Forecasting Irish Electricity Demand Using Smart Meter Data

Dylan Dijk, Rahil Morjaria & Codie Wood

TB2 2023

Abstract

Using a dataset comprised of smart meter readings from 2672 Irish households, this report aims to use the available data to forecast total electricity demand across households. Rather than relying on a single model fitted to the aggregated data, we adopt a sub-group modeling approach to capture the heterogeneity within the population. Inspired by previous research, we divide the households into “similar” sub-groups based on their characteristics. By constructing separate models for each sub-group and aggregating the predictions, we anticipate improved model fit compared to modeling an inhomogeneous population. We carry out hierarchical clustering to choose our clusters, and then compare the performance of models that are fit on clusters using different data. Comparing model predictions on a test set, we find that using clusters obtained using survey data and past history of electricity demand performs best compared to the other clusters, and significant improvement to a baseline model constructed using random cluster allocation.

Contents

1 Introduction

Electricity consumption is an unavoidable aspect of modern life, with electricity being used to power our homes and enabling various daily activities. The introduction of smart meters has meant the electricity industry faces the challenge of leveraging large amounts of data beyond billing purposes.

In particular, this smart meter data can be used for electricity demand forecasting. This is crucial for energy planning, resource management, and grid stability. Accurate forecasts of global demand can be used to enhance grid management. They can be used to optimize energy generation, pricing, and load balancing as well as to support demand response programs.

In this work, we aim to produce and implement a framework for forecasting total energy demand of a population, one day ahead. We have created a package called `DRCdemand` to aid in this analysis that utilises Stan and parallel programming, which can be found at `https://github.com/DylanDijk/DRCdemand`.[1]

1.1 Data

In this paper we will be exploring data from the Irish household electrical demand data set, published by the Commission for Energy Regulation [3], and contained in the `electBook` R package (`https://github.com/mfasiolo/electBook/blob/master/data/Irish.RData`).

The data set is composed of observations of 2672 Irish households in the year 2010. For each household we have a demand time series, stored in the `indCons` data frame, containing 48 half-hourly electricity demand observations per day. As well as this, we have `survey` data specifying the type of heating used, home ownership status, whether the windows are double-glazed, the number of appliances, social class, tariff allocation, stimulus allocation, and the year of building construction. We also have temperature and calendar information, stored in the `extra` data frame.

A number of days have been removed from the data set, such as Christmas Day, New Years Eve, and other holidays. This is because energy demand on these days are not reflective of non-holiday dates, and would only be useful if we have had several years of observations.

[1] All analysis scripts, along with code to produce all plots in the report, can be found in the sub-directory `https://github.com/DylanDijk/DRCdemand/analysis`.

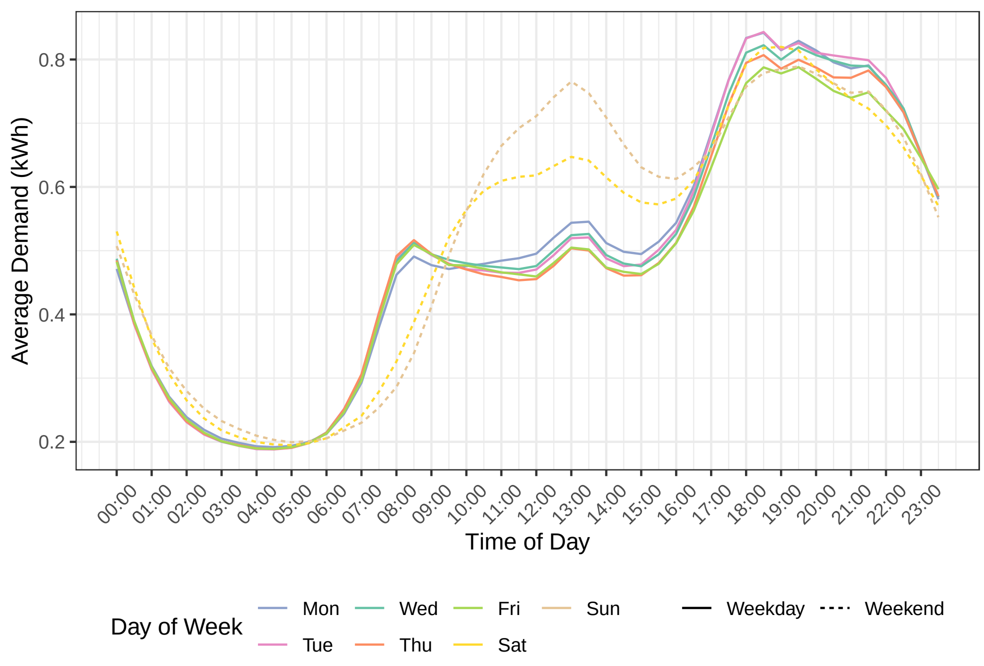


Figure 1: Average weekly profile of electricity demand, over all recorded households.

