EIENIV9
Information Visualisation

UNIVERSITÉ CÔTE D'AZUR

RAPPORT SUR LE PROJET :

AirBnB Information Visualisation

Groupe 6
Adam MIR-SADJADI
Florent BELOT
Benoît BARBIER
Massinissa ABBOUD

Site : https://airbnb-information-visualisation-g6-2025.onrender.com

Github : https://github.com/ADMR-S/AirBnB_Information_Visualisation_G6_2025

Année universitaire 2025 - 2026

TABLE DES MATIERES

Présentation du projet

Dataset

Le dataset utilisé pour notre projet représente les offres AirBnB aux Etats-Unis.
Il a été mis en ligne par l'utilisateur kritikseth et est disponible sur Kaggle à l'adresse suivante : https://www.kaggle.com/datasets/kritikseth/us-airbnb-open-data.

AirBnB regroupe des hôtes et des voyageurs sur une même application et permet aux hôtes de proposer leur logement à la location en indiquant une description, ses dates de disponibilité et un prix. Les utilisateurs peuvent laisser un avis après leur passage afin de renseigner les futurs intéressés sur la qualité des logements. Nous n'avons pas accès au contenu des avis utilisateurs dans ce dataset, seulement leur nombre.
Les données sont disponibles pour 2020 et 2023. Le dataset est de type table avec des données statiques (Les données extraites directement de la base AirBnB seraient normalement dynamiques mais en l'occurrence le dataset est statique).
Les types de données sont mixte, dont certaines peuvent être ordonnées quantitativement (price, number_of_reviews, availability_365…), les autres étant des données catégoriques non ordonnables (name, room_type, neighbourhood, id…).

Chaque entrée de la table représente un logement mis en location sur la plateforme et contient des coordonnées spatiales (latitude / longitude).

Profils & Tâches utilisateurs

Naturellement, nous pouvons identifier deux types d'utilisateurs sur la plateforme : Les hôtes et les voyageurs.
Pour chacun de ces profils, nous avons déterminé 5 tâches que l'utilisateur pourrait souhaiter réaliser à partir de ces données, et y avons associé un ou plusieurs types de visualisations.
Ainsi, un voyageur cherchera à :

- Trouver les offres les moins chères → Violin Chart
- Localiser les offres → Bubble Map
- Comparer la popularité (nombre de reviews) → Parallel coordinates chart
- Détecter les bons rapports prix/nb_reviews → Parallel coordinates chart
- Explorer la disponibilité annuelle → Treemap

Tandis qu'un hôte voudra :

- Effectuer un benchmark des prix par quartier/type de logement → Violin chart
- Suivre la densité concurrentielle → Bubble Map
- Comprendre la structure du marché → Violin Chart, Treemap
- Voir ses propres logements et les offres alentours sur la carte → BubbleMap
- Comparer les performances de ses logements → Parallel coordinates chart, violin chart

Choix techniques & Pipeline de visualisation

Nous utilisons la bibliothèque d3js pour réaliser ces visualisations. Afin de leur permettre d'interagir aisément, nous avons préconisé l'utilisation du framework web React plutôt que de se contenter d'un développement JS pur ou TypeScript. Le traitement des données a été effectué et documenté par un notebook Jupyter contenu dans le code source. Nous avons utilisé le gestionnaire de version git avec une branche de production dédiée. Le code source de l'application est disponible à cette adresse :
https://github.com/ADMR-S/AirBnB_Information_Visualisation_G6_2025.
L'application est hébergée sur Render à l'adresse suivante : https://airbnb-information-visualisation-g6-2025.onrender.com.

Le pipeline de visualisation de notre application peut être décrit ainsi (quelques variations selon le type de visualisation) :

1. Pré-traitement des données (code jupyter notebook)
2. Chargement : Récupération des données Airbnb depuis le csv
3. Filtrage : Application des filtres globaux
4. Filtrage secondaire (optionnel : Violin, Parallel coordinates chart, TreeMap)
5. Échantillonnage (Parallel coordinates Chart) / Agrégation : Réduction du nombre de lignes selon le mode choisi, clustering
6. Calcul des échelles / domaines des propriétés des variables visuelles
7. Mapping visuel / Projection (BubbleMap)
8. Rendu

Les interactions utilisateurs venant se positionner avant ou après les étapes qui suivent l'étape 3 selon le type de visualisation et l'interaction en question, re-déclenchant l'exécution du pipeline.

Traitement préalable des données

Pipeline de preprocessing

Le pipeline de traitement a consolidé et nettoyé deux années de données Airbnb américaines (2020 et 2023), représentant initialement 458 177 annonces à travers 27 villes américaines. L'ensemble du processus a été réalisé dans le notebook data_preparation/data_pipeline.ipynb en utilisant pandas et des bibliothèques géographiques spécialisées.

Acquisition et Harmonisation

Les fichiers CSV téléchargés via l'API kagglehub contenaient des schémas légèrement différents entre les deux années, nécessitant la suppression des colonnes incohérentes comme neighbourhood_group et number_of_reviews_ltm. Une colonne temporelle year a été ajoutée pour distinguer les millésimes avant leur fusion en un jeu unique de 17 colonnes. L'enrichissement géographique a mappé chaque ville vers son État correspondant, créant une hiérarchie géographique exploitable pour l'analyse.

