

1. Bar Plot (Diagramme en barres)

Quand l'utiliser ?

- Comparer des **catégories** ou des **sous-catégories**.
- Représenter des **moyennes**, des **totaux**, ou des **fréquences**.

Exemple :

Comparer les ventes par produit ou les revenus par région.

python

Copier le code

```
sns.barplot(data=df, x='catégorie', y='valeur', hue='sous-catégorie')
```

2. Point Plot

Quand l'utiliser ?

- Comparer des **moyennes** tout en montrant les variations entre les catégories.
- Visualiser les tendances dans un ensemble de données catégorielles.

Exemple :

Comparer les revenus moyens par année et par sexe.

python

Copier le code

```
sns.pointplot(data=df, x='année', y='revenu', hue='sexe')
```

3. Joint Plot

Quand l'utiliser ?

- Étudier la **relation entre deux variables numériques**, avec des histogrammes ou une densité en plus.
- Identifier des tendances ou des regroupements.

Exemple :

Analyser la relation entre le prix d'un produit et sa note.

python

Copier le code

```
sns.jointplot(data=df, x='prix', y='note', kind='scatter')
```

4. Pie Plot (Diagramme circulaire)

Quand l'utiliser ?

- Montrer la **proportion** ou la **composition d'un tout**.
- Utiliser uniquement si les catégories sont peu nombreuses.

Exemple :

Répartition du budget en pourcentage.

python

Copier le code

```
plt.pie(sizes, labels=categories, autopct='%1.1f%%')
```

5. Lm Plot (Linear Model Plot)

Quand l'utiliser ?

- Visualiser une **relation linéaire** entre deux variables numériques.
- Identifier si une régression est adaptée.

Exemple :

Analyser la relation entre le temps d'étude et les résultats aux examens.

python

Copier le code

```
sns.lmplot(data=df, x='temps_étude', y='résultats')
```

6. Kde Plot (Kernel Density Plot)

Quand l'utiliser ?

- Montrer la **distribution d'une variable continue**.
- Comparer plusieurs distributions.

Exemple :

Comparer les revenus de deux groupes sociaux.

python

Copier le code

```
sns.kdeplot(data=df, x='revenu', hue='groupe', fill=True)
```

7. Violin Plot

Quand l'utiliser ?

- Visualiser à la fois la **distribution** et les **valeurs aberrantes** d'une variable.
- Comparer plusieurs distributions catégoriques.

Exemple :

Analyser la répartition des salaires par secteur d'activité.

python

Copier le code

```
sns.violinplot(data=df, x='secteur', y='salaire', hue='sexe', split=True)
```

8. Heatmap (Carte thermique)

Quand l'utiliser ?

- Montrer des **relations** ou des **patterns** dans des données matricielles ou de corrélation.
- Identifier des valeurs élevées ou faibles dans une matrice.

Exemple :

Analyse de la corrélation entre plusieurs variables.

python

Copier le code

```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
```

9. Box Plot (Boîte à moustaches)

Quand l'utiliser ?

- Représenter la **distribution** d'une variable tout en montrant les valeurs aberrantes.
- Comparer plusieurs distributions catégoriques.

Exemple :

Comparer les notes des étudiants entre différentes classes.

python

Copier le code

```
sns.boxplot(data=df, x='classe', y='note')
```

10. Swarm Plot

Quand l'utiliser ?

- Afficher chaque point de données dans une distribution catégorique.
- Complément utile aux boxplots ou violin plots.

Exemple :

Visualiser les salaires individuels par département.

python

Copier le code

```
sns.swarmplot(data=df, x='département', y='salaire')
```

11. Pair Plot

Quand l'utiliser ?

- Explorer les **relations pairées** entre plusieurs variables numériques.
- Identifier des schémas ou des clusters.

Exemple :

Analyser les relations entre les caractéristiques des fleurs (longueur, largeur, etc.).

python

Copier le code

```
sns.pairplot(data=df, hue='espèce')
```

12. Count Plot

Quand l'utiliser ?

- Compter les occurrences de catégories dans une variable catégorielle.

Exemple :

Voir la répartition des genres (H/F) dans une classe.

python

Copier le code

```
sns.countplot(data=df, x='genre')
```

13. Cluster (Clustering)

Quand l'utiliser ?

