Finding Home in France: A Data-Driven Approach to City Selection

HarvardX PH125.9x Data Science Capstone — Choose Your Own Project

Adam Solomon

2026-02-27

Contents

1 Executive Summary

This project informs a real-life family relocation decision using data science. It combines a weighted composite scoring system across key lifestyle metrics with a machine learning analysis of the French housing market. My family is planning to leave Paris, but where should we go?

The project proceeds in two phases. First, a city screening phase evaluates 55 major French cities across six criteria my family believes are most impactful to our quality of life (sunshine, rainfall, age demographics, affluence, poverty, and political alignment) using weighted composite scoring to narrow the field to seven target cities.

Those seven cities: Toulouse, Marseille, Lyon, Annecy, Montpellier, Bordeaux, and Paris (included as a benchmark) then move on to the second phase of the project. In phase two we build predictive machine learning models from the DVF (Demandes de Valeurs Foncières) database, a comprehensive open dataset published by the French government containing every property transaction recorded by notaries across France. After filtering out apartments, deduplicating multi-parcel records, and removing outliers the final modeling dataset contains 59,373 transactions from our seven target cities. We then train three supervised algorithms (linear regression, Random Forest, and XGBoost) to predict house prices from property and location features, alongside K-means clustering to identify distinct market tiers across communes.

The best-performing model by R^2 was Random Forest (0.676), with XGBoost close behind (0.670). Both tree models explain roughly two-thirds of house price variance using nine features. The single most impactful improvement came not from algorithm selection but from feature engineering: adding commune-level median income data boosted tree model R^2 by approximately 7 to 8 points, nearly doubling the gap over linear regression. This finding reinforces a key lesson from the course that the quality of your features matters more than the sophistication of your algorithm.

The final city rankings blend lifestyle criteria from the screening phase with affordability and model predictability from the real estate analysis. The results confirm that no single city dominates in every dimension. Mediterranean cities offer the most sunshine but weaker economic profiles and a political environment that is less aligned with our family, while southwestern cities like Toulouse provide a stronger overall balance of our key lifestyle factors.

2 Introduction

2.1 Personal Motivation

I currently live in Paris with my family, and we are planning a relocation to another French city. We are looking for a place with more sunshine, a house with a garden (a rarity in Paris), a community that aligns with our values, and a middle-class to affluent population to support my wife's planned business. Rather than relying on anecdote or real estate agent advice, I wanted to approach this decision with data, which made it a natural fit for a capstone project.

This project is not a toy exercise on a well-known dataset. The DVF database is a rich, messy, real-world source that requires substantial cleaning and domain knowledge to work with. The personal stakes made every modeling choice feel consequential: the train/test split matters because I need predictions for *future* transactions. When we put in an offer on a house, running it through the predictive model and knowing if the house is overpriced, underpriced, or priced just right will be a competitive advantage for us. Also, feature importance matters because it tells me what actually drives prices in each city.

2.2 Dataset Description

The primary dataset is the **DVF (Demandes de Valeurs Foncières)**, published by the French government on data.gouv.fr. It contains every property transaction recorded by notaries across France, including the sale price, property type, surface area, number of rooms, land area, and location down to the commune (municipality) level. I used five years of data (2020–2024), filtered to house sales ("Maison") in and around seven target cities.

Supplementary datasets include:

- **INSEE Filosofi 2020**: Commune-level median disposable income and poverty rates, providing neighborhood-level economic context.
- **Météo France 30-year climate normals**: Annual sunshine hours, temperature, and rainfall for 55 French cities.
- **INSEE RP 2020**: Population by age bracket at the commune level, used for demographic screening.
- **2022 Presidential Election results**: First-round vote shares by department, used as a proxy for political alignment.
- **Communes reference file** (data.gouv.fr): Administrative codes linking communes to departments and regions.

2.3 Project Goal

The goal is twofold. First, I screen 55 French cities on lifestyle criteria (climate, demographics, economics, politics) to select a shortlist of candidates. Second, I build machine learning models to predict house prices in these cities. The predictive models serve a practical purpose: when my family finds a house we like, we can run it through the model to estimate whether it is overpriced, fairly priced, or a bargain relative to comparable properties.

As a final step, I combine the lifestyle scores from phase one with price analysis from phase two to produce a data-driven city ranking, though this synthesis relies more on straightforward data science than on the ML models themselves.

The rubric requires at least two models, with one more advanced than linear regression. I use three supervised approaches (linear regression, Random Forest, XGBoost) to predict house prices from property and location features. Supervised models are appropriate here because we have a clear target variable (sale price) and want to learn the relationship between features and outcomes. I also use K-means clustering as an unsupervised complement. Unlike the supervised models, K-means has no target variable; instead it reveals natural groupings in the data, in this case identifying distinct market tiers across communes that span city boundaries.

3 Methods and Analysis

3.1 City Selection Methodology

Before diving into real estate modeling, I needed to narrow down France's many cities to a manageable shortlist. I built a screening dataset of 55 major French cities with six normalized criteria:

1. **Sunshine** (higher = better): We like the sun and are outdoorsy people
2. **Rainfall** (lower = better): We prefer dry climates for an outdoor lifestyle
3. **Working-age population** (higher = better): Percentage of residents aged 25–54, from INSEE census data because socializing with our peers is important to us
4. **Affluence** (higher = better): Third-quartile disposable income from Filosofi 2020 to better support my wife's business
5. **Poverty** (lower = better): Poverty rate from Filosofi 2020, inverted, as economically healthy communities better support my wife's planned business
6. **Political alignment** (lower far-right vote = better): Combined Le Pen + Zemmour first-round vote share from the 2022 presidential election, inverted to align with my family's political leanings

Each criterion was min-max normalized to a 0–1 scale. I then computed a weighted composite score, giving the highest weight to political alignment (1.0), followed by sunshine, rainfall, and affluence (0.75 each), age demographics (0.5), and poverty (0.25). These weights reflect my personal priorities, someone with different values would produce a different ranking, and that is the point: the framework is reusable even if the weights change.

```r
# Composite scoring weights (iterated through 5 rounds - see Methods text)
weights <- c(
    far_right = 1.00, # Primary: political alignment
    sunshine  = 0.75, # Secondary: climate
    rainfall  = 0.75, # Secondary: dry climate
    affluent  = 0.75, # Secondary: affluent population
    age       = 0.50, # Tertiary: working-age demographics
    poverty   = 0.25 # Minimal weight
)

city_screening <- city_screening |>
    mutate(
        composite_score = (
            sunshine_norm * weights["sunshine"] +
                rainfall_norm * weights["rainfall"] +
                affluent_norm * weights["affluent"] +
                far_right_norm * weights["far_right"] +
                age_norm * weights["age"] +
                poverty_norm * weights["poverty"]
        ) / sum(weights)
    )
```

The full city screening pipeline including data acquisition, joins, min-max normalization, and composite scoring is implemented in Sections 1 through 2.9 of `cyo_script.R` (lines 64–394).