Nettoyage et Validation

Le processus de déduplication s'est déroulé en deux phases : d'abord l'identification de 971 doublons sémantiques basés sur la combinaison hôte-localisation-prix-année, puis l'élimination de 2 doublons d'ID stricts. Une valeur aberrante extrême (minimum_nights = 100 000 000) a été filtrée en appliquant une plage raisonnable de 1 à 2 000 nuits. Les métriques d'hôte ont été recalculées par année pour refléter précisément le nombre d'annonces gérées dans chaque période.
Le géocodage a transformé 1 459 quartiers initiaux en 1 411 emplacements lisibles en traduisant les codes postaux numériques via la bibliothèque pgeocode. Le jeu final de 457 203 annonces sur 18 colonnes a été validé pour la cohérence des coordonnées géographiques, l'intégrité référentielle hôte-annonce, et les distributions statistiques des variables numériques clés. L'audit a identifié des valeurs manquantes dans les champs non critiques (name, host_name, last_review, reviews_per_month) avant l'export vers data_preparation/out/dataset.csv.

Voici les caractéristiques finales et les informations notables après traitement :

- Rows: 457,203
- Columns: 18
- Features loaded:
  id, name, host_id, host_name, neighbourhood, latitude, longitude, room_type, price, minimum_nights, number_of_reviews, last_review, reviews_per_month, calculated_host_listings_count, availability_365, city, year, state

Statistical Notes :

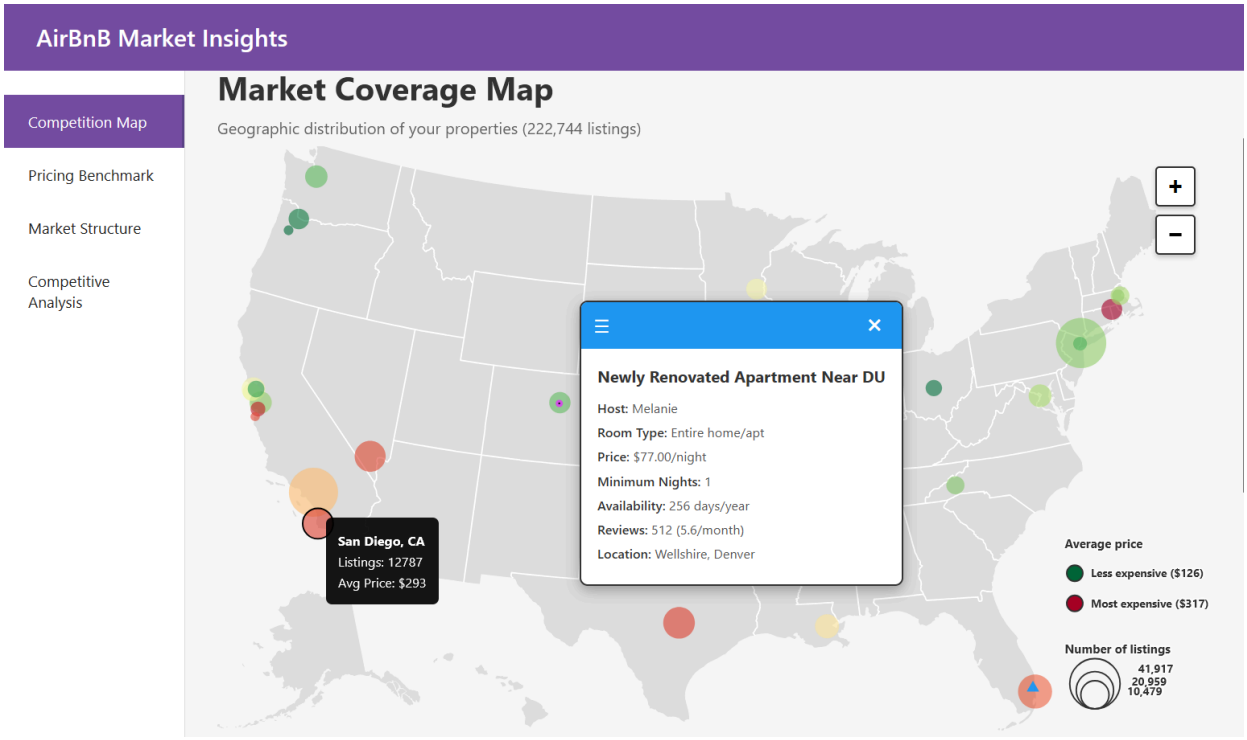
- Price: Range observed from 0 to 100,000$
- Minimum nights: max = 1250
- Reviews per month: max ≈ 101, potentially hotels like hosts.
- Availability: max 365, expected.

Types de visualisations

BubbleMap (Adam)

Les données étant concentrées aux USA, j'ai débuté mon travail sur la bubblemap en suivant le tutoriel de Mike Bostock, créateur de d3js, portant sur une bubblemap affichant la population aux USA et disponible à l'adresse suivante : https://bost.ocks.org/mike/bubble-map/.
En réalisant un makefile permettant de compiler les informations topologiques d'une carte téléchargée sur le site du Census Bureau des Etats-Unis à l'aide de topojson en utilisant la projection geo Albers Usa, j'ai pu obtenir une base de travail préprojetée (ce qui réduit les temps de calcul) au format json pour notre visualisation. Dans mon code, je m'inspire également de la logique d'affichage des bulles de population. Les coordonnées des listings de notre dataset sont projetées à l'exécution de la visualisation en suivant la même projection que celle de la carte d'origine (us.json).

