SY09 Printemps 2020
TP 02 — Visualisation de données

La visualisation est une part importante de l'analyse de données que ce soit lors de travaux préliminaires d'exploration pour mettre en évidence des informations cachées ou pour la présentation de résultats.

Il existe de nombreux outils permettant de faire de la visualisation. Parmi les solutions existantes en Python, nous utiliserons la bibliothèque `seaborn` qui a l'avantage d'être assez exhaustive tout en proposant une interface consistante. Elle s'appuie sur la bibliothèque `matplotlib` que nous utiliserons également pour adapter certaines figures.

Un alias couramment utilisé est le suivant :

```
import seaborn as sns
```

Nous utiliserons les jeux de données `titanic` et `iris`, présents dans `seaborn`, qu'on peut charger avec les instructions suivantes :

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

1 Visualisation univariée

La visualisation univariée consiste à étudier un seul prédicteur à la fois sans tenir compte des liens avec les autres prédicteurs.

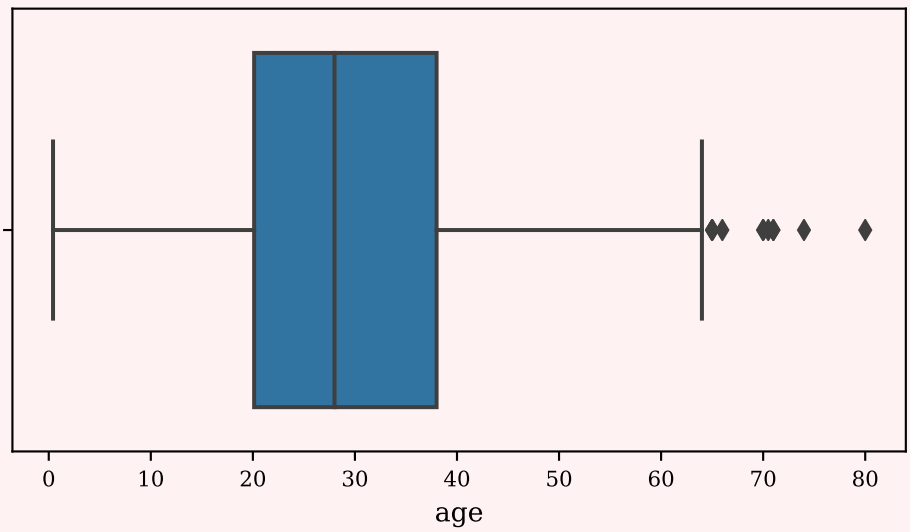
Prédicteur quantitatif

Pour les prédicteurs quantitatifs, on peut utiliser les boites à moustaches. Avec `seaborn`, on a la syntaxe suivante

```
sns.boxplot(x=<Série Pandas>)
```

(1) Tracer la boite à moustaches des âges des passagers du titanic. Quel est l'âge médian ?

```
In [1]: sns.boxplot(x=titanic.age)
        plt.show()
```



L'âge médian est autour de 28 ans.

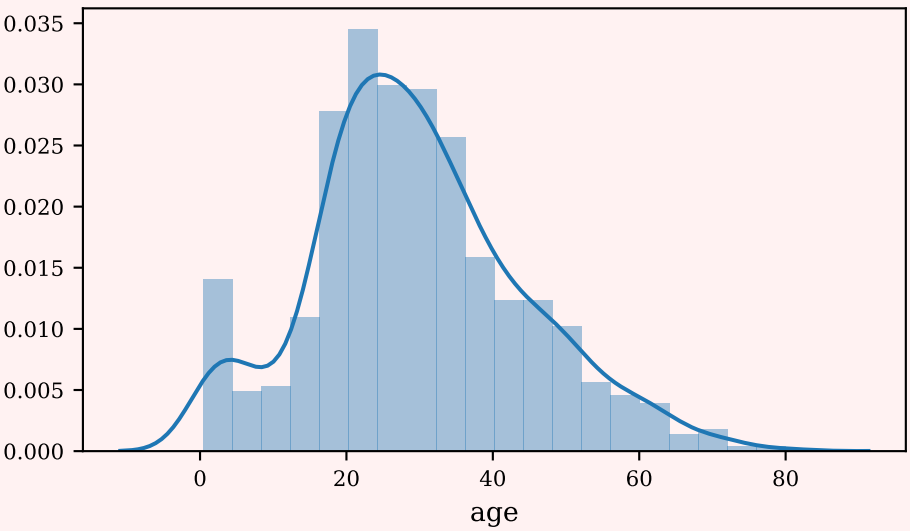
L'autre visualisation classique d'une variable quantitative est l'histogramme qu'on peut obtenir avec l'instruction suivante

```
sns.distplot(<Série Pandas>)
```

(2) Tracer l'histogramme des âges des passagers. Que font les arguments optionnels suivants ?

1. `bins`
2. `kde`
3. `rug`

```
In [2]: sns.distplot(titanic.age.dropna())
        plt.show()
```



On peut modifier l'apparence de l'histogramme avec les arguments optionnels suivant :

1. `bins` : permet de régler le nombre de boîtes.
2. `kde` : ajoute une courbe de densité
3. `rug` : ajoute une distribution unidimensionnelle des données

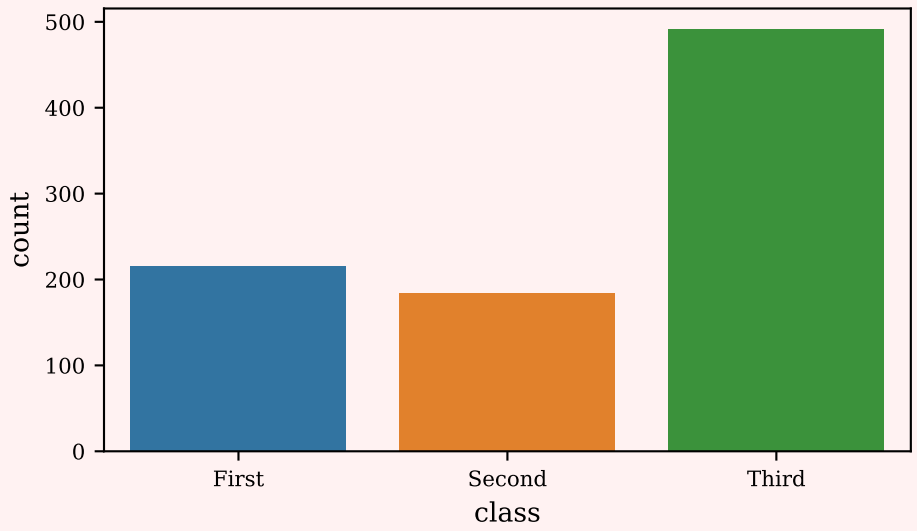
Prédicteur qualitatif

Pour les prédicteurs qualitatifs, on utilise un diagramme en barres avec les syntaxes possibles suivantes :

```
sns.countplot(<Série Pandas>)
sns.countplot(x=<Colonne>, data=<DataFrame Pandas>)
```

(3) Représenter la donnée des classes des passagers.

```
In [3]: sns.countplot(titanic["class"])
        plt.show()
```



2 Visualisation multivariée

La visualisation multivariée consiste à étudier les relations pouvant exister entre plusieurs prédicteurs. Les techniques utilisées varient principalement en fonction de la nature des prédicteurs : qualitatifs ou quantitatifs.