The top-ranked cities are shown below. I selected the top six candidates plus Paris as a benchmark, giving seven cities for deep analysis.

Table 1: Target cities ranked by composite lifestyle score

City	Sunshine (hrs/yr)	Far-Right %	Composite Score
Paris	1662	13.7	0.796
Toulouse	2047	25.1	0.585
Marseille	2858	37.0	0.566
Aix-en-Provence	2801	37.0	0.552
Lyon	2007	24.7	0.542
Villeurbanne	2007	24.7	0.542
Annecy	1909	28.3	0.517
Montpellier	2668	35.0	0.502
Beziers	2600	35.0	0.499
Bordeaux	2035	28.0	0.470

Looking at this list there is still a question to be answered: why not just take the top seven cities? Some factors that matter to our family do not lend themselves to quantitative scoring. For example, Bordeaux ranked 12th overall but its two-hour TGV connection to Paris keeps us close to family and friends. Aix-en-Provence scored well but shares a department with Marseille, so including both would double-count the same real estate market in the modeling phase. Due to factors like these we selected seven cities that best suited our particular family to build the ML models on.

3.2 DVF Data Preparation

3.2.1 Commune-Level Filtering

A key methodological choice was how to define "a city" in the DVF data. French cities like Lyon and Paris have very few house sales within the administrative city limits — most houses are in surrounding suburbs. I used IGN commune boundary polygons and the `sf::st_touches()` function (in a separate preprocessing script) to identify all communes adjacent to each target city. This produced a lookup table of 159 communes across 10 departments (Paris suburbs span departments 92, 93, and 94 beyond the city's own 75).

Each commune was assigned a "ring" indicator: ring 0 for the city proper, ring 1 for adjacent suburbs. This became an important modeling feature — suburban houses tend to be cheaper per square meter but larger.

3.2.2 Loading and Filtering

I loaded the five DVF files (2020 S2 through 2024), filtered to house sales ("Maison" + "Vente") in the 159 target communes, and joined the commune metadata. This produced 68,006 raw transactions.

```r
# Load the pre-processed clean dataset
dvf_houses <- read_csv(
    file.path(DATA_DIR, "dvf_houses_clean.csv"),
    show_col_types = FALSE,
    col_types = cols(insee_code = col_character())
)
```

3.2.3 Data Cleaning

The raw DVF data required several cleaning steps:

1. **Deduplication**: DVF repeats rows when a property spans multiple cadastral parcels. I grouped by mutation key and summed land area across parcels (68,006 → 61,133 rows).
2. **Missing values**: Dropped 46 transactions with missing prices.
3. **Outlier removal**: Applied conservative bounds — prices between €10,000 and €5,000,000, built area between 20 and 1,000 m^2, and 1 to 20 rooms — removing 504 rows.
4. **Price-per-m^2 trimming**: Removed the top and bottom 1% of price-per-m^2 values to catch remaining data quality issues (1,210 rows).
5. **Feature engineering**: Computed `prix_m2` (price per square meter) and `has_land` (binary indicator for properties with land area recorded).
6. **Income enrichment**: Joined commune-level median disposable income from INSEE Filosofi 2020, achieving 100% coverage across all transactions.

The cleaning pipeline removed 12.7% of rows, leaving **59,373** transactions for analysis.

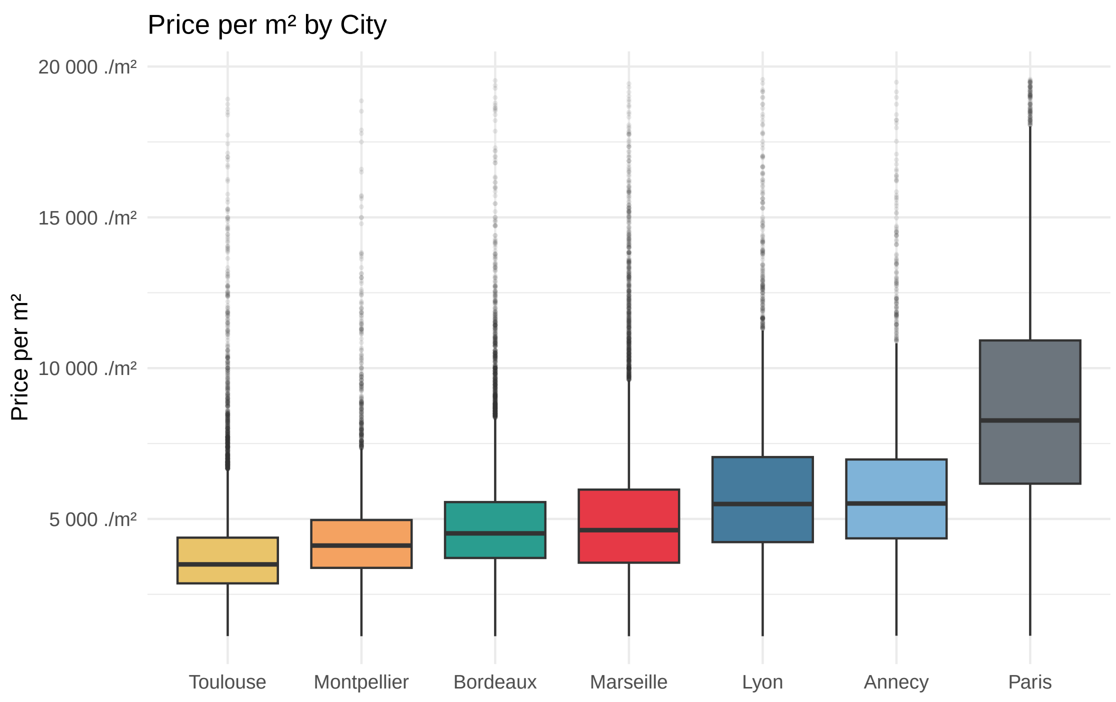
Table 2: Clean dataset summary by city

City	Transactions	Median Price	Median €/m^2	Median m^2
Toulouse	12070	€350,000	€3,492	98
Montpellier	6194	€410,000	€4,116	100
Bordeaux	16397	€417,142	€4,521	94
Marseille	10878	€400,000	€4,628	89
Lyon	5057	€590,000	€5,494	107
Annecy	2259	€604,500	€5,512	110
Paris	6518	€732,050	€8,265	90