Overview

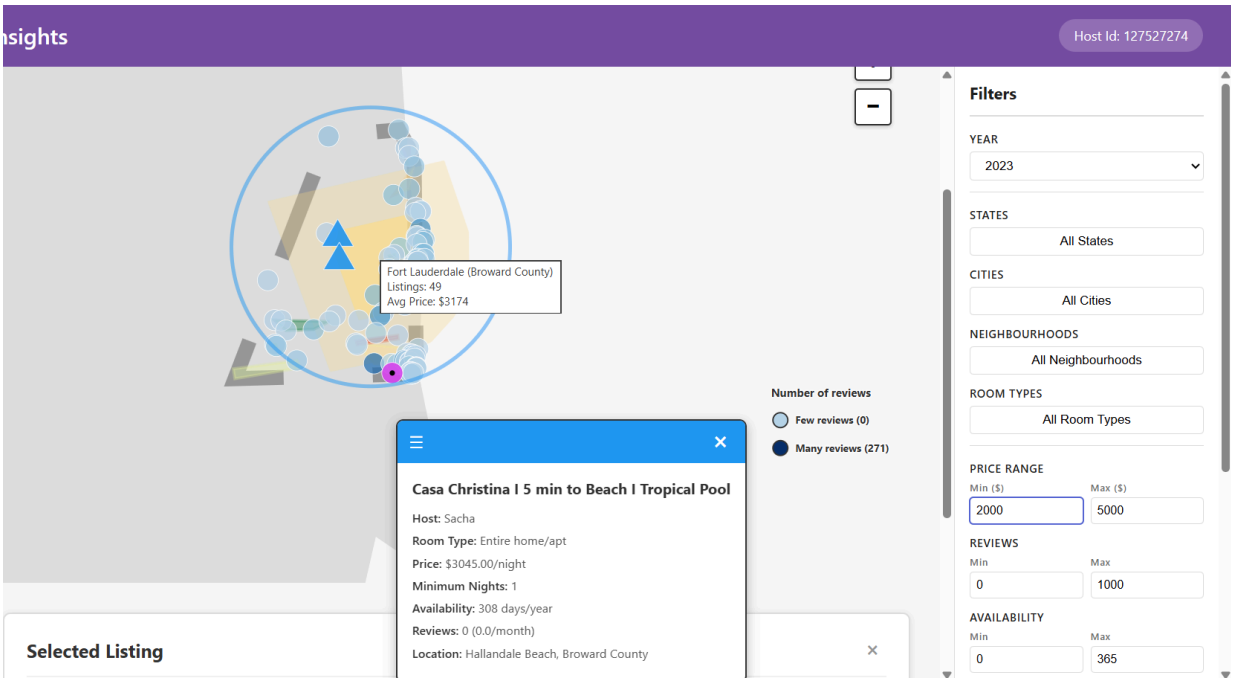


Le dataset dont nous disposons étant très dense, une agrégation des données par ville était nécessaire pour pouvoir réaliser l'overview de la carte. Cette vue permet aux voyageurs et aux hôtes de repérer rapidement les zones à forte ou faible densité de listings (taille du cercle) et les disparités de prix (échelle de couleur séquentielle du vert au rouge) compte tenu des filtres sélectionnés. Les hôtes peuvent également y voir afficher leurs propriétés représentées par des triangles bleus, et une offre sélectionnée dans une autre visualisation ou dans la vue de détail apparaîtra sous la forme d'un cercle violet comportant un point noir en son centre (ici accompagné d'un pop-up pour le detail on demand). Un tooltip affiche le nom de la ville, son nombre de listings et son prix moyen au survol d'une bulle. J'ai envisagé d'autres propriétés

pour les cercles afin de transmettre l'information autrement que par la couleur (avec des bordures variables ou en jouant sur l'opacité) mais la solution la plus simple a finalement semblé être visuellement la plus convenable si ce n'est en termes d'accessibilité (un travail approfondi pourrait être mené sur cet aspect).

Par la même occasion, nous effectuons un clustering par quartiers (donnée disponibles directement dans le dataset) qui nous permettra d'afficher des zones de prix dans la vue détaillée. Les "unincorporated areas" ont été ignorées pour éviter les aplats non significatifs trop larges, et un quartier est défini par au moins 2 listings dans le dataset filtré (on pourrait envisager de calculer une seule fois les informations de chaque quartier plutôt que de tenir compte des filtres mais ce procédé est intéressant également pour se concentrer sur une partie précise du dataset). A chaque filtrage des données, l'aggrégation a lieu à nouveau pour mettre à jour l'échelle de couleur et la taille des bulles de prix et de densité, ce qui pourrait être amélioré par un meilleur usage des structures de données impliquées afin d'alléger le traitement. Pour la quantité de données que nous possédons, cette implémentation s'est avérée suffisante mais pourrait souffrir de problèmes de scaling à l'avenir.
Cette vue reste active jusqu'à un niveau de magnification égal à 3x.