Figure 1 shows the average weekly electricity demand profile across all of the households. We see a distinct pattern of low demand at night, followed by a small peak at around 8am, a sustained level of energy usage during typical working hours, another small peak at 1pm, and a final larger demand peak at around 6pm. There is little variation in patterns across weekdays, however the profile differs on weekends, where we see that there is sustained higher demand during typical working hours. Whilst we have only shown here the average demand, we note that there is large variation in the demand profiles of individual households.

1.2 Objective and proposed methodology

Our objective, is to use the available data to forecast the total demand across households for the following day. As described previously, we have data for individual households, and an immediate approach would be to sum across all households, and fit a single model to the aggregated data. Alternatively, we can separate the population into “similar” sub-groups, and then construct models for each group, and then aggregate these predictions to obtain predictions for total demand.

The motivation for this, as highlighted in [1] (see, Chapter 9), is the hope that creating models for a population with similar characteristics will have an improved fit, as opposed to fitting a model to an inhomogenous population. In the following section, we describe the methodology used to choose the sub-populations, and in Section 3.4 we describe the model used for each of the clusters.

2 Clustering

Clustering refers to the process of grouping similar objects or data points together based on their characteristics or attributes. It is a statistical technique that is widely used to find patterns or structures within a dataset, where there may not be any predefined labels or categories.

The main goal of clustering is to identify groups or clusters of data points that share similar traits or properties. These traits can be numerical, categorical, or a combination of both. By organizing the data into clusters, we can gain insights into the underlying structure of the dataset and understand how different objects or individuals relate to each other.

Here, we use clustering with the aim of dividing the population of Irish households into sub-populations with similar characteristics. We will consider clustering based on the data available in the `survey` dataset and electricity demand profiles.

2.1 Hierarchical agglomerative clustering

Hierarchical agglomerative clustering (HAC) is a clustering algorithm that groups similar data points together in a hierarchical manner. It starts by considering each data point as an individual cluster, and progressively merges the closest clusters until all data points belong to a single cluster. A general algorithm for HAC is given in Algorithm 1.

Algorithm 1: Hierarchical Agglomerative Clustering

```
Input  : Dataset with $n$ data points
Output: Dendrogram representing hierarchical clusters
Initialize each data point as a separate cluster;
while number of clusters > 1 do
    1. Find the two closest clusters based on dissimilarity measure;
    2. Merge the closest clusters into a new cluster;
    3. Update the dissimilarity matrix;
end
return Dendrogram representing hierarchical clusters.
```

2.1.1 Gower's distance

In order to determine which clusters are closest to eachother, we first have to establish some notion of distance, referred to as *dissimilarity*, between individuals.

Traditional distance metrics, such as Euclidean distance or Manhattan distance, assume that all variables are continuous and numeric. However, our `survey` data includes a mixture of continuous and categorical variables. As such, we cannot apply these more standard distance measures.

Gower's distance [4] is a measure of similarity between individuals in a data set, commonly used when dealing with mixed-type data such as our `survey` data. The dissimilarity is always a number between 0, when individuals are identical, and 1, when individuals are maximally dissimilar.

Suppose we have n individuals and p variables, and let $\mathbf{X}$ denote our $n \times p$ data matrix. Individuals i and j may be compared on variable k and assigned a dissimilarity score $d(ij, k) \in [0, 1]$, taking 0 value when i and j are similar. Sometimes comparison is not possible because information is missing, or in the case of dichotomous variables because a variable is non-existent in both individuals i and j.

Let $\delta(ij, k)$ be an indicator of comparability of $\mathbf{X}_{i,k}$ and $\mathbf{X}_{j,k}$, that is

$$\delta(ij, k) = \begin{cases} 0, & \text{if } \mathbf{X}_{i,k} \text{ and } \mathbf{X}_{j,k} \text{ are not comparable} \\ 1, & \text{otherwise.} \end{cases}$$

The Gower's dissimilarity $D(i, j)$ between two individuals i and j is then calculated as the weighted sum of the dissimilarities between their corresponding variables. This is given by the expression

$$D(i, j) = \frac{\sum_{k=1}^{p} \delta(ij, k) d(ij, k)}{\sum_{k=1}^{p} \delta(ij, k)}.$$

For continuous variables, the dissimilarity $d(ij, k)$ is typically calculated using the absolute difference between the values of the two individuals, given by

$$d(ij, k) = \frac{|\mathbf{X}_{i,k} - \mathbf{X}_{j,k}|}{\max(\mathbf{X}_{\cdot,k}) - \min(\mathbf{X}_{\cdot,k})}.$$

For categorical variables, the distance is defined as 0 if the categories are the same and 1 otherwise, that is to say

$$d(ij, k) = \begin{cases} 1, & \text{if } \mathbf{X}_{i,k} = \mathbf{X}_{j,k} \\ 0, & \text{otherwise.} \end{cases}$$

The Gower's distance matrix is then defined to be the square matrix that stores the pairwise distances between all individuals in the data set. Each element of the matrix represents the dissimilarity between two individuals, indicating how different they are based on their variable values.

2.1.2 Linkage methods

With this notion of dissimilarity between individuals, we can proceed to define a metric to quantify the distance between clusters. We call this a linkage method, or linkage criterion.