- Identifier des **groupes similaires** dans les données.
- Utiliser le clustering pour regrouper des points avec des caractéristiques similaires.

Exemple :

Segmentation des clients selon leurs habitudes d'achat.

python

Copier le code

```
from sklearn.cluster import KMeans  
  
kmeans = KMeans(n_clusters=3).fit(data)
```

14. Venn Diagram (Diagramme de Venn)

Quand l'utiliser ?

- Montrer les **intersections** et les **différences** entre ensembles.
- Comparer deux ou trois ensembles au maximum.

Exemple :

Comparer les clients ayant acheté deux produits différents.

python

Copier le code

```
from matplotlib_venn import venn2  
  
venn2([set1, set2])
```

15. Donut Plot (Diagramme en anneau)

Quand l'utiliser ?

- Semblable au diagramme circulaire, mais avec une esthétique plus moderne.
- Montrer la proportion des catégories dans un tout.

Exemple :

Répartition des dépenses dans un budget.

python

Copier le code

```
plt.pie(data, labels=labels, wedgeprops={'width': 0.4})
```

16. Spider Chart (Diagramme en araignée)

Quand l'utiliser ?

- Comparer plusieurs dimensions pour un ou plusieurs objets.
- Utilisé en analyse de performance ou de compétences.

Exemple :

Évaluer les compétences techniques d'un candidat.

python

Copier le code

```
ax.fill(angles, valeurs, color='lightblue', alpha=0.5)
```

Résumé des utilisations selon les besoins

Type de graphique	Objectif principal
Bar Plot	Comparaison catégorielle.
Point Plot	Comparaison avec des tendances.
Joint Plot	Relation entre deux variables.
Pie Plot	Proportions d'un tout.
Lm Plot	Régression linéaire.
Kde Plot	Distribution continue.
Violin Plot	Distribution avec détails.
Heatmap	Relations matricielles.
Box Plot	Distribution et valeurs aberrantes.
Swarm Plot	Distribution détaillée.
Pair Plot	Relations multiples.

Type de graphique	Objectif principal
Count Plot	Fréquences catégoriques.
Cluster	Grouperments similaires.
Venn Diagram	Intersections d'ensembles.
Donut Plot	Proportions modernes.
Spider Chart	Comparaisons multidimensionnelles.

Si tu veux des exemples plus détaillés ou des explications supplémentaires, dis-le-moi ! 😊

40

Vous avez dit :

Plotly en fr

ChatGPT a dit :

ChatGPT

Plotly : Visualisation interactive de données en Python (en français)

Plotly est une bibliothèque Python puissante pour créer des visualisations interactives et dynamiques. Contrairement à des bibliothèques comme Matplotlib ou Seaborn, Plotly permet d'explorer les graphiques en temps réel (zoom, survol, sélection).

Installation

Pour installer Plotly, utilisez la commande suivante :

```
bash
```

Copier le code

```
pip install plotly
```

Importation et premiers pas

Avant de commencer à créer des graphiques, importez Plotly et, si nécessaire, chargez un jeu de données :

python

Copier le code

```
import plotly.express as px
```

```
import plotly.graph_objects as go
```

```
# Exemple de jeu de données
```

```
df = px.data.gapminder() # Données sur les pays, leur population, PIB, etc.
```

```
print(df.head())
```

Types de graphiques courants avec Plotly

1. Scatter Plot (Nuage de points)

Pour analyser la relation entre deux variables numériques.

python

Copier le code

```
fig = px.scatter(df, x='gdpPercap', y='lifeExp', color='continent',
```

```
                  size='pop', hover_name='country', title="Relation entre PIB et espérance de vie")
```

```
fig.show()
```

2. Line Plot (Graphique en courbes)

Pour montrer une tendance ou une série temporelle.

python

Copier le code

```
fig = px.line(df[df['country'] == 'France'], x='year', y='lifeExp', title="Espérance de vie en France au fil des années")
```

```
fig.show()
```

3. Bar Plot (Diagramme en barres)

Pour comparer des catégories ou des sous-catégories.

python

Copier le code

```
fig = px.bar(df[df['year'] == 2007], x='continent', y='pop', color='continent',  
             title="Population par continent en 2007", text='pop')  
  
fig.show()
```

4. Pie Chart (Diagramme circulaire)

Pour représenter la proportion des catégories dans un tout.

