `size` fait varier leur taille et `sex` change le marqueur (`style`).

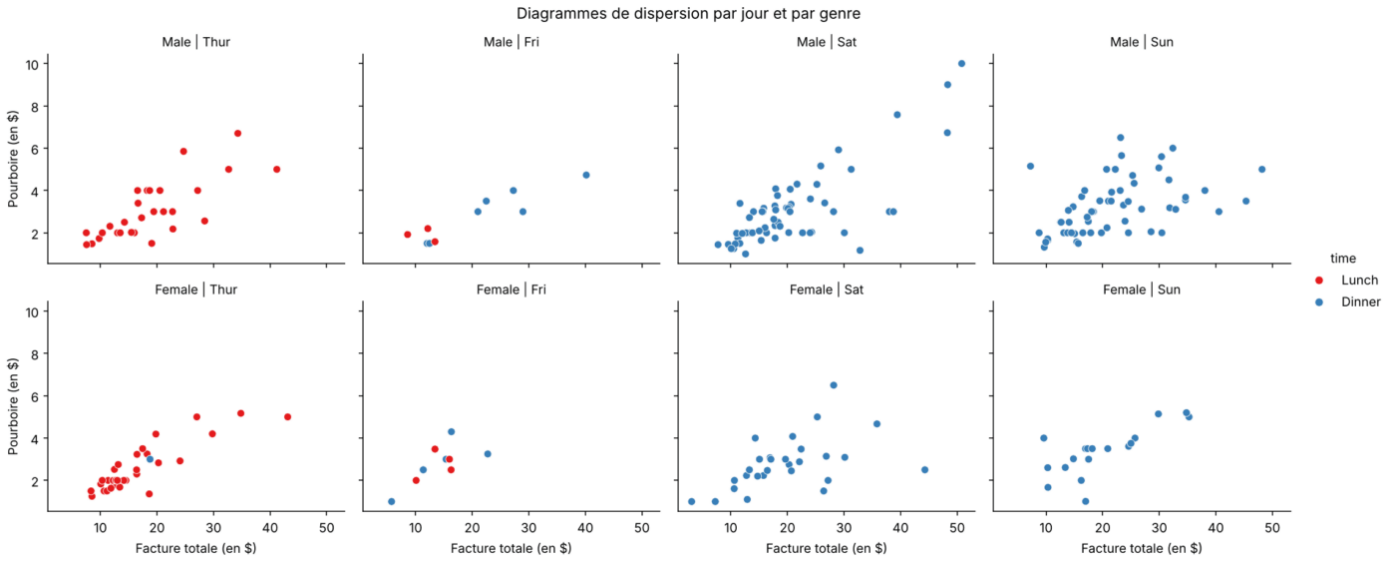
Grille à facettes avec col et row

Avec les paramètres `col` et `row`, `relplot` crée une grille à facettes pour visualiser les relations dans différents sous-ensembles de données. Affecte-leur des colonnes catégorielles :

```python
g = sns.relplot(
    x="total_bill",
    y="tip",
    data=tips,
    hue="time",
    col="day",        # un graphique par jour (colonnes)
    row="sex",        # un graphique par genre (lignes)
    palette="Set1",
    height=3,         # hauteur de chaque sous-graphique
    aspect=1.2        # rapport largeur / hauteur de chaque sous-graphique
)

# Titres et étiquettes des facettes
g.set_titles(col_template="{col_name}", row_template="{row_name}")
g.set_axis_labels("Facture totale (en $)", "Pourboire (en $)")
g.figure.suptitle("Diagrammes de dispersion par jour et par genre", y=1.03)
plt.show()
```

Python



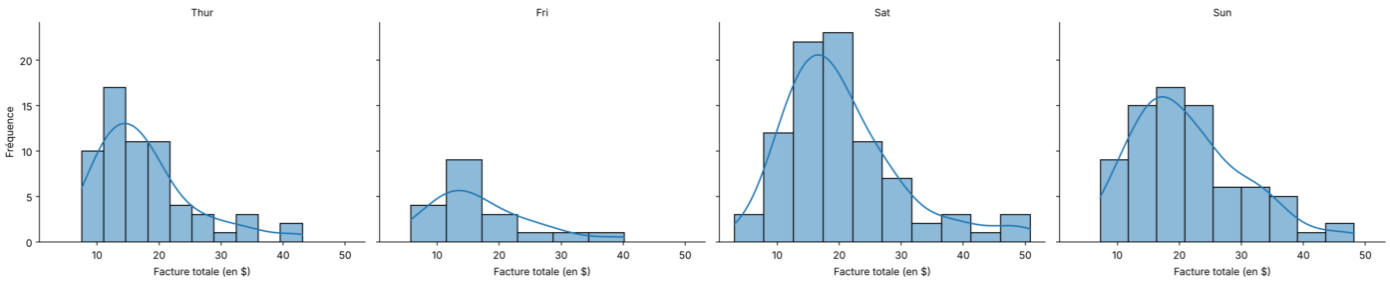
Facet Grid

Une grille à facettes (*FacetGrid*) crée une grille de sous-graphiques, chacun représentant un sous-ensemble différent de tes données. Elle sert à comparer des modèles ou des relations entre plusieurs variables dans différentes catégories.