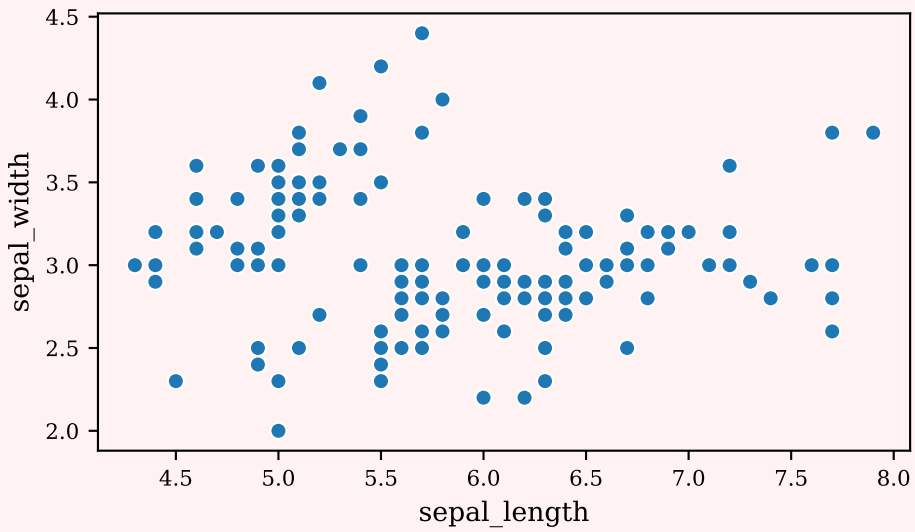
Quantitatif *vs* quantitatif

Diagramme de dispersion Le diagramme de dispersion permet de visualiser les relations existantes entre deux variables qualitatives. On trace un diagramme de dispersion avec l'instruction suivante

```
sns.scatterplot(
  x=<Colonne des abscisses>,
  y=<Colonne des ordonnées>,
  data=<DataFrame Pandas>
)
```

(4) Tracer le diagramme de dispersion de la largeur du sépale en fonction de sa longueur.

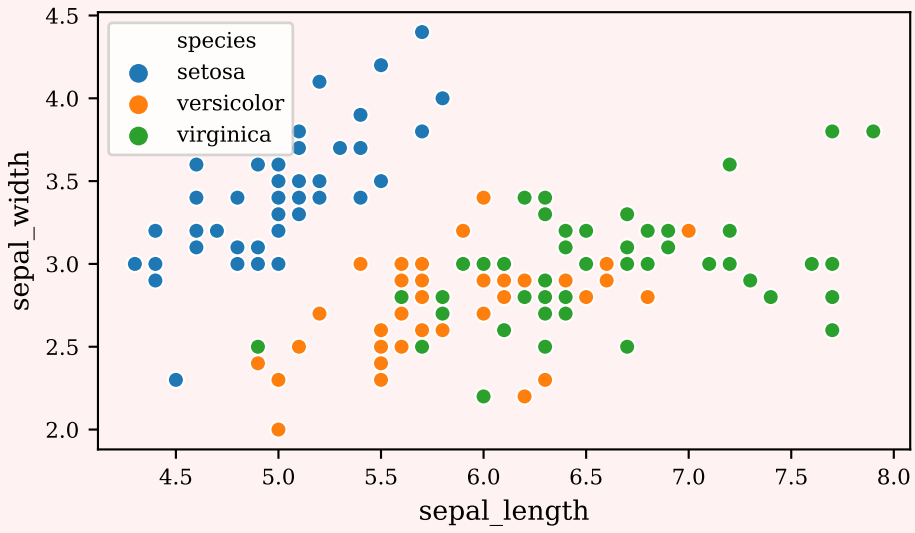
```
In [4]: sns.scatterplot(x="sepal_length", y="sepal_width", data=iris)
        plt.show()
```



Il est possible de visualiser d'autres variables en jouant sur la couleur, la forme ou la taille des points avec les arguments `hue`, `style` et `size`.

(5) Rajouter la donnée de classe avec la couleur.

```
In [5]: sns.scatterplot(x="sepal_length", y="sepal_width", hue="species",
        ↪ data=iris)
        plt.show()
```



(6) En plus de la donnée de classe, ajouter la longueur du pétale avec la taille du point.

```
In [6]: sns.scatterplot(x="sepal_length", y="sepal_width", hue="species",
        ↪ size="petal_length", data=iris)
        plt.show()
```

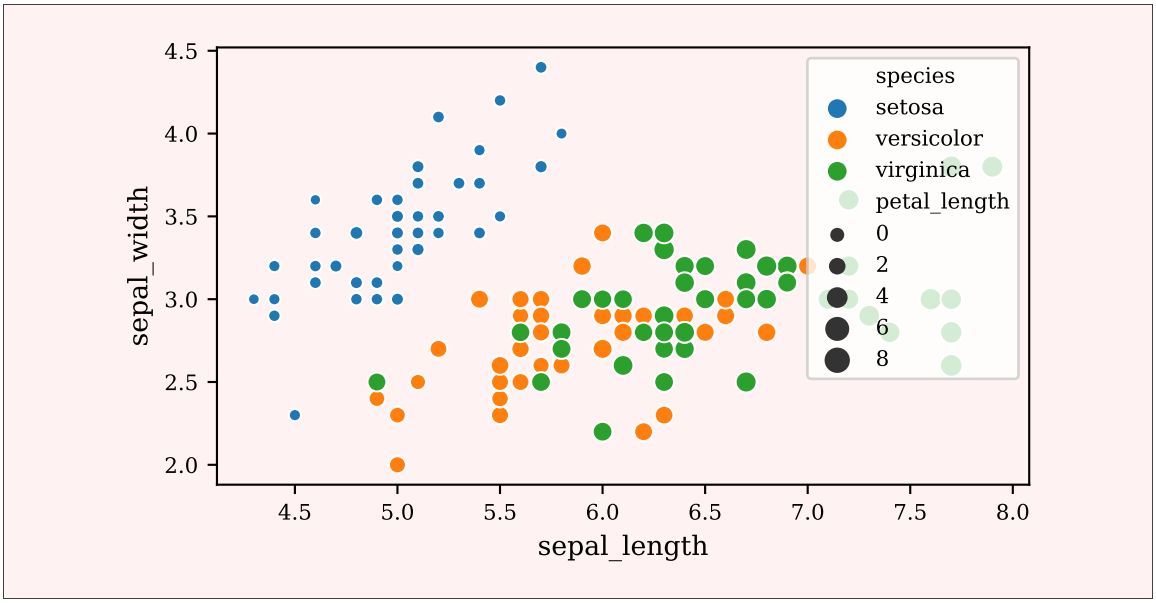
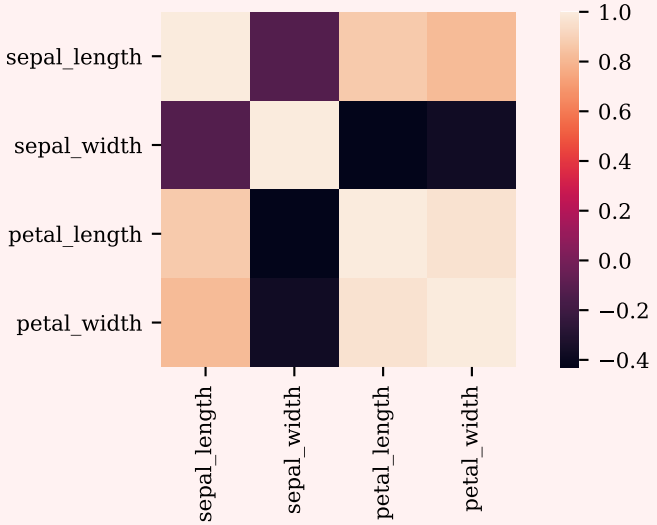