Détail



Au-delà de 3 niveaux de zoom, la vue change et donne le contrôle à l'utilisateur d'une loupe appliquant un fisheye permettant un focus + context qui permet de révéler les listings sur son passage en appliquant une distorsion légère pour faciliter l'isolement d'un listing précis. A cette échelle, les quartiers sont visibles sous forme de polygones même sans passage du fisheye et indiquent via une échelle de couleur (sur la même base que celle des bulles de ville) les disparités de prix moyen entre chaque quartier. Un tooltip affiche le détail précis pour le quartier

si l'utilisateur le survole longtemps (pour éviter de perturber les interactions avec les listings), et le quartier est mis en évidence par un agrandissement temporaire semi-opaque.

Les listings quant à eux possèdent une couleur attribuée par une échelle racine carrée (afin de marquer clairement la distinction entre 0 et 100 listings par exemple et contourner l'asymétrie des nombres de reviews qui contiennent peu de valeurs très élevées) afin de permettre aux voyageurs comme aux hôtes de rapidement repérer les listings attractifs. Un listing survolé est mis en évidence de la même manière qu'un quartier afin de donner du feedback à l'utilisateur quant au listing qu'il s'apprête à sélectionner, l'opération pouvant être fine en fonction de la densité de listing là où l'utilisateur le souhaite. La sélection d'un listing sur la carte déclenche sa mise en évidence (cercle violet avec centre noir) et l'ouverture d'un pop-up afin de pouvoir consulter le détail rapidement sans nécessité de naviguer jusqu'au panneau de détail situé sous la carte. Sans filtres, la densité de listing est extrêmement importante et a nécessité l'implémentation de plusieurs mesures pour éviter les problèmes de performance trop conséquents :

- Un throttle a été ajouté au mouvement de la souris, pour déclencher moins de mousemove events que sans et alléger la charge du processeur (throttle de 32ms)
- Les listings sont stockés dans des quadtrees pour permettre une recherche spatiale rapide. J'ai utilisé Claude Sonnet pour m'aider dans l'implémentation de ces derniers que je ne maîtrisais pas, et qui ont rendu cette vue viable.
- Un debouncing a été ajouté aux filtres, pour permettre de saisir une valeur entière avant de déclencher une nouvelle agrégation (amélioration de l'UX)

Remarques additionnelles

L'implémentation d'un contexte permettant de stocker en mémoire un listing sélectionné au niveau de l'application permet l'interaction entre les différentes visualisations. Ainsi, un listing sélectionné sur la carte sera mis en évidence dans les autres vues, et vice-versa (un listing sélectionné depuis le panneau du parallel coordinates chart sera sélectionné au retour sur la page de la carte).
N'ayant jamais utilisé React auparavant, je me suis aidé de Copilot pour construire l'architecture de fichiers que je souhaitais, gagner du temps pour les tâches répétitives sur les différentes versions du projet et déboguer les implémentations complexes (quadtree, fisheye…). Deux fichiers (BubbleMap.tsx et FisheyeUtils.ts) restent monolithiques (500 et 700 lignes), résultant du développement par trial and error et de l'utilisation de Copilot sur certaines sections malgré une volonté de conserver une architecture maintenable et nécessiteraient un refactoring. Avec cette première expérience, une conception mieux réfléchie dès le début du développement donnerait de meilleurs résultats.

Aller plus loin

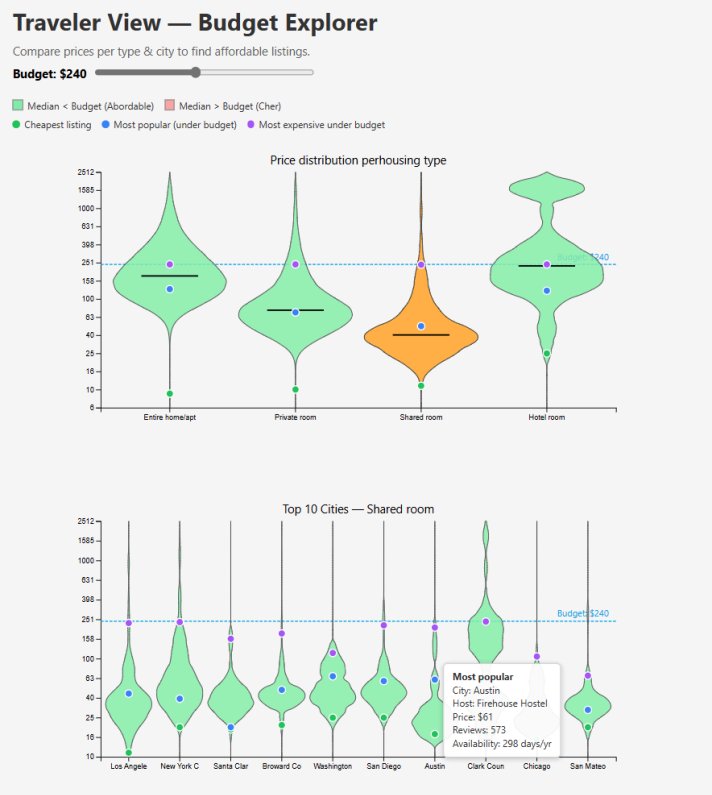
Avec plus de temps de développement, voici les pistes que nous aurions souhaité explorer :

- Un niveau de zoom supplémentaire serait souhaitable (la densité de listings pouvant rendre la sélection d'un listing précis complexe) tout comme déclencher le fisheye à un

niveau de zoom plus élevé afin d'assurer une meilleure scalabilité à l'application. Un travail d'optimisation préalable a été effectué mais toute amélioration est bienvenue considérant la taille des jeux de données potentiels.