The choice of linkage method can have a significant impact on the resulting clustering structure, as different linkage methods may lead to different cluster configurations and interpretations. There are several commonly used linkage methods in HAC.

Complete linkage calculates the distance between two clusters as the maximum distance between any pair of data points from each cluster. Let C_i and C_j denote two clusters. We have distance given by

$$d_{\mathrm{CL}}(C_i, C_j) = \max_{\ell \in C_i, k \in C_j} D(\ell, k).$$

Complete linkage focuses on the dissimilarity between the most dissimilar points in each cluster. This method tends to produce compact clusters and is less affected by outliers.

Single linkage, on the other hand, calculates the distance between two clusters as the minimum distance between any pair of data points from each cluster. For clusters C_i and C_j we have distance given by

$$d_{\mathrm{SL}}(C_i, C_j) = \min_{\ell \in C_i, k \in C_j} D(\ell, k).$$

Single linkage emphasizes the similarity between the closest points in each cluster. This method tends be sensitive to outliers and noise in the data.

Average linkage calculates the distance between two clusters as the average distance between all pairs of data points from each cluster. It takes into account the overall similarity between the data points in the clusters. Average linkage can be thought of as striking a balance between single and complete linkage, and is less sensitive to outliers.

2.1.3 Limitations of hierarchical clustering

We note here that HAC assumes a hierarchical structure in the data, however in some cases the underlying data may not exhibit a clear hierarchical structure. This can lead to less meaningful clustering results.

Further, HAC does not allow for specifying the desired size of clusters in advance, which often results to imbalanced cluster sizes. Large variation in cluster sizes can lead to difficulties in interpretation and analysis, as larger clusters may dominate the overall patterns, obscuring the characteristics of smaller clusters. Smaller numbers of individuals within clusters can also lead to large uncertainty of estimates when these clusters are used for model fitting.

Despite its limitations, due to the adaptability of HAC methods to take advantage of information from mixed-type data we choose to focus on this method of clustering here. Other clustering methods such as k-means clustering and spectral clustering could also be adapted and applied, but are outside of the scope of this work.

2.2 Implementation

We conducted HAC of our households, using Gower's distance with a complete linkage mechanism. We chose to use the complete linkage mechanism, as we wanted to emphasise the differences between different households. We performed HAC using three different sets of information about our households. We considered the following sets of variables:

1. **Fixed:** Including only fixed characteristics of the households, without any data on their demand patterns over time. This included all of the variables from the `survey` data frame, excluding the average demand.
2. **Demand:** Including only the weekly demand profile of the household and their average yearly demand, as calculated using the training data set.
3. **All:** Including all of the fixed and demand data of the households.

We chose to investigate these sets of variables in order to determine what information about households could reveal the most about the underlying groupings in the population, in order to motivate what data could be collected by energy companies in future to best suit their forecasting needs.

HAC is sensitive to the scaling and normalization of data. Differences in the scales or units of the variables can disproportionately influence clustering results. As such, when preprocessing the data, we used scaled and centered variables to ensure meaningful clustering.

For each of the sets of variables, we generated partitions of the data into first 4 and then 8 clusters. In order to have a better sense of the performance of our clustering algorithm, we also performed the same levels of partitioning using random clustering. Here, for each number of clusters k we randomly allocated households into a cluster, with a uniform probability.

3 Forecasting

Given a choice of clusters, we now want to forecast the total demand for each cluster, for the following day. As described in Section 1.1, we have 48 observations for each day. In order to model the total demand, and make predictions for the 48 time points of a given day, we have decided to fit an individual model to each of the time points.

The advantage of constructing separate models for each time point, is that we get predictions tailored to the unique characteristics of each time point, and therefore may acquire more accurate predictions, compared to a single model.

The disadvantage of separate models, are that we are not utilising the temporal nature of the observations, and may be losing information by not using the consecutive data points within a single model. Additionally, fitting 48 models will be more computationally expensive. To overcome this, we run the models in parallel.

The models we used to forecast electricity demand are fully Bayesian linear regression models. Before covering the exact implementation, we introduce Bayesian linear regression, and its relation to ridge regression. Then, in Section 3.4 the fully Bayesian model that we implemented is described.

3.1 Bayesian linear regression

For this section will be looking at linear regression models of the form:

$$y_i = x_i^T \beta + \epsilon$$

In the frequentist fixed design setting the squared loss used in classical regression can be motivated by assuming that the outcome observations are independently distributed from a Gaussian distribution. Finding the MLE estimate of the coefficient, is then equivalent to minimising the squared loss.

In the frequentist approach of ridge regression we solve:

$$\hat{\beta}_R := \underset{\beta}{\operatorname{argmin}} ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda||\beta||_2^2$$

The additional term penalising the size of the coefficients, helps to avoid overfitting. Additionally if $\mathbf{X}^T\mathbf{X}$ is not invertible, for example, in the case of multicollinearity between the predictors, the additional penalisation term ensures that it is invertible.