Diagramme de corrélation Le diagramme de corrélation est utile pour avoir une idée des liens de type linéaire entre des variables quantitatives. Le diagramme de corrélation n'est pas directement supporté par `seaborn` : il faut d'abord créer ce tableau à l'aide de la fonction `Pandas corr` et utiliser ensuite la visualisation `heatmap`.

```
corr = <DataFrame Pandas>.corr()
sns.heatmap(corr)
```

(7) Étudier les corrélations linéaires des variables quantitatives du jeu de données iris. Peut-on dire que les largeurs du sépale et du pétale sont corrélés négativement ? Étudier l'influence de l'espèce.

```
In [7]: iris_quant = iris.drop(columns="species")
        corr =  iris_quant.corr()
        sns.heatmap(corr, square=True)
        plt.show()
```



La corrélation globale est négative. Conditionnellement à la classe, la corrélation est positive (paradoxe de Simpson).

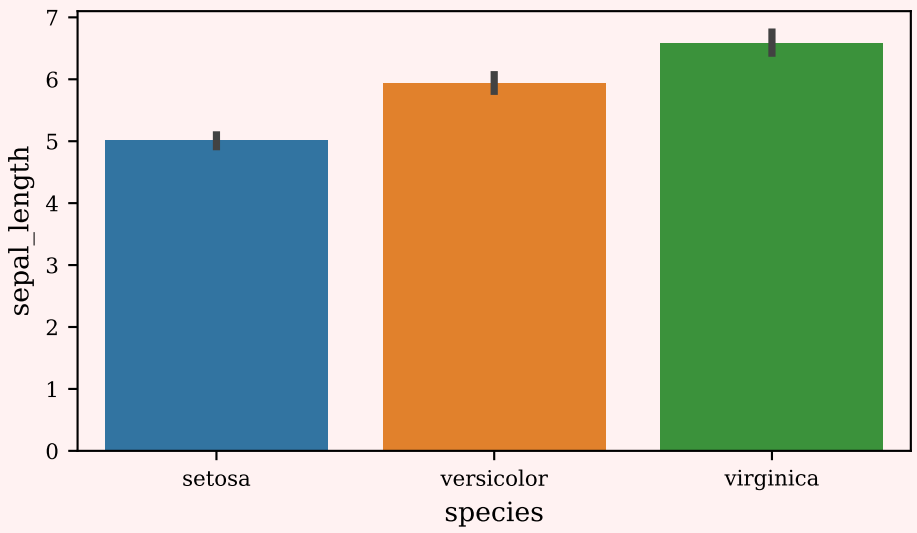
Qualitatif *vs* quantitatif

Diagramme en barres avec dispersion La fonction `barplot` permet de visualiser une variable quantitative avec une indication sur la dispersion des données sous forme d'un intervalle de confiance en fonction d'une colonne qualitative. La syntaxe est la suivante

```
sns.barplot(
  x=<Colonne qualitative>,
  y=<Colonne quantitative>,
  data=<DataFrame Pandas>
)
```

(8) Avec le jeu de données `iris`, visualiser la longueur des sépales en fonction de l'espèce.

```
In [8]: sns.barplot(x="species", y="sepal_length", data=iris)
        plt.show()
```

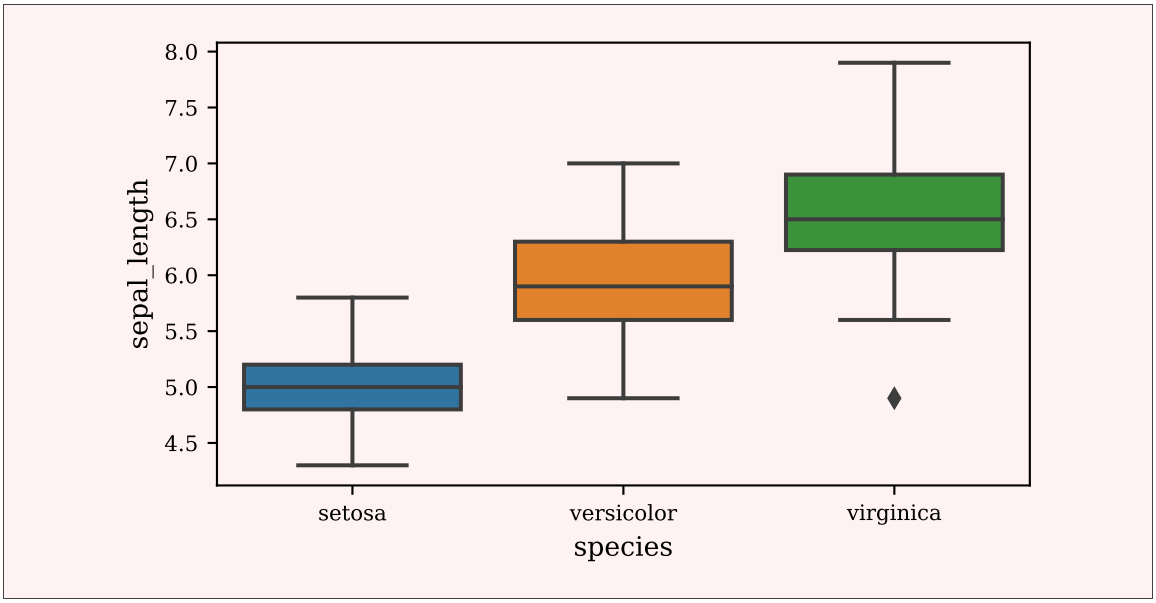


Boites à moustaches multiples Au lieu d'afficher un diagramme en bâtons, on peut afficher des boites à moustaches. La syntaxe est similaire.

```
sns.boxplot(
  x=<Colonne qualitative>,
  y=<Colonne quantitative>,
  data=<DataFrame Pandas>
)
```

(9) Avec le jeu de données `iris`, visualiser les boites à moustaches des longueur des sépales en fonction de l'espèce.

```
In [9]: sns.boxplot(x="species", y="sepal_length", data=iris)
        plt.show()
```