3.3 Exploratory Data Analysis

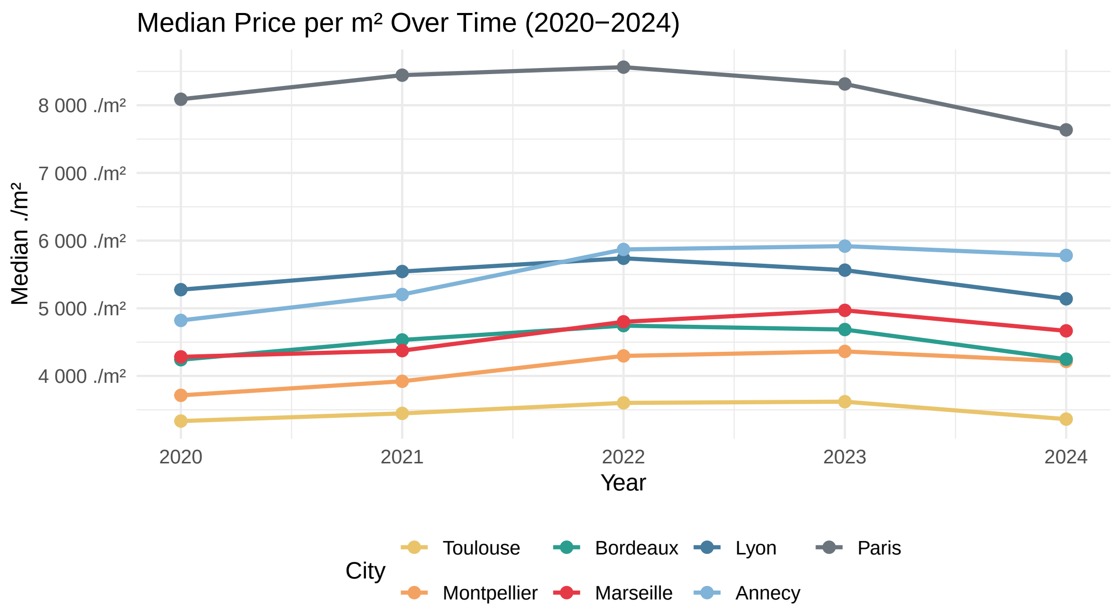
3.3.1 Price Distribution by City

The box plot below reveals the substantial price differences across cities. Paris and Annecy are the most expensive markets (median above €5,000/m^2), while Toulouse is the most affordable at roughly €3,500/m^2. The spread within each city is considerable — a sign that location-specific features (neighborhood, proximity to center) matter a great deal.



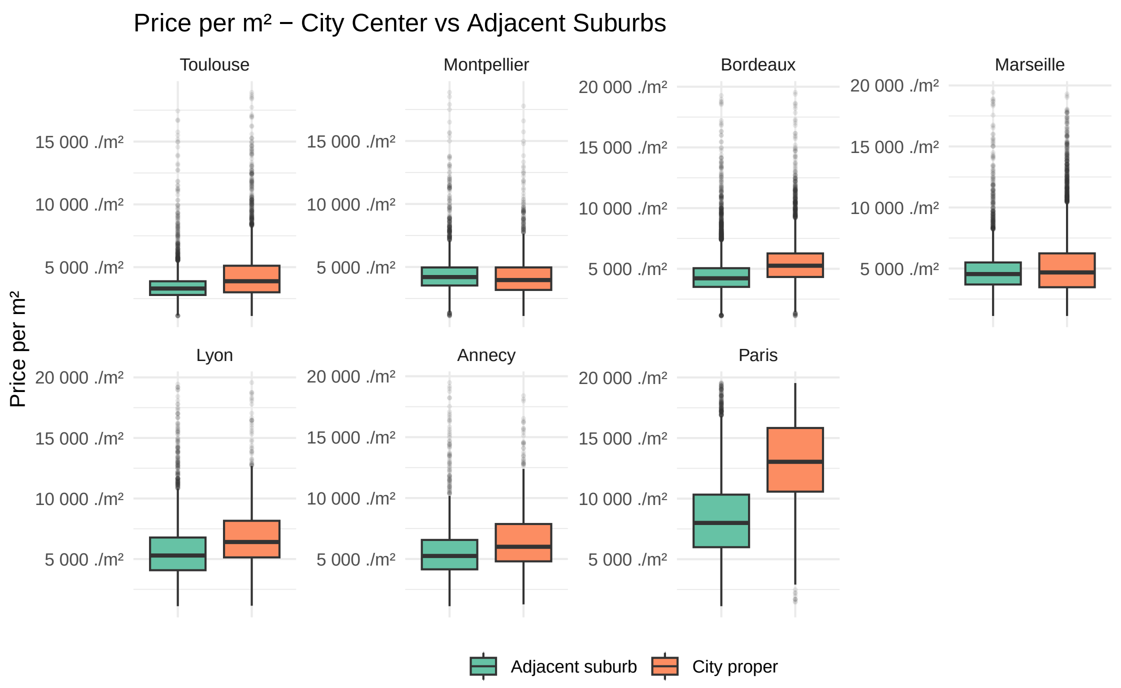
3.3.2 Temporal Trends

Median prices per m² were relatively stable across the five-year window, with some cities showing modest appreciation (Annecy, Paris) and others flattening or declining slightly after 2022 (Bordeaux, Toulouse). Despite this relatively low signal, including year as a feature allowed the model to account more easily for any gradual temporal drift and would likely become more important were the dataset to grow to span decades.