Ridge regression has a closed form solution:

$$\hat{\beta}_R = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_{p+1})^{-1}\mathbf{X}\mathbf{y}^T \tag{1}$$

Now, if we look at Bayesian linear regression, that is, we set the following likelihood and priors on the coefficients respectively:

$$\begin{gathered} \mathbf{y}\,|\beta; \mathbf{X} \sim \mathcal{N}(\mathbf{X}\beta, \mathbf{I}\sigma^2) \qquad (2) \\ \beta \sim \mathcal{N}(0, \mathbf{I}\sigma_\beta^2) \end{gathered}$$

For this choice of prior and likelihood, there is a closed form for the posterior distribution of β (see, [2], equations 3.53, 3.54):

$$\beta\,|\mathbf{y}; \mathbf{X} \sim \mathcal{N}\big((\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{y}^T,\ (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\sigma^2\big) \tag{3}$$

where $\lambda := \frac{\sigma^2}{\sigma_\beta^2}$.

Therefore, the mean of the posterior is the same as the ridge regression estimate (1), where larger λ introduces more regularisation. And in Bayesian regression, decreasing the variance in the prior σ_β^2 can be interpreted as increasing the amount of regularisation on the coefficients.

Now, to make predictions from the model given a new data point, $\mathbf{x}^*$, we can use the maximum a posterior probability (MAP) of the posterior of the coefficients. Alternatively, instead of just using the MAP estimate from the posterior of the coefficients, in this setting, the full predictive distribution can be evaluated (see, [2], equation 3.58):

$$y^* \,|\beta, \mathbf{y}; \mathbf{x}^*, \mathbf{X} \;\sim\quad \mathcal{N}\Big(\mathbf{x}^* \cdot \hat{\beta}_R, \sigma^2 + (\mathbf{x}^*)^T \sigma^2 \Big(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I}\Big)^{-1} \mathbf{x}^*\Big) \tag{4}$$

which also represents the uncertainty information in the prediction. A simpler way to retrieve a credible interval on the prediction would be to use the variance of the posterior $\beta\,|\mathbf{y}; \mathbf{X}$, and then look at the variability in $(\mathbf{x}^*)^T\beta$, however this does not then incorporate the variability in the outcome variable that is provided by the likelihood. In the predictive posterior (4) we can see that there is the additional σ^2 term.

In this setting (2) there are no priors on the variance, and they need to be chosen manually. In the frequentist approach, the regularisation parameter is often chosen by cross-validation, aiming to balance the trade-off in bias introduced by the penalisation and the reduction in prediction variance.

3.2 Fully Bayesian linear regression

In a fully Bayesian approach, priors are introduced for the variance components, σ_β^2 and σ^2 in the setup (2). In this case, evaluating quantities of interest becomes much harder, for example the integral for the predictive posterior becomes intractable.

In order to retrieve closed form expressions for quantities of interest in Bayesian linear regression a hierarchical prior can be used:

$$\begin{gathered} p(\beta, \sigma^2) = p(\beta|\sigma^2)p(\sigma^2) \\ \text{where} \\ \beta|\sigma^2 \sim \mathcal{N}(0, \mathbf{I}\sigma^2\lambda) \\ \sigma^2 \sim \text{InvGamma}(a_0, b_0) \end{gathered}$$

This choice of priors are conjugate priors, meaning that the posterior will have a Normal Inverse-Gamma distribution. That is, the posteriors of both $\beta|\sigma^2$ and σ^2 will belong to the same distribution as the prior. This then allows the posterior, marginal likelihood/model evidence, and predictive posteriors to be derived exactly.

However, we can avoid the derivations of exact Bayesian inference, that are only possible under certain prior structures with conjugacy, by using MCMC algorithms to solve integrals numerically. This will allow us to experiment with different priors and avoid having to worry about integrals being tractable.

3.3 Stan

In order to generate samples from the posterior of our coefficients, which we then will use to compute the predictions, and the intervals around these predictions, we will use Stan. Stan is a probabilistic programming language that implements full Bayesian statistical inference via Markov Chain Monte Carlo. The `rstan` package [5] provides an R interface to Stan, which links to the `StanHeaders` package that provides the header files for the Stan project.

The default MCMC algorithm used by the `rstan::stan()` function is the no-U-turn sampler (NUTS), which is an adaptive variant of the Hamiltonian Monte Carlo (HMC) algorithm. At a high level, the NUTS algorithm, is built on top of the HMC algorithm, and automates the tuning of certain parameters used by HMC.

We now give a brief overview of the HMC algorithm, and for this section we denote the target density as $\pi(\theta|\mathbf{y})$.

Hamiltonian mechanics is an abstract formulation of classical mechanics, it describes a system involving two time-evolving vectors θ and v. The Hamiltonian $H(\theta, v)$, represents the total energy of the system, and describes the time evolution of the system through Hamilton's equations:

$$\frac{d\theta_i}{dt} = \frac{\partial H}{\partial v_i} \qquad \frac{dv_i}{dt} = -\frac{\partial H}{\partial \theta_i} \quad \text{for } i = 1, \dots, p$$

For any time interval s, these equations then define a mapping, T_s from any time t to the state time $t+s$.

$$(\theta_{t+s}, v_{t+s}) = T_s(\theta_t, v_t)$$

To create an MCMC algorithm that explores the target distribution $\pi(\theta|\mathbf{y})$, we utilise these dynamics to describe a frictionless ball rolling around the negative log posterior distribution, subject to a gravitational pull. The vector θ denotes the position of the ball, and v the momentum.