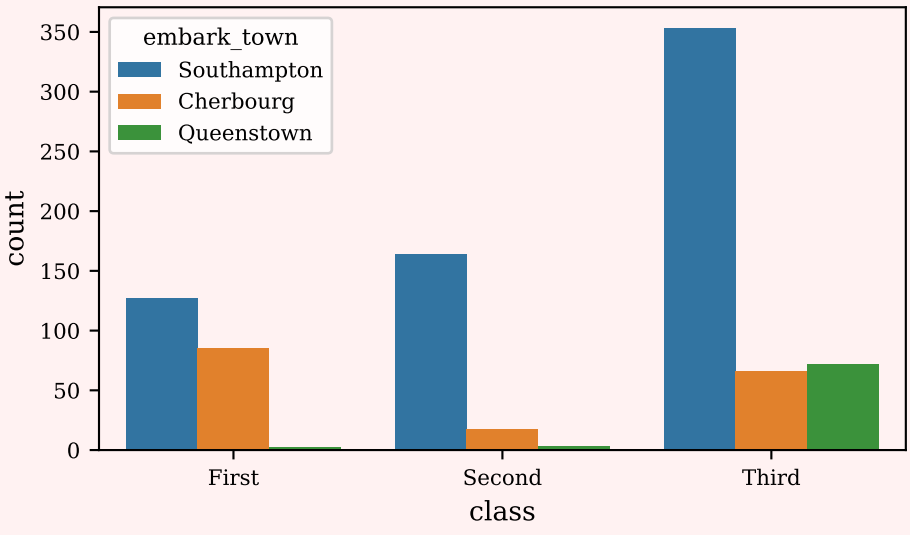
Qualitatif *vs* qualitatif

Lorsque les deux variables sont qualitatives, les représentations possibles sont centrées autour du tableau de contingence. On peut utiliser la fonction `countplot` en fournissant la deuxième variable qualitative en argument.

```
sns.countplot(
  x=<Colonne qualitative>,
  hue=<Colonne qualitative>,
  data=<DataFrame Pandas>
)
```

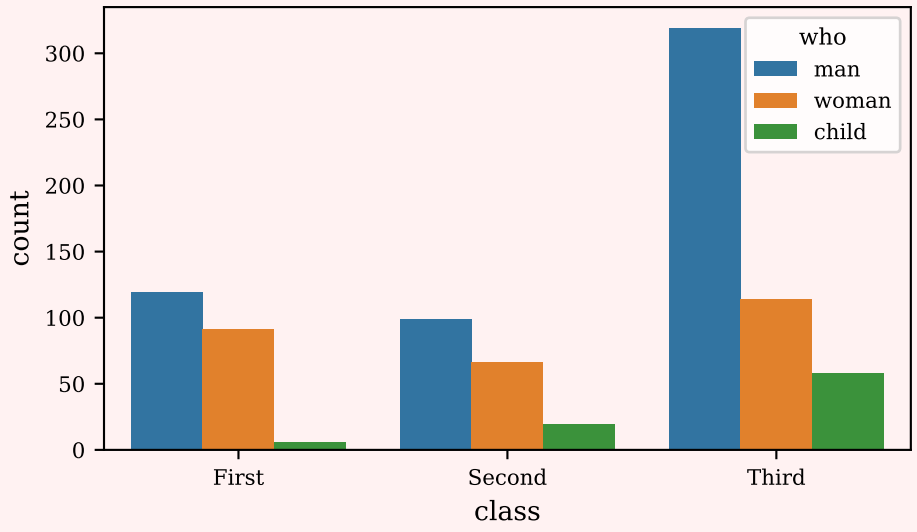
(10) Tracer les lieux d'embarquation en fonction de la classe.

```
In [10]: sns.countplot(x="class", hue="embark_town", data=titanic)
         plt.show()
```



(11) Visualiser le descripteur « who » en fonction de la classe.

```
In [11]: sns.countplot(x="class", hue="who", data=titanic)
         plt.show()
```



D'une manière générale, un jeu de données peut être divisé en plusieurs sous-jeux de données correspondant à une modalité fixe suivant un descripteur. Sur chacun de ces sous-jeux de données, on peut appliquer le même type de visualisation.

Pour réaliser cela avec `seaborn`, il faut appeler la fonction `FacetGrid` en spécifiant le jeu de données et la variable qualitative qui va partitionner ce jeu de données.

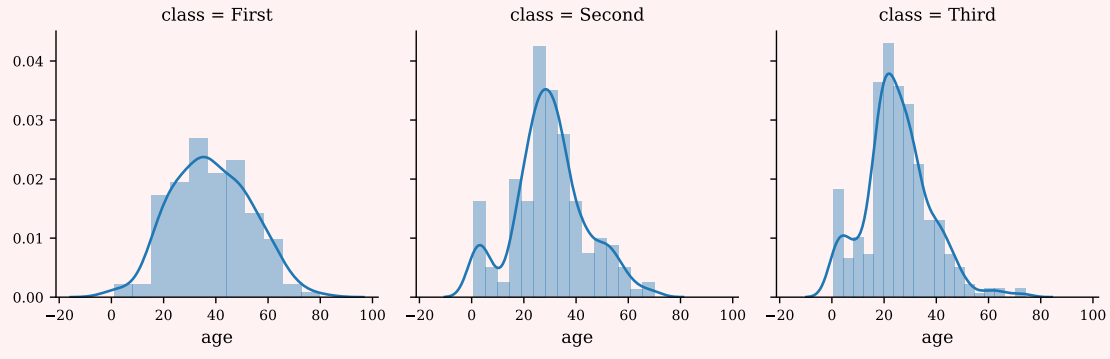
```
g = sns.FacetGrid(<DataFrame Pandas>, col=<Colonne qualitative>)
g = sns.FacetGrid(<DataFrame Pandas>, col=<Colonne qualitative1>, row=<Colonne
↪ qualitative2>)
```

On appelle ensuite la fonction `map` sur l'objet retourné qui va créer la même visualisation pour tous les sous-jeux de données. Par exemple, pour un histogramme, on exécutera

```
g.map(sns.distplot, <Argument d'un distplot>)
```

(12) Visualiser « age » en fonction de « class » avec trois histogrammes en utilisant `FacetGrid`.

```
In [12]: g = sns.FacetGrid(titanic, col="class")
         g.map(sns.distplot, "age")
Out [12]: <seaborn.axisgrid.FacetGrid object at 0x7fc138b1e6d0>
In [13]: plt.show()
```



(13) Visualiser « age » en fonction de « class » avec trois histogrammes en utilisant `FacetGrid`.

```
In  [14]: g = sns.FacetGrid(titanic, col="class", row="who")
          g.map(sns.distplot, "age")
Out [14]: <seaborn.axisgrid.FacetGrid object at 0x7fc138af55d0>
In  [15]: plt.show()
```

3 Visualisation du jeu de données `sy02-p2019.csv`

(14) Charger le jeu de données contenu dans le fichier `data/sy02-p2019.csv`. Quel est le type de la colonne « Note médian » ? Pourquoi ? On pourra utiliser l'argument `na_values`.

```
In  [16]: X = pd.read_csv("data/sy02-p2019.csv")
          X["Note médian"].dtype
Out [16]: dtype('O')
```