Utilise `sns.FacetGrid(data, col, row)` pour créer la grille : elle renvoie un objet grille. Associe-lui ensuite le tracé de ton choix avec `.map()` :

```python
# Grille à facettes d'histogrammes, un par jour
g = sns.FacetGrid(tips, col="day", height=4, aspect=1.2)
g.map(sns.histplot, "total_bill", kde=True)
g.set_axis_labels("Facture totale (en $)", "Fréquence")
g.set_titles(col_template="{col_name}")
plt.show()
```

Python



Tu peux mapper n'importe quel autre tracé pour créer d'autres types de FacetGrid.

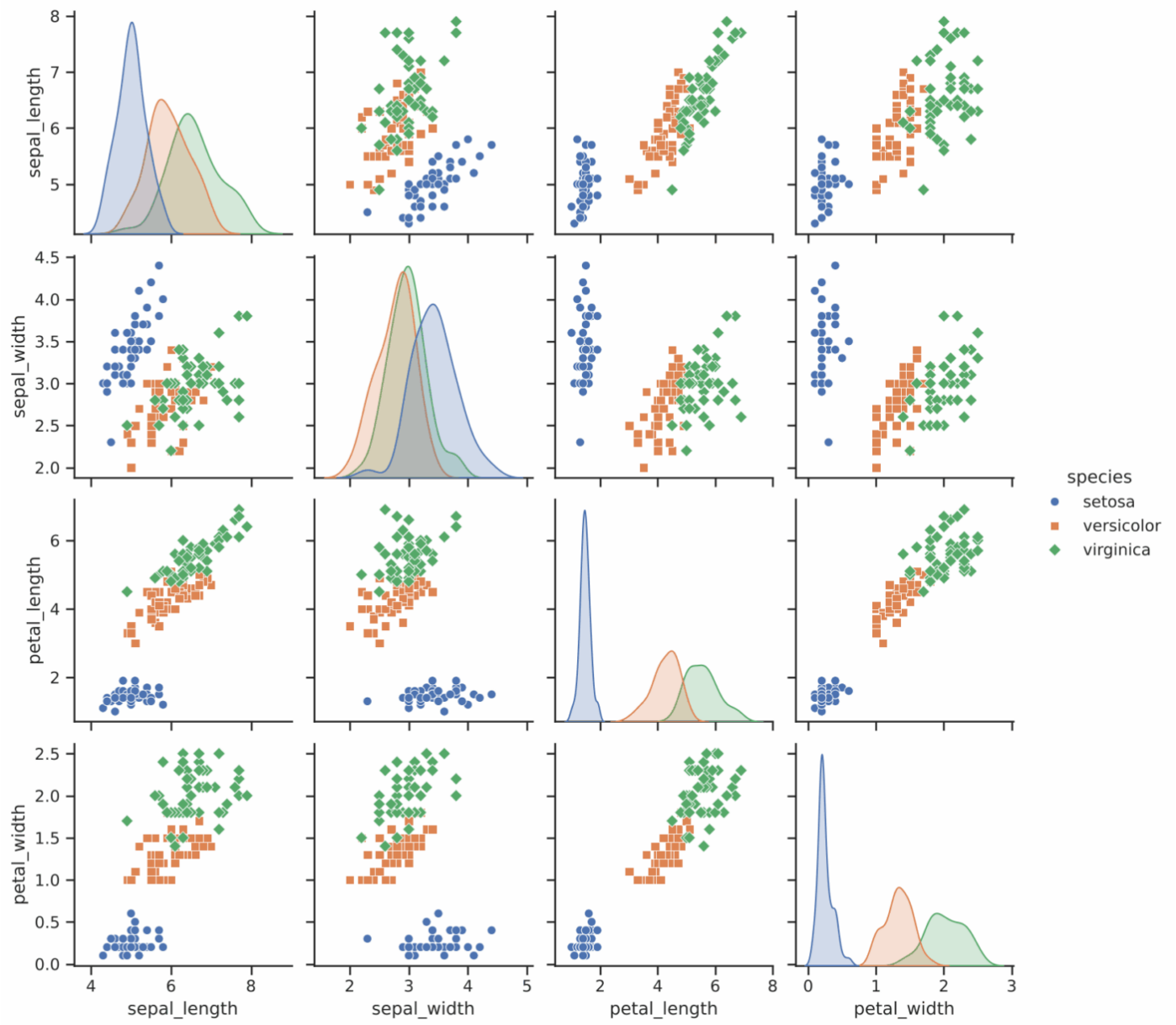
Pair Plot

Un pair plot (diagramme de paires, ou matrice de diagrammes de dispersion) fournit une grille de diagrammes où chacun montre la relation entre deux variables. Les cellules de la diagonale affichent généralement des histogrammes ou des KDE pour chaque variable seule, donc sa distribution. Les cellules hors diagonale affichent souvent des diagrammes de dispersion, qui montrent comment deux variables sont liées. Il est particulièrement utile pour comprendre les motifs, les corrélations et les distributions de plusieurs dimensions des données.