As mentioned, the Hamiltonian $H(\theta, v)$, represents the total energy of the system, so we can write the following, where $U(\theta)$ is potential energy, and $K(v)$ is kinetic energy,

$$H(\theta, v) = U(\theta) + K(v),$$

In practice, a mass matrix M is used, and the kinetic energy function is given as $K(v) = v^T M^{-1} v$. Therefore, if we let $\mu(v) = \mathcal{N}(0, M)$ and include the appropriate constants, we have that:

$$H(\theta, v) = -\log(\pi(\theta|\mathbf{y})) - \log(\mu(v))$$

Now, we define a joint distribution on the position and velocity vectors (θ, v), such that we can run Hamiltonian dynamics to obtain samples from $\pi(\theta|\mathbf{y})$. The following distribution is chosen:

$$\tilde{\pi}(\theta, v) := \pi(\theta|\mathbf{y})\mu(v)$$

Below the algorithm [6] presents the main idea of HMC, in the idealised case, when we can compute the map T_s for a given s.

Algorithm 2: HMC

```
Input  : Starting sample $\theta_0$, Target distribution $\pi(\theta|\mathbf{y})$, size of time interval $s > 0$,
         and mass PSD matrix $\mathbf{M}$
Output: Samples of $(\theta, v)$
for $t \geq 1$ do
    1. Sample velocity $\tilde{v}_t \sim \mathcal{N}(0, \mathbf{M}^{-1})$
    2. Compute the position of the particle starting from previous position $Z_{t-1}$
       $(\theta_t, v_t) = T_s(Z_{t-1}, \tilde{v}_t)$
end
return;
```

Generating samples in this fashion generates a Markov chain of samples of θ. Then, using the conservation, volume preservation, and reversibility properties of the Hamiltonian, it can be shown that $\pi(\theta|\mathbf{y})$ is the invariant distribution for the Markov kernel from the HMC algorithm (see Appendix A.2).

In practice, T_s is not computable and a discrete approximation is used $T_{L_s,\epsilon}$, where L_s and ϵ determine the approximation. In order for the algorithm to remain valid, under this approximation, a Metropolis-type accept step is used. The NUTS algorithm, chooses a different s at each iteration, which are chosen to optimise the mixing of the MC. In addition, NUTS carries out the tuning for the parameters ϵ and M, which is carried out in the warm-up iterations.

3.4 Model implementation

Using Stan allowed us to experiment with different models, and our final model was of the form:

$$\begin{aligned} \sigma^2 &\sim \text{InvGamma}(2, 0.1) \\ \beta &\sim \mathcal{N}(0, 100) \\ y &\sim \mathcal{N}(\mathbf{X}\beta, \sigma^2) \end{aligned} \tag{5}$$

In order to produce our models, we first split our data into a training and test set. We created our test set by randomly selecting two days from each month, leading to a test set consisting of 24 days total, each with demand data across 48 time points for each household. We then assigned cluster labels to each household, based on the different clustering methods described in Section 2.2. We then calculated the total demand for each of these clusters, giving us a time series for each cluster. We could then fit a model for each of these time series by transforming the time series into a data set in which each row was a certain day and each column a time point (between 0-47).

For each cluster we then proceeded to fit a model for each time point (column of the data set) of the form in Equation (5) (as the underlying affects for each time point would likely be different for each time of day) using only the training data. The features included in the model were 'Time of Year' as a numeric value between 0 and 1, 'Temperature', a factor variable indicating whether it was a 'Weekday' or 'Weekend' and the total demand data of the cluster from the day before (or in regards to our training data the previous day in our data set). All the data was centered (apart from the factor variables) and then the column means stored to be used for our forecasting.

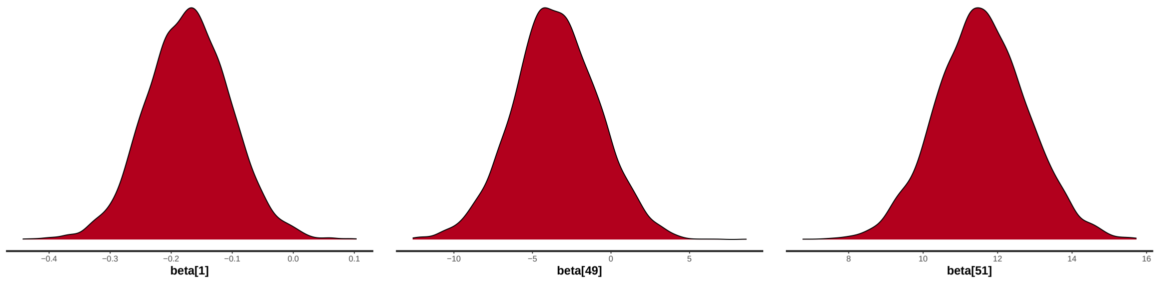


Figure 2: Density graphs showing distribution of coefficients time 0, TimeOfYear and Weekday

.

