

Lorand julien

Master EnviTerr

Compte rendus Analyse de données

Séance 2

Lors de la séance 2, on a vraiment fait nos premiers pas dans la manipulation de données avec Python. L'idée était d'apprendre comment organiser des informations, comment y accéder facilement, et comment réaliser nos premiers calculs et graphiques. En résumé, c'était une séance pour comprendre comment Python peut nous aider à travailler avec des données sans se compliquer la vie.

On a commencé par découvrir les différentes manières de stocker des données : des listes, des tuples, mais aussi les tableaux Numpy. Même si ces structures ressemblent toutes à des collections de valeurs, elles n'ont pas le même fonctionnement. Par exemple, une liste peut être modifiée alors qu'un tuple ne peut pas l'être. Les tableaux Numpy, eux, sont faits spécialement pour le calcul scientifique : ils permettent d'effectuer des opérations très rapidement. C'est important de comprendre ces différences, car le choix de la structure change la façon dont on travaille ensuite.

Une fois qu'on savait comment ranger l'information, on a appris à aller chercher ce qu'on voulait dedans. Grâce aux indices en Python, on peut accéder directement à une valeur précise ou sélectionner toute une partie d'un tableau. Ce n'est pas seulement pratique, c'est indispensable : c'est comme savoir "où regarder" dans ses données. Quand on devra analyser ou nettoyer un jeu de données plus complexe, cette compétence sera incontournable.

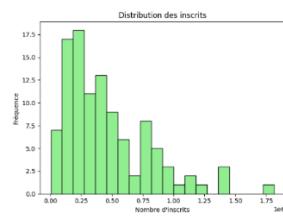
Ensuite, on a exploré comment Python pouvait nous aider à faire des calculs automatiquement. Avec Numpy, on a vu qu'on pouvait calculer une moyenne, une variance, ou même multiplier toutes les valeurs d'un tableau en une seule instruction. Là où on ferait des dizaines de calculs à la main, Python fait tout à notre place en quelques millisecondes. C'est là qu'on réalise la vraie puissance du langage.

On a aussi appris à vérifier la structure des données : combien il y a d'éléments, sous quelle forme ils sont présentés, et de quel type ils sont. Cette étape paraît simple, mais elle est fondamentale : souvent, quand un programme "ne marche pas", c'est parce qu'on n'a pas compris la forme réelle des données. En vérifiant la taille, les dimensions ou les types, on évite beaucoup d'erreurs.

Pour terminer, on a abordé la visualisation. C'était notre premier contact avec Matplotlib, et on a vu qu'en quelques lignes, on pouvait transformer des chiffres en graphiques. On a tracé des courbes, des histogrammes, des diagrammes en barres... Et rien que ça permet déjà de voir les données autrement. Une distribution devient visible, une tendance devient évidente, une anomalie saute aux yeux. C'est exactement ce qu'on cherche quand on commence à analyser des données.

Au final, cette séance 2 pose les bases de tout ce qu'on fera ensuite. On comprend comment sont organisées les données, comment y accéder, comment les transformer et comment les représenter visuellement. C'est une séance fondamentale, car elle donne les réflexes et les outils dont on aura besoin pour toutes les étapes suivantes : explorer, nettoyer, analyser et modéliser les données.

Question n°	Explication	Résultat / Utilité + Exemple de code
Question n°1	Comprendre comment stocker des données en Python (listes, tuples, tableaux Numpy).	Permet de manipuler efficacement des données numériques ou textuelles. Exemples : <code>liste = [1, 2, 3] tuple1 = (4, 5, 6) arr = np.array([1, 2, 3])</code>
Question n°2	Sert à accéder à des éléments dans une structure de données.	Sert à sélectionner, modifier ou extraire des valeurs précises. Exemples : <code>liste[0] arr[1:3]</code>
Question n°3	Sert à effectuer des opérations mathématiques simples.	Permet de faire des calculs automatiques sur les vecteurs. Exemples : <code>np.mean(arr) np.std(arr) arr * 2</code>
Question n°4	Comprendre les dimensions, tailles et formes des tableaux.	Indispensable pour bien gérer les manipulations de données et éviter les erreurs. Exemples : <code>arr.shape arr.size</code>
Question n°5	Visualiser rapidement les données avec des graphiques simples.	Aide à repérer tendances, distribution ou anomalies. Exemples de code : Ligne simple : <code>plt.plot(arr) plt.savefig("ligne.png")</code> Histogramme : <code>plt.hist(arr) plt.savefig("hist.png")</code>