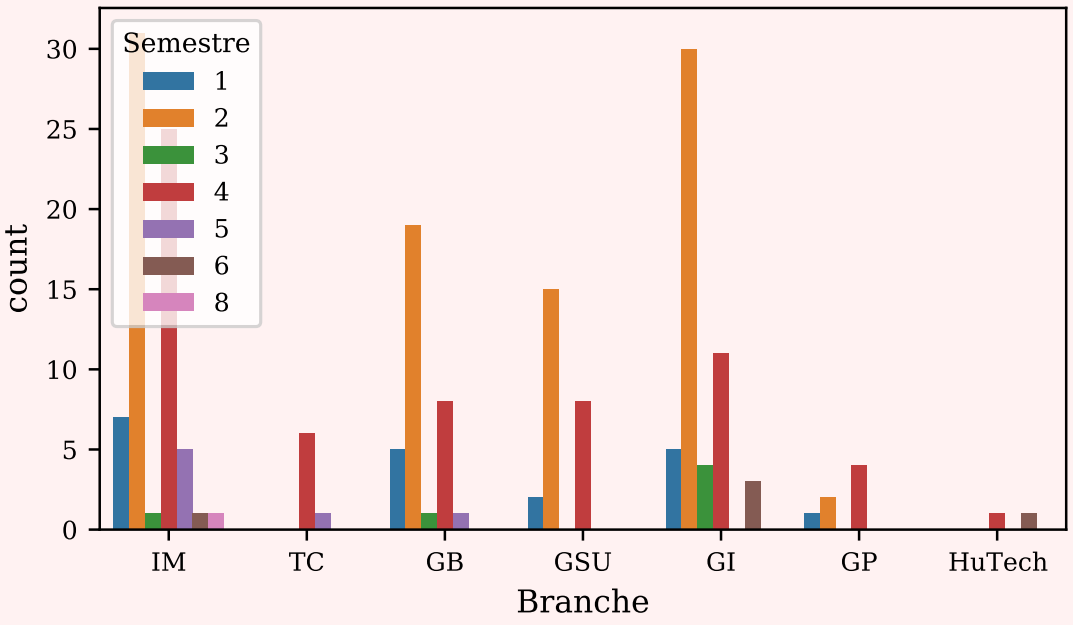
Les données ne sont pas correctement chargées. La colonne correspondant à la note de médian n'est pas reconnue comme quantitative à cause du « ABS ».

```
In  [17]: X = pd.read_csv("data/sy02-p2019.csv", na_values="ABS")
          X["Note médian"].dtype
Out [17]: dtype('float64')
```

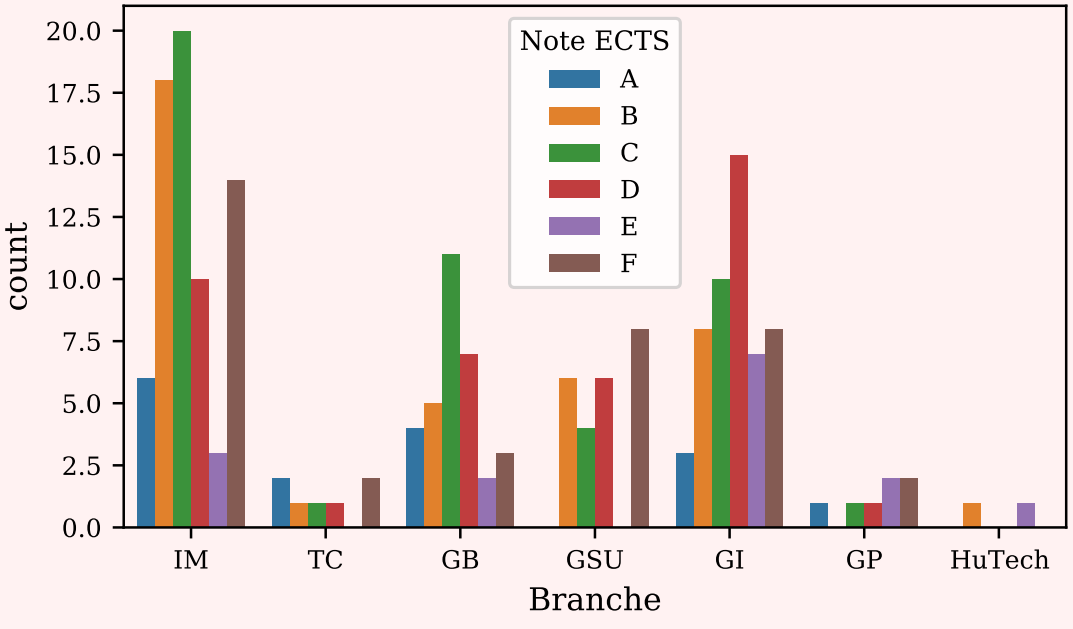
(15) Visualiser les effectifs des différentes branches. Rajouter

1. l'information de « niveau »
2. la note ECTS. Les modalités sont-elles dans l'ordre ?

```
In  [18]: sns.countplot(x="Branche", hue="Semestre", data=X)
          plt.show()
```



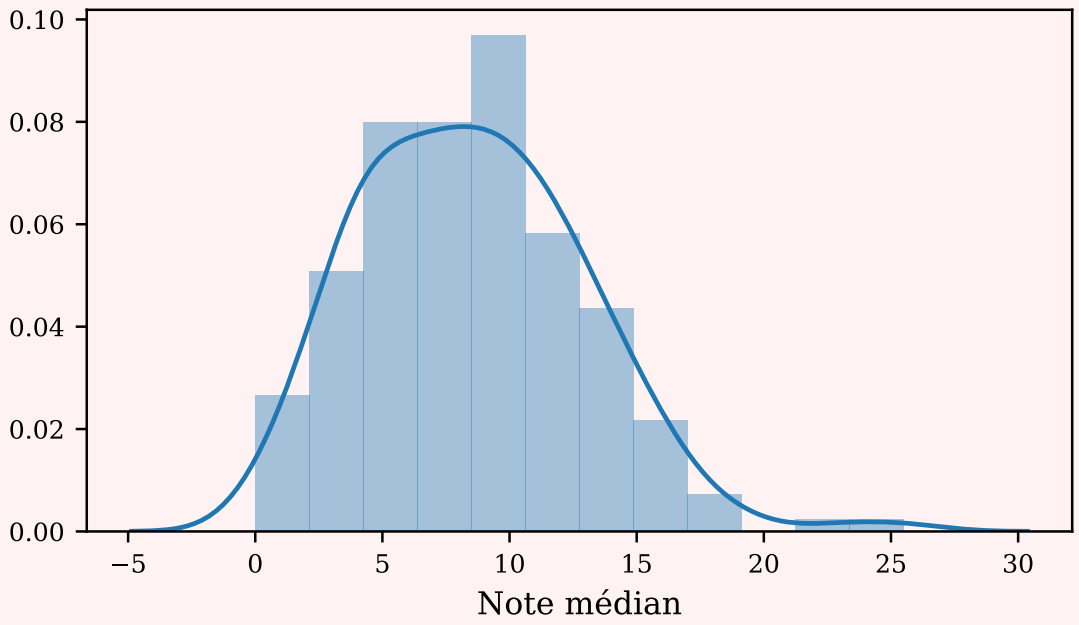

```
In  [19]: X["Note ECTS"] = pd.Categorical(X["Note ECTS"], ordered=True)
          sns.countplot(x="Branche", hue="Note ECTS", data=X)
          plt.show()
```