3.3.3 The Ring Effect: City Center vs. Suburbs

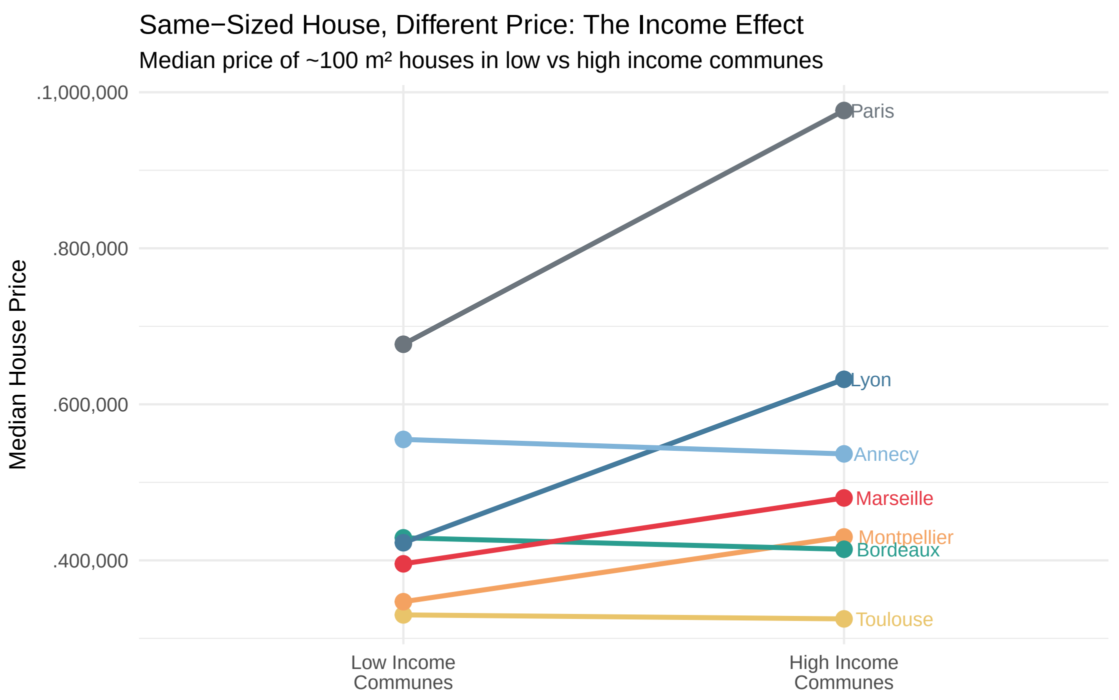
One of the more interesting patterns is how the price gap between city proper (ring 0) and adjacent suburbs (ring 1) varies by city. In Paris, the gap is enormous with city-center houses commanding roughly double the price per m² of suburban ones. In Montpellier, the pattern reverses: suburbs are slightly more expensive. What does this mean for our modeling? A single ring coefficient in linear regression would average the Paris premium with the Montpellier reversal and get both wrong. This is why the linear model uses interaction terms between city and ring, giving each city its own urban premium. Tree-based models discover these city-specific patterns automatically through splits.



3.3.4 The Income Effect

Perhaps the most telling EDA finding is what I call the “income effect.” When I restrict the data to similarly-sized houses (~100 m²) and split each city’s communes into low-income and high-income halves, the price gap is striking. In Lyon, a ~100 m² house in a high-income commune costs roughly 50% more than the same-sized house in a low-income commune. Paris shows a similar pattern at 44%, and even mid-range cities like Montpellier and Marseille show 20–25% premiums. This motivated the addition of commune-level median income as a model feature.

However, the pattern is not universal. In Toulouse, Bordeaux, and Annecy the relationship reverses slightly, with low-income communes showing marginally higher prices for the same-sized house. This suggests that commune-level income, while powerful, is too coarse a proxy in some markets. A commune might have low overall income but contain a pocket of desirable houses near a lake, a historic center, or good schools that command premiums the income data cannot capture. I suspect that more granular location data at the neighborhood or street level would resolve these reversals, a puzzle I will return to in the Limitations section.



3.3.5 Feature Correlations

Table 3: Correlation matrix of numeric features

	price	surface	rooms	land	price_m2	year
price	1.00	0.64	0.46	0.18	0.67	0.00
surface	0.64	1.00	0.74	0.23	-0.03	-0.02
rooms	0.46	0.74	1.00	0.16	-0.04	-0.01
land	0.18	0.23	0.16	1.00	0.04	0.00
price_m2	0.67	-0.03	-0.04	0.04	1.00	0.02
year	0.00	-0.02	-0.01	0.00	0.02	1.00

The correlation matrix confirms several expected relationships and reveals some useful patterns. Surface area is the strongest predictor of price among features used in the model ($r = 0.64$), followed by rooms ($r = 0.46$). Note that price per m² shows a higher correlation with price ($r = 0.67$) but is excluded from modeling because it is derived directly from the target variable (price / surface), which would constitute data leakage.

Surface and rooms are highly correlated with each other ($r = 0.74$), meaning they carry a lot of overlapping information. In a linear model, this multicollinearity can inflate standard errors and make individual coefficients harder to interpret. I considered dropping rooms to reduce redundancy, but ultimately kept both because tree-based models handle correlated features naturally and rooms still contributes meaningful predictive power (11.8% of XGBoost gain) despite the overlap with surface.

Land area has a weaker relationship with price ($r = 0.18$), suggesting that lot size plays a secondary role compared to the house itself. Year shows essentially zero correlation with all other variables, reinforcing the observation from the temporal trends that the market was relatively stable over this five-year window.

3.4 Modeling Approach

3.4.1 Train/Test Split

I used a **temporal split**: training on 2020–2023 and testing on 2024. This is the natural choice for real estate because prices are path-dependent, a random split would leak future market conditions into the training data. A family making a purchase decision in 2024 would only have access to historical data, so the temporal split mirrors the real-world use case.

```
train <- dvf_houses |> filter(year <= 2023)
test <- dvf_houses |> filter(year == 2024)
```

This produced a 49,269/10,104 split (83/17%). The split is roughly proportional across cities — each city contributes 16–19% of its transactions to the test set.

3.4.2 Feature Selection

I selected nine features based on domain knowledge and EDA insights:

Feature	Type	Description
surface	numeric	Built area in square meters
rooms	numeric	Number of main rooms
land	numeric	Land area in square meters, 0 where no land
target_city	factor (7 levels)	City group: Toulouse, Marseille, Lyon, Annecy, Montpellier, Bordeaux, Paris
ring	integer 0/1	0 = city proper, 1 = adjacent suburb
year	numeric	Transaction year (2020-2024)
quarter	integer 1-4	Quarter of year
has_land	integer 0/1	Whether property has recorded land area
median_income	numeric	Commune-level median disposable income from INSEE Filosofi 2020

Target variable: price (Valeur fonciere) — the sale price in euros.

Nine features were selected for modeling based on the EDA findings and domain knowledge. Column names were cleaned for compatibility with the randomForest and xgboost packages, which do not handle spaces in variable names. Categorical variables were cast to appropriate types: target_city as a factor with seven levels, and ring, quarter, and has_land as integers. I deliberately excluded prix_m2 (price per square meter) because it is derived from the target variable — including it would constitute data leakage. The commune-level median income was the last feature added and proved to be the most impactful single addition (see Results). The full feature preparation code is in Section 4.2 of cyo_script.R.

3.4.3 Evaluation Metrics

I evaluate each model using three metrics: RMSE (root mean squared error, in euros), MAE (mean absolute error, more robust to outliers), and R^2 (proportion of variance explained). All metrics are computed on the held-out 2024 test set. I also report per-city breakdowns to ensure no city is systematically underserved by the pooled model.