Séance 3 : Les paramètres statistiques élémentaires

La séance 3 nous a plongés dans une étape essentielle de tout travail d'analyse : explorer et nettoyer les données. C'est un moment important, car avant de faire des statistiques ou des graphiques avancés, il faut d'abord s'assurer que les données sont fiables, cohérentes et bien structurées. En quelque sorte, cette séance nous apprend à "mettre de l'ordre" avant de commencer à analyser.

On a commencé par regarder comment un tableau de données est organisé. On apprend à examiner un jeu de données dès son chargement : quelles sont les colonnes, quels types de données elles contiennent, s'il manque des valeurs, ou si certains éléments semblent incohérents. C'est un peu comme se familiariser avec un nouveau dossier : on regarde ce qu'il contient et si quelque chose semble anormal. Les commandes comme `head()`, `info()` ou `describe()` deviennent vite des réflexes, parce qu'elles donnent un premier aperçu de la qualité des données.

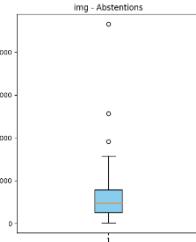
Une grande partie de la séance a été consacrée aux valeurs manquantes. Elles sont très fréquentes dans les jeux de données réels, et il est indispensable de savoir les repérer et les gérer. On apprend que certaines valeurs manquantes peuvent être supprimées, tandis que d'autres doivent être remplacées, selon leur importance et la nature de l'analyse. C'est une vraie compétence : trop supprimer peut fausser l'analyse, trop remplacer peut la rendre moins fiable. L'idée est de trouver un juste équilibre en fonction du contexte.

Ensuite, on a travaillé sur la normalisation et l'harmonisation des données. Ce sont de petites transformations, mais elles ont un impact énorme : mettre du texte en minuscules, supprimer des espaces, convertir des nombres en entiers, arrondir des valeurs trop précises, etc. Ces opérations ressemblent à du "ménage", mais sont essentielles pour éviter des erreurs. Si une colonne mélange les chaînes de caractères et les nombres ou contient des formats incohérents, tout le reste de l'analyse peut tomber à l'eau. La séance nous a montré que de petites incohérences peuvent provoquer de grandes frustrations.

Une fois les données nettoyées, on a pu commencer à construire les premières représentations graphiques exploratoires. C'est presque la partie la plus satisfaisante, car on voit immédiatement ce que les données "racontent". Les histogrammes permettent de comprendre la distribution d'une variable, les boxplots mettent en évidence les valeurs extrêmes, et les diagrammes en barres montrent la répartition des catégories. Ces graphiques simples jouent un rôle clé : ils nous permettent de repérer d'un coup d'œil des tendances, des anomalies ou des structures invisibles dans les valeurs brutes.

Dans l'ensemble, cette séance a montré qu'avant de faire de l'analyse statistique ou des modèles avancés, il est indispensable de bien comprendre, préparer et nettoyer les données. C'est souvent la partie la plus longue dans un projet Data, mais aussi celle qui garantit la fiabilité des résultats. À partir de cette séance, on comprend que la qualité d'une analyse dépend directement de la qualité du nettoyage qui précède.

Retour sur l'utilité du codage pour cette séance.

Question n°	Explication	Résultat / Utilité + Exemple de code + Graphique
Q°1	Identifier le type de variable (qualitative, quantitative, discrète, continue).	Permet de choisir les bonnes méthodes d'analyse et les bons graphiques.
Q°2	Regrouper et structurer les valeurs d'une variable.	Permet de créer un vecteur propre et manipulable. Exemple : <code>x = np.arange(0, len(df))</code>
Q°3	Transformer des données brutes en valeurs exploitable.	Génère des valeurs utilisables pour l'analyse (nettoyées, converties, harmonisées). Exemples : <code>df["age"] = df["age"].astype(int)</code> <code>df["revenu"] = df["revenu"].fillna(0)</code>
Q°4	Harmoniser les données pour éviter les erreurs et les incohérences.	Uniformise le format des données : minuscule, suppression des espaces, arrondi, conversion, etc. Exemples : <code>df["nom"] = df["nom"].str.lower()</code> <code>df["revenu"] = np.round(df["revenu"], 2)</code>
Q°5	Préparer les données pour la visualisation ou l'analyse statistique	Génère des graphiques pour explorer les données. 