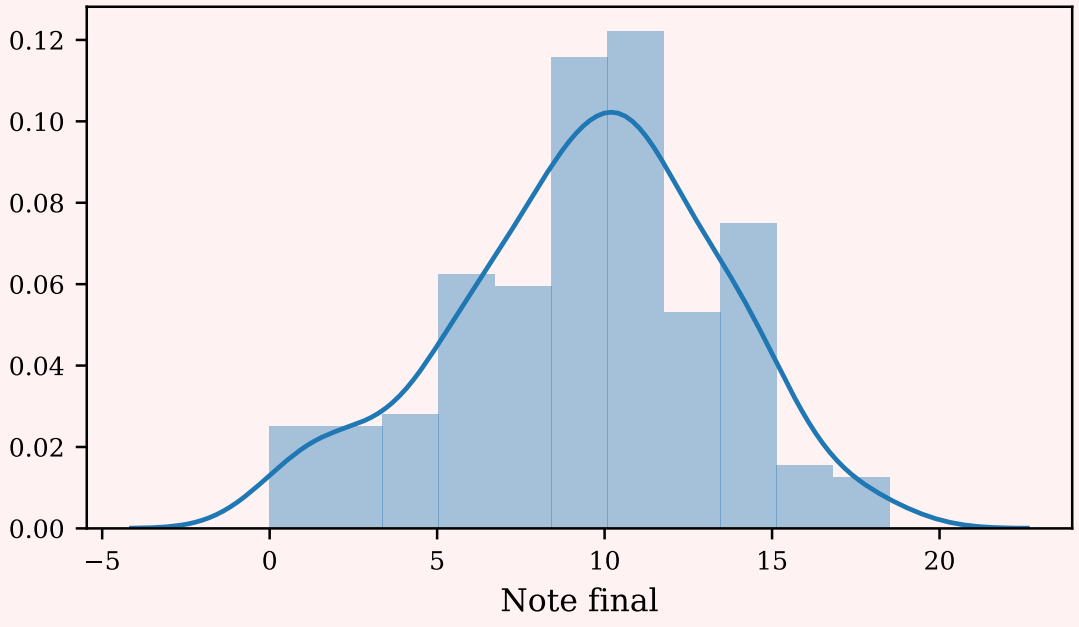
(16) Visualiser les distributions des notes de médian et des notes de final. Afficher le diagramme de dispersion.

Synthétiser les trois visualisations précédentes en utilisant `sns.jointplot`.

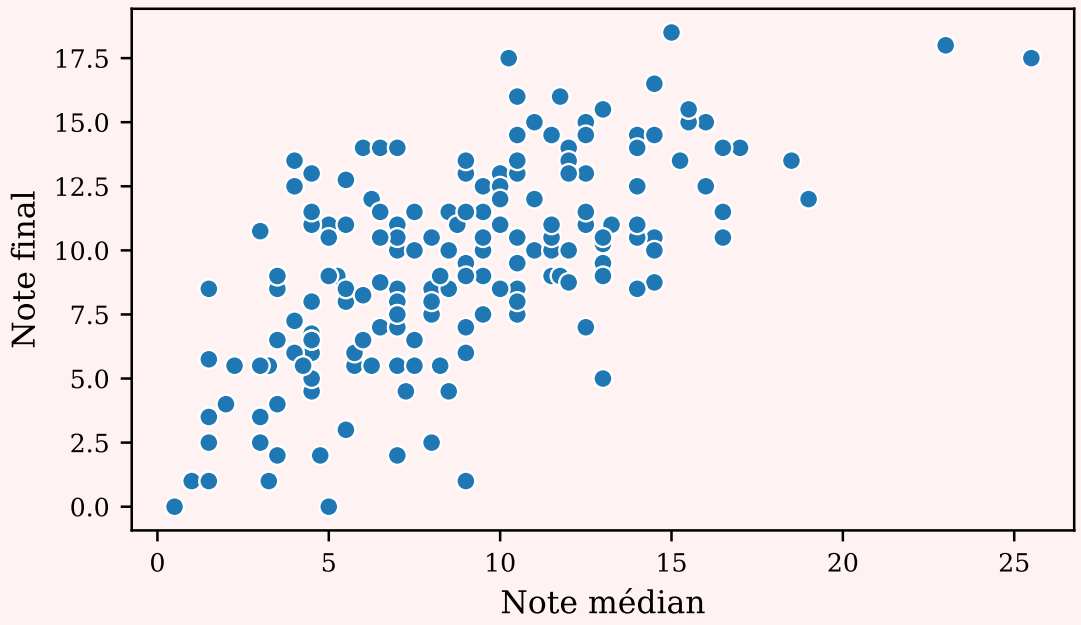
```
In [20]: sns.distplot(X["Note médian"])
         plt.show()
```



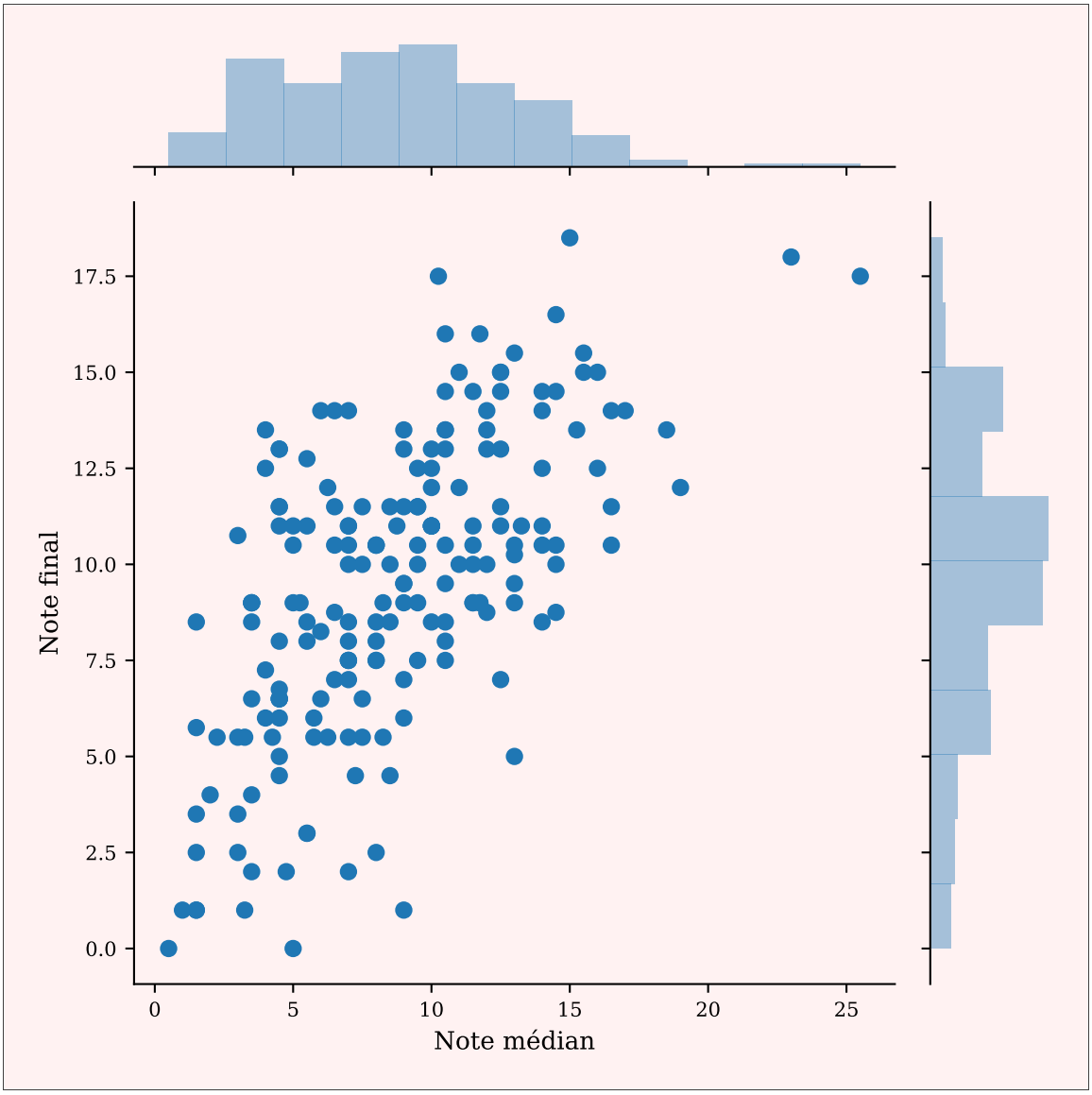

```
In [21]: sns.distplot(X["Note final"])
         plt.show()
```




```
In [22]: sns.scatterplot(x="Note médian", y="Note final", data=X)
         plt.show()
```

