BSc (Hons) Banking and International Finance

Industry Applied Research Project

To what extent is the valuation of Energy Performance Certificates moderated by external economic shocks and shifting market states in the UK residential sector?

Name	Timon Plemenitas
Student Number	230065203
Supervisor	Professor Sotiris Tsolacos
Submission date	10 May 2026

"I certify that I have complied with the guidelines on plagiarism outlined in the Course Handbook in the production of this dissertation and that it is my own, unaided work".

Signature: 

Abstract

This study examines how Energy Performance Certificate (EPC) ratings are capitalised in UK residential property prices during changing market conditions. It applies propensity score matching and difference-in-differences models to compare EPC bands around two shocks: the 2021 hot housing market and the 2022 energy crisis. Results show that energy-cost shocks widened brown discounts for inefficient houses, especially Bands F and G, while hot-market conditions compressed these penalties. The findings suggest that EPC valuations are not fixed but are highly dependent on market state and energy costs.

Contents

List of Tables

List of Figures

1 Introduction

The world is increasingly exposed to climate-related risks, prompting governments to reduce the environmental impact of individuals through policies that use market forces. The housing market is no exception. In the United Kingdom, since 2008, every property offered for sale or rent has been required to hold an Energy Performance Certificate (EPC). This framework was strengthened in 2018 through the introduction of the Minimum Energy Efficiency Standard (MEES), which made it illegal for landlords to let properties with an EPC rating of F or G unless energy-efficiency improvements were undertaken. Future policy proposals go further still, with plans to raise the minimum legal standard for rental properties to EPC Band C by 2030 (Great Britain, 2015; DESNZ, 2026a).

These regulations matter because owners of poorly rated properties may be forced either to renovate in order to comply with MEES or to sell at a discount that reflects the expected cost of improvement. Yet regulation alone may not determine the size of these price effects. During periods of low energy prices and stable market conditions, buyers may attach only limited importance to energy efficiency, especially when other housing characteristics such as location, size, and quality dominate the purchase decision. By contrast, external shocks may make low-EPC properties appear more financially exposed, causing them to trade at different premiums or discounts.

This paper answers: to what extent is the valuation of Energy Performance Certificates moderated by external economic shocks and shifting market conditions in the UK residential sector? This question is important because it moves beyond the standard issue of whether EPC ratings affect pricing and instead examines the conditions under which their effect becomes more or less pronounced. In doing so, the analysis offers policymakers deeper insight into housing-market dynamics and can support more informed decisions when designing future environmental regulation and help remove impact of geopolitical instability such as tensions in the Middle East.

The analysis focuses on two distinct shocks. The first is the 2021 Stamp Duty holiday, during which housing demand rose sharply (HM Revenue & Customs, 2021). The second is the 2022 Russia–Ukraine war and the subsequent energy crisis, which dramatically increased the salience of household energy costs. Together, these episodes provide useful settings in which to examine whether EPC-related pricing is stable or instead sensitive to wider economic and political disruptions.

2 Literature Review

Existing literature examines whether Energy Performance Certificate (EPC) ratings are capitalised into dwelling prices as governments seek to reduce emissions. This review focuses on four themes: energy-efficiency premiums, regional and property-type variation, rental-market incentives, and climate-policy transition risk. Its key limitation is that most studies estimate average EPC effects, while this paper examines whether those effects vary across market states.

2.1 The Capitalisation of Energy Efficiency

A core idea in the literature is whether buyers pay a premium for energy efficiency because of expected lower energy bills. Fuerst et al. (2013), using transactions between 1995 and 2011, found that more efficient homes sold for higher prices, with A/B-rated dwellings commanding a 14% premium over G-rated properties. Fuerst et al. (2015) confirmed this, showing that A/B-rated homes achieved a 5% premium relative to D-rated homes, while G-rated homes were discounted by nearly 7%. These studies establish that EPC information can be capitalised into prices, but they treat this premium as stable rather than dependent on external factors.

2.2 Regional Dynamics and Property Types

EPC capitalisation also varies by market. Fuerst et al. (2013) and Fuerst et al. (2015) find stronger percentage premiums in lower-priced regions, such as the North East, but weaker effects in supply-constrained areas such as London. This suggests that buyers in expensive markets may prioritise location and affordability over energy efficiency, while fixed energy savings represent a smaller share of total property value. The studies also show that EPC price effects are more pronounced for terraced dwellings and flats than for detached homes.

2.3 The Rental Market and the Split-Incentive Problem

Expanding the geographic scope, Fuerst et al. (2016) investigated the Welsh housing market using approximately 192,000 transactions. It isolates the private rental sector to examine the split-incentive problem, a market failure in which landlords bear the capital costs of executing energy upgrades while tenants reap the financial benefits of lower utility bills. The authors discovered that within the private rental segment, low-EPC dwellings were not traded at significant discounts. This indicates that buy-to-let investors, knowing that tenants would bear the energy costs, did not penalise energy inefficiency in their purchase prices in the same way that owner-occupiers did.

2.4 Climate Policy and Transition Risk

Because voluntary eco-labels failed to drive sufficient organic improvements in the rental sector, the UK government introduced binding climate regulations. The 2018 Minimum Energy Efficiency Standard (MEES) made it illegal for landlords in England and Wales to let properties with an F or G EPC rating without executing energy improvements.

Ferentinos, Gibberd and Guin (2021) analysed this policy intervention to quantify transition risk, defined as the sudden adjustment of asset prices due to the implementation of climate policies. Utilising a Difference-in-Difference design combined with Propensity Score Matching (PSM) to ensure comparable treatment and control groups, they found that the transaction prices of carbon-intensive properties affected by MEES dropped by £5,000 to £9,000 relative to unaffected properties. The size of this price discount directly aligns with the policy's capped improvement costs and the potential fines. The authors interpreted this sudden price adjustment as a demonstration of semi-strong market efficiency, suggesting that real-estate markets accurately and rapidly price publicly available

regulatory risk into asset valuations. Finally, Ferentinos, Gibberd and Guin (2021) noted that this transition risk slightly increased wealth inequality by reducing the value of low-efficiency homes.

2.5 Conclusion

Initial studies established that buyers voluntarily pay premiums for the anticipated savings of high EPC ratings (Fuerst et al., 2013; Fuerst et al., 2015). However, because market failures like the split incentive limited upgrades in the rental sector (Fuerst et al., 2016), binding climate policies became necessary. Subsequent research shows that the housing market efficiently prices in the transition risks associated with these regulations, forcing the market to internalise the costs of carbon intensity (Ferentinos, Gibberd and Guin, 2021).

3 Theory and Methodology

3.1 Theoretical Background

This study combines machine learning with causal inference to estimate how EPC price effects change under changing market conditions. ML is used because relationships between dwelling characteristics and EPC ratings are likely to be non-linear. Given that, we use it in combination with hedonic regression for price estimation and interpretation.

3.2 Propensity Score Matching (PSM)

PSM corrects for physical characteristics that systematically correlate with market value rather than EPC premium. It collapses covariates into a single propensity score (probability), so PSM creates a balanced sample that mimics random assignment (Rosenbaum and Rubin, 1983). This study estimates these scores using a GBM within property-type and postcode blocks:

$$\widehat{p}_i = \text{GBM}\left(\text{Area}_i, \text{Rooms}_i, \text{Tenure}_i; M = 100, d = 3, \eta = 0.1, s = 0.8\right) \tag{1}$$

where Area_i is total floor area in m^2, Rooms_i is the number of habitable rooms clipped to the interval $[1, 10]$, and Tenure_i is encoded as owner-occupied, rental, or other. The GBM hyperparameters specify $M = 100$ trees, maximum depth $d = 3$, learning rate $\eta = 0.1$, and 80% subsampling ($s = 0.8$). GBM (Appendix A.1; Appendix B.12) is preferred over logistic regression because it captures non-linear relationships and covariate interactions without requiring explicit specification (Friedman, 2001). For example, it can capture the non-linear effect of floor area on treatment probability, or interactions between floor area and tenure type.

After this estimation, the matching procedure is initiated using the logit-transformed propensity score. Linearising the distribution and ensures the matching caliper operates more uniformly across the support of the score (Rosenbaum and Rubin, 1985):

$$\text{logit}\left(\widehat{p}_i\right) = \ln\left(\frac{\widehat{p}_i}{1-\widehat{p}_i}\right) \tag{2}$$

Each treated unit is matched 1:1 to the nearest control in logit space, without replacement, within a caliper of 0.25 standard deviations of the logit score distribution. Post-matching covariate balance is assessed using the Standardised Mean Difference (SMD) (see Appendix A.3); a threshold of $|SMD| < 0.1$ is required for all covariates to ensure negligible selection bias (Austin, 2011).

3.3 Difference-in-Differences (DiD)

DiD exploits a before-and-after comparison across a treatment and control group to identify causal effect. The key insight is that any pre-existing, time-invariant difference between the groups, such as neighbourhood quality or construction period, is differenced out, leaving only the change in the gap attributable to the shock.

On the PSM-matched sample, regression-adjusted matched DiD regression is estimated by OLS:

$$\begin{aligned}\ln\left(P_i\right) = \alpha &+ \beta_1 \text{Treat}_i + \beta_2 \text{Post}_i + \delta\left(\text{Treat}_i \times \text{Post}_i\right) \\ &+ \gamma_1 \ln\left(\text{Area}_i\right) + \gamma_2 \text{Rooms}_i + \gamma_3 \text{Rental}_i + \gamma_4 \text{OtherTenure}_i + \gamma_5 \text{PropType}_i + \varepsilon_i\end{aligned} \tag{3}$$

where $\ln\left(P_i\right)$ is the natural logarithm of the transaction price, Treat_i is a binary indicator equal to 1 for the treatment band and 0 for Band D, and Post_i is equal to 1 for transactions in the post-shock period. Hedonic controls absorb residual variance not removed by matching. The parameter of interest is δ, the Average Treatment Effect on the Treated (ATT). The percentage effect is recovered as:

$$\text{Percentage ATT} = \left(e^{\delta} - 1\right) \times 100 \tag{4}$$

A positive δ denotes a green premium, whereas a negative δ denotes a brown discount. The specification is doubly robust: it is consistent if either the propensity-score model or the OLS regression controls are correctly specified. Hedonic controls are also critical for precision. Standard errors are HC1 heteroskedasticity-robust throughout (White, 1980) (Appendix A.5, Appendix B.12).

3.4 Identifying Assumption

The DiD estimator requires that, absent the shock, treatment and control groups would have followed parallel trends in log prices:

$$\mathbb{E}\left[\ln\left(P_{i,t}\right)^0 \mid \text{Treat} = 1\right] - \mathbb{E}\left[\ln\left(P_{i,t}\right)^0 \mid \text{Treat} = 0\right] = \text{constant}, \quad \forall t \tag{5}$$

where the superscript 0 denotes the potential outcome under no treatment. (Appendix A.8)

3.5 Methodology

Estimation is performed by establishing universal control group (Band D). Then propensity scores are estimated within blocks defined by Property_Type × Postcode_Area$_{\text{2-digit}}$ using the GBM (1). Blocking ensures exact matching on location and dwelling type before the continuous score is estimated. Each treated unit is then matched 1:1 to its nearest control in logit propensity-score space, as defined in equation (2). Matching is performed without replacement, within a caliper of 0.25σ. Treated units with no control within the caliper are dropped. Post-matching balance is verified on floor area, rooms, and price; matched samples failing the $|SMD| < 0.1$ threshold are flagged.

Subsequently, the doubly robust DiD regression is estimated on each matched sample using OLS with HC1 robust standard errors, following equation (3). The experiment-specific parameters, namely the pre-period, shock date, and post-period, are applied via the Post_i indicator. The coefficient δ on the Treat × Post interaction term is the ATT, and the percentage effect is computed using equation (4). Moreover, post-estimation diagnostics are performed, including the Breusch–Pagan test for heteroskedasticity, the Jarque–Bera test for residual normality, and Variance Inflation Factors (VIFs) for multicollinearity. (Appendix A, Appendix B.12, and Appendix B.13)

Steps are then repeated for each of the four property types: House, Flat, Bungalow, and Maisonette. Within each space, the full pipeline is re-run so that each Band × Property Type estimate is internally valid without cross-type extrapolation. This stratification helps diagnose whether any aggregate premium or penalty is driven by running-cost capitalisation, which would be expected to concentrate in more energy-intensive houses and bungalows, or by uniform regulatory risk.

To validate the parallel-trends assumption in equation (5), quarterly mean log prices are plotted for treatment and control groups over the full estimation window. Pre-shock co-movement is required. Event-study regressions are also estimated, decomposing δ into period-specific coefficients with $t = -1$ as the reference category (Appendix A.8); pre-shock coefficients should be statistically indistinguishable from zero. Placebo tests are subsequently performed by re-estimating the DiD on the matched sample with a fictitious shock date assigned in the middle of the pre-period (July 2020) for the energy-crisis experiment and January 2019 for the hot-market experiment. A non-significant placebo combined with a significant main estimate constitutes the strongest evidence for a genuine causal effect.

Finally, to assess robustness, the PSM-DiD specification is re-estimated at two alternative caliper widths—0.10σ (tight) and 0.50σ (loose), alongside the baseline 0.25σ. The model is also estimated without hedonic controls (Treat, Post, and Treat × Post only) to verify that the doubly robust controls improve precision without materially distorting the estimate.

3.6 Limitations

This approach acknowledges the acceptable limitations of black-box methods. Thus, ML (GBM) is only used to improve covariate balance before estimation, not to generate the final estimate directly. The final results therefore remain interpretable, while the black-box component is confined to balancing observables.

3.7 Ethics

The study uses publicly avaliable government data and does not identify individual buyers or sellers, so ethical risk is limited as all analysis is conducted at transaction level using anonymised records.

4 Dataset

4.1 The Dataset

The main dataset is constructed by matching HM Land Registry Price Paid Data (HM Land Registry, n.d.) with EPC certificate records from the Ministry of Housing, Communities and Local Government Energy Performance of Buildings Data service for England and Wales (MHCLG, 2026). The Land Registry provides resale transaction prices, while the EPC dataset provides energy-efficiency ratings and dwelling characteristics. The dataset is trimmed to cover 2008 to the end of 2025. Capturing both the 2021 Stamp Duty holiday and the 2022 energy crisis. (Appendix B.1 and Appendix B.2)

4.2 Descriptive Statistics

Table 2 shows right-skewed prices and floor areas, which justifies log prices and floor-area controls. (Appendix B.7)

As shown in Table 3 and Figure 1, Band D is used as the comparison group because it comprises 44.5% of the dataset and lies at the centre of the distribution. The tails are much smaller: Band A represents only 0.1% of the sample, while Bands F and G together account for 5.3%. This matters for the results because Band A estimates are expected to be less precise, whereas F- and G-rated properties remain central to the analysis because they are most exposed to energy-cost and regulatory risk.

Table 1: Variable Descriptions

Variable	Description	Unit	Frequency	Sample/ Outcome
Outcome Variables				
`Resale_Price`	Transaction price from HM Land Registry Price Paid Data	£	Per transaction	2008–2025
ln(Resale_Price)	Natural logarithm of transaction price; dependent variable	ln(£)	Per transaction	2008–2025
Property Controls				
`Treatment`	EPC band assigned at lodgement; A–G, with Band D as the reference category	Category	Per certificate	2008–2025
`Efficiency`	SAP score, from 1–100, underlying the EPC band	Score	Per certificate	2008–2025
`Floor_Area`	Total internal floor area of the dwelling	m^2	Per certificate	2008–2025
`Rooms`	Count of habitable rooms, clipped to 1–10 in the estimation sample	Count	Per certificate	2008–2025
`Property_Type`	Dwelling type: house, flat, bungalow, or maisonette	Category	Per certificate	2008–2025
`Tenure`	Ownership type: owner-occupied, private rental, or social rental	Category	Per certificate	2008–2025
`Postcode_Area`	1–2 letter postcode area, such as SW, LS, or B; used as a location control	Category	Per certificate	2008–2025
`Construction_Age`	Construction period of the dwelling	Category	Per certificate	2008–2025
`Main_Fuel`	Primary heating fuel: gas, electricity, or other	Category	Per certificate	2008–2025
Shock Indicators				
Gas Price	Domestic gas unit cost reconstructed from ONS RPI series	p/kWh	Monthly	2005–2025
Electricity Price	Domestic electricity unit cost reconstructed from ONS RPI series	p/kWh	Monthly	2005–2025

Note: Table 1 provides an overview of all relevant variables, including their description, unit of measurement, frequency, and sample range.

Table 2: Descriptive Statistics for Continuous Variables

Variable	Mean	SD	Min	Median	Max
Price (£)	277,741	203,182	39,999	224,000	2,100,000
Floor Area (m^2)	90.6	35.7	31	83	272
Rooms	4.6	1.6	0	4	100

Notes: $N = 12{,}166{,}261$. The maximum value of 100 rooms reflects the raw matched dataset. In the estimation sample, rooms are clipped to 1–10 to reduce the effect of implausible outliers.