python

Copier le code

```
fig = px.pie(df[df['year'] == 2007], values='pop', names='continent',  
            title="Répartition de la population mondiale par continent (2007)")  
  
fig.show()
```

5. Histogramme

Pour analyser la distribution d'une variable.

python

```
fig = px.histogram(df, x='lifeExp', nbins=20, title="Distribution de l'espérance de vie")  
  
fig.show()
```

6. Box Plot (Boîte à moustaches)

Pour visualiser la distribution et les valeurs aberrantes.

python

```
fig = px.box(df, x='continent', y='lifeExp', color='continent',  
             title="Distribution de l'espérance de vie par continent")  
  
fig.show()
```

7. Heatmap (Carte thermique)

Pour visualiser des relations entre des données matricielles (corrélation, intensité, etc.).

python

Copier le code

```
import numpy as np  
  
import pandas as pd  
  
# Exemple de matrice de corrélation  
  
data = df[df['year'] == 2007][['lifeExp', 'pop', 'gdpPercap']]  
  
correlation = data.corr()
```

```
fig = go.Figure(data=go.Heatmap(z=correlation.values,  
                                x=correlation.columns, y=correlation.columns,  
                                colorscale='Viridis'))  
  
fig.update_layout(title="Matrice de corrélation")  
  
fig.show()
```

8. Bubble Chart (Nuage de points avec taille variable)

Pour analyser plusieurs dimensions en même temps.

python

Copier le code

```
fig = px.scatter(df[df['year'] == 2007], x='gdpPercap', y='lifeExp', size='pop', color='continent',  
                hover_name='country', log_x=True, title="PIB vs Espérance de vie (2007)")  
  
fig.show()
```

9. Sunburst Chart (Diagramme en rayons solaires)

Pour visualiser une hiérarchie.

python

Copier le code

```
fig = px.sunburst(df[df['year'] == 2007], path=['continent', 'country'], values='pop',  
                 title="Répartition de la population par continent et pays (2007)")  
  
fig.show()
```

10. Map (Carte géographique)

Pour afficher des données géographiques.

python

Copier le code

```
fig = px.choropleth(df[df['year'] == 2007], locations='iso_alpha', color='lifeExp',  
                   hover_name='country', color_continuous_scale='Viridis',  
                   title="Espérance de vie par pays (2007)")  
  
fig.show()
```

11. Violin Plot

Pour combiner la distribution et les statistiques descriptives.

python

Copier le code

```
fig = px.violin(df, y='lifeExp', x='continent', color='continent', box=True,  
               title="Distribution de l'espérance de vie par continent")  
  
fig.show()
```

Personnalisation des graphiques

Les graphiques dans Plotly peuvent être personnalisés à l'infini. Voici quelques exemples :

1. Ajout de titres et légendes :

python

Copier le code

```
fig.update_layout(title="Mon graphique personnalisé", xaxis_title="Axe X", yaxis_title="Axe Y")
```

2. Changement des couleurs :

python

Copier le code

```
fig.update_traces(marker=dict(color='crimson'))
```

3. Thèmes prédéfinis :

python

Copier le code

```
fig.update_layout(template='plotly_dark') # Autres thèmes : plotly, ggplot2, seaborn, etc.
```

Quand utiliser Plotly ?

- Lorsque l'interactivité est essentielle (zoom, survol, filtres).
 - Lorsque vous souhaitez présenter des visualisations complexes dans un format dynamique (tableaux de bord, rapports interactifs).
 - Pour des présentations ou des tableaux de bord web.
-

Résumé des graphiques selon leur usage

Type de graphique	Quand l'utiliser ?
Scatter Plot	Étudier la relation entre deux variables continues.
Line Plot	Montrer des tendances dans une série temporelle.
Bar Plot	Comparer des catégories ou sous-catégories.
Pie Chart	Représenter les proportions d'un tout.
Histogram	Montrer la répartition d'une variable.
Box Plot	Analyser la distribution avec des valeurs aberrantes.
Heatmap	Visualiser des relations matricielles.
Bubble Chart	Étudier plusieurs dimensions avec des tailles de points.
Sunburst Chart	Visualiser des hiérarchies ou des répartitions complexes.
Map	Représenter des données géographiques.
Violin Plot	Combiner densité et statistiques descriptives.