We identified that a variance of 100 for our coefficients β was necessary to capture the significance of the factor variable 'Weekday' and 'Weekend' which were included in the model using a dummy variable and during feature selection was found to reduce our overall residual mean squared error (RMSE).

We used 2000 iterations of 4 chains (with 1000 warm-up iterations), as we found this allowed our coefficients to converge. As an example, for the 8 Cluster Hierarchical clustering (using all household information data) we have convergence as seen in Figure 3.

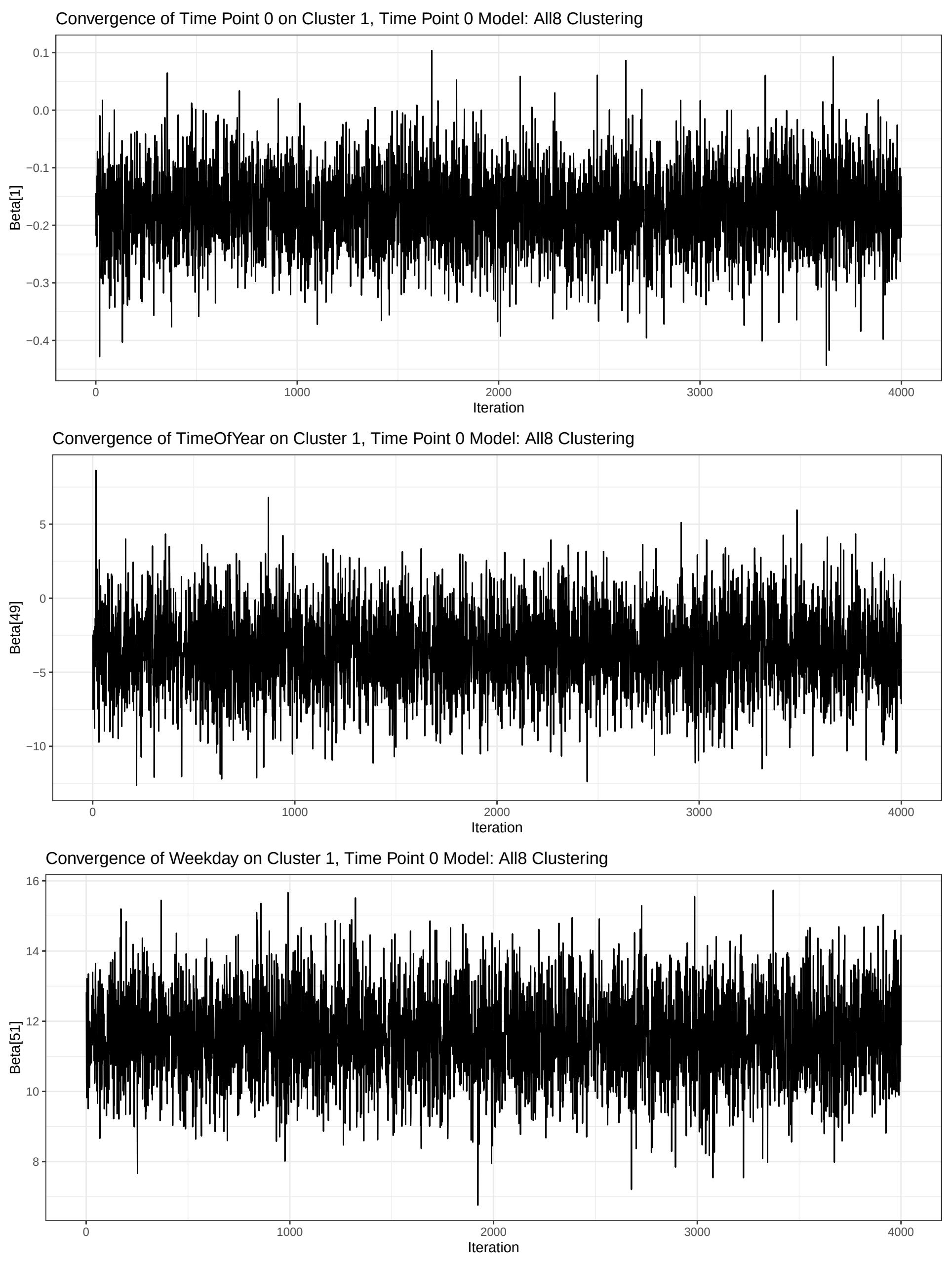


Figure 3: Plot showing convergence of previous day time point 0, TimeOfYear and Weekday on our Cluster 1, Time Point 0 Model.

4 Results and discussion

For each clustering method we fit 48 models, one for each time point, in each cluster. We used the testing data set to estimate the daily total demand for each clustering method by summing our estimates and confidence intervals (which are calculated using the mean posterior for each time point and cluster) over all clusters in the clustering method.