Séance 04

La séance 4 marque une étape importante dans le cours, car elle nous fait entrer dans le monde des distributions statistiques. Cette séance nous apprend à reconnaître, comprendre et représenter plusieurs types de lois de probabilité, ce qui est essentiel pour analyser des phénomènes aléatoires. L'idée est d'apprendre à "lire" la forme d'une distribution, ce qu'elle raconte, et comment la représenter correctement grâce au code Python.

Nous avons d'abord découvert les distributions discrètes, c'est-à-dire celles qui prennent des valeurs séparées, généralement des entiers. Parmi elles, on retrouve la loi de Dirac, qui représente un événement certain concentré sur une seule valeur, ou encore la loi uniforme discrète qui attribue la même probabilité à chaque valeur possible. Une place importante a aussi été donnée à la loi binomiale, qui modélise le nombre de réussites dans une série d'essais, ainsi qu'à la loi de Poisson, très utilisée pour représenter des événements rares dans un intervalle donné. Enfin, la séance a abordé la loi de Zipf ou Zipf–Mandelbrot, une loi particulière qui apparaît dans de nombreux phénomènes naturels comme la fréquence des mots dans une langue ou la popularité des villes. Toutes ces distributions ont été illustrées graphiquement afin de mieux visualiser leur forme et leur comportement.

La séance s'est également concentrée sur plusieurs distributions continues, celles dont les valeurs possibles remplissent un intervalle entier. Nous avons notamment vu la loi normale, probablement la plus célèbre, qui apparaît dans une multitude de phénomènes naturels et humains. La loi log-normale, qui représente des valeurs positives très dispersées, la loi uniforme continue, où tout intervalle a la même probabilité, et la loi du chi-deux, souvent utilisée pour mesurer des variations ou tester des hypothèses statistiques, ont aussi été étudiées. Enfin, nous avons exploré la loi de Pareto, connue pour décrire des situations où une minorité d'éléments représente une grande part d'un phénomène (comme la règle des 80/20).

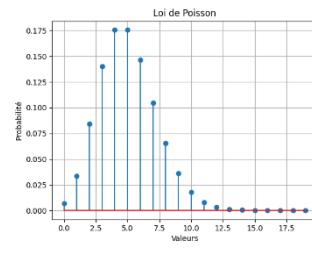
Pour chaque loi, nous avons appris à produire son graphique à l'aide de Python et de la bibliothèque SciPy, en construisant soit une fonction de masse (pour les lois discrètes), soit une densité de probabilité (pour les lois continues). Le travail a consisté à choisir un ensemble de valeurs pertinentes, à calculer les probabilités ou les densités correspondantes, puis à construire une représentation visuelle claire et lisible. Cette étape nous permet de mieux comprendre comment une distribution se comporte, où se situent les valeurs les plus probables, et comment elle peut être utilisée dans une analyse réelle.

Ce qui ressort de cette séance, c'est que chaque distribution a une forme qui lui est propre, et cette forme traduit un comportement particulier de la variable étudiée. Savoir reconnaître ces lois et savoir les représenter graphiquement est un élément fondamental pour analyser des données, modéliser des phénomènes ou même faire des prédictions.

En conclusion, la séance 4 nous a apporté une vision structurée des principales distributions de probabilité, discrètes et continues, ainsi que les outils pour les illustrer grâce au code. Elle constitue une base indispensable pour comprendre le fonctionnement des données aléatoires et prépare le terrain pour des analyses statistiques plus avancées.

Retour sur l'utilité du codage pour cette séance.