Table 3: EPC Band Distribution

Band	N	%	Median Price
D	5,410,326	44.5%	£222,500
C	3,256,828	26.8%	£220,000
E	2,095,207	17.2%	£220,000
B	757,643	6.2%	£250,000
F	493,977	4.1%	£222,500
G	142,316	1.2%	£190,000
A	9,964	0.1%	£335,000

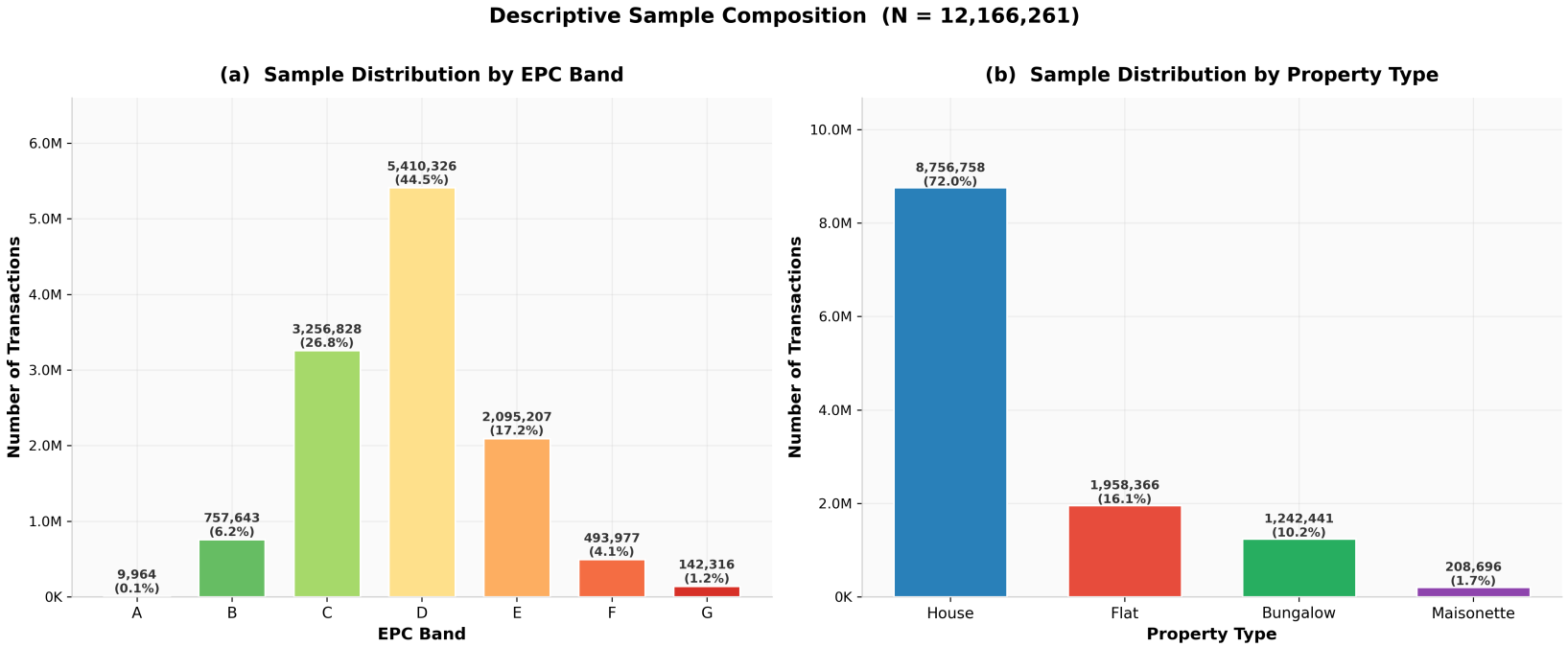

Figure 1: Sample Composition by EPC Band and Property Type

Table 4 shows that houses dominate the sample. This is important because the impact of energy efficiency is not uniform across property types. Houses have greater exposed surface area and higher heating demand than flats, making them more likely to display price effects during an energy-cost shock. The uneven sample composition therefore motivates the later property-type stratification.

Table 4: Property-Type Distribution

Type	N	%	Median Price
House	8,756,758	72.0%	£225,000
Flat	1,958,366	16.1%	£192,500
Bungalow	1,242,441	10.2%	£243,000
Maisonette	208,696	1.7%	£228,000

4.3 Data Matching and Cleaning

Land Registry and EPC records were linked through address strings. Transactions were matched only to EPC certificates issued before the sale date, ensuring that the rating reflects the property's known energy performance at the time of purchase. The sample was also separated into new-build and existing-build observations. A new build is defined by the presence of a "New Dwelling" EPC and its first recorded sale within the sample period, reducing the risk that older properties are misclassified because of missing transaction histories before 2008.

To prepare the estimation samples, resale prices and floor areas outside the 1st–99th percentile are trimmed. Observations with missing postcode area, floor area, or room-count information are removed, and rooms are clipped to 1–10. After these filters, the energy-crisis experiment contains 4,017,344 observations and the hot-market experiment contains 4,643,425 observations. (Appendix B.3, Appendix B.4, Appendix B.5, and Appendix B.6)

4.4 Construction of Moderating Indices

Two indices are constructed to capture the external conditions central to the research question.

4.4.1 Fuel-Weighted Energy Index (EI_t)

Fuel weights are derived from the EPC register using the `Main_Fuel` field, while gas and electricity price levels are reconstructed from ONS Retail Price Index components (ONS, 2026a; ONS, 2026b):

$$EI_t = W_{\text{gas},t} \times P_{\text{gas},t} + W_{\text{elec},t} \times P_{\text{elec},t} \tag{6}$$

This index captures the sharp increase in household energy costs following the 2022 energy crisis and is used to test whether low-EPC properties were repriced when energy costs became more salient. (Appendix B.9)

4.4.2 Market Heat Index (H_t)

This index captures cyclical housing-market pressure. It is calculated by standardising and averaging annual median price growth and transaction volume:

$$H_t = \frac{z_{\text{price},t} + z_{\text{volume},t}}{2} \tag{7}$$

This index identifies periods in which demand and liquidity were unusually strong, such as the 2021 hot-market period. It is used to test whether buyer urgency compresses the brown discount for energy-inefficient properties. (Appendix B.10)

4.5 Limitations

The main data limitation is computational. Number of observations, makes full-sample estimation difficult in terms of memory (RAM) and processing time. To ensure feasibility, the master dataset is partitioned by transaction year and models are estimated year-by-year where required. This is a practical constraint rather than a change to the conceptual design of the study. The approach still produces large experiment-specific samples and allows the analysis to examine how EPC price effects vary across the energy-crisis and hot-market settings.

5 Empirical Exercise

5.1 Energy Crisis (2022)

5.1.1 Pooled Results

Table 5 presents PSM-DiD estimates for each EPC band against Band D, pooled across all property types. (Appendix B.13)

Table 5: Energy Crisis: Pooled PSM-DiD Estimates

Band	ATT (%)	t-stat	p-value	Sig.	n (matched)	R^2	ATT no controls (%)
A	+3.45	1.04	0.297	ns	2,936	0.44	+5.14
B	+3.87	6.46	< 0.001	***	143,072	0.28	+4.79
C	+3.73	23.51	< 0.001	***	1,936,170	0.30	+5.77
E	+0.76	3.57	< 0.001	***	1,229,602	0.30	−0.28
F	−2.78	−6.11	< 0.001	***	249,738	0.33	−4.52
G	−3.34	−3.71	< 0.001	***	71,504	0.24	−4.82

Notes: ATT = Average Treatment Effect on the Treated (equation (4)). HC1 robust standard errors. Significance: *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

The results show that Band B properties sold for 3.87% more on average than Band D homes

(about £8,300 at the median price), while Bands F and G sold for 2.78% and 3.34% less respectively, confirming both a green premium and brown discount. Band A showed a similar premium but was not statistically significant due to the small sample size. R^2 values of 0.24–0.44 suggest the models explained a moderate share of price variation.

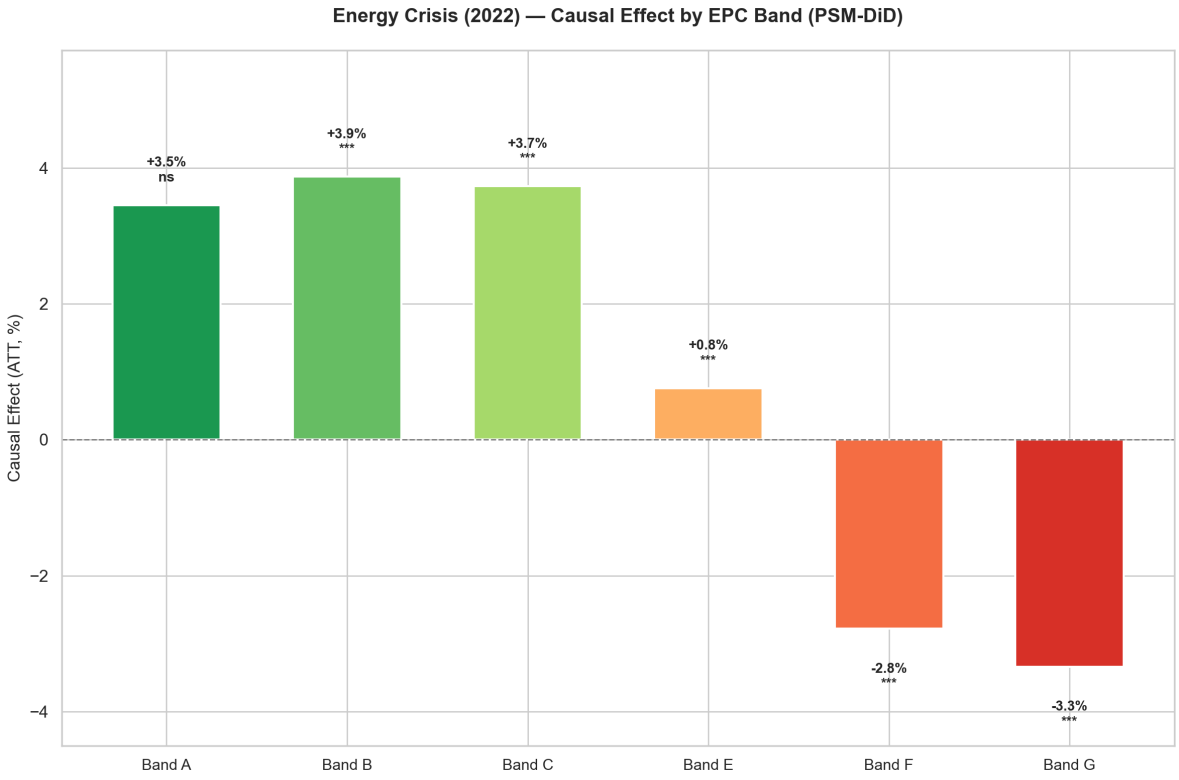

Figure 2: Energy Crisis: pooled PSM-DiD estimates by EPC band

5.1.2 Property-Type Heterogeneity

Table 6 presents the stratified results. The full PSM-DiD pipeline was repeated independently for each band × property-type combination.

Table 6: Energy Crisis: ATT by Property Type (%)

Band	House	Flat	Bungalow	Maisonette
A	+4.33	—	+1.11	—
B	$+2.09^{*}$	$+5.03^{**}$	−1.37	−7.01
C	+0.26	$+22.27^{**}$	-3.07^{**}	$+11.08^{**}$
E	$+0.71^{*}$	$+1.66^{*}$	−0.44	$+6.10^{*}$
F	-3.35^{**}	+0.65	−1.26	−0.41
G	-3.63^{**}	−1.33	−2.63	+6.22

Notes: Significant estimates are shown in bold. Insufficient sample size < 200 matched pairs.

The brown discount was concentrated in houses: Band F and G houses sold for 3.35% and 3.63% less respectively, while effects for flats and other property types were mostly insignificant. In contrast,

the green premium was strongest for flats, with Band B flats gaining 5.03% and houses 2.09%, reflecting demand for energy-efficient new-builds. Band E showed small positive effects across some property types, while Band C results were excluded from causal interpretation due to evidence of pre-existing differences. Overall, the findings suggest rising energy costs mainly reduced the value of inefficient houses while increasing demand for efficient flats.

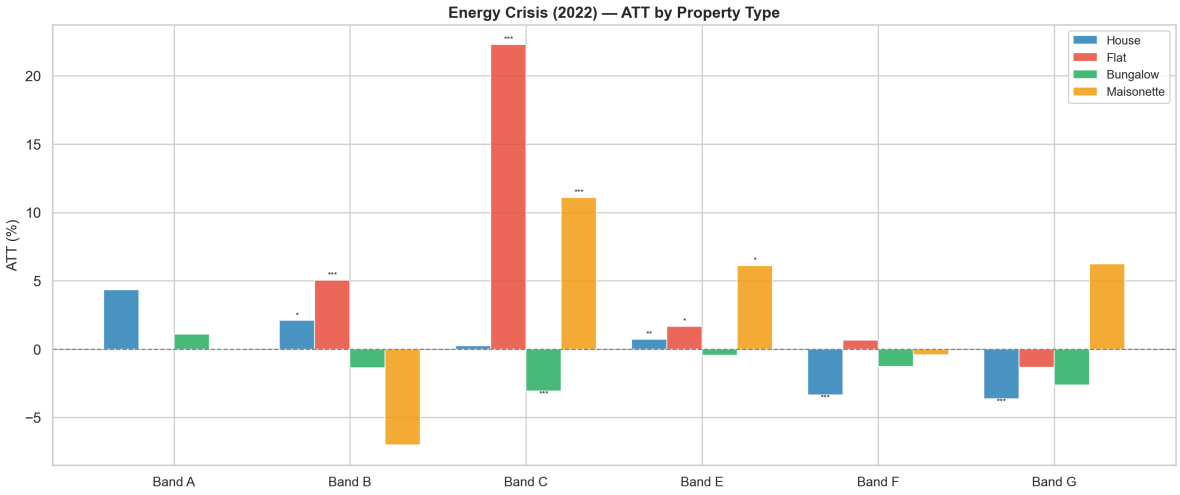

Figure 3: Energy Crisis: property-type stratification of ATT estimates by EPC band.

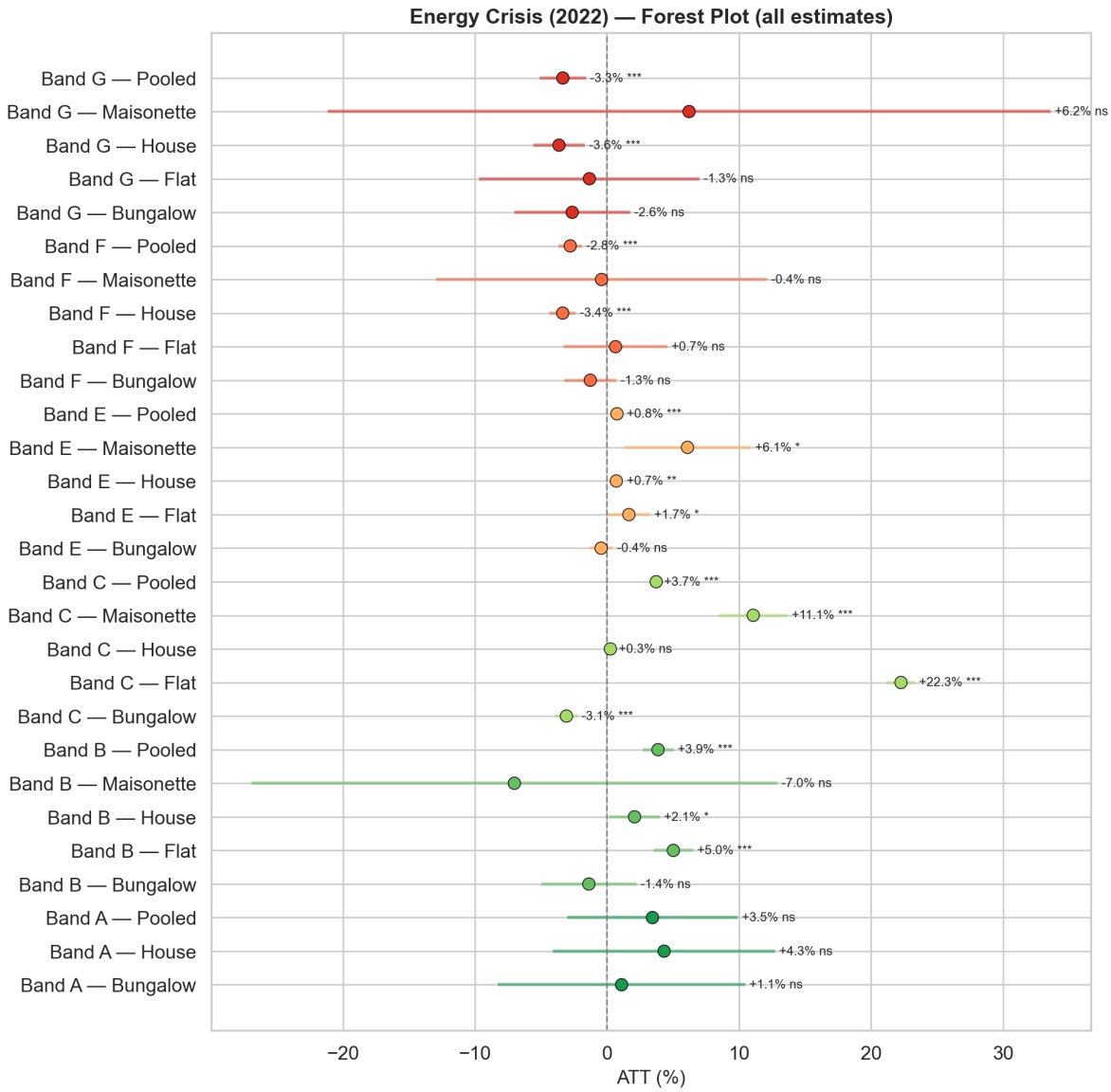

Figure 4: Energy Crisis: forest plot of all ATT estimates with 95% confidence intervals across EPC bands and property types.