The evaluation functions are implemented in Section 4.3 of cyo_script.R.

3.4.4 Model 1: Linear Regression (Baseline)

Linear regression provides a transparent baseline. To handle the fact that different cities have different price structures, I included interaction terms: `target_city × surface` (each city gets its own price-per-m^2 slope) and `target_city × ring` (each city gets its own urban premium). Without these interactions, a single pooled slope would be dominated by high-volume cities like Bordeaux.

```r
lm_formula <- price ~ target_city * (surface + ring) + rooms + land +
    year + quarter + has_land + median_income

lm_fit <- lm(lm_formula, data = train_model)
lm_pred_test <- predict(lm_fit, newdata = test_model)
lm_results <- evaluate_model(
    test_model$price, lm_pred_test,
    as.character(test_model$target_city), "Linear Regression"
)
all_results <- bind_rows(all_results, lm_results)
```

3.4.5 Model 2: Random Forest

Random Forest handles non-linear relationships and feature interactions without requiring manual specification. Where linear regression needed explicit interaction terms (city × surface, city × ring), the 500-tree ensemble discovers these patterns on its own through recursive partitioning. If land area matters more in Toulouse than in Paris, the model learns that from the data. I don't have to tell it.

```r
set.seed(42)
rf_fit <- randomForest(price ~ ., data = train_model, ntree = 500, importance = TRUE)
rf_pred_test <- predict(rf_fit, newdata = test_model)
rf_results <- evaluate_model(
    test_model$price, rf_pred_test,
    as.character(test_model$target_city), "Random Forest"
)
all_results <- bind_rows(all_results, rf_results)
```

3.4.6 Model 3: XGBoost

XGBoost builds an ensemble of shallow decision trees sequentially, where each new tree corrects the errors of the ones before it. Unlike Random Forest, XGBoost requires numeric input. I converted all factor variables to one-hot encoded columns using model.matrix.

One important methodological detail: XGBoost uses early stopping to determine how many boosting rounds to run. If I had used the test set for this, the test set would have influenced model selection, biasing the final evaluation metrics upward. To avoid this data leakage, I held out 20% of the training data as a separate validation set for early stopping, keeping the test set completely untouched until final evaluation.

```r
# Illustrative - see cyo_script.R Section 4.6 for full implementation
xgb_fit <- xgb.train(
    params = list(
        objective = "reg:squarederror",
        max_depth = 6, eta = 0.1,
        subsample = 0.8, colsample_bytree = 0.8
    ),
    data = dtrain,
    nrounds = 1000,
    evals = list(train = dtrain, val = dval),
    early_stopping_rounds = 50
)
```

3.4.7 Model 4: K-Means Clustering (Unsupervised)

As a complement to the supervised models, I used K-means clustering to segment communes by their real estate market characteristics. I aggregated transactions to the commune level (median price/m^2, surface, rooms, land area, transaction volume, and median income), scaled the six features, and tested k = 2 through 8 using both the elbow method (within-cluster sum of squares) and silhouette analysis. The silhouette-optimal k was 6, but this produced singleton and near-singleton clusters (one or two communes each), making them uninterpretable. I chose k = 4 instead: its silhouette score (0.404) is nearly identical to k = 6 (0.414), but the resulting clusters are large enough to represent meaningful market tiers.

```
commune_summary <- dvf_houses |>
    group_by(insee_code, commune_name, target_city, ring) |>
    summarise(
        median_prix_m2 = median(prix_m2, na.rm = TRUE),
        median_surface = median(`Surface reelle bati`, na.rm = TRUE),
        median_rooms = median(`Nombre pieces principales`, na.rm = TRUE),
        median_land = median(`Surface terrain`, na.rm = TRUE),
        n_transactions = n(),
        median_income = median(median_income_commune, na.rm = TRUE),
        .groups = "drop"
    ) |>
    filter(n_transactions >= 10, !is.na(median_income))

cluster_features <- c(
    "median_prix_m2", "median_surface", "median_rooms",
    "median_land", "n_transactions", "median_income"
)
commune_scaled <- scale(commune_summary[, cluster_features])

FINAL_K <- 4
set.seed(42)
km_fit <- kmeans(commune_scaled, centers = FINAL_K, nstart = 25)
commune_summary$cluster <- factor(km_fit$cluster)
```

4 Results

4.1 Overall Model Comparison

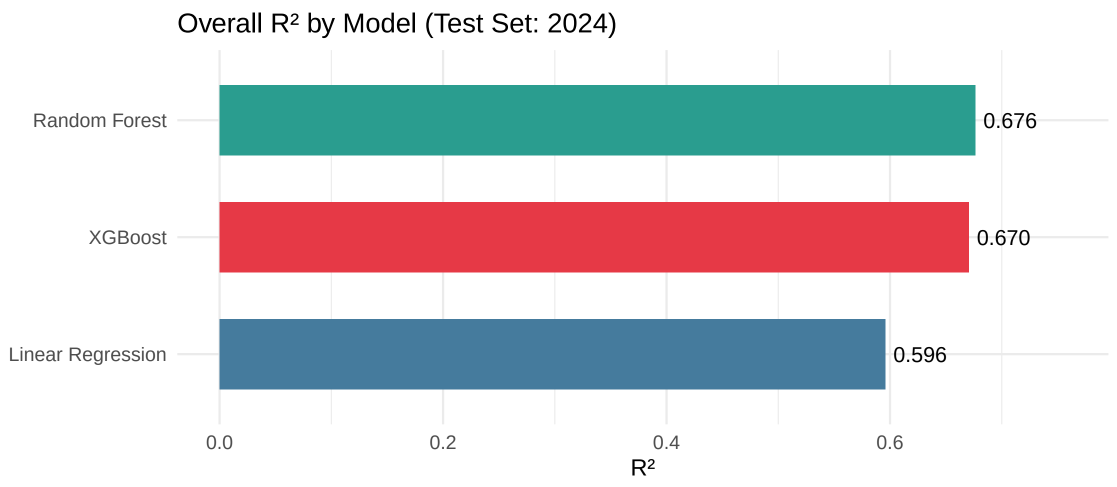
Table 5: Overall model performance on 2024 test set

Model	n	RMSE	MAE	R^2
Linear Regression	10104	€250,599	€166,492	0.596
Random Forest	10104	€224,212	€140,149	0.676

