**La data visualisation**

**Quelle figure pour quel type de variable ?**

Nous allons ici présenter des graphes créés avec les librairies **Matplotlib, Pandas et Seaborn**.

Le but est de présenter clairement **quelles sont les méthodes** que nous pouvons appeler afin de visualiser nos variables en fonction que celles-ci soient numériques et/ou catégoriques. Donc nous verrons tout d’abord la visualisation des variables numériques, puis celles des variables catégoriques et enfin nous verrons comment visualiser des variables numériques et catégoriques sur un même graphe.

**1. Comment présenter des variables numériques ?**

Les variables numériques peuvent être visualisées sous forme **d’histogrammes, de diagrammes en boîte, de courbes et de nuages de points**.

**Les histogrammes**

L’intérêt de visualiser les variables numériques seules sous forme d’histogrammes est de voir la **distribution de ces variables.** Sur l’axe des abscisses nous avons les différentes valeurs ordonnées que peut prendre le *tip*, et en ordonnées nous avons les effectifs, c’est-à-dire le nombre de mêmes valeurs de *tip* rassemblées (*counts*).

Avec les trois librairies **nous pouvons jouer avec les *bins*** afin de déterminer des intervalles de classe plus ou moins grands. Plus les *bins* augmentent, c’est-à-dire plus le nombre d’intervalle augmente et sont plus petits, plus resserrés, plus le graphique est précis. Par défaut les méthodes Matplotlib, Pandas et Seaborn calculent seules un nombre de *bins* suffisamment adéquat.

L’avantage avec Seaborn, c’est que la méthode *sns.distplot()* fait également apparaître la courbe kde : **kernel density estimate**, c’est-à-dire l’estimation de la densité de probabilité de la variable (estimation par noyau ou encore méthode de Parzen).

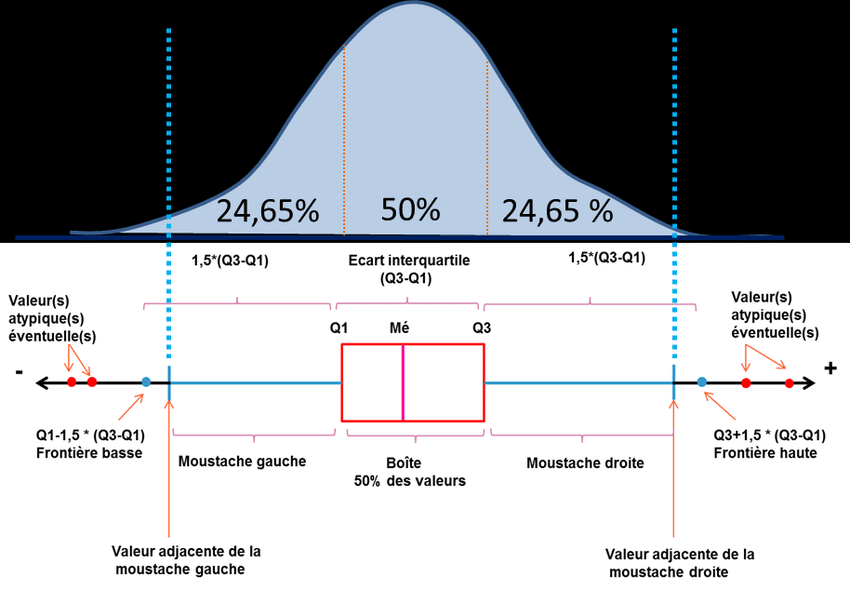
*La méthode de Parzen est une généralisation de la méthode d'estimation par histogramme. Dans un histogramme, la densité en un point x est estimée par la proportion d'observations x1, x2, ..., xN qui se trouvent à proximité de x. Pour cela, on trace une boîte en x et dont la largeur est gouvernée par un paramètre de lissage h ; on compte ensuite le nombre d'observations qui appartiennent à cette boîte. Cette estimation, qui dépend du paramètre de lissage h, présente de bonnes propriétés statistiques mais est par construction non-continue.*

*La méthode du noyau consiste à retrouver la continuité : pour cela, on remplace la boîte centrée en x et de largeur h par une gaussienne centrée en x. Plus une observation est proche du point de support x plus la courbe en cloche lui donnera une valeur numérique importante. À l'inverse, les observations trop éloignées de x se voient affecter une valeur numérique négligeable. L'estimateur est formé par la somme (ou plutôt la moyenne) des courbes en cloche. Comme indiqué sur l'image suivante, il est clairement continu.*

**Les diagrammes en boîte (ou boîtes à moustache)**

L’intérêt particulier de ces graphes est le nombre d’information qu’il contient par rapport à la répartition de valeurs de la ou les variables affichée(s).