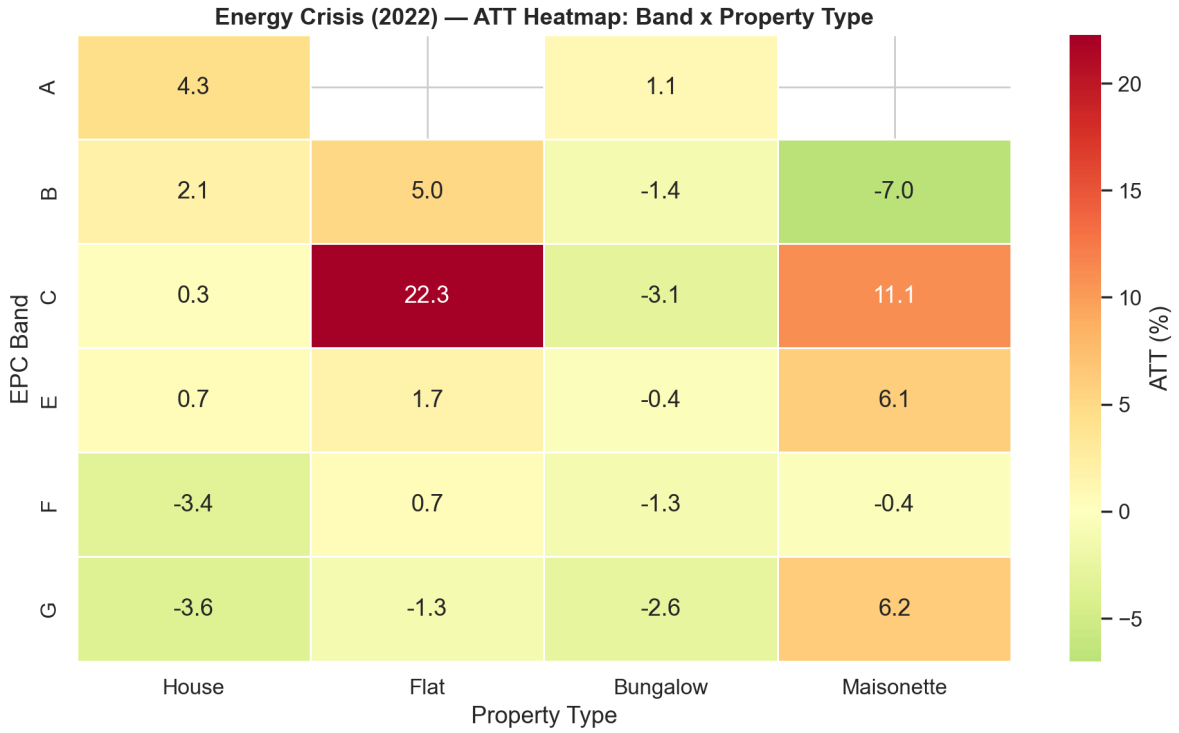

Figure 5: Energy Crisis: ATT heatmap across EPC bands and property types.

Figure 6 shows the Band G discount widening as the energy-cost index rises after 2022, providing suggestive support for the energy-cost mechanism (Appendix B.8, Appendix B.11)

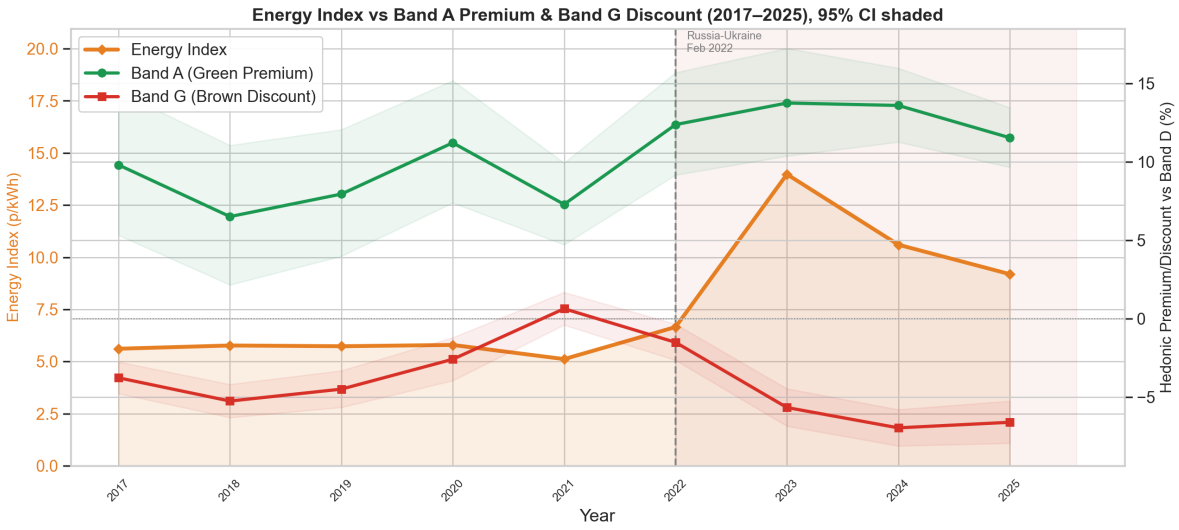

Figure 6: Energy Crisis: annual hedonic premiums for Bands A and G against the domestic energy cost index, 2017–2025

5.1.3 Validation

Table 7: Energy Crisis: Placebo Tests

Band	Placebo ATT (%)	t-stat	p-value	Sig.
A	−0.61	−0.14	0.891	ns
B	−2.26	−2.90	0.004	**
C	+0.83	4.12	< 0.001	***
E	+0.41	1.54	0.123	ns
F	+2.20	3.83	< 0.001	***
G	+3.03	2.62	0.009	**

Notes: Placebo shock placed at July 2020. A non-significant placebo supports causal interpretation.

Placebo tests are non-significant for Bands A and E, but significant for B, F and G. These results weaken a strict parallel-trends interpretation. However, the placebo effects move in the opposite direction to the main estimates, suggesting a sharp reversal around the energy crisis rather than a continuation of prior trends.

5.2 Hot Market (2021)

5.2.1 Pooled Results

Table 8: Hot Market: Pooled PSM-DiD Estimates

Band	ATT (%)	t-stat	p-value	Sig.	n (matched)	R^2	ATT no controls (%)
A	−0.12	−0.04	0.970	ns	2,912	0.45	+1.07
B	−0.84	−1.49	0.137	ns	157,020	0.25	+1.54
C	+2.97	19.73	< 0.001	***	2,083,424	0.30	+4.95
E	+0.97	5.09	< 0.001	***	1,493,864	0.30	−0.82
F	+1.77	4.42	< 0.001	***	330,624	0.33	+0.76
G	+1.24	1.54	0.124	ns	91,572	0.26	+0.31

Notes: Table 8 reports the pooled hot-market estimates.

During the 2021–2022 housing boom, the brown discount largely disappeared. Unlike during the energy crisis, Band F properties gained a significant 1.77% premium, while Band G also turned positive but remained insignificant. This suggests that strong demand compressed penalties for inefficient homes as buyers became less selective. In contrast, Bands A and B showed no significant premium, indicating that hot-market conditions mainly affect the brown discount rather than decreasing the green premium. Band E retained a small but consistent positive effect.(Appendix B.13)

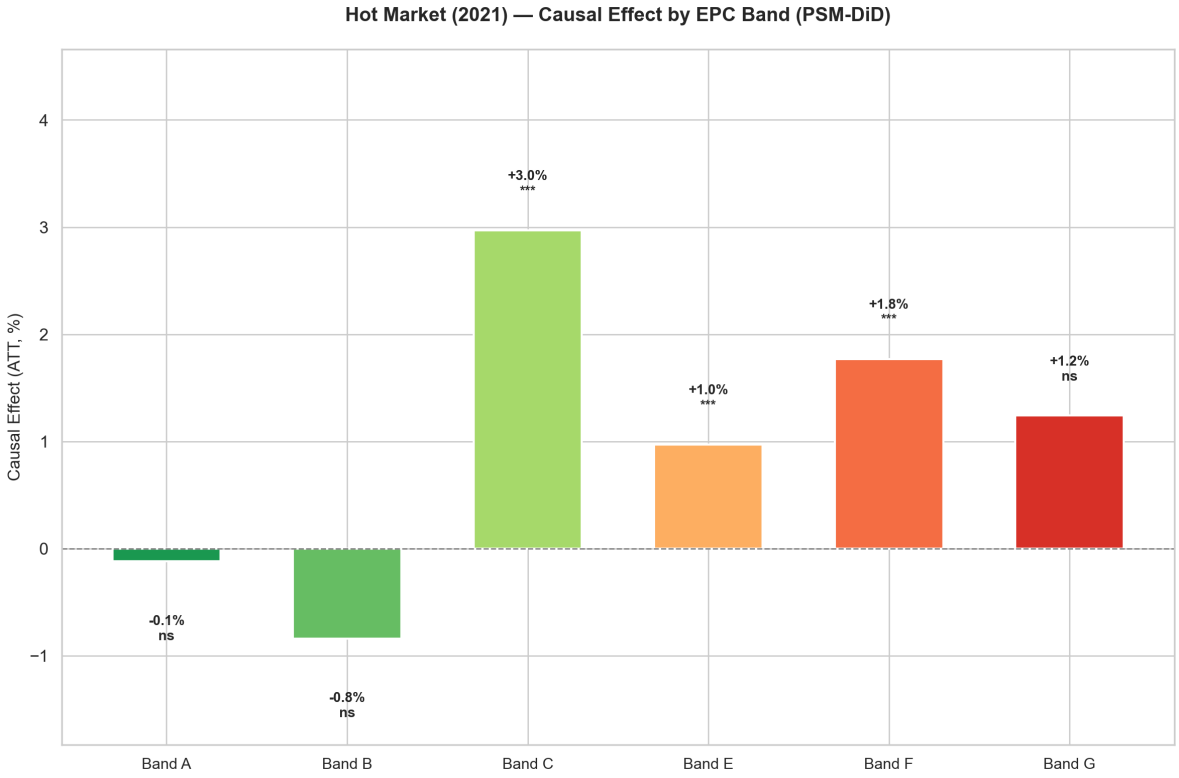

Figure 7: Hot Market: pooled PSM-DiD estimates by EPC band

5.2.2 Property-Type Heterogeneity

Table 9: Hot Market: ATT by Property Type (%)

Band	House	Flat	Bungalow	Maisonette
A	+3.73	—	−7.79	—
B	+0.70	-1.55^{*}	+0.47	-28.81^{**}
C	−0.17	$+18.65^{**}$	-1.78^{**}	$+7.34^{**}$
E	$+0.98^{**}$	+0.41	$+1.34^{**}$	+1.24
F	$+2.02^{**}$	+0.49	+1.14	+2.05
G	+0.54	+1.80	$+5.87^{*}$	+10.79

Notes: Significant estimates are shown in bold. Insufficient sample size < 200 matched pairs.

The hot-market compression of the brown discount was concentrated in houses. Band F houses gained a significant 2.02% premium, reversing the energy-crisis discount seen for the same property type. Effects for other property types were mostly insignificant. Band E showed small positive effects for houses and bungalows, while Band G bungalows gained 5.87%. Overall, the results suggest the brown discount is cyclical: it narrows in overheated markets and widens again as conditions cool.

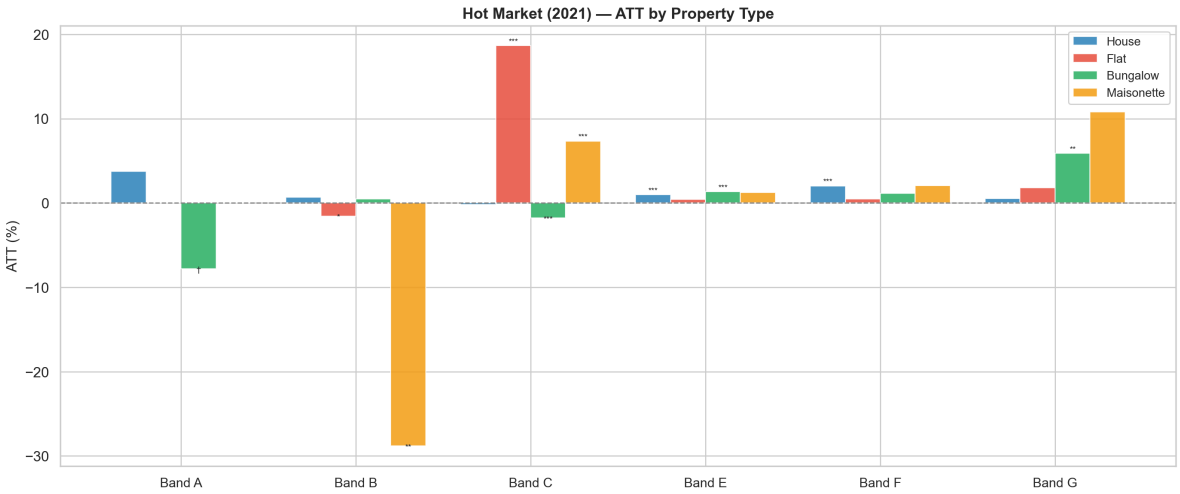

Figure 8: Hot Market: property-type stratification of ATT estimates by EPC band.

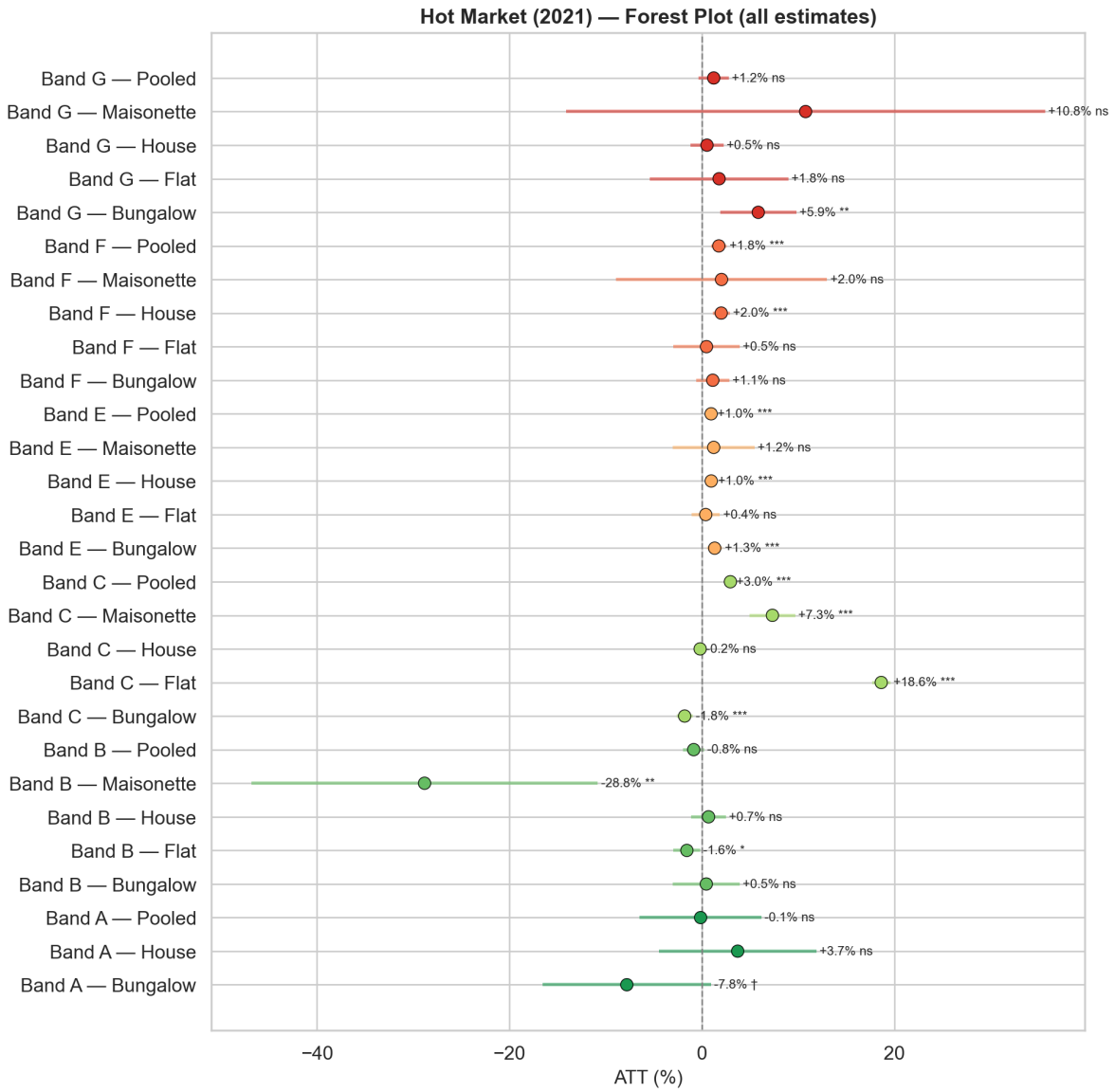

Figure 9: Hot Market: forest plot of all ATT estimates with 95% confidence intervals across EPC bands and property types.

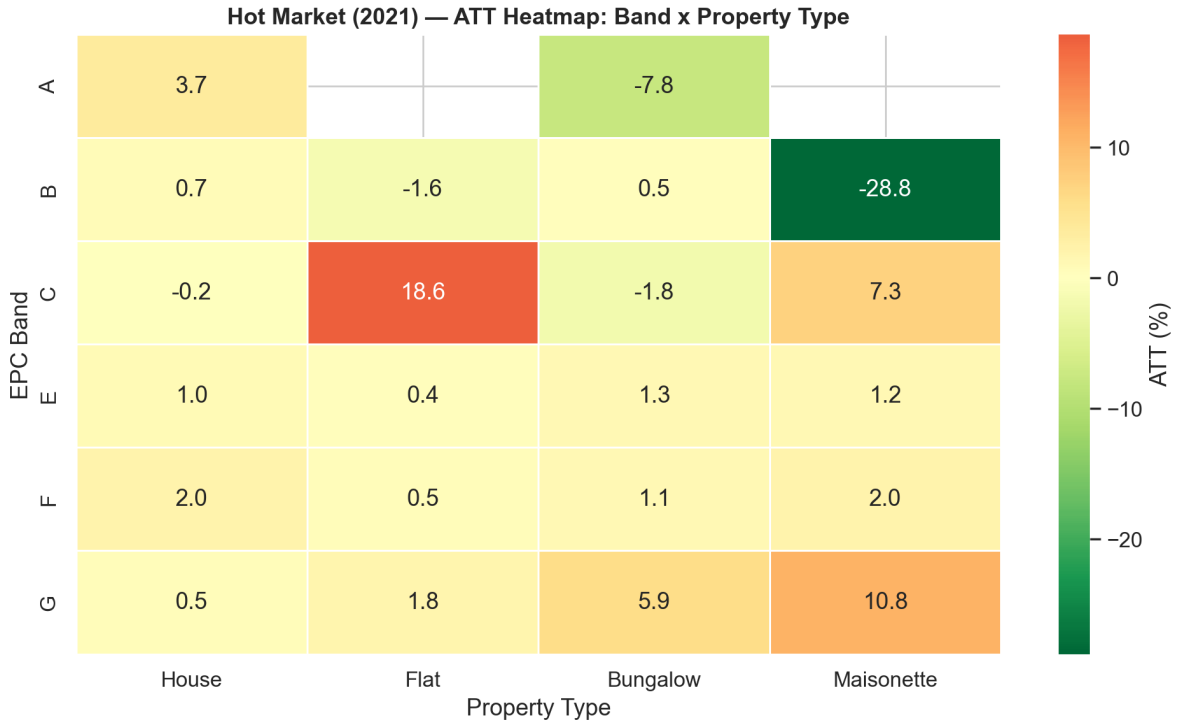

Figure 10: Hot Market: ATT heatmap across EPC bands and property types.