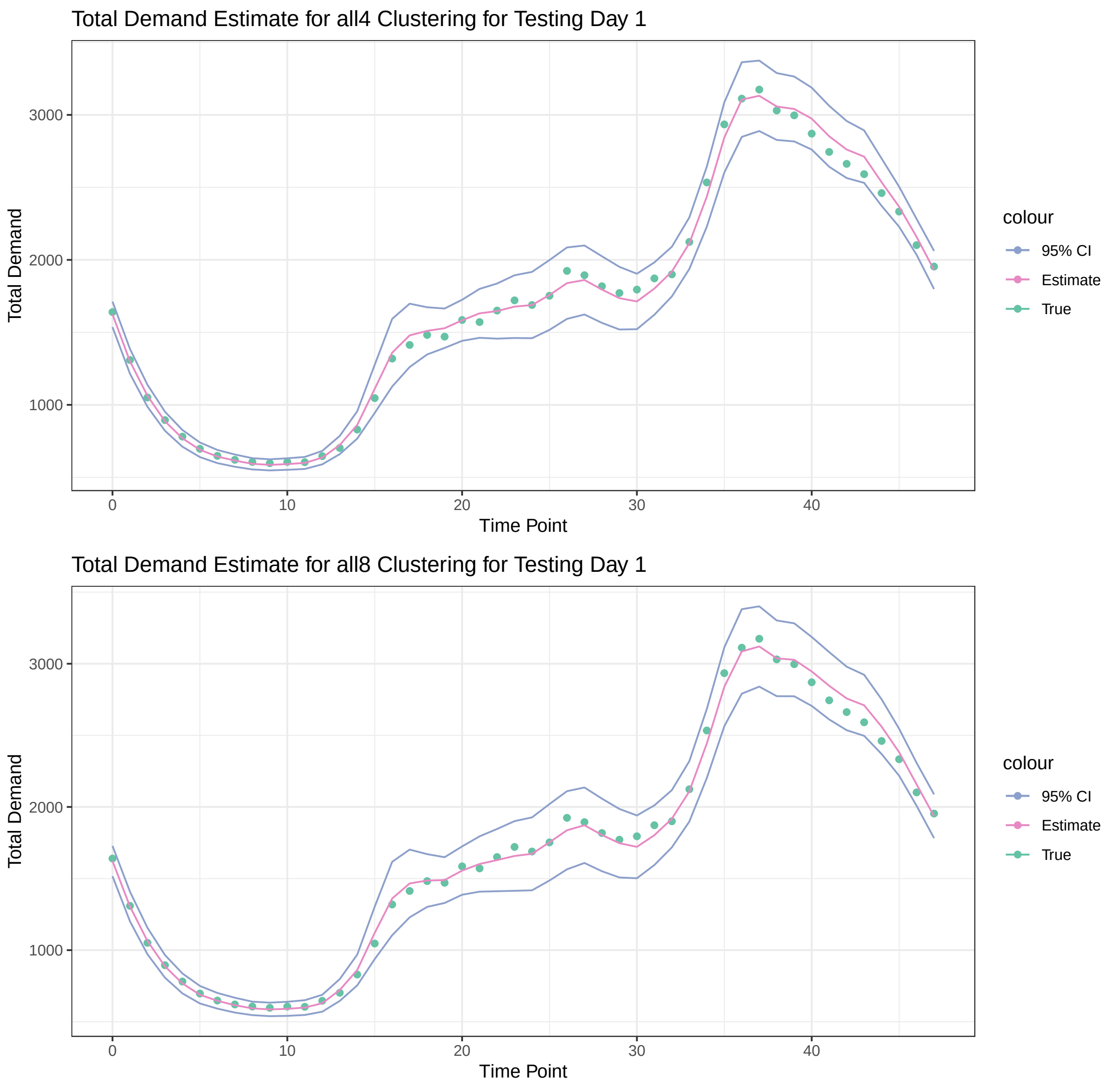


Figure 4: Plot comparing total demand estimate and confidence intervals for all data clustering method with 4 and 8 clusters respectively.