La boîte en elle-même contient 50 % de l’effectif de la variable (cf schéma du powerpoint).



On remarque qu’**avec Pandas et Seaborn**, sans mettre d’argument dans les parenthèses il est possible d’obtenir des boîtes à moustache de toutes les variables numériques de notre dataframe d’un seul coup, ce qui peut être très intéressant.

Par contre **Seaborn** permet, elle, de visualiser la répartition d’une variable numérique en fonction d’une ou deux variables catégorique(s), ce qui nous apporte des informations nouvelles que les deux autres librairies ne nous fournissent pas.

**Les courbes**

**Avec la méthode *df.plot.line(),*** Pandas est la seule librairie à proposer de visualiser toutes les variables numériques selon des courbes indépendantes sur le même axe. Nous pouvons ainsi comparer leurs variations au fur et à mesure de leur enregistrement.

Lorsque l’on met 1 variable dans les parenthèses de la méthode de Pandas, on visualise toutes les autres variables du dataset en fonction de cette dite variable.

**Lorsque l’on compare les plots de mise en relation de deux variables numériques** avec les méthodes linéaires des trois librairies qui nous concernent, nous remarquons aisément que **Seaborn** se distingue des deux autres. En effet, alors que Matplotlib et Pandas relient par une courbe les différentes valeurs suivant leur ordre d’apparition dans le dataframe, seul Seaborn traite automatiquement les données et les ordonne selon des valeurs croissantes afin de faire apparaître une corrélation entre les deux variables visualisées si elle existe.

**Les points reliés**

Cette méthode n’est disponible qu’avec la librairie **Seaborn**. Elle permet de visualiser une variable numérique ou de mettre en relation une variable numérique avec 1 à 2 autres variables, quelles soient numériques ou catégoriques.

L’intérêt particulier de cette méthode réside dans le fait que les points représentent, par défaut, **la moyenne** des valeurs de la variable numérique, et les barres superposées sur ses points représentent l’intervalle de confiance.

Toutefois nous pouvons modifier cette méthode en lui demandant non pas la moyenne mais la médiane par ex. en appliquant *sns.pointplot(…..)****.estimate=median***, après avoir importé la **librairie Numpy**.

**Les nuages de points**

Les méthodes *scatter* des librairies **Matplotlib et Pandas** prennent 2 variables numériques comme argument.

A plot identiques entre les 3 librairies (avec 2 arguments) on obtient le même résultat, un peu plus esthétique tout de même avec Seaborn.

Mais la plus grande différence avec la **librairie Seaborn** c’est qu’elle permet de visualiser une variable numérique seule, permettant de voir sa distribution. Et avec les arguments *hue, size et style*, il est possible d’ajouter jusqu’à trois variables de plus, qu’elles soient numériques ou catégoriques.

Nous pouvons également visualiser des nuages de points avec ***sns.stripplot***, mais lorsque les données d’une variable sont trop nombreuses elles deviennent difficilement lisibles.

Il en est de même avec la méthode ***sns.swarmplot****.* C’est pourquoi ces deux dernières méthodes sont à privilégier pour visualiser nos variables seules, pour voir leur distribution.

Notons tout de même la supériorité du modèle fourni par ***swarm*** puisqu’il permet de voir clairement la quantité, le nombre de même valeurs dans la répartition de la variable.

**Les hexbins**

Ces graphes sont à interpréter à partir du principe de densité de population. En effet, à travers l’intensité colorimétrique on peut voir que plus la couleur est foncée plus le nombre de données, de valeurs est important en ces points.