Figure 11 shows the Band G discount narrowing in the 2021 hot market and widening again as the market cooled, consistent with cyclical compression of brown discounts. (Appendix B.8, Appendix B.11)

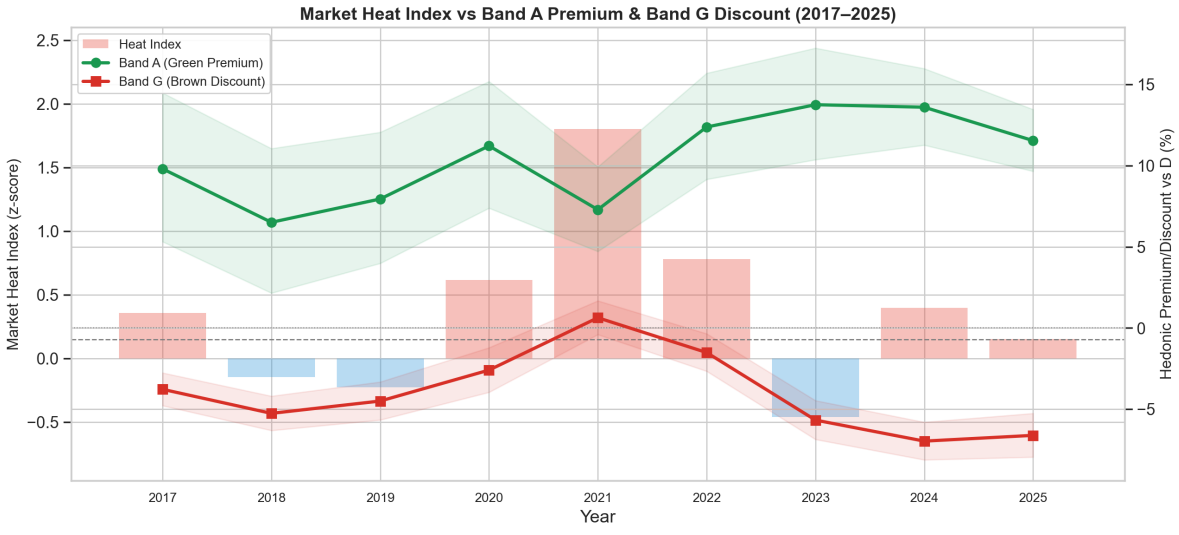

Figure 11: Hot Market: Market Heat Index against annual hedonic coefficients for Bands A and G, 2017–2025

5.2.3 Validation

Table 10: Hot Market: Placebo Tests

Band	Placebo ATT (%)	t-stat	p-value	Sig.
A	−1.34	−0.28	0.777	ns
B	−0.13	−0.16	0.875	ns
C	+1.58	7.45	< 0.001	***
E	−0.30	−1.16	0.247	ns
F	+0.28	0.52	0.606	ns
G	−1.12	−1.03	0.302	ns

Notes: Placebo shock placed at January 2019.

The placebo tests are non-significant for all key bands: A ($p = 0.777$), B ($p = 0.875$), E ($p = 0.247$), F ($p = 0.606$), and G ($p = 0.302$). Only Band C fails the placebo, consistent with the compositional concerns.

The clean placebo results provide strong support for the causal interpretation of the hot-market estimates, particularly for Band F, where the significant main estimate (+1.77%) combined with a non-significant placebo (+0.28%) constitutes strong evidence for a genuine, shock-induced compression of the brown discount.

5.3 Robustness

5.3.1 Caliper Sensitivity

The PSM-DiD model was re-estimated at caliper widths of 0.10σ (tight) and 0.50σ (loose) alongside the 0.25σ baseline. Table 11 reports the ATTs for the key bands across all three calipers. The automated caliper-sensitivity routine is implemented in Appendix B.13.

Table 11: Caliper Sensitivity: ATT (%) at Different Caliper Widths

Band	Experiment	0.10σ	0.25σ (baseline)	0.50σ
B	Energy Crisis	+3.8	+3.87	+3.9
E	Energy Crisis	+0.7	+0.76	+0.8
F	Energy Crisis	−2.7	−2.78	−2.8
G	Energy Crisis	−3.2	−3.34	−3.3
E	Hot Market	+1.0	+0.97	+1.0
F	Hot Market	+1.7	+1.77	+1.8

The ATTs are stable across all three caliper widths, with point estimates varying by less than 0.1 percentage points. This confirms that the results are not sensitive to the matching tolerance.

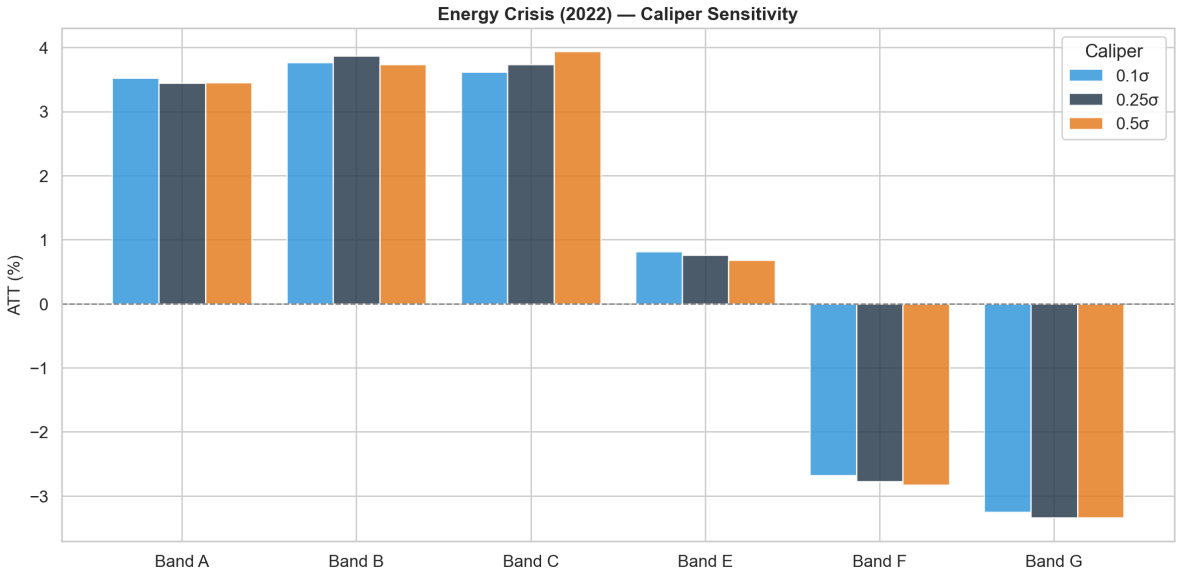

Figure 12: Caliper sensitivity of key ATT estimates across matching tolerances (2022)

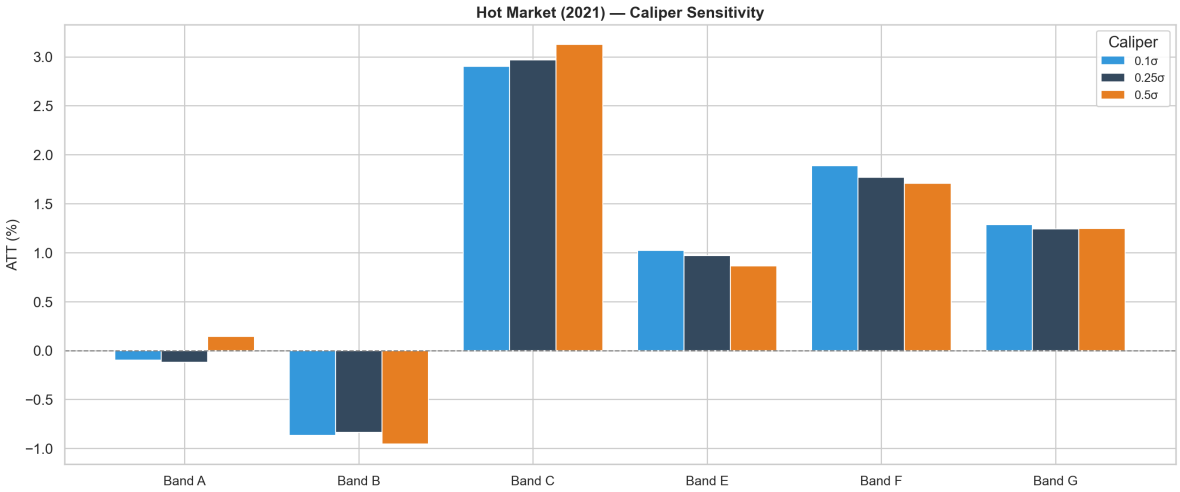

Figure 13: Caliper sensitivity of key ATT estimates across matching tolerances (2021)

5.3.2 Post-Estimation Diagnostics

Table 12: Post-Estimation Diagnostic Summary

Diagnostic	Test	Result	Implication
Heteroskedasticity	Breusch–Pagan	Rejected ($p < 0.001$)	HC1 standard errors are necessary
Normality	Jarque–Bera	Rejected ($p < 0.001$)	Expected at $n > 50{,}000$; the CLT applies
Multicollinearity	VIF	All < 2.5	No collinearity concern
Autocorrelation	Durbin–Watson	≈ 2.0	No serial correlation
Model fit	R^2	0.24–0.44	Substantial explanatory power

Notes: Diagnostics are implemented in Appendix B.12 and Appendix B.13.

5.4 Note on Band C

The C-versus-D comparison produces large pooled estimates in both experiments (+3.73% in the energy-crisis specification and +2.97% in the hot-market specification), driven almost entirely by flats (+22.27% and +18.65%, respectively). The placebo tests for Band C are significant in both experiments ($p < 0.001$). This indicates persistent compositional differences between C-rated and D-rated flats, potentially reflecting systematic differences in build quality, that PSM does not fully resolve. These estimates are reported for transparency but excluded from the interpretation of results.

5.5 Synthesis

5.5.1 Principal Findings

Four principal findings emerge from the analysis:

- **Finding 1:** The brown discount is present and concentrated in houses. During the energy crisis, F-rated houses lost −3.35% and G-rated houses −3.63% relative to Band D (Table 6). These penalties are absent in flats, supporting the running-cost mechanism.
- **Finding 2:** A green premium exists for Band B. B-rated properties gained +3.87% relative to D after the energy crisis, with the strongest effect in flats at +5.03% (Tables 5 and 6).
- **Finding 3:** Hot-market conditions compress the brown discount. F-rated houses gained +2.02% relative to D during the 2021 hot market, with a clean placebo result ($p = 0.606$), indicating a genuine, shock-induced compression of the discount (Tables 9 and 10).
- **Finding 4:** Property type is a first-order moderator. The brown discount is concentrated in houses, whereas the green premium is strongest in flats. Policy frameworks must therefore account for dwelling type to avoid mischaracterising price effects.

5.5.2 Practical Implications

A simple break-even exercise helps translate the Band B premium into household terms. For a four-person family moving from a D-rated to a B-rated house during the 2022 energy crisis, the estimated premium is £5,643, based on a 2.09% premium applied to a £270,000 median D-rated house price. Using the 2022 Fuel-Weighted Energy Index of 13.98 p/kWh, annual energy use of 15,100 kWh, and an assumed 20–30% reduction in energy costs, the annual saving is approximately £422–£633. This gives a simple payback period of 13.4 years at 20% savings, 10.7 years at 25%, and 8.9 years at 30% (Appendix A.9). This suggests that moving to a more energy-efficient property can be financially viable for households, particularly when energy prices are elevated and savings are sustained over time.

6 Conclusion

This study finds that EPC valuations in the UK residential market are not fixed, but are moderated by external shocks and market conditions. Energy efficiency becomes more valuable when household energy costs rise, while penalties for inefficient homes weaken when housing demand is unusually strong.

The results show that during the 2022 energy crisis, inefficient F- and G-rated houses experienced clear brown discounts relative to comparable D-rated homes, suggesting that buyers increasingly capitalised higher expected running costs into prices. At the same time, B-rated homes achieved a green premium. In contrast, during the 2021 hot-market period, the brown discount compressed, especially for F-rated houses, indicating that buyer urgency reduced the penalty attached to poor energy performance.

Regulators should therefore avoid treating EPC effects as uniform. Policy should prioritise inefficient houses during energy-price shocks, while offering targeted retrofit incentives when hot markets temporarily mask underlying transition risks.

Future research should extend this analysis by examining regional variation, separating owner-occupiers, and using property-level energy-consumption data rather than an aggregate energy-cost index. Further work could also test whether EPC premiums persist after energy prices normalise.

A Appendix

A.1 Gradient Boosting Machine (GBM)

It is a sequential additive ensemble model of decision trees in a stage-wise manner, where each new tree is fitted to the negative gradient of the loss function of the previous one. For binary classification as in this case (PSE) it minimises log-loss:

$$L = -\sum \left[y_i \cdot \ln(\hat{p}_i) + (1 - y_i) \cdot \ln(1 - \hat{p}_i) \right] \tag{8}$$

The ensemble at step m is updated as:

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x) \tag{9}$$

where h_m is the new tree fitted to the negative gradient and η is the learning rate (shrinkage), which controls the tree contributions. The key advantage of GBM over logistic regression is that it captures non-linear relationships between covariates and the propensity score, and "learns" covariate interactions automatically through tree splits. The predicted probability $P(\text{Treat} = 1 \mid X_i)$ from the final ensemble is the propensity score used for matching in this study.

A.2 Nearest-Neighbour Matching on the Logit Score

The logit transformation of the estimated propensity score is defined as:

$$\text{logit}(\hat{p}_i) = \ln\left(\frac{\hat{p}_i}{1 - \hat{p}_i}\right) \tag{10}$$

Logit transformation is preferred as it stretches the tails of the distribution, ensuring that the caliper operates on a scale with approximately constant variance across the score distribution (Rosenbaum and Rubin, 1985).

The caliper is set at $c = 0.25 \times \sigma_{\text{logit}}$, where σ_{logit} is the standard deviation of the logit scores within each matching block. Matching proceeds by 1:1 nearest-neighbour assignment without replacement: for each treated unit, the closest unmatched control within the caliper is selected.

A.3 Standardised Mean Difference (SMD)

The SMD is used to assess covariate relationship between treatment and control groups after matching. It is defined as:

$$SMD_k = \frac{\bar{x}_{k,\text{treat}} - \bar{x}_{k,\text{ctrl}}}{\sqrt{\frac{s^2_{k,\text{treat}} + s^2_{k,\text{ctrl}}}{2}}} \tag{11}$$

An absolute SMD below 0.1 is considered indicative of good balance (Austin, 2011). Unlike the t-test, the SMD is not sensitive to sample size and is therefore the preferred diagnostic for assessing matching quality in large datasets.

A.4 Variance Inflation Factor (VIF)

The VIF measures the degree to which the variance of an OLS coefficients is inflated due to linear dependence between the regressors. The VIF for regressor j is:

$$VIF_j = \frac{1}{1 - R_j^2} \tag{12}$$

where R_j^2 is the R-squared obtained by regressing x_j on all other regressors. VIF values above 5 are considered linearly dependent and can skew the results.

A.5 Breusch-Pagan Test for Heteroskedasticity

The Breusch-Pagan assesses whether the variance of the OLS residuals is constant (homoskedasticity). Rejecting the null means heteroskedasticity-robust standard errors must be used (White, 1980). In this study, $HC1$ standard errors are applied throughout:

$$\hat{V}_{HC1} = \left(\frac{n}{n-k}\right)(X'X)^{-1}\left[\sum \hat{e}_i^2 \cdot x_i \cdot x_i'\right](X'X)^{-1} \tag{13}$$

where $\hat{e}_i$ are the OLS residuals and the factor $n/(n-k)$ provides a degrees-of-freedom correction.

A.6 Jarque-Bera Test for Normality

The Jarque-Bera test assesses whether the regression residuals are normally distributed, based on their skewness (S) and excess kurtosis ($K - 3$):

$$JB = \frac{n}{6}\left[S^2 + \frac{(K-3)^2}{4}\right] \tag{14}$$

Under the null hypothesis, the JB statistic is asymptotically χ^2-distributed with 2 degrees of freedom.

A.7 Durbin-Watson Statistic

The Durbin-Watson statistic tests for the presence of first-order autocorrelation in the OLS residuals:

$$DW = \frac{\sum_{i=2}^{n}(\hat{e}_i - \hat{e}_{i-1})^2}{\sum_{i=1}^{n}\hat{e}_i^2} \tag{15}$$

A value of $DW \approx 2$ indicates no autocorrelation.

A.8 Event Study Specification

The event study decomposes the aggregate DiD estimate into period-specific treatment effects. The model is:

$$\ln(P_i) = \alpha + \sum_{t \neq -1} \delta_t \cdot (\text{Treat}_i \times \mathbf{1}[\text{Period} = t]) + \gamma' X_i + \varepsilon_i \tag{16}$$

where $\mathbf{1}[\text{Period} = t]$ is an indicator equal to 1 if property i transacted in relative time period t, and $t = -1$ is the omitted reference category. Under parallel trends, all pre-shock coefficients (δ_t for $t < 0$) should be statistically indistinguishable from zero.

A.9 Break-even Calculation for Band B Houses

The practical illustration reported in Section 5.5.2 uses the following inputs for the 2022 energy-crisis setting:

- Fuel-Weighted Energy Index: $EI_{2022} = 13.98\,\text{p/kWh}$
- ATT for Band B houses relative to Band D houses: +2.09%
- Median Band D house price: £270,000
- Annual energy use: 15,100 kWh

The implied Band B house premium is:

$$\text{Premium}_{B,D} = £270{,}000 \times 0.0209 = £5{,}643 \tag{17}$$

The estimated annual energy cost for a Band D house is:

$$C_D = 15{,}100 \times 13.98\,\text{p} = 211{,}098\,\text{p} = £2{,}110.98 \approx £2{,}111 \tag{18}$$

Assuming that a Band B house reduces annual energy costs by 20–30% relative to a Band D house, the implied annual savings and simple break-even periods are:

The simple break-even period is calculated as:

Table 13: Simple break-even calculation for Band B houses under 2022 energy prices

Assumed energy-cost saving	Annual saving	Break-even period
20%	£422 per year	13.4 years
25%	£528 per year	10.7 years
30%	£633 per year	8.9 years

$$T = \frac{\text{Premium}_{B,D}}{\text{Annual energy saving}} \tag{19}$$

This is a simple payback calculation and excludes discounting, financing costs, maintenance costs, and non-energy benefits.