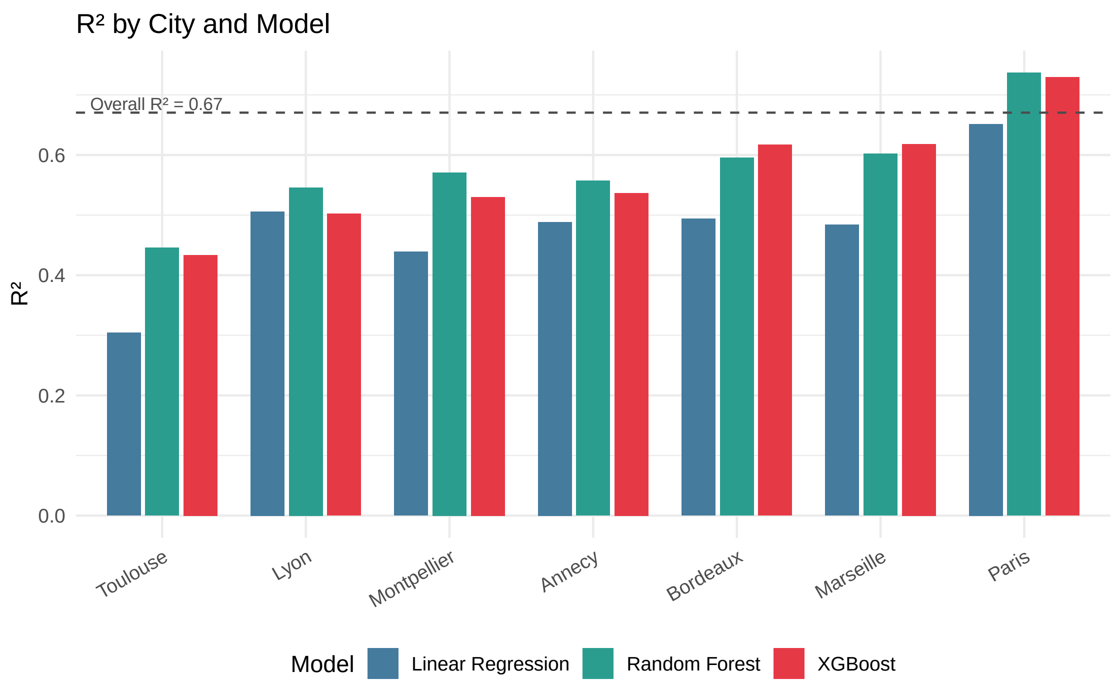
Model	n	RMSE	MAE	R^2
XGBoost	10104	€226,259	€138,695	0.670

Random Forest achieved the best performance with an R^2 of 0.676, closely followed by XGBoost (0.670). Both tree models substantially outperformed linear regression, reducing RMSE by approximately 11% and MAE by 17%.

The narrow gap between Random Forest and XGBoost (R^2 difference of just 0.006) suggests that the performance ceiling is driven by *missing features* rather than model complexity. The DVF dataset does not include property condition, renovation status, energy performance rating, proximity to transit, or school quality — all of which influence house prices. A more sophisticated algorithm cannot recover information that is not in the data.



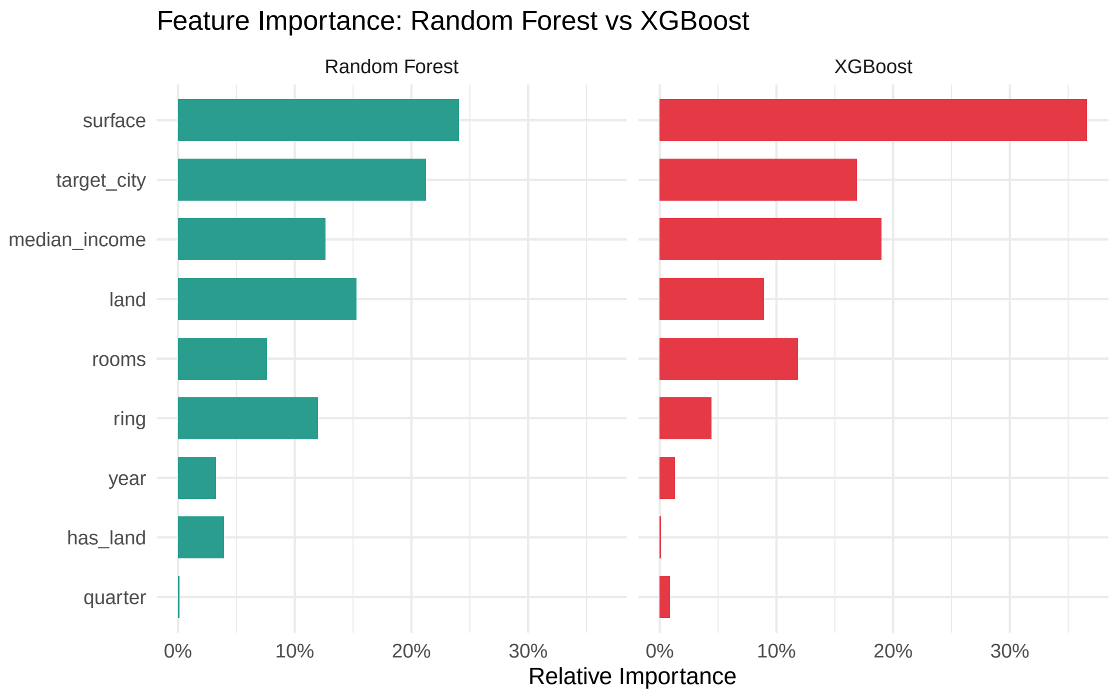
4.2 Per-City Performance



The per-city results reveal important heterogeneity. Paris is the most predictable market (R^2 0.73 for both tree models), likely because the large income variation across its many communes provides a strong price signal. Toulouse is the hardest to predict (R^2 0.43 for XGBoost) despite having the most training data. Its market is simply more heterogeneous, with price variation driven by factors our features cannot capture.

Notably, Annecy (the smallest sample with only 419 test transactions) performs mid-pack (R^2 0.54), suggesting that sample imbalance is not the primary driver of per-city performance differences.

4.3 Feature Importance



Both models agree that surface area is the dominant predictor, which is unsurprising since bigger houses cost more. The more interesting finding is that commune-level median income ranks as the second most important feature in XGBoost (19% of total gain), ahead of the city identifier and rooms. This confirms the EDA insight: neighborhood wealth is a powerful proxy for desirability, school quality, and amenities that our other features cannot capture. Rooms contributes meaningful predictive power in both models despite its high correlation with surface, validating the decision to keep both features. The city identifier and ring provide geographic context, while land area and temporal features play smaller supporting roles.