Question n°	Explication	Résultat / Utilité + Exemple de code
Q°1	Identifier si la variable est discrète ou continue pour choisir la bonne distribution.	Permet d'appeler la bonne fonction. Exemple: <code>pmf = binom(n, p).pmf(x)</code>
Q°2	Regrouper et structurer les valeurs d'une variable.	Permet de créer un vecteur adapté. Exemple: <code>x = np.arange(0, n + 1)</code>
Q°3	Transformer une réalité en valeurs exploitable.	Génère les probabilités. Exemple: <code>pmf = poisson(l).pmf(x)</code>
Q°4	Harmoniser les données pour éviter les erreurs.	Normalise ou arrondit. Exemple: <code>pdf = poisson(l).pmf(np.round(x))</code>
Q°5	Préparer les données pour la visualisation ou analyse.	Sauvegarde un graphique. Exemple: <code>save_discrete(x, pmf, "Loi binomiale", "binomiale.png")</code>



Seance 5

La statistique inférentielle cherche à comprendre une population à partir d'un échantillon. L'échantillonnage consiste donc à sélectionner une partie de la population mère afin d'en tirer des informations générales. Il est rarement possible d'interroger l'ensemble des individus, soit parce que la population est trop vaste, soit par manque de temps ou de moyens. On a alors recours à différentes méthodes d'échantillonnage, comme le tirage aléatoire simple, l'échantillonnage stratifié ou systématique. Le choix de la méthode dépend de la taille de la population, de sa structure et des objectifs de l'étude.

Un estimateur est un outil statistique calculé à partir de l'échantillon, dont le but est d'approcher un paramètre inconnu de la population. L'estimation correspond à la valeur obtenue grâce à cet estimateur. Par exemple, la moyenne calculée sur un échantillon permet d'estimer la moyenne réelle de la population. Comme on ne travaille que sur une partie des données, toute estimation comporte une part d'incertitude.

Il est important de distinguer l'intervalle de fluctuation de l'intervalle de confiance. L'intervalle de fluctuation décrit les variations possibles des résultats lorsque l'on répète des tirages d'échantillons à partir d'une population connue. L'intervalle de confiance, en revanche, s'applique lorsque la population est inconnue : il permet d'encadrer une estimation et d'indiquer dans quelle mesure on peut lui faire confiance. Les deux notions répondent donc à des logiques différentes, mais complémentaires.

En statistique, on parle de biais lorsqu'un estimateur produit de manière systématique des valeurs trop élevées ou trop faibles par rapport au paramètre réel. Ces biais peuvent provenir d'un échantillon mal choisi, d'une méthode d'estimation inadaptée ou de données imparfaites. Limiter les biais est un enjeu essentiel pour garantir la fiabilité des résultats.

Lorsqu'on travaille sur la population entière, on ne fait plus de statistique inférentielle mais de la statistique descriptive exhaustive. Cette approche est aujourd'hui souvent associée aux données massives, ou big data, qui permettent d'analyser des volumes de données très importants. Toutefois, disposer de beaucoup de données ne signifie pas automatiquement produire de bons résultats : la qualité et la pertinence des données restent fondamentales.

Le choix d'un estimateur est déterminant, car il influence directement la précision et la stabilité des résultats. Un bon estimateur doit être aussi précis que possible, peu sensible aux variations d'échantillon et cohérent avec les hypothèses statistiques retenues. Plusieurs méthodes d'estimation existent, comme la méthode des moments ou celle du maximum de vraisemblance, et leur sélection dépend du type de données et du cadre théorique de l'analyse.

Les tests statistiques servent à aider à la prise de décision. Ils permettent de vérifier si un résultat observé peut être attribué au hasard ou s'il traduit un effet réel. Mettre en place un test suppose de formuler une hypothèse nulle, de choisir une statistique de test, une loi de référence et un seuil de décision. Le type de test dépend de la nature des variables étudiées et des hypothèses sur leur distribution.

Enfin, la statistique inférentielle fait régulièrement l'objet de critiques. On lui reproche notamment une dépendance forte aux hypothèses, un usage parfois mécanique des tests et une mauvaise interprétation des résultats. Ces critiques rappellent que les outils statistiques doivent être utilisés avec prudence, esprit critique et en tenant compte du contexte de recherche, en particulier en géographie et en sciences humaines.