***sns.relplot()***

Pour rester encore un peu avec Seaborn, voici une dernière méthode permettant de visualiser une variable numérique seule, la relation entre deux variables numériques, ou encore les relations entre 2 variables numériques et 1 à 3 autres variables numériques ou catégoriques.

Le style par défaut est kind=’scatter’ mais nous pouvons aussi lui attribuer le style ‘line’.

**2. Comment présenter des variables catégoriques ?**

Les variables catégoriques peuvent être visualisées à travers des **diagrammes en secteurs** ou des **barres.**

**Les diagrammes en secteur**

Les camemberts permettent de visualiser la distribution d’une variable catégoriques en pourcentage. La totalité des enregistrements de notre variable (la colonne en entier) est donc égale à 100 % et il va falloir faire appel à la méthode pandas ***df[‘variable’].value\_counts()*** pour regrouper les valeurs identiques de notre variable. Nous obtenons alors les noms de nos variables en index et les effectifs (counts) en values.

A partir de là nous pouvons plotter la méthode ***plt.pie*** avec Matplotlib ou ***plot.pie*** avec Pandas.

Cf code.

Pour créer un donut avec Matplotlib il suffit de créer un cercle blanc au centre de la pie.

Cf code.

**Les barres**

Pour produire des diagrammes en barres visualisant des variables catégoriques avec Matplotlib et Pandas il faut faire comme précédemment avec le camembert et préparer ces données avec la méthode pandas ***df[‘variable’].value\_counts().***

Seaborn propose elle une méthode faisant le calcul automatiquement grâce à la méthode ***sns.countplot(x, hue)***. Notons par ailleurs que nous pouvons même visualiser deux variables catégoriques à la fois, sur le même axe.

**3. Comment présenter des variables numériques et catégoriques ?**

Pour visualiser des variables numériques et catégoriques à la fois afin d’analyser leurs relations entre elles, nous pouvons créer des **barres**, des **diagrammes en boîtes**, des **nuages de points** et quelques autres dérivés.

**Les barres**

Comme précédemment pour Matplotlib et Pandas, nous faisons appel aux méthodes ***plt.bar()*** ou ***plot.bar*** ou ***barh*** pour avoir les barres de manière horizontale, en mettant cette fois 1 variable catégorique et 1 variable numérique (et plus seulement une seule variable catégorique).

Avec Seaborn, nous pouvons ici faire appel à la méthode ***sns.barplot().***Elle prend pour argument 1 variable numérique et 1 variable catégorique minimum. Par la suite il est possible d’ajouter jusqu’à deux autres variables catégoriques ou bien numériques.

**Les diagrammes en boîte et variantes**

Seul **Seaborn** permet de visualiser des variables catégoriques associées à une variable numérique du dataframe sous forme de **boîtes à moustache.**

Pour cela il faut toujours utiliser la méthode ***sns.boxplot(x,y,data)***. L'une des deux variables **x** ou **y** doit obligatoirement être numérique et l'autre catégorique. Il est ensuite possible d'ajouter une seconde variable catégorique avec ***hue***.

Deux variantes de la boîte à moustache classique s'exécutent avec les méthodes ***sns.boxenplot(x,y,data=df)*** et ***sns.violinplot(x,y,data=df)***. Il est toujours possible de rajouter une variable catégorique avec *hue*.

**Les nuages de points**

Pour visualiser des nuages de points mettant en relation une une ou deux variables numériques avec une à trois variables catégoriques nous appelons donc les méthodes ***sns.stripplot()*** et ***sns.swarplot()*** de la librairie Seaborn, comme nous l’avons vu auparavant.

**Conclusion** :

Nous voyons bien que **Seaborn** est la seule librairie qui offre des méthodes capables de *plotter* en une seule fois, et surtout une seule ligne de code, des variables à la fois numériques et catégoriques dans le même graphe. De plus, elle propose pour la plupart de ses méthodes d’intégrer jusqu’à 5 variables, même s’il faut se poser la question de la pertinence et de la lisibilité de nos visualisations dans ces cas-là.

Il faut savoir également que Seaborn propose la méthode sns.catplot() comme méthode généraliste, qui permet en son sein de déterminer directement le style de plot désiré avec l’attribut ***kind****.*

**kind = ‘strip (par défaut), swarm, box, boxen, violin, point, bar, ou count.**