4.4 Impact of the Income Feature

Adding commune-level median income was the single most impactful modeling decision. The table below shows the before-and-after comparison (the "before" values come from a prior run of the pipeline without the income feature):

Table 6: Impact of adding commune-level median income feature

Model	R^2 Before	R^2 After	Δ R^2
Linear Regression	0.554	0.596	+0.042
Random Forest	0.597	0.676	+0.079
XGBoost	0.599	0.670	+0.071

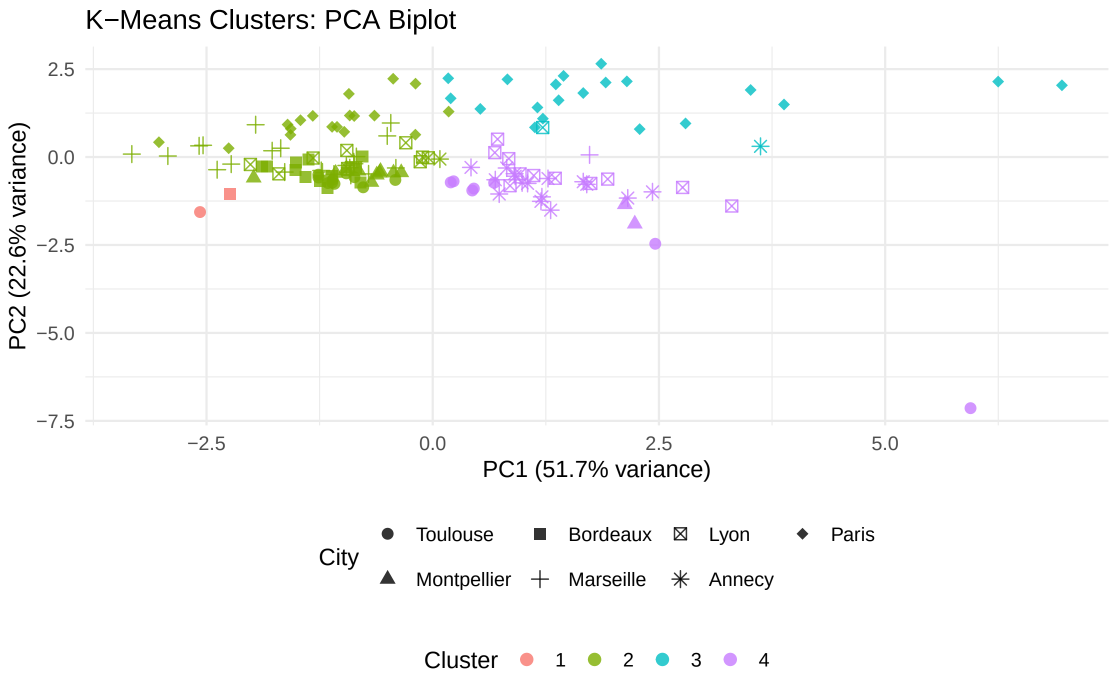
Tree models gained 7 to 8 R^2 points, far exceeding linear regression's 4-point improvement. The reason is that tree models can use income to create non-linear interaction effects (e.g., "if income > €25,000 AND surface > 100 m^2, predict a premium"), while linear regression only gets a single additive coefficient.

4.5 K-Means Market Clustering

The K-means analysis (k=4) reveals four distinct market tiers across the seven cities:

Table 7: K-means cluster centroids (k=4)

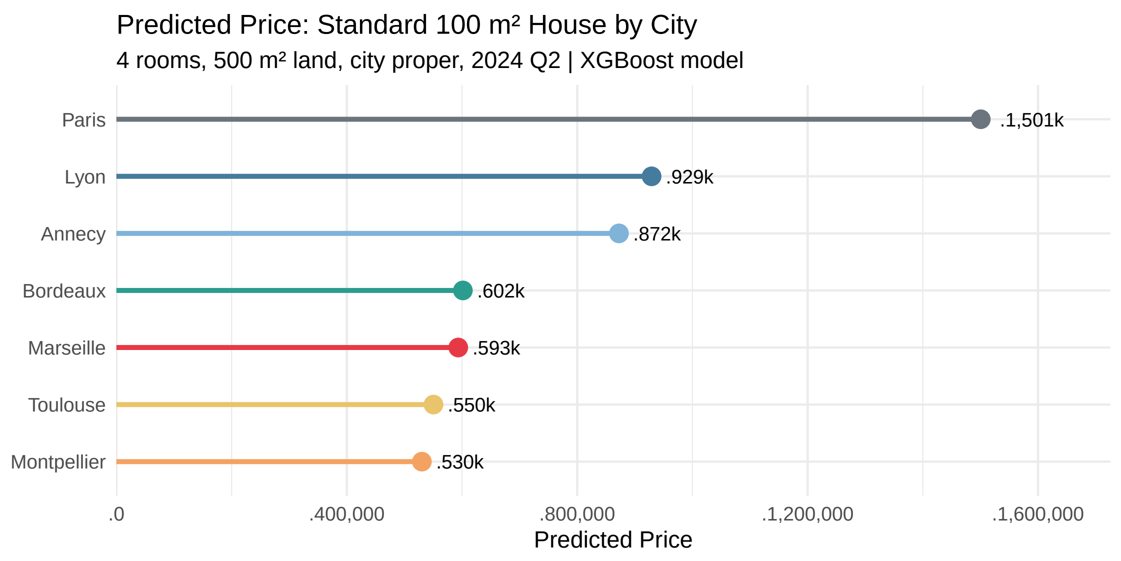
Cluster	Communes	Median €/m²	Median Surface	Avg Transactions	Median Income
3	22	11328	131	106	€32,366
4	37	5236	122	203	€30,149
2	83	5120	90	458	€22,476
1	2	4566	96	5724	€22,795



The PCA biplot shows that clusters cut across city boundaries. Paris communes appear in multiple tiers, and communes from Lyon, Annecy, and Marseille overlap in the same clusters despite belonging to different cities. This cross-city clustering adds nuance to the relocation decision. It is not just about choosing a city, but about choosing the right neighborhood tier within that city.

4.6 City Rankings and Predictions

To produce a practical recommendation, I used the XGBoost model to predict the price of a "standard house" (100 m², 4 rooms, 500 m² land, city proper, 2024 Q2) in each city, using city-specific median commune income.



4.6.1 Multi-Criteria Ranking

The final ranking combines three dimensions with explicit weights: lifestyle (50%, from the composite screening score), affordability (30%, inverted predicted price), and model predictability (20%, XGBoost R^2). Including predictability rewards cities where our model's predictions are more reliable, which is useful when the prediction will inform an actual purchase decision.