Question code : séance 5

Question de cours / Objectif	Ce que fait le code	Lignes de code clés (exemples)
1. Charger un fichier de données	Ouvre un fichier CSV et le transforme en DataFrame Pandas	<code>pd.read_csv() ouvrirUnFichier("./data/Echantillonage-100-Echantillons.csv")</code>
2. Calculer les moyennes d'échantillons	Calcule la moyenne de chaque colonne (opinion) sur 100 échantillons	<code>donnees[colonne].mean() round(moyenne, 0)</code>
3. Calculer les fréquences d'un échantillon	Transforme des effectifs moyens en proportions	<code>frequence = moyenne / somme_moyennes round(frequence, 2)</code>
4. Calculer les fréquences de la population mère	Passe des effectifs totaux aux fréquences réelles	<code>effectif / total_population</code>
5. Comparer échantillon et population	Mesure l'écart entre fréquences estimées et réelles	<code>abs(freq_ech - freq_pop)</code>

6. Calculer un intervalle de fluctuation	Encadre la fréquence théorique à partir de la population mère	<pre>math.sqrt((p*(1-p))/n) borne_inf = p - z*écart_type borne_sup = p + z*écart_type</pre>
7. Extraire un échantillon individuel	Sélectionne une ligne précise du DataFrame	<code>donnees.iloc[0]</code>
8. Convertir un objet Pandas en liste	Facilite les calculs classiques en Python	<code>list(premier_echantillon_pandas)</code>
9. Calculer les fréquences observées	Calcule les proportions dans un seul échantillon	<code>premier_echantillon[i] / n_échantillon</code>
10. Calculer un intervalle de confiance	Encadre une estimation issue d'un échantillon	$p_{\text{hat}} \pm z * \sqrt{(p_{\text{hat}} * (1 - p_{\text{hat}})) / n}$
11. Vérifier si la valeur réelle est dans l'IC	Teste la cohérence de l'estimation	<code>borne_inf <= freq_pop <= borne_sup</code>
12. Charger des données pour un test statistique	Importe des distributions à tester	<code>pd.read_csv()</code>
13. Tester la normalité (Shapiro-Wilk)	Vérifie si une distribution suit une loi normale	<code>scipy.stats.shapiro(data)</code>
14. Interpréter la p-value	Décide si la normalité est rejetée ou non	<code>if p_value < 0.05:</code>

Seance 6

La statistique d'ordre, aussi appelée statistique ordinaire, concerne l'étude des variables qualitatives hiérarchisées. Elle consiste à classer des objets, des individus ou des territoires selon un rang, sans supposer que les écarts entre ces rangs soient mesurables ou constants. Elle s'oppose à la statistique nominale, qui traite des catégories sans ordre interne. En géographie, la statistique d'ordre occupe une place essentielle, car de nombreux phénomènes spatiaux s'expriment sous forme de classements : hiérarchies urbaines, rangs des États selon la population ou la densité, attractivité des territoires. Ces classements permettent de matérialiser des hiérarchies spatiales, révélatrices de rapports de domination, de concentration ou de marginalisation dans l'espace.

Dans les statistiques d'ordre, l'ordre croissant, qualifié d'ordre naturel, est généralement privilégié. Il facilite l'analyse des distributions, l'identification des valeurs extrêmes et l'étude des lois statistiques associées. Toutefois, certaines approches géographiques, comme la loi rang-taille, utilisent un ordre décroissant plus pertinent pour mettre en évidence les phénomènes de concentration. L'essentiel est que l'ordre retenu soit cohérent, explicite et constant, car toute comparaison repose sur la stabilité des rangs.

Comparer des classements implique de distinguer la corrélation des rangs et la concordance de classements. La corrélation des rangs mesure le degré de dépendance entre deux hiérarchies, en évaluant si les objets bien classés selon un critère le sont également selon un autre. La concordance, quant à elle, repose sur l'analyse des paires concordantes et discordantes et permet d'évaluer le niveau d'accord global entre deux classements. Ces deux approches sont complémentaires : l'une mesure l'intensité de la relation, l'autre la cohérence de l'ordre.