- Enrichir la carte en fond via des API telles que Leaflet ou Google Maps permettrait aux utilisateurs de mieux situer les listings.
- Certains listings sont superposés (même propriétaire, même immeuble…) : nous devrions proposer une interaction au survol qui permette d'éclater la vue (sur un cercle par exemple) et de laisser l'utilisateur choisir précisément celui qu'il souhaite.
- Diversifier les propriétés des variables visuelles (bordure, bicolorisation, niveau de luminosité…) peut permettre de rendre l'application plus accessible et enrichir la vue de nouvelles informations sémantiques.
- Un mécanisme d'ajout de listings aux favoris (représentés par des étoiles sur la carte) constituerait un apport interactif intéressant pour les deux profils d'utilisateurs.
- Une interface de fonction "show on map" serait à développer pour pouvoir rediriger la vue sur la carte et la centrer sur le listing en question depuis les autres visualisations.

Violin Chart (Massinissa)



Données utilisées et prétraitement:

Pour visualiser les violons, nous n'avons gardé que les colonnes pertinentes à l'analyse des prix et de la disponibilité :

price, room_type, city, host_id, host_name, availability_365, number_of_reviews, calculated_host_listings_count.

Avant le tracé, les données sont filtrées afin d'exclure les logements non disponibles (**availability_365 = 0**).
Chaque valeur de prix est ensuite transformée en échelle logarithmique pour obtenir une meilleure lisibilité des écarts, notamment entre les logements très bon marché et ceux plus luxueux.

Calculs statiques et variables visuelles:

Pour chaque catégorie (type de logement ou ville), les indicateurs suivants sont calculés dynamiquement :

- Nombre d'annonces : via un groupement et un `count`.
- Valeurs extrêmes (min, max) des prix.
- Moyenne et médiane des prix.
- Pourcentage de logements sous le budget utilisateur**.**
- Densité de distribution des prix, estimée via un **Kernel Density Estimation (KDE)**.

Ces mesures sont ensuite encodées visuellement dans le graphe en violon :

- Prix (échelle logarithmique) → position verticale (axe Y) → *variable quantitative continue.*
- Type de logement ou ville → position horizontale (axe X) → *variable catégorielle.*
- Densité de prix → largeur du violon → *variable quantitative (distribution).*
- Médiane → ligne horizontale noire → *valeur centrale du groupe.*
- Budget utilisateur → ligne horizontale bleue → *référence de comparaison globale.*
- Statut d'abordabilité → couleur du violon : vert (médiane < budget), rouge (médiane > budget), doré pour le violon sélectionné.

Conception visuelle et interactions

Structure et niveaux de zoom :

La visualisation se compose de deux niveaux hiérarchiques :

- **Vue globale :** distribution des prix par *type de logement* (Entire Home, Private Room, etc.).
  L'utilisateur visualise instantanément les catégories les plus abordables et celles plus chères.
- **Vue détaillée :** en cliquant sur un violon, la vue se met à jour pour afficher la distribution des prix par *ville* (top 10 villes les plus représentées pour ce type).

Les deux niveaux sont synchronisés avec les **filtres globaux du projet** (via `useFilterStore` et `useFilteredData`) et se régénèrent dynamiquement lorsque les filtres ou le budget changent.

Interactivité et exploration :

Plusieurs interactions améliorent l'expérience d'exploration :

- **Glissière de budget :**
  L'utilisateur définit un budget maximal (par défaut 150 $).
  Cette valeur :
  - Modifie dynamiquement la couleur des violons (vert/rouge).
  - Fait apparaître une **ligne bleue horizontale** représentant le seuil de budget sur l'axe Y.
  - Met à jour les pourcentages de logements "sous budget" dans les info-bulles.

Survol (tooltip) : au survol d'un violon, une info-bulle affiche les statistiques calculées pour la catégorie :

```
Type : Entire Home
Listings : 12 300
Min : $30
Médiane : $140
Moyenne : $155
Max : $500
Sous budget (150 $) : 34 %
```

Cela fournit un *detail-on-demand* clair sans surcharger la vue principale.

Points représentatifs :
Trois cercles sont ajoutés sur chaque violon pour illustrer des exemples concrets de logements:

- 🟢 **Le plus abordable** (prix minimal).
- 🔵 **Le plus populaire** (plus grand nombre d'avis sous le budget).
- 🟣 **Le plus cher sous le budget.**

En survolant ces points, une info-bulle affiche les détails du logement :

```
Host : JohnDoe
City : Miami
Price : $120
Reviews : 48
Availability : 210 days/yr
```

Zoom local : Le clic sur un violon agit comme un *filtre local* et génère une seconde visualisation détaillant les distributions par ville.
Si l'utilisateur clique sur un autre violon, la vue détaillée est régénérée instantanément.
Le violon sélectionné dans la vue globale reste **doré**, conservant ainsi le repère visuel même lors des changements de budget.