Table 8: Final city ranking (50% lifestyle + 30% affordability + 20% predictability)

Rank	City	Blended Score	Lifestyle Score	Affordability	Predictability
1	Marseille	0.688	0.566	0.935	0.624
2	Bordeaux	0.637	0.470	0.927	0.619
3	Montpellier	0.616	0.502	1.000	0.326
4	Paris	0.598	0.796	0.000	1.000
5	Toulouse	0.586	0.585	0.979	0.000
6	Annecy	0.522	0.517	0.647	0.349
7	Lyon	0.494	0.542	0.589	0.232

Marseille emerges as the top-ranked city (0.688), driven by strong affordability (0.935) and solid predictability (0.624) despite a middling lifestyle score. Bordeaux follows closely (0.637) with a similar profile. Montpellier ranks third, boosted by the highest affordability score (1.000) but held back by lower predictability. The top three cities all share a pattern: they score well on affordability and adequately on predictability, compensating for lifestyle scores below Paris.

Paris, despite having the highest lifestyle score (0.796) and the best model predictability (1.000), ranks fourth because its affordability score is zero. It is the most expensive market by a wide margin. Lyon finishes last, penalized by both high prices and low predictability. This confirms the intuition that motivated the project: Paris is a great city to leave, not to buy a house in, and the ranking framework makes that reasoning explicit.

5 Conclusion

5.1 Summary

This project demonstrated a complete data science pipeline applied to a real-world decision: choosing the best French city to relocate to. Starting with a screening of 55 cities across six lifestyle dimensions, I narrowed the field to seven candidates and built predictive models using nearly 60,000 house transactions from the DVF open dataset.

The key technical findings are: (1) Random Forest and XGBoost substantially outperform linear regression for house price prediction, explaining roughly two-thirds of price variance on the 2024 test set; (2) commune-level median income is a remarkably powerful feature, boosting tree model R^2 by 7 to 8 points; and (3) model performance varies significantly across cities, with Paris being most predictable and Toulouse most challenging, in ways that are not explained by sample size alone.

The practical conclusion is that Marseille offers the strongest overall balance of affordability, predictability, and lifestyle factors under the blended ranking, with Bordeaux and Montpellier close behind. The framework itself, weighted composite scoring combined with ML-based affordability analysis, is reusable for anyone with different preferences, and different weights would produce a different winner.

5.2 Limitations

Several limitations affect the analysis:

The best R^2 of 0.676 means roughly one-third of price variance remains unexplained. The DVF dataset lacks critical property attributes: house condition, renovation status, energy performance rating (DPE), proximity to public transit, and school catchment quality. These omissions likely explain much of the residual error and the heterogeneity in per-city performance.

The commune-level income data is from a single year (Filosofi 2020) and is applied uniformly to transactions from 2020 through 2024. In reality, neighborhood incomes evolve over time, and any communes that gentrified or declined during this window would introduce noise.

The DVF data for 2020 covers only the second semester, introducing a mild seasonal imbalance in the first year of the training set.

The city screening weights are subjective. Different weights would produce a different ranking. Someone who prioritizes sunshine over politics would favor Marseille or Montpellier. The framework is transparent about this subjectivity, but it is a limitation nonetheless.

Finally, the temporal train/test split, while realistic, means the model is trained on a period of modest price changes (2020–2023) and tested on a relatively stable 2024. A significant future market shift could degrade performance.

5.3 Future Work

Several extensions could improve the analysis. Incorporating DPE (energy performance) ratings, which are increasingly available via French open data, would add a significant predictor. Geospatial features such as distance to nearest metro station, proximity to green space, and school locations could be computed from OpenStreetMap data and would likely reduce the remaining R^2 gap, though by how much is hard to say without testing.

On the modeling side, a stacked ensemble (combining linear regression, RF, and XGBoost predictions) could squeeze out marginal gains. City-specific models rather than a single pooled model might improve performance for difficult markets like Toulouse, at the cost of smaller training samples.

For the relocation decision itself, a natural next step is to use the trained model as a property valuation tool. Feeding in specific listings would let us identify undervalued houses in whichever city we choose.

6 References

- DVF (Demandes de Valeurs Foncières): https://www.data.gouv.fr/fr/datasets/demandes-de-valeurs-foncieres/
- INSEE Filosofi 2020 (commune-level income): https://www.insee.fr/fr/statistiques/6692392
- INSEE RP 2020 (population by age): https://www.insee.fr/
- Météo France climate normals: https://meteofrance.com/
- 2022 Presidential Election results: https://www.data.gouv.fr/datasets/election-presidentielle-des-10-et-24-avril-2022-resultats-definitifs-du-1er-tour
- IGN commune boundaries (for adjacency analysis): https://www.data.gouv.fr/
- R Core Team (2025). R: A language and environment for statistical computing. https://www.R-project.org/
- Wickham H, et al. (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686.
- Liaw A, Wiener M (2002). Classification and Regression by randomForest. R News, 2(3), 18-22.
- Chen T, Guestrin C (2016). XGBoost: A Scalable Tree Boosting System. Proc. 22nd ACM SIGKDD, 785-794.
- Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K (2023). cluster: Cluster Analysis Basics and Extensions. R package. https://CRAN.R-project.org/package=cluster
- Pebesma E (2018). Simple Features for R: Standardized Support for Spatial Vector Data. The R Journal, 10(1), 439-446.

7 Appendix: Session Info

```r
sessionInfo()
```

```
## R version 4.5.1 (2025-06-13)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.7.3
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/Paris
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] here_1.0.2      scales_1.4.0    knitr_1.50      lubridate_1.9.4
##  [5] forcats_1.0.1   stringr_1.5.2   dplyr_1.1.4     purrr_1.1.0
##  [9] readr_2.1.5     tidyr_1.3.1     tibble_3.3.0    ggplot2_4.0.0
## [13] tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.6       compiler_4.5.1     tidyselect_1.2.1   yaml_2.3.10
##  [5] fastmap_1.2.0      R6_2.6.1           labeling_0.4.3     generics_0.1.4
##  [9] rprojroot_2.1.1    pillar_1.11.1      RColorBrewer_1.1-3 tzdb_0.5.0
## [13] rlang_1.1.6        stringi_1.8.7      xfun_0.53          S7_0.2.0
## [17] timechange_0.3.0   cli_3.6.5          withr_3.0.2        magrittr_2.0.4
## [21] digest_0.6.37      grid_4.5.1         rstudioapi_0.17.1  hms_1.1.3
## [25] lifecycle_1.0.4    vctrs_0.6.5        evaluate_1.0.5     glue_1.8.0
## [29] farver_2.1.2       rmarkdown_2.30     tools_4.5.1        pkgconfig_2.0.3
## [33] htmltools_0.5.8.1
```