Utilise `sns.pairplot(data)`. Tu peux changer le type de tracé (`kind`), les couleurs, les marqueurs, etc., et modifier les tracés de la diagonale avec `diag_kind`.

```python
sns.set_theme(style="ticks")
sns.pairplot(iris, hue="species", markers=["o", "s", "D"])
plt.show()
```

Python

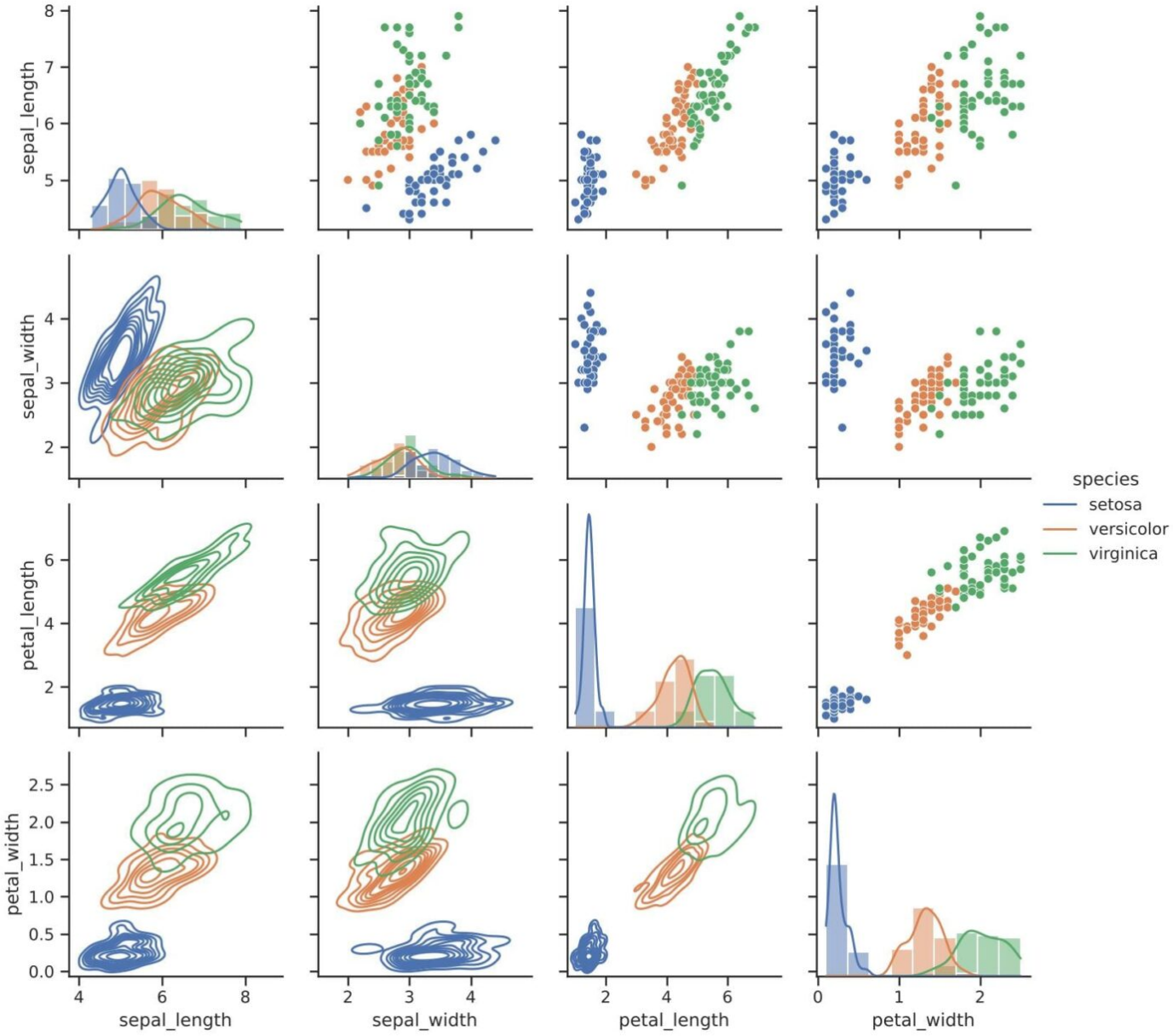


Pair Grid

Un `PairGrid` permet de personnaliser séparément les tracés du triangle inférieur, du triangle supérieur et de la diagonale.

```python
g = sns.PairGrid(iris, hue="species")
g.map_upper(sns.scatterplot)
g.map_diag(sns.histplot, kde=True)
g.map_lower(sns.kdeplot)
g.add_legend()
plt.show()
```

Python



La suite de la série porte sur les diagrammes matriciels avec Seaborn. Documentation : grilles et graphiques multiples avec Seaborn.

Pour aller plus loin, consulte aussi notre article sur Seaborn pour la data science.


SEABORN