Analyse et apports de la visualisation et lien avec les user goals:

Cette visualisation en violon apporte une réelle valeur ajoutée pour les deux profils d'utilisateurs ciblés :

- **Pour les hôtes**:
  - Comprendre la structure du marché et la concurrence.
  - Identifier les catégories où les prix sont stables ou au contraire très dispersés.
  - Analyser comment leur offre se positionne par rapport aux médianes locales.

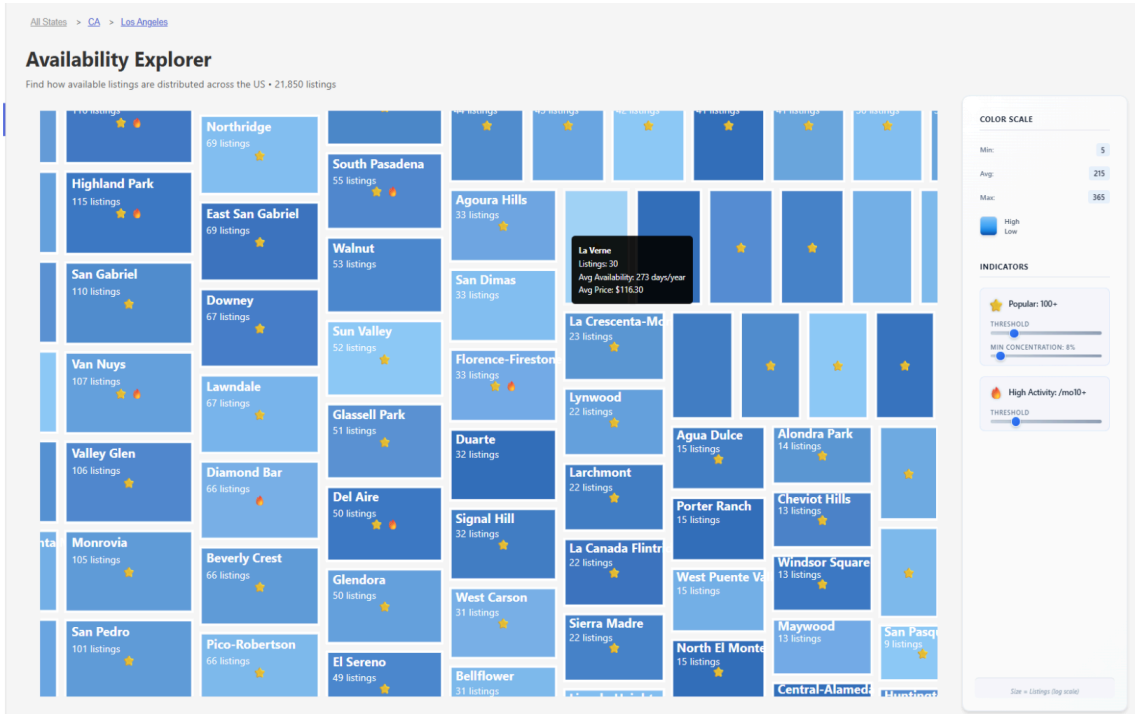
- **Pour les voyageurs**:

  - Repérer en un coup d'œil les types de logements accessibles selon un budget donné.
  - Explorer les différences de prix entre les villes d'un même type.
  - Découvrir, via les trois points représentatifs, des exemples concrets de logements correspondant à leurs critères.
  - Ajuster leur budget et voir en temps réel l'évolution de la proportion d'offres "abordables".

TreeMap (Benoît)

Exploration Hiérarchique par Treemap

La visualisation treemap offre une exploration interactive multiniveaux des annonces Airbn. Deux perspectives analytiques complémentaires permettent aux utilisateurs d'analyser soit la structure du marché (vue hôte), soit les opportunités de disponibilité (vue voyageur).



L'agrégation calcule dynamiquement pour chaque nœud : nombre d'annonces, prix/disponibilité moyennes, statistiques d'avis. L'algorithme de D3 optimise ensuite la disposition spatiale avec padding interne/externe constant, produisant des rectangles équilibrés plutôt que des longs

traits. La couleur s'applique via une échelle séquentielle basée sur ces métriques agrégées, tandis que les badges se déclenchent selon des seuils ajustables via curseurs en temps réel.

Hiérarchie

Navigation à 5 niveaux :

- Niveau 0-2 : État → Ville → Quartier (hiérarchie géographique standard)
- Niveau 3 : Catégorie (Hôte : taille de portefeuille | Voyageur : disponibilité annuelle)
- Niveau 4 : Type de chambre (feuille non-drillable)

Chaque regroupement s'effectue dynamiquement selon le niveau actuel, permettant des transitions fluides entre 19 états, 27 villes et 1 411 quartiers.

Encodage Visuel et Badges

Vue Hôte - Structure Tarifaire :

- Couleur : Dégradé violet inversé mappé au prix moyen
- Interprétation : Violet foncé = luxe, violet clair = économique
- Badges : (Destinations populaires + avis) et (Haute activité + avis/mois)

Vue Voyageur - Disponibilité :

- Couleur : Dégradé bleu direct mappé à la disponibilité moyenne
- Interprétation : Bleu foncé = toute l'année, bleu clair = saisonnière
- Mêmes badges révélant propriétés actives à forte rotation

Les domaines basés sur les percentiles réduisent l'influence des valeurs aberrantes.