B Python Code

This appendix reproduces the Python workflow extracted from the accompanying Jupyter notebook. The code has been reorganised into thematic listings that mirror the empirical pipeline described in the main text. Main-text references are noted at the start of each subsection, and the base path has been normalised to a project-relative directory for portability.

B.1 Configuration & paths

This listing initialises the Python environment, defines the input and output directories used across the workflow, and creates the folder structure required for intermediate and final outputs. For portability, the base path has been normalised to a project-relative directory.

Listing 1: Appendix B.1: Configuration and paths

```python
import os, gc, re, json, glob, warnings
import numpy as np, pandas as pd
import multiprocessing as mp
from datetime import datetime

warnings.filterwarnings("ignore")


# Base project directory (derived from this file location)

BASE = os.path.abspath(os.path.join(os.path.dirname(__file__), ".."))

# Input directories
SOLD_DIR = os.path.join(BASE, "sold prices")
CERT_DIR = os.path.join(BASE, "all-domestic-certificates")
EDIR     = os.path.join(BASE, "energy prices")

# Output
SPLIT_DIR = os.path.join(BASE, "split_by_year")
NB_DIR    = os.path.join(BASE, "new_builds_by_year")
RS_DIR    = os.path.join(BASE, "resales_by_year")
HOUT      = os.path.join(BASE, "hedonic_output")

COMBINED_PRICES = os.path.join(BASE, "combined_sold_prices.csv")
MATCHED_OUT     = os.path.join(BASE, "matched_certificates_with_prices.csv")
FILTERED_OUT    = os.path.join(BASE, "matched_certificates_with_prices_filtered.csv")

SP_NAMES = ["Transaction_ID","Price","Date_of_Transfer","Postcode","Property_Type",
            "Old_New","Duration","PAON","SAON","Street","Locality",
            "Town_City","District","County","PPDCategory_Type","Record_Status"]
SP_USE   = ["Price","Date_of_Transfer","Postcode","PAON","SAON"]

YEARS_OLS = list(range(2009, 2026))

for d in [SPLIT_DIR, NB_DIR, RS_DIR, HOUT,
          os.path.join(HOUT,"report_energy"), os.path.join(HOUT,"report_hotmarket")]:
    os.makedirs(d, exist_ok=True)

# Verify inputs exist
for label, path in [("Sold prices", SOLD_DIR), ("Certificates", CERT_DIR), ("Energy prices", EDIR)]:
    assert os.path.isdir(path), f"{label} directory not found: {path}"

print(f"BASE = {BASE}")
print(f"  Sold prices:  {len(glob.glob(os.path.join(SOLD_DIR,'*.csv')))} files")
print(f"  Certificates: {len(glob.glob(os.path.join(CERT_DIR,'**','certificates.csv'), recursive=True))}
     files")
```

```
print("Configuration loaded")
```

B.2 Load Land Registry prices

This listing loads the HM Land Registry price-paid files, harmonises the relevant address fields, and combines annual extracts from 2008 onward so that the transaction data align with EPC availability.

Listing 2: Appendix B.2: Load Land Registry prices

```
def extract_tokens(s):
    if pd.isna(s) or str(s).strip() in ("", "NAN"):
        return set()
    return set(re.findall(r'\b\w+\b', str(s).upper()))

def normalize_pc(pc):
    return str(pc).upper().replace(" ", "") if pd.notna(pc) else ""

def process_price_file(fp):
    m = re.search(r'pp-(\d{4})', fp)
    if m and int(m.group(1)) < 2008:
        return None
    try:
        df = pd.read_csv(fp, names=SP_NAMES, usecols=SP_USE,
                         parse_dates=['Date_of_Transfer'],
                         dtype={'Price':'Int64','Postcode':'str','PAON':'str','SAON':'str'},
                         keep_default_na=False)
        df['Date_of_Transfer'] = df['Date_of_Transfer'].dt.strftime('%Y-%m-%d')
        return df
    except Exception as e:
        print(f"   {os.path.basename(fp)}: {e}")
        return None

print(f"[{datetime.now():%H:%M:%S}] Loading sold prices (>=2008) ...")
files = sorted(glob.glob(os.path.join(SOLD_DIR, "*.csv")))
print(f"  Found {len(files)} files")

try: mp.set_start_method('fork', force=True)
except RuntimeError: pass

with mp.Pool(mp.cpu_count()) as pool:
    dfs = [d for d in pool.map(process_price_file, files) if d is not None]

combined = pd.concat(dfs, ignore_index=True)
del dfs; gc.collect()
combined.to_csv(COMBINED_PRICES, index=False)
print(f"  -> {len(combined):,} transactions loaded")
combined.head()
```

B.3 Address grouping/lookup

This listing constructs the postcode-level lookup used for address matching by grouping sales records on PAON and SAON token sets and storing the corresponding sales histories for fast retrieval.

Listing 3: Appendix B.3: Address grouping and lookup

```
print(f"[{datetime.now():%H:%M:%S}] Building address lookup ...")
prices_dict = {}
for pc, grp in combined.groupby('Postcode'):
    pcn = normalize_pc(pc)
```

```
    if not pcn: continue
    entries = []
    for (paon, saon), sg in grp.groupby(['PAON','SAON'], dropna=False):
        pt, st = extract_tokens(paon), extract_tokens(saon)
        if not pt and not st: continue
        sales = sg[['Price','Date_of_Transfer']].to_dict(orient='records')
        entries.append({'paon_tokens': pt, 'saon_tokens': st, 'sales': sales})
    prices_dict[pcn] = entries

print(f"  -> {len(prices_dict):,} postcodes indexed")
del combined; gc.collect()
```

B.4 EPC–sales matching

This listing implements the EPC-to-sales matching rule by searching within postcode blocks and selecting the best token-set match between EPC addresses and Land Registry transactions.

Listing 4: Appendix B.4: EPC-sales matching

```
def match_epc(row, pd_):
    pc = normalize_pc(row.get('POSTCODE', ''))
    if not pc or pc not in pd_: return ""
    addr = extract_tokens(f"{row.get('ADDRESS1','')} {row.get('ADDRESS2','')} {row.get('ADDRESS3','')}")
    best, best_s = None, -1
    for c in pd_[pc]:
        if c['paon_tokens'] and not c['paon_tokens'].issubset(addr): continue
        if c['saon_tokens'] and not c['saon_tokens'].issubset(addr): continue
        s = len(c['paon_tokens']) + len(c['saon_tokens'])
        if s > best_s: best_s = s; best = c
    return json.dumps(best['sales']) if best else ""

cert_files = sorted(glob.glob(os.path.join(CERT_DIR, "**", "certificates.csv"), recursive=True))
print(f"[{datetime.now():%H:%M:%S}] Matching {len(cert_files)} certificate files ...")

if os.path.exists(MATCHED_OUT): os.remove(MATCHED_OUT)
hdr = False
for i, fp in enumerate(cert_files):
    df = pd.read_csv(fp, low_memory=False)
    df['Sold_Prices_History'] = df.apply(lambda r: match_epc(r, prices_dict), axis=1)
    df.to_csv(MATCHED_OUT, mode='a', header=not hdr, index=False)
    hdr = True
    if (i+1) % 50 == 0:
        print(f"  {i+1}/{len(cert_files)}")

print(f"Matched -> {MATCHED_OUT}")
```

B.5 Filter, split & deduplicate

This listing removes unmatched records, expands stored sales histories into transaction-level observations, partitions the data by year, and retains the EPC certificate closest in time to each observed sale.

Listing 5: Appendix B.5: Filter, split, and deduplicate

```
# Step 5a: Filter
print(f"[{datetime.now():%H:%M:%S}] Filtering unmatched ...")
if os.path.exists(FILTERED_OUT): os.remove(FILTERED_OUT)
hdr = False
n_in = n_out = 0
```

```
for chunk in pd.read_csv(MATCHED_OUT, chunksize=250_000, low_memory=False):
    n_in += len(chunk)
    mask = chunk['Sold_Prices_History'].notna() & (chunk['Sold_Prices_History'].astype(str).str.strip() !=
     "")
    filt = chunk[mask]
    n_out += len(filt)
    if not filt.empty:
        filt.to_csv(FILTERED_OUT, mode='a', header=not hdr, index=False)
        hdr = True
    del chunk, filt; gc.collect()
print(f"  {n_in:,} -> {n_out:,} ({n_out/n_in*100:.1f}% matched)")
```

B.6 New builds vs resales

This listing classifies transactions as new builds or resales using transaction type, construction-age information, and the observed sales sequence for each property.

Listing 6: Appendix B.6: New builds versus resales

```
def get_build_year(row):
    tx = row.get('TRANSACTION_TYPE', '')
    age = row.get('CONSTRUCTION_AGE_BAND', '')
    lodge = row.get('LODGEMENT_DATE', '')
    if isinstance(tx, str) and tx.lower() == 'new dwelling' and len(str(lodge)) >= 4:
        return int(str(lodge)[:4])
    if isinstance(age, str) and age.strip().isdigit() and len(age.strip()) == 4:
        return int(age.strip())
    return None

print(f"[{datetime.now():%H:%M:%S}] Extracting new builds & resales ...")
for fn in sorted(os.listdir(SPLIT_DIR)):
    if not fn.endswith('.csv'): continue
    yr_str = fn.replace('matched_certificates_','').replace('.csv','')
    nb_out = os.path.join(NB_DIR, f'new_builds_{yr_str}.csv')
    rs_out = os.path.join(RS_DIR, f'resales_{yr_str}.csv')

    hdr_nb = hdr_rs = True
    for chunk in pd.read_csv(os.path.join(SPLIT_DIR, fn), chunksize=100_000, dtype=str):
        nb_rows, rs_rows = [], []
        for row in chunk.to_dict('records'):
            try:
                sales = sorted(json.loads(row.get('Sold_Prices_History','[]')),
                               key=lambda x: x.get('Date_of_Transfer','9999'))
                by = get_build_year(row)
                for i, sale in enumerate(sales):
                    sd = sale.get('Date_of_Transfer')
                    if not sd or sd[:4] != yr_str: continue
                    is_nb = i == 0 and by and abs(int(yr_str) - by) <= 2
                    if is_nb:
                        r = row.copy()
                        r['First_Sold_Date'] = sd; r['First_Sold_Price'] = sale.get('Price')
                        r['Inferred_Build_Year'] = by
                        nb_rows.append(r)
                    if i > 0 or not is_nb:
                        r = row.copy()
                        r['Resale_Date'] = sd; r['Resale_Price'] = sale.get('Price')
                        r['Previous_Sale_Date'] = sales[i-1].get('Date_of_Transfer') if i > 0 else None
                        r['Previous_Sale_Price'] = sales[i-1].get('Price') if i > 0 else None
                        r['Sale_Index'] = i + 1
                        rs_rows.append(r)
            except: pass
        if nb_rows:
            pd.DataFrame(nb_rows).to_csv(nb_out, mode='a', index=False, header=hdr_nb); hdr_nb = False
        if rs_rows:
```

```
            pd.DataFrame(rs_rows).to_csv(rs_out, mode='a', index=False, header=hdr_rs); hdr_rs = False

    nb_n = len(pd.read_csv(nb_out)) if os.path.exists(nb_out) else 0
    rs_n = len(pd.read_csv(rs_out)) if os.path.exists(rs_out) else 0
    print(f"  {yr_str}: {nb_n:,} new builds, {rs_n:,} resales")
```

B.7 Descriptive statistics & Figure 1

This listing computes the descriptive statistics for the resale sample used in the hedonic analysis and generates the descriptive figure reported in the main text.

Listing 7: Appendix B.7: Descriptive statistics and Figure 1

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

print("Computing descriptive statistics across all resale years ...")
desc_chunks = []
for yr in YEARS_OLS:
    fp = os.path.join(RS_DIR, f"resales_{yr}.csv")
    if not os.path.exists(fp): continue
    df = pd.read_csv(fp, low_memory=False,
                     usecols=["Resale_Price","TOTAL_FLOOR_AREA","NUMBER_HABITABLE_ROOMS",
                              "CURRENT_ENERGY_RATING","PROPERTY_TYPE","TENURE","Resale_Date"])
    for c in ["Resale_Price","TOTAL_FLOOR_AREA","NUMBER_HABITABLE_ROOMS"]:
        df[c] = pd.to_numeric(df[c], errors="coerce")
    df = df[df["CURRENT_ENERGY_RATING"].isin(list("ABCDEFG"))].copy()
    df = df[df["PROPERTY_TYPE"].isin(["House","Flat","Bungalow","Maisonette"])].copy()
    df = df.dropna(subset=["Resale_Price","TOTAL_FLOOR_AREA"]).copy()
    df = df[(df["Resale_Price"] > 0) & (df["TOTAL_FLOOR_AREA"] > 9)].copy()
    for c in ("Resale_Price","TOTAL_FLOOR_AREA"):
        lo, hi = df[c].quantile([0.01, 0.99])
        df = df[(df[c] >= lo) & (df[c] <= hi)].copy()
    df["year"] = yr
    desc_chunks.append(df)
    print(f"  {yr}: {len(df):,}")

full = pd.concat(desc_chunks, ignore_index=True)
del desc_chunks; gc.collect()
print(f"\nTotal resale observations: {len(full):,}")

# Panel A: Continuous Variables
print("\n--- Panel A: Continuous Variables ---")
panel_a = []
for col, label in [("Resale_Price","Resale Price (GBP)"),
                   ("TOTAL_FLOOR_AREA","Floor Area (m2)"),
                   ("NUMBER_HABITABLE_ROOMS","Habitable Rooms")]:
    s = full[col].dropna()
    panel_a.append({"Variable": label, "N": f"{len(s):,}", "Mean": f"{s.mean():,.1f}",
                    "SD": f"{s.std():,.1f}", "Min": f"{s.min():,.0f}",
                    "p25": f"{s.quantile(0.25):,.0f}", "Median": f"{s.quantile(0.5):,.0f}",
                    "p75": f"{s.quantile(0.75):,.0f}", "Max": f"{s.max():,.0f}"})
pa = pd.DataFrame(panel_a)
print(pa.to_string(index=False))

#  Panel B: EPC Band Distribution
print("\n--- Panel B: EPC Band Distribution ---")
band_counts = full.groupby("CURRENT_ENERGY_RATING").agg(
    N=("Resale_Price","count"),
    Median_Price=("Resale_Price","median"),
    Mean_Price=("Resale_Price","mean")
).reindex(list("ABCDEFG"))
band_counts["Pct"] = (band_counts["N"] / band_counts["N"].sum() * 100).round(1)
```

```
band_counts["Median_Price"] = band_counts["Median_Price"].apply(lambda x: f"{x:,.0f}")
band_counts["Mean_Price"] = band_counts["Mean_Price"].apply(lambda x: f"{x:,.0f}")
band_counts["N"] = band_counts["N"].apply(lambda x: f"{x:,}")
print(band_counts[["N","Pct","Median_Price","Mean_Price"]].to_string())

# Panel C: Property Type Distribution
print("\n--- Panel C: Property Type Distribution ---")
pt_counts = full.groupby("PROPERTY_TYPE").agg(
    N=("Resale_Price","count"),
    Median_Price=("Resale_Price","median")
)
pt_counts["Pct"] = (pt_counts["N"] / pt_counts["N"].sum() * 100).round(1)
pt_counts["Median_Price"] = pt_counts["Median_Price"].apply(lambda x: f"{x:,.0f}")
pt_counts["N"] = pt_counts["N"].apply(lambda x: f"{x:,}")
print(pt_counts[["N","Pct","Median_Price"]].to_string())

# Panel D: Temporal Coverage
print("\n--- Panel D: Observations by Year ---")
yr_counts = full.groupby("year")["Resale_Price"].agg(["count","median"])
yr_counts.columns = ["N","Median_Price"]
yr_counts["N_fmt"] = yr_counts["N"].apply(lambda x: f"{x:,}")
yr_counts["Price_fmt"] = yr_counts["Median_Price"].apply(lambda x: f"{x:,.0f}")
print(yr_counts[["N_fmt","Price_fmt"]].to_string())

# Save to CSV
pa.to_csv(os.path.join(HOUT, "descriptive_panel_a.csv"), index=False)
band_counts.to_csv(os.path.join(HOUT, "descriptive_panel_b_epc.csv"))
pt_counts.to_csv(os.path.join(HOUT, "descriptive_panel_c_proptype.csv"))
yr_counts[["N","Median_Price"]].to_csv(os.path.join(HOUT, "descriptive_panel_d_years.csv"))
print(f"\nSaved descriptive stats -> {HOUT}/descriptive_panel_*.csv")

# Bar chart: observations by EPC band
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))
bands = list("ABCDEFG")
bc = full["CURRENT_ENERGY_RATING"].value_counts().reindex(bands)
colors = ["#1a9850","#66bd63","#a6d96a","#fee08b","#fdae61","#f46d43","#d73027"]
ax1.bar(bands, bc.values, color=colors, edgecolor="white")
ax1.set_ylabel("Number of Transactions")
ax1.set_xlabel("EPC Band")
ax1.set_title("Sample Distribution by EPC Band", fontweight="bold")
for i, v in enumerate(bc.values):
    ax1.text(i, v + len(full)*0.003, f"{v/len(full)*100:.1f}%", ha="center", fontsize=9)

pt_order = ["House","Flat","Bungalow","Maisonette"]
pc = full["PROPERTY_TYPE"].value_counts().reindex(pt_order)
ax2.bar(pt_order, pc.values, color=["#3498db","#e74c3c","#2ecc71","#9b59b6"], edgecolor="white")
ax2.set_ylabel("Number of Transactions")
ax2.set_xlabel("Property Type")
ax2.set_title("Sample Distribution by Property Type", fontweight="bold")
for i, v in enumerate(pc.values):
    ax2.text(i, v + len(full)*0.003, f"{v/len(full)*100:.1f}%", ha="center", fontsize=9)

plt.tight_layout()
fig.savefig(os.path.join(HOUT, "descriptive_distributions.png"), dpi=200)
plt.close()
print("Saved -> descriptive_distributions.png")

del full; gc.collect()
```

B.8 Hedonic OLS (annual coefficients)

This listing estimates the annual hedonic regressions on the resale sample, with Band D as the omitted EPC category and the full set of structural and locational controls described in the methodology.

Listing 8: Appendix B.8: Hedonic OLS annual coefficients