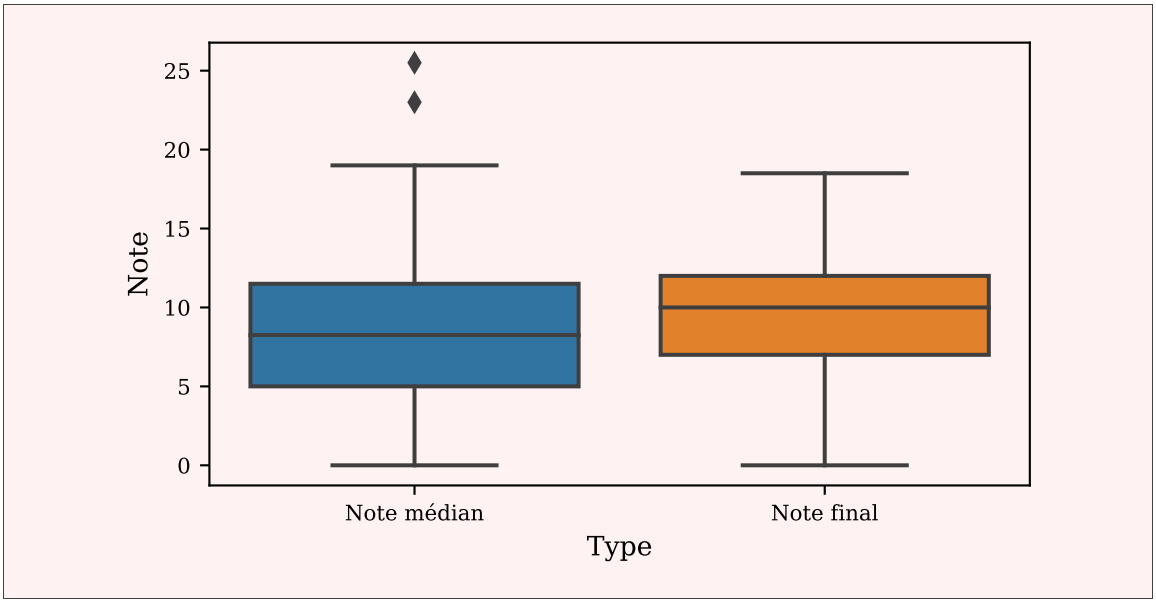



```
In [23]: sns.jointplot(x="Note médian", y="Note final", data=X)
         plt.show()
```



(17) Tracer en parallèle les boites à moustaches des notes de médian et de final. On pourra utiliser la fonction `melt`.

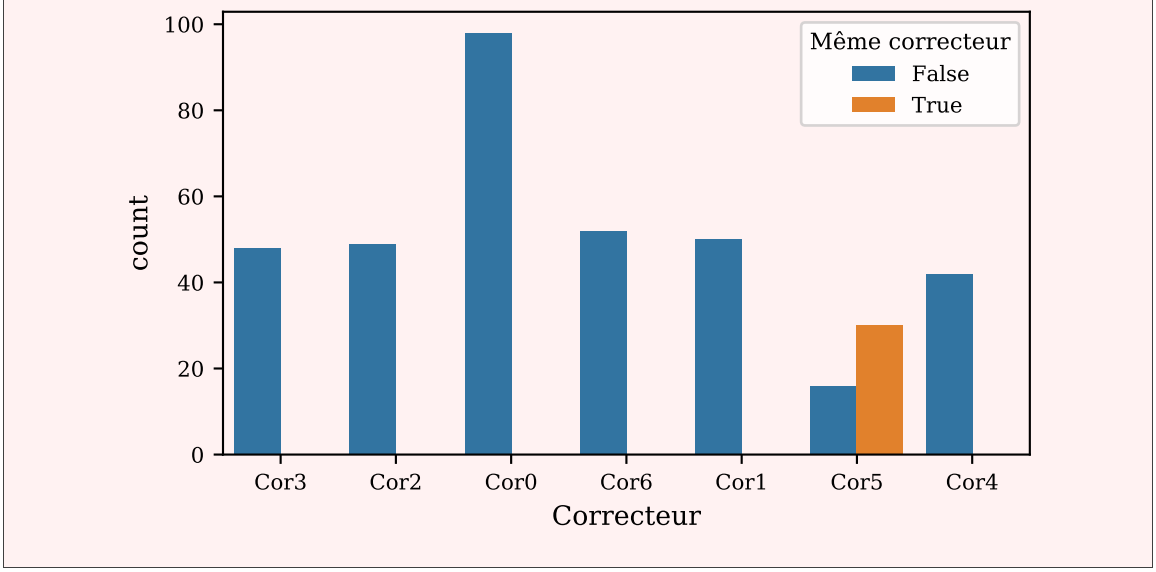
```
In [24]: X1 = X.melt(
             value_vars=["Note médian", "Note final"], var_name="Type",
             ↪  value_name="Note"
         )
         sns.boxplot(x="Type", y="Note", data=X1)
         plt.show()
```