Interactions

Fonctionnalités principales :

- Zoom D3 avec réinitialisation au double-clic
- Visibilité adaptative : Badges visibles à faible zoom, titres et comptages progressifs à zoom plus élevé
- Tooltips contextuels affichant statistiques agrégées au survol
- Navigation fil d'Ariane pour remonter hiérarchie instantanément
- Slider des badges permettent de faire ressortir la présence de listings ayant certaines caractéristique et ce même à un au niveau d'agrégation.

Parallel Coordinates Chart (Florent)

Tâches utilisateur supportées:

Le Parallel Coordinates Chart (PCC) permet d'afficher simultanément plusieurs dimensions quantitatives (Price, Reviews, Reviews/month, Availability, Min nights, Host listings) et de révéler les corrélations entre ces variables. Cette visualisation permet de répondre aux tâches utilisateurs suivantes :

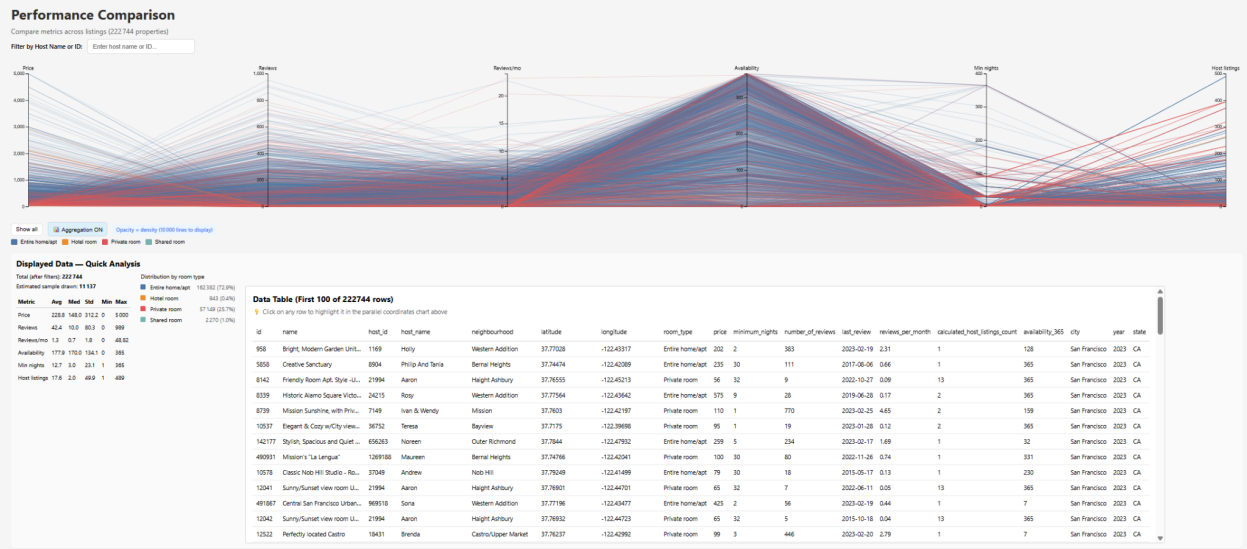
<u>Pour les clients :</u>

- Comparer la popularité des logements : En visualisant simultanément le nombre de reviews et les reviews par mois, les clients peuvent identifier les logements fréquemment réservés par d'autres voyageurs.
- Détecter les bons rapports prix/nombre de reviews : La représentation parallèle permet de tracer visuellement une ligne entre le prix et la popularité, révélant les offres sous-évaluées (prix bas, reviews élevées) ou surévaluées.
- Explorer la disponibilité annuelle : L'axe de disponibilité sur 365 jours aide à identifier les logements régulièrement disponibles versus ceux rarement accessibles, permettant une planification optimale.

<u>Pour les hôtes :</u>
- Comparer les performances de leurs logements avec la concurrence : Grâce au filtrage par nom/ID d'hôte, les propriétaires peuvent isoler leurs propres annonces et les comparer visuellement aux logements concurrents dans leur zone géographique et gamme de prix.
- Analyser les métriques clés de manière holistique : La visualisation simultanée de 6 dimensions (prix, reviews, disponibilité, durée minimale, nombre de logements) révèle des patterns qu'une analyse dimension par dimension ne montrerait pas, comme la corrélation entre professionnalisation (nombre de listings) et stratégie tarifaire.
- Identifier des stratégies gagnantes : En observant les lignes des logements les plus performants (reviews élevées, bonne disponibilité), les hôtes peuvent déduire les combinaisons optimales de prix, durée minimale de séjour et disponibilité.

Ces tâches s'inscrivent dans une démarche d'exploration et de comparaison multivariée, permettant aux utilisateurs de formuler des hypothèses ("Les logements avec minimum_nights élevé ont-ils moins de reviews ?") et de les valider visuellement en quelques secondes.