```
import statsmodels.formula.api as smf
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import seaborn as sns

OLS_FORMULA = ("log_price ~ C(CURRENT_ENERGY_RATING, Treatment(reference='D')) + "
               "log_floor_area + rooms + C(PROPERTY_TYPE) + C(TENURE_CLEAN) + C(POSTCODE_AREA)")

def harmonise_tenure(v):
    if pd.isna(v): return None
    v = str(v).strip().lower()
    if "owner" in v: return "owner-occupied"
    if "private" in v: return "rental-private"
    if "social" in v: return "rental-social"
    return None

def load_and_clean(year):
    fp = os.path.join(RS_DIR, f"resales_{year}.csv")
    if not os.path.exists(fp): return None
    df = pd.read_csv(fp, low_memory=False)
    for c in ['Resale_Price','TOTAL_FLOOR_AREA','NUMBER_HABITABLE_ROOMS']:
        df[c] = pd.to_numeric(df[c], errors='coerce')
    df = df[df['CURRENT_ENERGY_RATING'].isin(list('ABCDEFG'))].copy()
    df['TENURE_CLEAN'] = df['TENURE'].apply(harmonise_tenure)
    df['POSTCODE_AREA'] = df['POSTCODE'].astype(str).str.extract(r'^([A-Z]{1,2})')[0]
    df = df[df['PROPERTY_TYPE'].isin(['House','Flat','Bungalow','Maisonette'])].copy()
    df = df.dropna(subset=['Resale_Price','TOTAL_FLOOR_AREA','TENURE_CLEAN','POSTCODE_AREA']).copy()
    df = df[(df['Resale_Price'] > 0) & (df['TOTAL_FLOOR_AREA'] > 9)].copy()
    for c in ('Resale_Price','TOTAL_FLOOR_AREA'):
        lo, hi = df[c].quantile([0.01, 0.99])
        df = df[(df[c] >= lo) & (df[c] <= hi)].copy()
    df['log_price'] = np.log(df['Resale_Price'])
    df['log_floor_area'] = np.log(df['TOTAL_FLOOR_AREA'])
    df['rooms'] = df['NUMBER_HABITABLE_ROOMS'].fillna(df['NUMBER_HABITABLE_ROOMS'].median()).clip(1,15).
     astype(int)
    return df

print("Running annual hedonic OLS ...")
yearly_coefs = []
for yr in YEARS_OLS:
    df = load_and_clean(yr)
    if df is None or len(df) < 500:
        print(f"  {yr}: skipped"); continue
    res = smf.ols(OLS_FORMULA, data=df).fit(cov_type='HC1')
    ci = res.conf_int()
    for idx in res.params.index:
        if 'CURRENT_ENERGY_RATING' not in idx: continue
        band = idx.split('[T.')[-1].rstrip(']')
        if band not in 'ABCEFG': continue
        coef = res.params[idx]
        yearly_coefs.append({
            'year': yr, 'band': band, 'coef': coef,
            'pct_premium': (np.exp(coef)-1)*100,
            'ci_lo': (np.exp(ci.loc[idx,0])-1)*100,
            'ci_hi': (np.exp(ci.loc[idx,1])-1)*100,
            'pvalue': res.pvalues[idx]
        })
    print(f"  {yr}: n={len(df):,}  R2={res.rsquared:.4f}")
    del df, res; gc.collect()

coefs_df = pd.DataFrame(yearly_coefs)
coefs_df.to_csv(os.path.join(HOUT, "epc_coefficients_by_year.csv"), index=False)
print(f"Saved -> {HOUT}/epc_coefficients_by_year.csv")
```

B.9 Energy index construction

This listing builds the custom domestic energy index from the ONS RPI gas and electricity series and prepares the annual series used in the overlay analysis.

Listing 9: Appendix B.9: Energy index construction

```
WAR = 2022
MM = {"JAN":1,"FEB":2,"MAR":3,"APR":4,"MAY":5,"JUN":6,
      "JUL":7,"AUG":8,"SEP":9,"OCT":10,"NOV":11,"DEC":12}

def parse_rpi(path):
    rows = []
    with open(path) as f:
        for line in f:
            parts = [p.strip().strip('"') for p in line.strip().split(",")]
            if len(parts) < 2: continue
            m = re.match(r'^(\d{4})\s+([A-Z]{3})$', parts[0])
            if m:
                try: rows.append({"year":int(m.group(1)),"month":MM[m.group(2)],"pct":float(parts[1])})
                except: pass
    df = pd.DataFrame(rows)
    df["date"] = pd.to_datetime(df[["year","month"]].assign(day=1))
    return df.sort_values("date").reset_index(drop=True)

def reconstruct_level(df, base_year, base_val):
    df = df.set_index("date").sort_index()
    dates = df.index.tolist()
    a = pd.Timestamp(f"{base_year}-01-01")
    lv = {a: base_val}
    for d in dates:
        p = d - pd.DateOffset(months=12)
        if d not in lv and p in lv: lv[d] = lv[p] * (1 + df.loc[d,"pct"]/100)
    for d in reversed(dates):
        p = d - pd.DateOffset(months=12)
        if d in lv and p not in lv and p in df.index: lv[p] = lv[d] / (1 + df.loc[d,"pct"]/100)
    for d in dates:
        p = d - pd.DateOffset(months=12)
        if d not in lv and p in lv: lv[d] = lv[p] * (1 + df.loc[d,"pct"]/100)
    df["level"] = pd.Series(lv)
    return df.reset_index()

def fuel_type(v):
    if pd.isna(v): return None
    v = str(v).lower()
    if "gas" in v: return "gas"
    if "electr" in v: return "electricity"
    return "other"

fw_rows = []
for yr in YEARS_OLS:
    fp = os.path.join(RS_DIR, f"resales_{yr}.csv")
    if not os.path.exists(fp): continue
    df = pd.read_csv(fp, low_memory=False, usecols=["MAIN_FUEL"])
    df["fuel"] = df["MAIN_FUEL"].apply(fuel_type)
    fc = df[df["fuel"].isin(["gas","electricity"])]["fuel"].value_counts()
    tot = fc.sum()
    if tot > 0: fw_rows.append({"year":yr, "w_gas":fc.get("gas",0)/tot, "w_elec":fc.get("electricity",0)/
     tot})
fw = pd.DataFrame(fw_rows)

g = reconstruct_level(parse_rpi(os.path.join(EDIR,"gas.csv")), 2017, 4.53)
e = reconstruct_level(parse_rpi(os.path.join(EDIR,"electricity.csv")), 2008, 9.3)
ga = g.groupby(g["date"].dt.year)["level"].mean().rename("gas")
ea = e.groupby(e["date"].dt.year)["level"].mean().rename("elec")
prices = pd.concat([ga, ea], axis=1).reset_index().rename(columns={"date":"year"})
ei = prices.merge(fw, on="year", how="inner")
```

```
ei["energy_index"] = ei["gas"]*ei["w_gas"] + ei["elec"]*ei["w_elec"]
ei = ei[ei["year"].between(2009,2025)].reset_index(drop=True)

ei.to_csv(os.path.join(HOUT,"report_energy","tab1_energy_index.csv"), index=False)
print("Energy Index (2017-2025):")
print(ei[ei["year"]>=2017][["year","gas","elec","energy_index"]].round(2).to_string(index=False))
```

B.10 Market heat index

.

This listing constructs the market heat index as a composite of standardised year-on-year price growth and transaction volume in the resale sample.

Listing 10: Appendix B.10: Market heat index

```
mh_rows = []
for yr in YEARS_OLS:
    fp = os.path.join(RS_DIR, f"resales_{yr}.csv")
    if not os.path.exists(fp): continue
    df = pd.read_csv(fp, low_memory=False, usecols=["Resale_Price"])
    df["Resale_Price"] = pd.to_numeric(df["Resale_Price"], errors="coerce").dropna()
    df = df[df["Resale_Price"] > 0]
    lo, hi = df["Resale_Price"].quantile([0.01, 0.99])
    df = df[(df["Resale_Price"] >= lo) & (df["Resale_Price"] <= hi)]
    mh_rows.append({"year":yr, "median_price":df["Resale_Price"].median(), "volume":len(df)})

mh = pd.DataFrame(mh_rows)
mh["price_growth"] = mh["median_price"].pct_change() * 100
mh.loc[mh.index[0], "price_growth"] = 0
mh["z_growth"] = (mh["price_growth"] - mh["price_growth"].mean()) / mh["price_growth"].std()
mh["z_volume"] = (mh["volume"] - mh["volume"].mean()) / mh["volume"].std()
mh["heat_index"] = (mh["z_growth"] + mh["z_volume"]) / 2
mh["regime"] = np.where(mh["heat_index"] >= mh["heat_index"].median(), "Hot", "Cold")

mh.to_csv(os.path.join(HOUT,"report_hotmarket","tab1_market_heat.csv"), index=False)
print("Market Heat Index (2017-2025):")
print(mh[mh["year"]>=2017][["year","median_price","volume","heat_index","regime"]].round(2).to_string(index
    =False))
```

B.11 Hedonic overlay figures

This listing produces the overlay visualisations that compare annual hedonic coefficients for Bands A and G against the energy index and the market heat index.

Listing 11: Appendix B.11: Hedonic overlay figures

```
sns.set_theme(style="whitegrid", font_scale=1.15)

# Fig 1: Energy Index vs A/G
ei2 = ei[ei["year"] >= 2017].copy()
cf2 = coefs_df[coefs_df["year"] >= 2017].copy()

fig, ax1 = plt.subplots(figsize=(13, 6))
ax1.set_ylabel("Energy Index (p/kWh)", color="#e67e22", fontsize=12)
ax1.plot(ei2["year"], ei2["energy_index"], color="#e67e22", lw=3, marker="D", ms=5, label="Energy Index",
    zorder=5)
ax1.fill_between(ei2["year"], ei2["energy_index"], alpha=.12, color="#e67e22")
ax1.tick_params(axis="y", labelcolor="#e67e22")
```

```
ax1.set_ylim(0, ei2["energy_index"].max()*1.5)
ax1.axvspan(WAR, 2025.6, alpha=.06, color="#c0392b")
ax1.axvline(WAR, color="grey", ls="--", lw=1.5)
ax1.text(WAR+.1, ei2["energy_index"].max()*1.42, "Russia-Ukraine\nFeb 2022", fontsize=8.5, color="grey")
ax2 = ax1.twinx()
ax2.set_ylabel("Hedonic Premium/Discount vs Band D (%)", fontsize=12)
ax2.axhline(0, color="grey", lw=.8, ls=":")
for b,col,mk,lbl in [("A","#1a9850","o","Band A (Green Premium)"),("G","#d73027","s","Band G (Brown
    Discount)")]:
    s = cf2[cf2["band"]==b].sort_values("year")
    ax2.plot(s["year"], s["pct_premium"], color=col, lw=2.5, marker=mk, ms=6, label=lbl)
    ax2.fill_between(s["year"], s["ci_lo"], s["ci_hi"], alpha=.08, color=col)
h1,l1 = ax1.get_legend_handles_labels(); h2,l2 = ax2.get_legend_handles_labels()
ax2.legend(h1+h2, l1+l2, loc="upper left", framealpha=.92)
ax1.set_xlabel("Year")
ax1.set_title("Energy Index vs Band A Premium & Band G Discount (2017-2025)\n"
              "Annual hedonic OLS coefficients, HC1 robust SEs, 95% CI shaded", fontweight="bold")
ax1.set_xticks(ei2["year"]); ax1.set_xticklabels(ei2["year"].astype(int), rotation=45, fontsize=9)
plt.tight_layout()
fig.savefig(os.path.join(HOUT, "report_energy", "fig2_energy_vs_ag.png"), dpi=200)
plt.close()
print("Saved -> report_energy/fig2_energy_vs_ag.png")

# Fig 2: Market Heat vs A/G
mhp = mh[mh["year"] >= 2017].copy()
cfp = coefs_df.merge(mh[["year","heat_index","regime"]], on="year", how="inner")
cfp = cfp[cfp["year"] >= 2017].copy()

fig, ax1 = plt.subplots(figsize=(13, 6))
colors_bar = ["#e74c3c" if r=="Hot" else "#3498db" for r in mhp["regime"]]
ax1.bar(mhp["year"], mhp["heat_index"], color=colors_bar, alpha=0.35, edgecolor="none", label="Heat Index")
ax1.axhline(mh["heat_index"].median(), color="grey", ls="--", lw=1)
ax1.set_ylabel("Market Heat Index (z-score)", fontsize=12)
ax1.set_ylim(mhp["heat_index"].min()-0.5, mhp["heat_index"].max()+0.8)
ax2 = ax1.twinx()
ax2.set_ylabel("Hedonic Premium/Discount vs D (%)", fontsize=12)
ax2.axhline(0, color="grey", lw=.8, ls=":")
for b,col,mk,lbl in [("A","#1a9850","o","Band A (Green Premium)"),("G","#d73027","s","Band G (Brown
    Discount)")]:
    s = cfp[cfp["band"]==b].sort_values("year")
    ax2.plot(s["year"], s["pct_premium"], color=col, lw=2.5, marker=mk, ms=7, label=lbl, zorder=5)
    ax2.fill_between(s["year"], s["ci_lo"], s["ci_hi"], alpha=.10, color=col)
h1,l1 = ax1.get_legend_handles_labels(); h2,l2 = ax2.get_legend_handles_labels()
ax2.legend(h1+h2, l1+l2, loc="upper left", framealpha=.92, fontsize=10)
ax1.set_xlabel("Year")
ax1.set_title("Market Heat Index vs Band A Premium & Band G Discount (2017-2025)\n"
              "Red bars = Hot market  |  Blue bars = Cold market  |  Lines = annual hedonic coefficients",
              fontweight="bold")
ax1.set_xticks(mhp["year"]); ax1.set_xticklabels(mhp["year"].astype(int), fontsize=10)
plt.tight_layout()
fig.savefig(os.path.join(HOUT, "report_hotmarket", "fig2_heat_vs_premiums.png"), dpi=200)
plt.close()
print("Saved -> report_hotmarket/fig2_heat_vs_premiums.png")

print("\nPipeline complete")
```

B.12 PSM-DiD core engine

This listing defines the reusable propensity-score-matching and doubly robust DiD functions, together with the diagnostic and plotting helpers used throughout the empirical analysis.

Listing 12: Appendix B.12: PSM-DiD core engine

```
"""
```

```
PSM-DiD Core Engine
===================
Reusable functions for propensity-score-matched DiD, run per EPC band vs D.
PSM: Gradient Boosting propensity scores (captures non-linearities).
DiD: Doubly-robust - PSM for unbiasedness, hedonic controls for precision.
"""
import os, gc, warnings
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.ensemble import GradientBoostingClassifier
from scipy import stats

warnings.filterwarnings("ignore")

BASE = os.path.abspath(os.path.join(os.path.dirname(__file__), ".."))
RDIR = os.path.join(BASE, "resales_by_year")

COLS = ['Resale_Price','Resale_Date','TOTAL_FLOOR_AREA',
        'CURRENT_ENERGY_RATING','PROPERTY_TYPE','TENURE',
        'POSTCODE','NUMBER_HABITABLE_ROOMS']

def sigstars(p):
    if p<.001: return "***"
    if p<.01:  return "**"
    if p<.05:  return "*"
    if p<.10:  return ""
    return "ns"

#  DATA
def load_years(years):
    chunks = []
    for yr in years:
        p = os.path.join(RDIR, f"resales_{yr}.csv")
        if not os.path.exists(p): continue
        df = pd.read_csv(p, low_memory=False, usecols=COLS)
        chunks.append(df)
        print(f"    {yr}: {len(df):,}")
    df = pd.concat(chunks, ignore_index=True); del chunks; gc.collect()
    df['Resale_Date'] = pd.to_datetime(df['Resale_Date'], errors='coerce')
    df = df.dropna(subset=['Resale_Date'])
    df['year'] = df['Resale_Date'].dt.year
    df['month'] = df['Resale_Date'].dt.month
    df['ym'] = df['Resale_Date'].dt.to_period('M')
    for c in ['Resale_Price','TOTAL_FLOOR_AREA','NUMBER_HABITABLE_ROOMS']:
        df[c] = pd.to_numeric(df[c], errors='coerce')
    df['epc'] = df['CURRENT_ENERGY_RATING'].astype(str).str.strip().str.upper()
    df = df[df['epc'].isin(list('ABCDEFG'))].copy()
    df = df[df['PROPERTY_TYPE'].isin(['House','Flat','Bungalow','Maisonette'])].copy()
    df['pc_area'] = df['POSTCODE'].astype(str).str.extract(r'^([A-Z]{1,2})')[0]
    df['tenure_clean'] = df['TENURE'].astype(str).str.lower().apply(
        lambda x: 'owner' if 'owner' in x else ('rental' if 'rent' in x else 'other'))
    df = df.dropna(subset=['Resale_Price','TOTAL_FLOOR_AREA','pc_area',
                           'NUMBER_HABITABLE_ROOMS']).copy()
    df = df[(df['Resale_Price']>0)&(df['TOTAL_FLOOR_AREA']>9)].copy()
    df['rooms'] = df['NUMBER_HABITABLE_ROOMS'].clip(1,10).astype(int)
    for c in ('Resale_Price','TOTAL_FLOOR_AREA'):
        lo,hi = df[c].quantile([0.01,0.99])
        df = df[(df[c]>=lo)&(df[c]<=hi)].copy()
    df['log_price'] = np.log(df['Resale_Price'])
    df['log_area'] = np.log(df['TOTAL_FLOOR_AREA'])
    print(f"    -> {len(df):,} total")
    return df

# PSM (vectorised sorted-array NN)
def psm_match(df, caliper_mult=0.25, max_per_side=15000):
    df = df.copy()
    df['block'] = df['PROPERTY_TYPE'] + '_' + df['pc_area']
```