Le test de Spearman est un test non paramétrique qui applique le principe de la corrélation aux rangs. Il est particulièrement adapté lorsque les classements sont complets et sans ex æquo. Le test de Kendall, fondé sur le comptage des paires concordantes et discordantes, est plus robuste et se généralise facilement à plusieurs classements. Tous deux permettent de tester l'hypothèse d'indépendance entre variables ordinaires et sont invariants par transformation monotone.

Enfin, les coefficients de Goodman-Kruskal et de Yule servent à mesurer l'association entre variables qualitatives ordinaires. Le coefficient Γ de Goodman-Kruskal exprime le surplus de concordance entre deux classements, tandis que le coefficient Q de Yule constitue un cas particulier applicable aux tableaux de contingence 2×2 . Ces outils sont particulièrement utiles lorsque les données ne satisfont pas les hypothèses de la statistique paramétrique et sont fréquemment mobilisés en géographie pour analyser les relations entre catégories hiérarchisées.

Question codage : seance-06

Question de cours / Objectif	Ce que fait le code	Lignes de code clés (exemples)
1. Charger un fichier de données	Ouvre un fichier CSV et le transforme en DataFrame Pandas	<code>pd.read_csv() ouvrirUnFichier("./data/echantillon nage-100-echantillons.csv")</code>
2. Calculer les moyennes d'échantillons	Calcule la moyenne de chaque colonne (opinion) sur 100 échantillons	<code>moyenne = colonne.mean()round(moyenne, 0)</code>
3. Calculer les fréquences d'un échantillon	Transforme des effectifs moyens en proportions	<code>frequence = moyenne / somme_moyennes round(frequence, 2)</code>
4. Calculer les fréquences de population mère	Passe des effectifs totaux aux fréquences réelles	<code>frequence = effectif / total_population</code>
5. Comparer échantillon et population	Mesure l'écart entre fréquences estimées et fréquences réelles	<code>abs(freq_ech - freq_pop)</code>
6. Calculer un intervalle de fluctuation	Encadre la fréquence théorique à partir de la population mère	<code>math.sqrt((p*(1-p))/n) borne_inf = p - 1.96*ecart_type borne_sup = p + 1.96*ecart_type</code>
7. Extraire un échantillon individuel	Sélectionne une ligne précise du DataFrame	<code>donnees.iloc[0]</code>
8. Convertir un objet Pandas en liste	Facilite les calculs classiques en Python	<code>list(premier_echantillon_pandas)</code>
9. Calculer les fréquences observées	Calcule les proportions dans un seul échantillon	<code>premier_echantillon[i] / n_echantillon</code>

Réflexion personnelle sur les sciences des données et les humanités numériques :

Les exercices réalisés tout au long de ce parcours m'ont permis de mieux comprendre la place croissante des sciences des données dans les sciences humaines et sociales, et plus particulièrement dans le champ des humanités numériques. À travers la manipulation de jeux de données, la construction de classements, le calcul de fréquences et l'usage de tests statistiques, j'ai pris conscience que le numérique ne se limite pas à un simple outil technique, mais qu'il structure profondément la manière de produire, d'analyser et d'interpréter les connaissances. Même si la partie de codage reste assez compliquer à mettre en place.

Sur le plan de l'apprentissage, la principale difficulté réside dans la double exigence des humanités numériques : il ne suffit pas de maîtriser les concepts des sciences humaines et de savoir coder, mais il faut être capable de faire dialoguer les deux. Les exercices montrent bien que le traitement quantitatif des données ne remplace pas l'interprétation, mais qu'il la prépare et l'encadre. Les résultats statistiques (coefficients de Spearman ou de Kendall, par exemple) n'ont de sens que s'ils sont replacés dans une réflexion potentiellement plus pousser.

Enfin, ce parcours met en évidence l'un des enjeux majeurs des humanités numériques : éviter une approche purement techniciste ou utilitariste du numérique. Les outils de calcul et de visualisation facilitent l'analyse et renforcent la rigueur méthodologique, mais ils comportent aussi le risque de privilégier les résultats chiffrés au détriment de l'esprit critique. Les sciences des données apparaissent ainsi comme un levier puissant, à condition qu'elles soient intégrées dans une démarche réflexive, critique et interprétative, fidèle aux principes fondamentaux des sciences humaines et sociales.