Fonctionnalités/interactions:



En plus des filtres globaux cette visualisation permet également de faire d'autres ajustement :

Scaling dynamique des axes:

Les axes s'ajustent automatiquement aux données filtrées pour maximiser la lisibilité :
- Recalcul du domaine (min/max) basé sur les données actuellement affichées
- Arrondissement des bornes pour des valeurs lisibles (ex: 9.7 → 10)
- Exception pour `availability_365` : domaine fixe [0, 365] pour cohérence sémantique

Cela permet de garder les colonnes bien proportionnées par rapport aux filtres choisis.

Agrégation spatiale par binning:

Face au défi de dessiner 200 000+ lignes (temps de rendu >3 secondes, voir plus en fonction de la puissance de l'ordinateur/serveur), une technique d'agrégation spatiale a été implémentée à partir d'une méthode trouvée sur internet:
Chaque dimension est divisée en 20 intervalles équidistants, la position de chaque logement y est calculée (0–19), puis une clé composite unique combine les 6 dimensions pour regrouper les logements similaires ; une seule ligne représentative est tracée par groupe, avec une opacité proportionnelle au logarithme du nombre d'éléments regroupés.

Réduction de plus de 90 % des lignes affichées, avec un rendu bien plus rapide tout en préservant les patterns de corrélation et les zones de densité.
Cette méthode d'agrégation est activée par défaut et peut-être désactivée/activée à volonté grâce à un bouton.

Échantillonnage stratifié:

Un échantillonnage stratifié à également été rajouté pour permettre au ordinateur n'ayant pas assez de puissance d'afficher une représentation approximative du dataset. Le fonctionnement de l'échantillonnage : les données sont regroupées par type de logement, un quota proportionnel (minimum 5 échantillons) est attribué à chaque catégorie, puis un mélange aléatoire évite les biais. Cette approche maintient la visibilité de toutes les catégories, même avec un échantillonnage agressif, et peut être activée ou désactivée librement via un bouton.

Filtrage par hôte (Details on demand)

La vue côté Host intègre une recherche ciblée avec filtrage par nom ou ID d'hôte, mise à jour en temps réel du graphique et des statistiques, bouton de réinitialisation rapide et affichage du nombre de résultats trouvés.
Cette fonctionnalité permet aux propriétaires d'analyser rapidement leurs propres logements et de les comparer à la concurrence locale.

Listing interactif:

Un listing est disponible en dessous du PCC. Le listing permet un tri sur les colonnes : un premier clic active le tri croissant, un second le tri décroissant, et un troisième rétablit l'ordre original. Cliquer sur une ligne du tableau permet de mettre la ligne en surbrillance sur le PCC avec une indication visuel "1 selected" et un bouton pour la déselection rapide.
Le listing n'affiche que les 100 premières données pour ne pas surcharger la page et faire lag l'utilisateur.

Panel de statistiques descriptives:

Pour chaque dimension, les métriques suivantes sont calculées et affichées dans l'analyse rapide en dessous du PCC : moyenne, médiane, écart-type, minimum et maximum.
Ces statistiques se mettent à jour automatiquement selon les filtres actifs, permettant une analyse quantitative rapide.

Une visualisation complémentaire affiche :
- Le nombre et le pourcentage de chaque type de logement (Entire home/apt, Private room, Shared room)
- Des légendes colorées cohérentes avec le graphique principal
- Une mise à jour dynamique en fonction des filtres globaux

Cette vue permet de comprendre rapidement la composition du marché visualisé.

Perspectives d'amélioration:

- Brushing multi-axes : Permettre la sélection de plages de valeurs directement sur les axes, créant ainsi des filtres interactifs qui se propagent aux autres visualisations
- Configuration dynamique : Ajouter un sélecteur de dimensions permettant de choisir quelles variables afficher parmi toutes celles disponibles
- Export de données : Permettre l'export des données filtrées au format CSV pour analyse externe

Conclusion

Ce projet de visualisation d'informations sur les données Airbnb aux États-Unis nous a permis d'explorer de manière complète la chaîne de conception d'un outil de visualisation interactif appliquée à un grand jeu de données. Nos choix de visualisations semblent à ce point complémentaires et permettent de répondre aux tâches utilisateurs prévues initialement, bien que des améliorations soient toujours possibles.

Le travail de prétraitement a été essentiel pour garantir une base de travail commune saine. Sur cette base solide, nous avons pu développer un pipeline de visualisation adaptatif autour de filtres globaux et de contextes assurant la mémorisation entre les vues. Certains membres du groupe ont pu s'initier à un nouveau framework par la même occasion et acquérir de l'expérience sur plusieurs domaines.

Le travail a été réparti équitablement entre les membres du groupe concernant les parties communes (le groupe remercie Benoît pour l'architecture de base du projet React).

Les bases qui permettent l'interaction entre les différentes vues sont posées et se limitent pour le moment à passer le listing sélectionné d'une vue à l'autre, mais mériteraient d'être enrichies avec une plus grande variété d'interactions (sélection de groupe sur la carte par exemple).

La scalabilité de l'application pour des jeux de données 10 fois plus grands est cependant à mettre en doute pour certaines visualisations, et nous travaillerions à améliorer ce point avec plus de temps de développement.

Nous sommes néanmoins satisfaits du résultat global obtenu et avons le sentiment d'avoir beaucoup appris, ne serait-ce qu'en ce qui concerne l'interprétation de certaines visualisations qui n'étaient pas nécessairement évidentes de prime abord.