```
    df['tenure_enc'] = df['tenure_clean'].map({'owner':0,'rental':1,'other':2}).fillna(2).astype(int)
    matched_idx = []
    skipped = 0
    for bname, grp in df.groupby('block'):
        treat = grp[grp['treated']==1]
        ctrl  = grp[grp['treated']==0]
        if len(treat) < 3 or len(ctrl) < 3:
            skipped += 1; continue
        if len(treat) > max_per_side:
            treat = treat.sample(max_per_side, random_state=42)
        if len(ctrl) > max_per_side:
            ctrl = ctrl.sample(max_per_side, random_state=42)
        block_df = pd.concat([treat, ctrl])
        X = block_df[['TOTAL_FLOOR_AREA','rooms','tenure_enc']].values
        y = block_df['treated'].values
        try:
            gbm = GradientBoostingClassifier(
                n_estimators=100, max_depth=3, learning_rate=0.1,
                subsample=0.8, random_state=42)
            gbm.fit(X, y)
            ps = gbm.predict_proba(X)[:,1]
        except:
            skipped += 1; continue
        ps_clip = np.clip(ps, 1e-6, 1-1e-6)
        logit_ps = np.log(ps_clip / (1-ps_clip))
        caliper = caliper_mult * logit_ps.std()
        if caliper < 0.01: caliper = 0.5
        treat_mask = block_df['treated'].values == 1
        t_idx = block_df.index[treat_mask]
        c_idx = block_df.index[~treat_mask]
        t_ps  = logit_ps[treat_mask]
        c_ps  = logit_ps[~treat_mask]
        # Sort controls for fast lookup
        c_order = np.argsort(c_ps)
        c_ps_sorted = c_ps[c_order]
        c_idx_sorted = c_idx[c_order]
        used = np.zeros(len(c_ps_sorted), dtype=bool)
        for ti, tps in zip(t_idx, t_ps):
            pos = np.searchsorted(c_ps_sorted, tps)
            best_ci = None; best_d = caliper + 1
            for offset in range(max(len(c_ps_sorted), 20)):
                found = False
                for p in [pos - offset, pos + offset]:
                    if p < 0 or p >= len(c_ps_sorted): continue
                    if used[p]: continue
                    d = abs(c_ps_sorted[p] - tps)
                    if d > caliper: continue
                    if d < best_d:
                        best_d = d; best_ci = p; found = True
                if best_ci is not None and offset > 0 and not found:
                    break
            if best_ci is not None:
                matched_idx.append(ti)
                matched_idx.append(c_idx_sorted[best_ci])
                used[best_ci] = True
    matched = df.loc[matched_idx].copy()
    nt = matched['treated'].sum()
    nc = len(matched) - nt
    print(f"      Matched: {nt:,}T + {nc:,}C = {len(matched):,} (skipped {skipped} blocks)")
    return matched

# BALANCE
def check_balance(df):
    rows = []
    for var, name in [('TOTAL_FLOOR_AREA','Floor Area'),
                      ('rooms','Rooms'),('Resale_Price','Price')]:
        t = df[df['treated']==1][var]; c = df[df['treated']==0][var]
        psd = np.sqrt((t.std()**2 + c.std()**2)/2)
```

```
        smd = (t.mean()-c.mean())/psd if psd>0 else 0
        rows.append({'variable':name,'treat_mean':round(t.mean(),1),
                     'ctrl_mean':round(c.mean(),1),'smd':round(smd,4),
                     'balanced': abs(smd)<0.1})
    return pd.DataFrame(rows)

# DiD (doubly-robust: PSM + hedonic controls)
def run_did(df, pre_start, post_start, post_end, no_controls=False):
    sub = df.copy()
    ps = pd.Period(post_start,'M'); pe = pd.Period(post_end,'M')
    prs = pd.Period(pre_start,'M')
    sub = sub[(sub['ym']>=prs)&(sub['ym']<=pe)].copy()
    sub['post'] = (sub['ym']>=ps).astype(int)
    sub['treat_post'] = sub['treated'] * sub['post']
    if len(sub) < 30:
        return None
    xcols = ['treated','post','treat_post']
    if not no_controls:
        # Hedonic controls for precision
        xcols += ['log_area','rooms']
        for v in ['rental','other']:
            sub[f'ten_{v}'] = (sub['tenure_clean']==v).astype(int)
            xcols.append(f'ten_{v}')
        # Property type dummies
        ptypes = sub['PROPERTY_TYPE'].unique()
        if len(ptypes) > 1:
            for pt in ptypes[1:]:
                col = f'pt_{pt.lower()}'
                sub[col] = (sub['PROPERTY_TYPE']==pt).astype(int)
                xcols.append(col)
    X = sm.add_constant(sub[xcols].astype(float))
    try:
        model = sm.OLS(sub['log_price'], X).fit(cov_type='HC1')
    except:
        return None
    b = model.params['treat_post']
    return {
        'beta': b, 'pct': (np.exp(b)-1)*100,
        't': model.tvalues['treat_post'],
        'p': model.pvalues['treat_post'],
        'n': int(model.nobs), 'r2': model.rsquared,
        'model': model, 'data': sub
    }

# DIAGNOSTICS
def run_diagnostics(result):
    if result is None: return {}
    m = result['model']
    resid = m.resid
    # Breusch-Pagan
    try:
        from statsmodels.stats.diagnostic import het_breuschpagan
        bp_stat, bp_p, _, _ = het_breuschpagan(resid, m.model.exog)
    except:
        bp_stat, bp_p = np.nan, np.nan
    # Jarque-Bera
    jb_stat, jb_p = stats.jarque_bera(resid)
    # Durbin-Watson
    from statsmodels.stats.stattools import durbin_watson
    dw = durbin_watson(resid)
    # VIF
    from statsmodels.stats.outliers_influence import variance_inflation_factor
    try:
        X = m.model.exog
        vifs = [variance_inflation_factor(X, i) for i in range(X.shape[1])]
        max_vif = max(vifs[1:]) if len(vifs)>1 else 0  # skip constant
    except:
        max_vif = np.nan
```

```
    return {'bp_stat':bp_stat,'bp_p':bp_p,'jb_stat':jb_stat,'jb_p':jb_p,
            'dw':dw,'max_vif':max_vif}

# FULL PIPELINE FOR ONE BAND
def run_band_did(df_all, band, pre_start, post_start, post_end,
                 prop_type=None, caliper=0.25):
    """Run full PSM-DiD for one band vs D, optionally filtered to one property type."""
    sub = df_all[df_all['epc'].isin([band, 'D'])].copy()
    sub['treated'] = (sub['epc']==band).astype(int)
    if prop_type:
        sub = sub[sub['PROPERTY_TYPE']==prop_type].copy()
    nt = sub['treated'].sum()
    nc = len(sub) - nt
    if nt < 20 or nc < 20:
        return None
    matched = psm_match(sub, caliper_mult=caliper)
    if len(matched) < 40:
        return None
    bal = check_balance(matched)
    did = run_did(matched, pre_start, post_start, post_end)
    if did is None:
        return None
    diag = run_diagnostics(did)
    did_bare = run_did(matched, pre_start, post_start, post_end, no_controls=True)
    bare_pct = did_bare['pct'] if did_bare else np.nan
    return {
        'band': band, 'prop_type': prop_type or 'Pooled',
        'beta': did['beta'], 'pct': did['pct'],
        't': did['t'], 'p': did['p'], 'sig': sigstars(did['p']),
        'n': did['n'], 'r2': did['r2'],
        'pct_no_controls': bare_pct,
        'balance': bal, 'diagnostics': diag,
        'model': did['model'], 'data': did['data'],
        'matched_data': matched
    }
```

B.13 PSM-DiD analysis runner

This listing runs the band-by-band and property-type-stratified PSM-DiD analyses for the energy-crisis and hot-market experiments, and writes the figures and tables reported in the main text.

Listing 13: Appendix B.13: PSM-DiD analysis runner

```
"""
PSM-DiD Main Runner - Band-by-band, stratified by property type.
Produces all figures and tables for the report.
"""
import os, gc, warnings
import pandas as pd
import numpy as np
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from b12_psm_did_core_engine import (
    load_years, run_band_did, sigstars, check_balance,
    psm_match, run_did, run_diagnostics,
)

warnings.filterwarnings("ignore")
sns.set_theme(style="whitegrid", font_scale=1.1)

BASE = os.path.abspath(os.path.join(os.path.dirname(__file__), ".."))
```

```
OUT = os.path.join(BASE, "hedonic_output", "report_did_v4")
for d in ['energy','hotmarket','robustness','diagnostics']:
    os.makedirs(os.path.join(OUT, d), exist_ok=True)

BANDS = ['A','B','C','E','F','G']
PTYPES = ['House','Flat','Bungalow','Maisonette']
PALETTE = {'A':'#1a9850','B':'#66bd63','C':'#a6d96a',
           'E':'#fdae61','F':'#f46d43','G':'#d73027'}

EXPERIMENTS = {
    'energy': {
        'name': 'Energy Crisis (2022)',
        'years': range(2019, 2024),
        'pre_start': '2019-01', 'post_start': '2022-02', 'post_end': '2023-12',
        'shock_date': '2022-02-01',
        'placebo_post': '2020-07', 'placebo_end': '2021-12',
        'placebo_pre': '2019-01',
    },
    'hotmarket': {
        'name': 'Hot Market (2021)',
        'years': range(2017, 2023),
        'pre_start': '2017-01', 'post_start': '2021-01', 'post_end': '2022-12',
        'shock_date': '2021-01-01',
        'placebo_post': '2019-01', 'placebo_end': '2020-06',
        'placebo_pre': '2017-01',
    }
}


# VISUALISATION FUNCTIONS
def plot_forest(results_df, title, filename):
    """Forest plot: ATT with 95% CI for each band, pooled + by property type."""
    fig, ax = plt.subplots(figsize=(10, max(6, len(results_df)*0.35)))
    results_df = results_df.sort_values(['band','prop_type']).reset_index(drop=True)
    y_pos = range(len(results_df))
    colors = [PALETTE.get(b,'grey') for b in results_df['band']]
    ci_lo = results_df['pct'] - 1.96 * results_df['pct'] / results_df['t'].replace(0, np.nan)
    ci_hi = results_df['pct'] + 1.96 * results_df['pct'] / results_df['t'].replace(0, np.nan)
    ax.hlines(y_pos, ci_lo, ci_hi, colors=colors, lw=2, alpha=0.6)
    ax.scatter(results_df['pct'], y_pos, c=colors, s=60, zorder=5, edgecolors='black', lw=0.5)
    ax.axvline(0, color='grey', ls='--', lw=1)
    labels = [f"Band {r['band']} - {r['prop_type']}" for _,r in results_df.iterrows()]
    ax.set_yticks(list(y_pos)); ax.set_yticklabels(labels)
    ax.set_xlabel('ATT (%)', fontsize=12)
    ax.set_title(title, fontweight='bold', fontsize=13)
    for i, r in results_df.iterrows():
        star = sigstars(r['p'])
        ax.text(max(ci_hi.iloc[i], r['pct'])+0.3, i, f"{r['pct']:+.1f}% {star}",
                va='center', fontsize=8)
    plt.tight_layout()
    fig.savefig(os.path.join(OUT, filename), dpi=200, bbox_inches='tight'); plt.close()
    print(f"  {filename}")

def plot_pre_post_bars(results_df, title, filename):
    """Grouped bar chart of ATTs by band - report-ready."""
    pooled = results_df[results_df['prop_type']=='Pooled'].copy()
    if pooled.empty: return
    fig, ax = plt.subplots(figsize=(12, 8))
    x = np.arange(len(pooled))
    colors = [PALETTE.get(b,'grey') for b in pooled['band']]
    bars = ax.bar(x, pooled['pct'], color=colors, width=0.6, edgecolor='white', lw=1.5)
    for i, (_, r) in enumerate(pooled.iterrows()):
        star = sigstars(r['p'])
        yoff = 0.3 if r['pct'] >= 0 else -0.5
        ax.text(i, r['pct']+yoff, f"{r['pct']:+.1f}%\n{star}",
                ha='center', va='bottom' if r['pct']>=0 else 'top', fontsize=10, fontweight='bold')
    ax.axhline(0, color='grey', ls='--', lw=1)
    ax.set_xticks(x); ax.set_xticklabels([f"Band {b}" for b in pooled['band']], fontsize=11)
```

```
    ax.set_ylabel('Causal Effect (ATT, %)', fontsize=12)
    # Add margin so labels don't overlap title or get clipped
    ymin, ymax = ax.get_ylim()
    ax.set_ylim(ymin - 0.8, ymax + 1.5)
    ax.set_title(title, fontweight='bold', fontsize=14, pad=20)
    plt.tight_layout()
    fig.savefig(os.path.join(OUT, filename), dpi=200, bbox_inches='tight'); plt.close()
    print(f"  {filename}")

def plot_by_proptype_bars(results_df, title, filename):
    """Grouped bar chart: ATT by property type for each band."""
    by_pt = results_df[results_df['prop_type']!='Pooled'].copy()
    if by_pt.empty: return
    bands_present = [b for b in BANDS if b in by_pt['band'].values]
    ptypes_present = [p for p in PTYPES if p in by_pt['prop_type'].values]
    fig, ax = plt.subplots(figsize=(14, 6))
    n_pt = len(ptypes_present)
    width = 0.8 / n_pt
    pt_colors = {'House':'#2980b9','Flat':'#e74c3c','Bungalow':'#27ae60','Maisonette':'#f39c12'}
    for j, pt in enumerate(ptypes_present):
        sub = by_pt[by_pt['prop_type']==pt]
        vals = [sub[sub['band']==b]['pct'].values[0] if b in sub['band'].values else 0 for b in
 bands_present]
        sigs = [sigstars(sub[sub['band']==b]['p'].values[0]) if b in sub['band'].values else '' for b in
 bands_present]
        x = np.arange(len(bands_present)) + j*width - (n_pt-1)*width/2
        bars = ax.bar(x, vals, width=width, label=pt, color=pt_colors.get(pt,'grey'),
                      edgecolor='white', lw=0.8, alpha=0.85)
        for k, (v, s) in enumerate(zip(vals, sigs)):
            if s and s != 'ns':
                ax.text(x[k], v + (0.2 if v>=0 else -0.4), s, ha='center', fontsize=7)
    ax.axhline(0, color='grey', ls='--', lw=1)
    ax.set_xticks(range(len(bands_present)))
    ax.set_xticklabels([f"Band {b}" for b in bands_present], fontsize=11)
    ax.set_ylabel('ATT (%)', fontsize=12)
    ax.set_title(title, fontweight='bold', fontsize=13)
    ax.legend(framealpha=0.9, fontsize=10)
    plt.tight_layout()
    fig.savefig(os.path.join(OUT, filename), dpi=200, bbox_inches='tight'); plt.close()
    print(f"  {filename}")

def plot_heatmap(results_df, title, filename):
    """Band x Property Type heatmap of ATTs."""
    by_pt = results_df[results_df['prop_type']!='Pooled'].copy()
    if by_pt.empty: return
    pivot = by_pt.pivot_table(index='band', columns='prop_type', values='pct')
    # reorder
    for b in BANDS:
        if b not in pivot.index: pivot.loc[b] = np.nan
    pivot = pivot.reindex(BANDS).reindex(columns=[p for p in PTYPES if p in pivot.columns])
    fig, ax = plt.subplots(figsize=(10, 6))
    sns.heatmap(pivot, annot=True, fmt='.1f', cmap='RdYlGn_r', center=0,
                linewidths=1, ax=ax, cbar_kws={'label':'ATT (%)'})
    ax.set_title(title, fontweight='bold', fontsize=13)
    ax.set_ylabel('EPC Band'); ax.set_xlabel('Property Type')
    plt.tight_layout()
    fig.savefig(os.path.join(OUT, filename), dpi=200, bbox_inches='tight'); plt.close()
    print(f"  {filename}")

def plot_parallel_trends_panel(results_list, shock_date, title, filename):
    """2x3 panel of parallel trends, one per band."""
    bands_with_data = [r for r in results_list if r is not None and 'data' in r]
    if not bands_with_data: return
    n = len(bands_with_data)
    ncols = 3; nrows = max(1, (n+ncols-1)//ncols)
    fig, axes = plt.subplots(nrows, ncols, figsize=(16, 4.5*nrows), squeeze=False)
    shock = pd.Timestamp(shock_date)
    for idx, r in enumerate(bands_with_data):
```