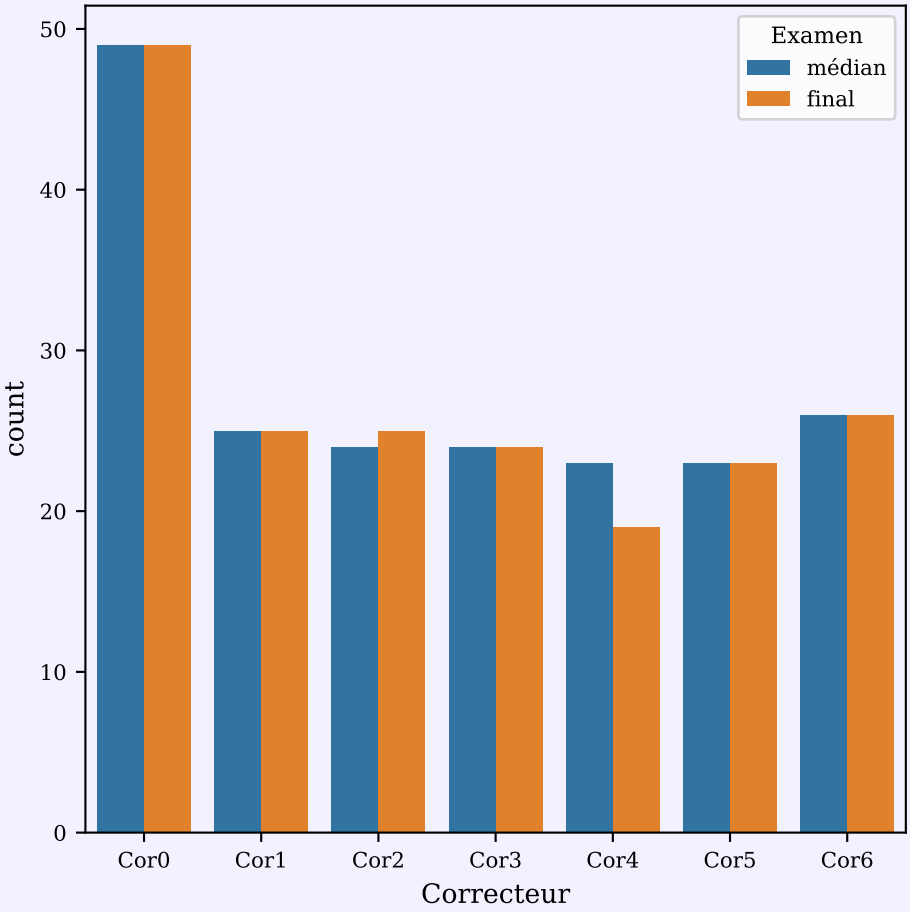
(18) Trouver une visualisation du nombre de copies corrigées sur tout le semestre par correcteur en indiquant la part des copies qui correspondent à un même étudiant du médian au final.

```
In [25]: X["Même correcteur"] = X["Correcteur médian"] == X["Correcteur final"]
         X1 = X.melt(
             id_vars=["Même correcteur"],
             value_vars=["Correcteur final", "Correcteur médian"],
             var_name="Examen",
             value_name="Correcteur",
         )

         sns.countplot(x="Correcteur", hue="Même correcteur", data=X1)
         plt.show()
```



(19) Comment obtenir la figure suivante ?



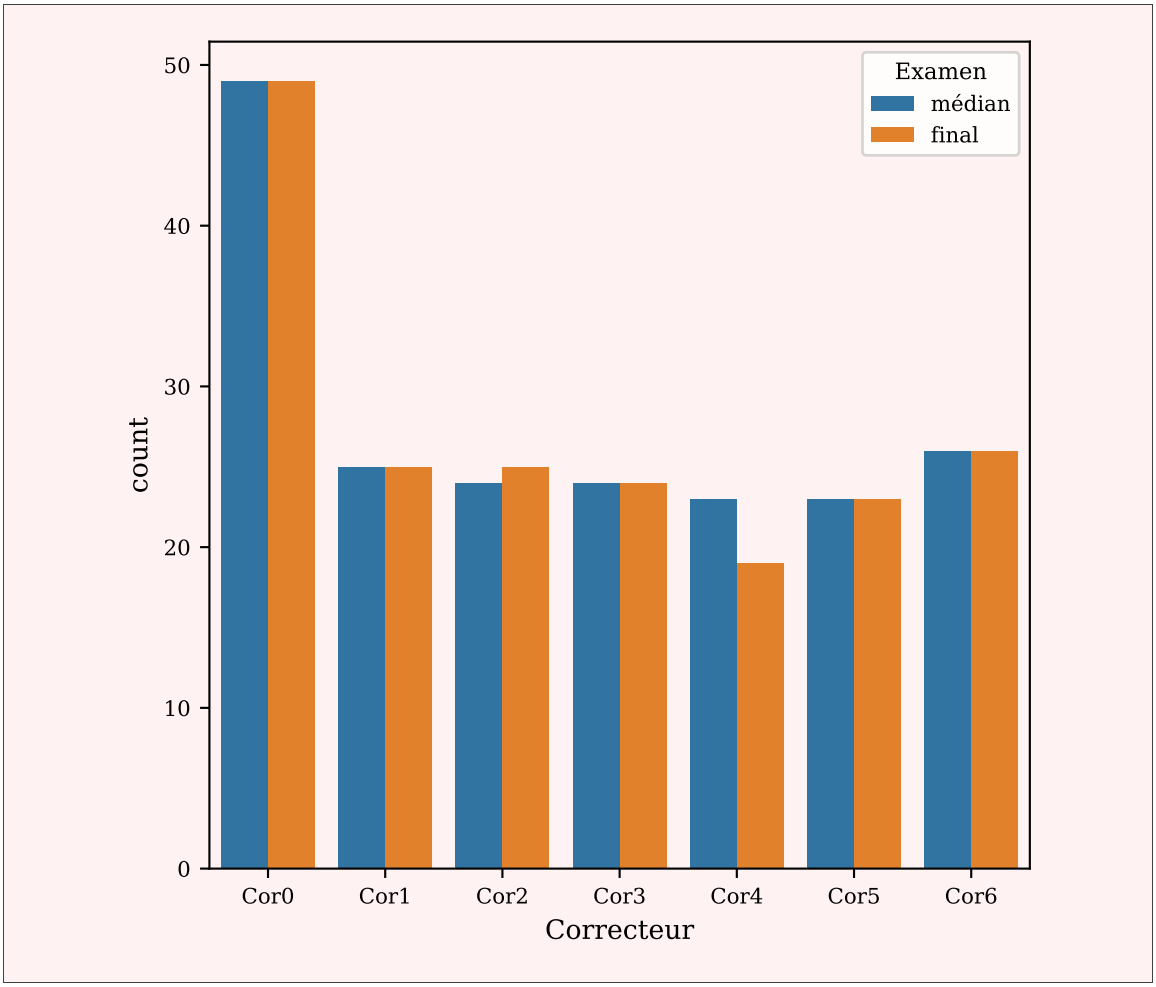

```
In [26]: X1 = X.rename(columns={"Correcteur médian": "médian", "Correcteur
         ↪ final": "final"})

         X1 = X1.melt(
             id_vars=["Note médian", "Note final"],
             value_vars=["médian", "final"],
             var_name="Examen",
             value_name="Correcteur",
         )

         X1["Note"] = np.where(X1["Examen"] == "médian", X1["Note médian"],
         ↪ X1["Note final"])

         X1 = X1.drop(columns=["Note médian", "Note final"])

         sns.countplot(x="Correcteur", hue="Examen", data=X1)
         plt.show()
```


Examen
médian
final
count
0
10
20
30
40
50
Cor0
Cor1
Cor2
Cor3
Cor4
Cor5
Cor6
Correcteur