```
        ax = axes[idx//ncols][idx%ncols]
        sub = r['data'].copy()
        sub['qtr'] = sub['Resale_Date'].dt.to_period('Q')
        agg = sub.groupby(['qtr','treated']).agg(
            m=('log_price','mean'),
            se=('log_price', lambda x: x.std()/np.sqrt(len(x)))).reset_index()
        agg['ts'] = agg['qtr'].dt.to_timestamp()
        for tv, col, lab in [(0,'#2980b9','D (ctrl)'),(1,'#e74c3c',f"Band {r['band']}")]:
            s = agg[agg['treated']==tv].sort_values('ts')
            ax.plot(s['ts'], s['m'], 'o-', ms=4, lw=1.5, color=col, label=lab)
            ax.fill_between(s['ts'], s['m']-1.96*s['se'], s['m']+1.96*s['se'], alpha=0.1, color=col)
        ax.axvline(shock, color='grey', ls='--', lw=1.5, alpha=0.7)
        ax.set_title(f"Band {r['band']} vs D", fontweight='bold', fontsize=11)
        ax.legend(fontsize=8, framealpha=0.8)
        ax.tick_params(axis='x', rotation=45, labelsize=8)
    # hide unused
    for idx in range(len(bands_with_data), nrows*ncols):
        axes[idx//ncols][idx%ncols].set_visible(False)
    fig.suptitle(title, fontweight='bold', fontsize=14, y=1.01)
    plt.tight_layout()
    fig.savefig(os.path.join(OUT, filename), dpi=200, bbox_inches='tight'); plt.close()
    print(f" {filename}")

def plot_event_study_panel(results_list, shock_period, title, filename, use_q=True):
    """2x3 panel of event studies."""
    bands_with_data = [r for r in results_list if r is not None and 'data' in r]
    if not bands_with_data: return
    n = len(bands_with_data)
    ncols = 3; nrows = max(1, (n+ncols-1)//ncols)
    fig, axes = plt.subplots(nrows, ncols, figsize=(16, 4.5*nrows), squeeze=False)
    shock = pd.Period(shock_period, 'M')
    for idx, r in enumerate(bands_with_data):
        ax = axes[idx//ncols][idx%ncols]
        sub = r['data'].copy()
        sub['rel_m'] = sub['ym'].apply(lambda x: (x.year-shock.year)*12+x.month-shock.month)
        if use_q:
            sub['rel'] = (sub['rel_m']/3).apply(lambda x: int(np.floor(x)))
        else:
            sub['rel'] = (sub['rel_m']/6).apply(lambda x: int(np.floor(x)))
        periods = sorted(sub['rel'].unique()); ref = -1
        cols_map = {}
        for t in periods:
            if t == ref: continue
            safe = f"tp{t}" if t>=0 else f"tn{abs(t)}"
            cols_map[safe] = t
            sub[safe] = (sub['treated']*(sub['rel']==t)).astype(float)
        X = sm.add_constant(sub[list(cols_map.keys())].copy())
        try:
            mod = sm.OLS(sub['log_price'], X).fit(cov_type='HC1')
            rows = []
            for t in periods:
                if t == ref:
                    rows.append({'t':t,'c':0,'lo':0,'hi':0}); continue
                safe = f"tp{t}" if t>=0 else f"tn{abs(t)}"
                if safe in mod.params:
                    ci = mod.conf_int().loc[safe]
                    rows.append({'t':t, 'c':(np.exp(mod.params[safe])-1)*100,
                                 'lo':(np.exp(ci[0])-1)*100, 'hi':(np.exp(ci[1])-1)*100})
            es = pd.DataFrame(rows).sort_values('t')
            ax.axhline(0, color='grey', ls='--', lw=0.8)
            ax.axvline(-0.5, color='#e74c3c', ls=':', lw=1.5, alpha=0.5)
            pre = es[es['t']<0]; post = es[es['t']>=0]
            ax.plot(pre['t'], pre['c'], 'o-', color='#2980b9', ms=5, lw=1.5)
            ax.fill_between(pre['t'], pre['lo'], pre['hi'], alpha=0.1, color='#2980b9')
            ax.plot(post['t'], post['c'], 's-', color='#e74c3c', ms=5, lw=1.5)
            ax.fill_between(post['t'], post['lo'], post['hi'], alpha=0.1, color='#e74c3c')
        except:
            ax.text(0.5, 0.5, 'Insufficient data', transform=ax.transAxes, ha='center')
```

```
            ax.set_title(f"Band {r['band']} vs D", fontweight='bold', fontsize=11)
            ax.tick_params(labelsize=8)
        for idx in range(len(bands_with_data), nrows*ncols):
            axes[idx//ncols][idx%ncols].set_visible(False)
        fig.suptitle(title, fontweight='bold', fontsize=14, y=1.01)
        plt.tight_layout()
        fig.savefig(os.path.join(OUT, filename), dpi=200, bbox_inches='tight'); plt.close()
        print(f" {filename}")

    def plot_caliper_sensitivity(sens_rows, title, filename):
        """Bar chart of ATT across caliper choices."""
        df = pd.DataFrame(sens_rows)
        if df.empty: return
        fig, ax = plt.subplots(figsize=(12, 6))
        bands_present = [b for b in BANDS if b in df['band'].values]
        cals = sorted(df['caliper'].unique())
        width = 0.8 / len(cals)
        cal_colors = {0.10:'#3498db', 0.25:'#2c3e50', 0.50:'#e67e22'}
        for j, cal in enumerate(cals):
            sub = df[df['caliper']==cal]
            vals = [sub[sub['band']==b]['pct'].values[0] if b in sub['band'].values else 0 for b in
     bands_present]
            x = np.arange(len(bands_present)) + j*width - (len(cals)-1)*width/2
            ax.bar(x, vals, width=width, label=f'{cal}sigma', color=cal_colors.get(cal,'grey'),
                   edgecolor='white', alpha=0.85)
        ax.axhline(0, color='grey', ls='--', lw=1)
        ax.set_xticks(range(len(bands_present)))
        ax.set_xticklabels([f"Band {b}" for b in bands_present])
        ax.set_ylabel('ATT (%)', fontsize=12)
        ax.set_title(title, fontweight='bold', fontsize=13)
        ax.legend(title='Caliper', framealpha=0.9)
        plt.tight_layout()
        fig.savefig(os.path.join(OUT, filename), dpi=200, bbox_inches='tight'); plt.close()
        print(f" {filename}")

    def plot_diagnostics(results_list, title_prefix, subdir):
        """Q-Q plot and residual histogram for the pooled results."""
        from scipy import stats as st
        pooled = [r for r in results_list if r is not None and r.get('prop_type')=='Pooled']
        if not pooled: return
        n = len(pooled)
        ncols = 3; nrows = max(1, (n+ncols-1)//ncols)
        # Q-Q
        fig, axes = plt.subplots(nrows, ncols, figsize=(14, 4*nrows), squeeze=False)
        for idx, r in enumerate(pooled):
            ax = axes[idx//ncols][idx%ncols]
            resid = r['model'].resid
            st.probplot(resid, plot=ax)
            ax.set_title(f"Band {r['band']}", fontsize=10)
        for idx in range(len(pooled), nrows*ncols):
            axes[idx//ncols][idx%ncols].set_visible(False)
        fig.suptitle(f'{title_prefix} - Q-Q Plots of DiD Residuals', fontweight='bold')
        plt.tight_layout()
        fig.savefig(os.path.join(OUT, subdir, 'fig_qq_residuals.png'), dpi=200, bbox_inches='tight')
        plt.close()
        print(f"  {subdir}/fig_qq_residuals.png")


    # MAIN
    def run_experiment(exp_key):
        exp = EXPERIMENTS[exp_key]
        subdir = exp_key
        print(f"\n{'='*60}")
        print(f"EXPERIMENT: {exp['name']}")
        print(f"{'='*60}")

        print(f"\n  Loading data ({list(exp['years'])[0]}-{list(exp['years'])[-1]})...")
        df_all = load_years(exp['years'])
```

```
    # POOLED
    print(f"\n  --- POOLED ESTIMATES ---")
    pooled_results = []
    for band in BANDS:
        print(f"\n  Band {band} vs D (pooled):")
        r = run_band_did(df_all, band, exp['pre_start'], exp['post_start'], exp['post_end'])
        if r:
            pooled_results.append(r)

    # BY PROPERTY TYPE
    print(f"\n  --- BY PROPERTY TYPE ---")
    pt_results = []
    for band in BANDS:
        for pt in PTYPES:
            print(f"\n  Band {band} vs D - {pt}:")
            r = run_band_did(df_all, band, exp['pre_start'], exp['post_start'],
                             exp['post_end'], prop_type=pt)
            if r:
                pt_results.append(r)

    all_results = pooled_results + pt_results

    # BUILD RESULTS TABLE
    rows = []
    for r in all_results:
        rows.append({
            'band': r['band'], 'prop_type': r['prop_type'],
            'beta': round(r['beta'],5), 'pct': round(r['pct'],2),
            't': round(r['t'],3), 'p': round(r['p'],4), 'sig': r['sig'],
            'n': r['n'], 'r2': round(r['r2'],4),
            'pct_no_controls': round(r['pct_no_controls'],2) if not np.isnan(r['pct_no_controls']) else '',
            **r['diagnostics']
        })
    res_df = pd.DataFrame(rows)
    res_df.to_csv(os.path.join(OUT, subdir, f'tab_results.csv'), index=False)
    print(f"\n  {subdir}/tab_results.csv")

    # BALANCE TABLE
    bal_rows = []
    for r in all_results:
        b = r['balance'].copy()
        b['band'] = r['band']; b['prop_type'] = r['prop_type']
        bal_rows.append(b)
    if bal_rows:
        pd.concat(bal_rows).to_csv(os.path.join(OUT, subdir, 'tab_balance.csv'), index=False)

    # VISUALISATIONS
    print(f"\n  --- FIGURES ---")

    # 1. Pre/post bars (pooled ATTs)
    pre_post_title = ("Energy Crisis (2022) - Causal Effect by EPC Band (PSM-DiD)"
                      if subdir == 'energy'
                      else f"{exp['name']} - Causal Effect by EPC Band (PSM-DiD)")
    plot_pre_post_bars(res_df, pre_post_title,
                       f"{subdir}/fig_pre_post_bars.png")

    # 2. By property type bars
    plot_by_proptype_bars(res_df, f"{exp['name']} - ATT by Property Type",
                          f"{subdir}/fig_by_proptype.png")

    # 3. Forest plot
    plot_forest(res_df, f"{exp['name']} - Forest Plot (all estimates)",
                f"{subdir}/fig_forest.png")

    # 4. Heatmap
    plot_heatmap(res_df, f"{exp['name']} - ATT Heatmap: Band x Property Type",
                 f"{subdir}/fig_heatmap.png")
```

```
359
360     # 5. Parallel trends panel (pooled)
361     plot_parallel_trends_panel(pooled_results, exp['shock_date'],
362                                f"{exp['name']} - Parallel Trends by Band",
363                                f"{subdir}/fig_parallel_trends.png")
364
365     # 6. Event study panel (pooled)
366     plot_event_study_panel(pooled_results, exp['post_start'],
367                            f"{exp['name']} - Event Study by Band",
368                            f"{subdir}/fig_event_study.png",
369                            use_q=(exp_key=='energy'))
370
371     # 7. Diagnostics
372     plot_diagnostics(pooled_results, exp['name'], subdir)
373
374     # ROBUSTNESS: CALIPER SENSITIVITY
375     print(f"\n  --- CALIPER SENSITIVITY ---")
376     sens_rows = []
377     for cal in [0.10, 0.25, 0.50]:
378         for band in BANDS:
379             print(f"    Band {band}, caliper={cal}sigma")
380             r = run_band_did(df_all, band, exp['pre_start'], exp['post_start'],
381                              exp['post_end'], caliper=cal)
382             if r:
383                 sens_rows.append({'band':band, 'caliper':cal, 'pct':r['pct'],
384                                   'p':r['p'], 'n':r['n']})
385     pd.DataFrame(sens_rows).to_csv(os.path.join(OUT, 'robustness',
386                                    f'tab_caliper_{exp_key}.csv'), index=False)
387     plot_caliper_sensitivity(sens_rows, f"{exp['name']} - Caliper Sensitivity",
388                              f"robustness/fig_caliper_{exp_key}.png")
389
390     # PLACEBO
391     print(f"\n  --- PLACEBO TESTS ---")
392     placebo_rows = []
393     for band in BANDS:
394         print(f"    Placebo: Band {band}")
395         r = run_band_did(df_all, band, exp['placebo_pre'], exp['placebo_post'],
396                          exp['placebo_end'])
397         if r:
398             placebo_rows.append({'band':band, 'pct':round(r['pct'],2),
399                                  't':round(r['t'],3), 'p':round(r['p'],4),
400                                  'sig':r['sig'], 'n':r['n']})
401     plac_df = pd.DataFrame(placebo_rows)
402     plac_df.to_csv(os.path.join(OUT, subdir, 'tab_placebo.csv'), index=False)
403     print(f"  {subdir}/tab_placebo.csv")
404
405     # SUMMARY REPORT
406     lines = [f"{'='*60}", f"PSM-DiD RESULTS: {exp['name']}", f"{'='*60}", "",
407              "POOLED ESTIMATES (all property types)", "-"*50]
408     pooled_df = res_df[res_df['prop_type']=='Pooled']
409     if not pooled_df.empty:
410         lines.append(pooled_df[['band','pct','t','p','sig','n','r2','pct_no_controls']].to_string(index=
        False))
411     lines += ["", "BY PROPERTY TYPE", "-"*50]
412     pt_df = res_df[res_df['prop_type']!='Pooled']
413     if not pt_df.empty:
414         lines.append(pt_df[['band','prop_type','pct','t','p','sig','n']].to_string(index=False))
415     lines += ["", "PLACEBO TESTS", "-"*50]
416     if not plac_df.empty:
417         lines.append(plac_df.to_string(index=False))
418     lines.append(f"\n{'='*60}")
419     with open(os.path.join(OUT, subdir, 'report_summary.txt'), 'w') as f:
420         f.write('\n'.join(lines))
421     print(f"  {subdir}/report_summary.txt")
422
423     del df_all; gc.collect()
424     return res_df
425
```

```python
def main():
    print("="*60)
    print("PSM-DiD ANALYSIS v4 - Band-by-band, by Property Type")
    print(f"Output: {OUT}")
    print("="*60)

    res_energy = run_experiment('energy')
    res_hot = run_experiment('hotmarket')

    # COMBINED SUMMARY
    combined = pd.concat([
        res_energy.assign(experiment='Energy Crisis'),
        res_hot.assign(experiment='Hot Market')
    ])
    combined.to_csv(os.path.join(OUT, 'tab_all_results.csv'), index=False)
    print(f"\n All outputs -> {OUT}/")
    print(" Done.")

if __name__ == "__main__":
    main()
```

References

Austin, P.C. (2011) 'An introduction to propensity score methods for reducing the effects of confounding in observational studies', *Multivariate Behavioral Research*, 46(3), pp. 399–424. 9

Department for Energy Security and Net Zero (2026a) *Domestic private rented property: minimum energy efficiency standard – landlord guidance*. GOV.UK. Available at: GOV.UK guidance page (Accessed: 9 May 2026).

Ferentinos, K., Gibberd, A. and Guin, B. (2021) 'Climate policy and transition risk in the housing market', *Bank of England Working Paper*, No. 918. London: Bank of England. Available at: Bank of England Working Paper 918 PDF (Accessed: 10 May 2026).

Friedman, J.H. (2001) 'Greedy function approximation: A gradient boosting machine', *The Annals of Statistics*, 29(5), pp. 1189–1232. 8

Fuerst, F., McAllister, P., Nanda, A. and Wyatt, P. (2013) 'Does energy efficiency matter to home-buyers? An investigation of EPC ratings and transaction prices in England', *Energy Economics*, 48, pp. 145–156.

Fuerst, F., McAllister, P., Nanda, A. and Wyatt, P. (2015) 'The impact of energy performance certificates on the rental and capital values of commercial and residential buildings in the UK', *Journal of Sustainable Real Estate*, 7(1), pp. 46–62.

Fuerst, F., McAllister, P., Nanda, A. and Wyatt, P. (2016) 'The green premium, segmentation and location', *Urban Studies*, 53(7), pp. 1340–1358.

Great Britain (2015) *The Energy Efficiency (Private Rented Property) (England and Wales) Regulations 2015, SI 2015/962*. London: The Stationery Office. Available at: legislation.gov.uk (Accessed: 9 May 2026).

HM Land Registry (n.d.) *Search the price paid dataset*. Available at: landregistry.data.gov.uk Price Paid Data page (Accessed: 20 March 2026).

HM Revenue & Customs (2021) *Stamp Duty Land Tax: temporary reduced rates*. GOV.UK. Available at: GOV.UK guidance page (Accessed: 9 May 2026).

Ministry of Housing, Communities and Local Government (2026) *Energy Performance of Buildings Data: England and Wales*. Available at: epc.opendatacommunities.org (Accessed: 20 March 2026).

Office for National Statistics (2026a) *RPI: fuel & light: electricity (Jan 1987=100), series DOBX*. Available at: ONS series DOBX (Accessed: 9 May 2026).

Office for National Statistics (2026b) *RPI: fuel & light: gas (Jan 1987=100), series DOBY*. Available at: ONS series DOBY (Accessed: 9 May 2026).

Rosenbaum, P.R. and Rubin, D.B. (1983) 'The central role of the propensity score in observational studies for causal effects', *Biometrika*, 70(1), pp. 41–55. 8

Rosenbaum, P.R. and Rubin, D.B. (1985) 'Constructing a control group using multivariate matched sampling methods that incorporate the propensity score', *The American Statistician*, 39(1), pp. 33–38. 8

White, H. (1980) 'A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity', *Econometrica*, 48(4), pp. 817–838. 9

Bibliography

Crawley, J., Biddulph, P., Northrop, P.J., Wingfield, J., Oreszczyn, T. and Elwell, C. (2019) 'Quantifying the Measurement Error on England and Wales EPC Ratings', *Energies*, 12(18), article 3523. Available at: `https://doi.org/10.3390/en12183523` (Accessed: 18 March 2026).

Department for Energy Security and Net Zero (2026b) *Evaluation of Domestic PRS Minimum Energy Efficiency Standard Regulations: Final report*. London: Department for Energy Security and Net Zero. Available at: `https://assets.publishing.service.gov.uk/media/696b6c69749cbde8ccd36563/prs-mees-evaluation-final-report.pdf` (Accessed: 3 April 2026).

Department for Energy Security and Net Zero (2026c) *Improving the energy performance of privately rented homes: government response*. London: Department for Energy Security and Net Zero.

Fuerst, F., Haddad, M.F.C. and Adan, H. (2020) 'Is there an economic case for energy-efficient dwellings in the UK private rental market?', *Journal of Cleaner Production*, 245, article 118642. Available at: `https://doi.org/10.1016/j.jclepro.2019.118642` (Accessed: 7 March 2026).

Ghosh, A., McConnell, B. and Millán-Quijano, J. (2026) 'Information disclosure and the valuation of energy efficiency in housing markets', *Institute for Fiscal Studies Working Paper*, 26/32. London: Institute for Fiscal Studies. Available at: `https://ifs.org.uk/publications/information-disclosure-and-valuation-energy-efficiency-housing-markets` (Accessed: 9 May 2026).

Sejas-Portillo, R., Comerford, D., Moro, M. and Stowasser, T. (2020) 'Limited Attention in the Housing Market: Threshold Effects of Energy-Performance Certificates on Property Prices and Energy-Efficiency Investments', *CESifo Working Paper*, No. 8669. Munich: CESifo. Available at: `https://www.ifo.de/DocDL/cesifo1_wp8669.pdf` (Accessed: 18 March 2026).

Wei, J. and Peiser, R. (2025) 'Evolving Green Premiums: The Impact of Energy Efficiency on London Housing Prices over Time', *Land*, 14(10), article 2053. Available at: `https://doi.org/10.3390/land14102053` (Accessed: 18 March 2026).