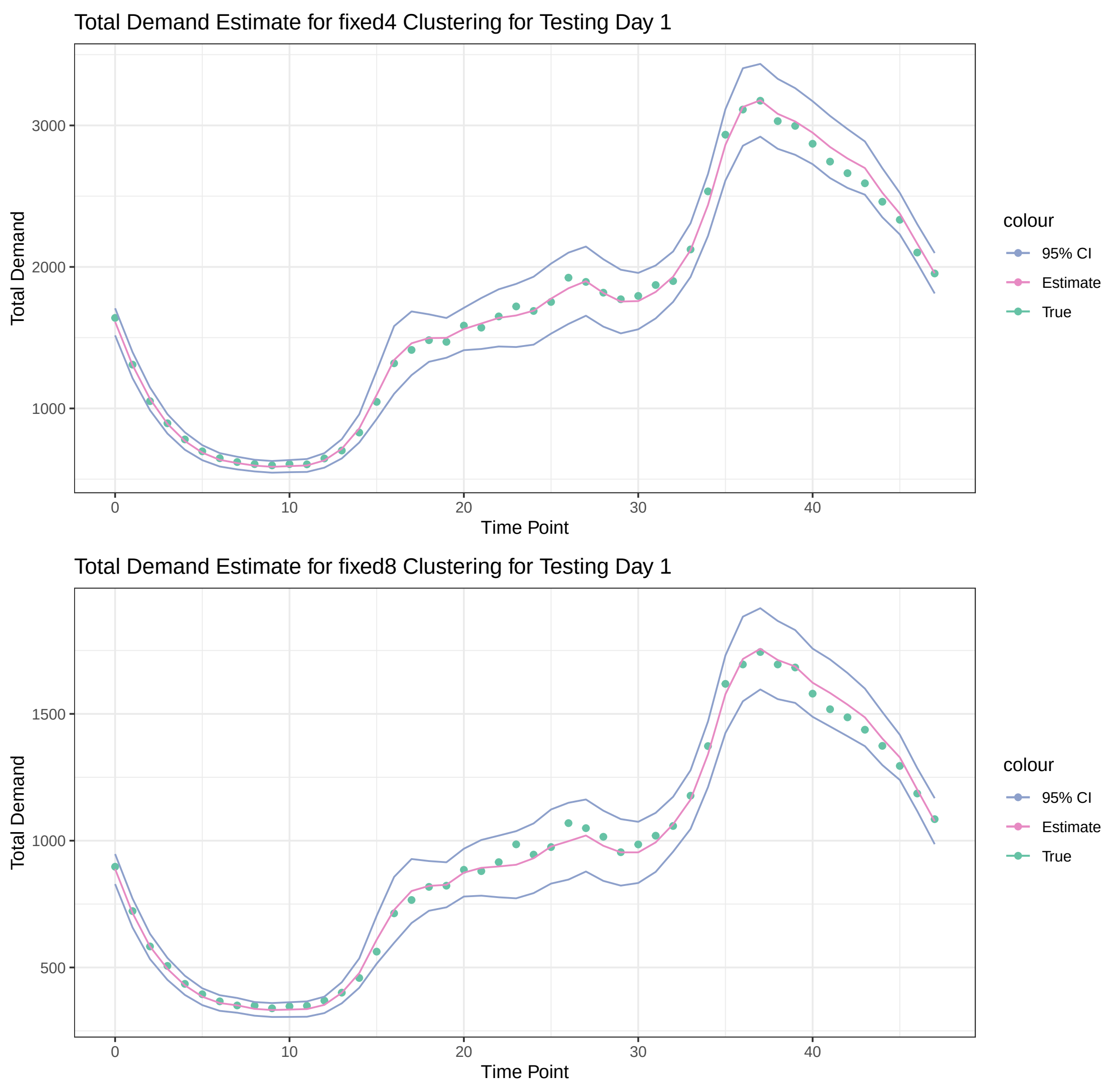
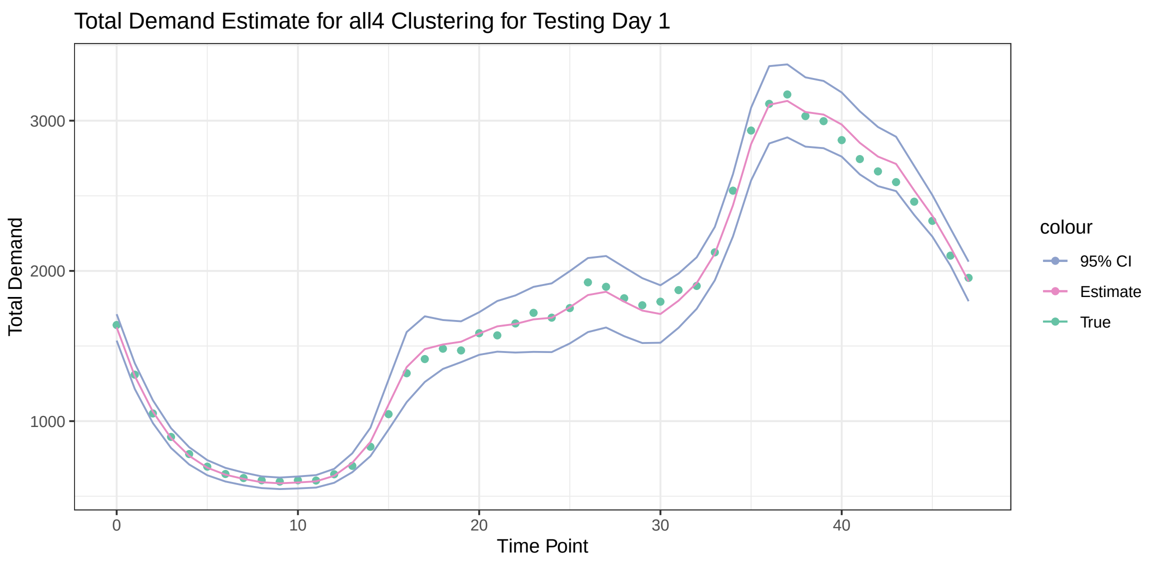


Figure 5: Plot comparing total demand estimate and confidence intervals for fixed data clustering method with 4 and 8 clusters respectively.



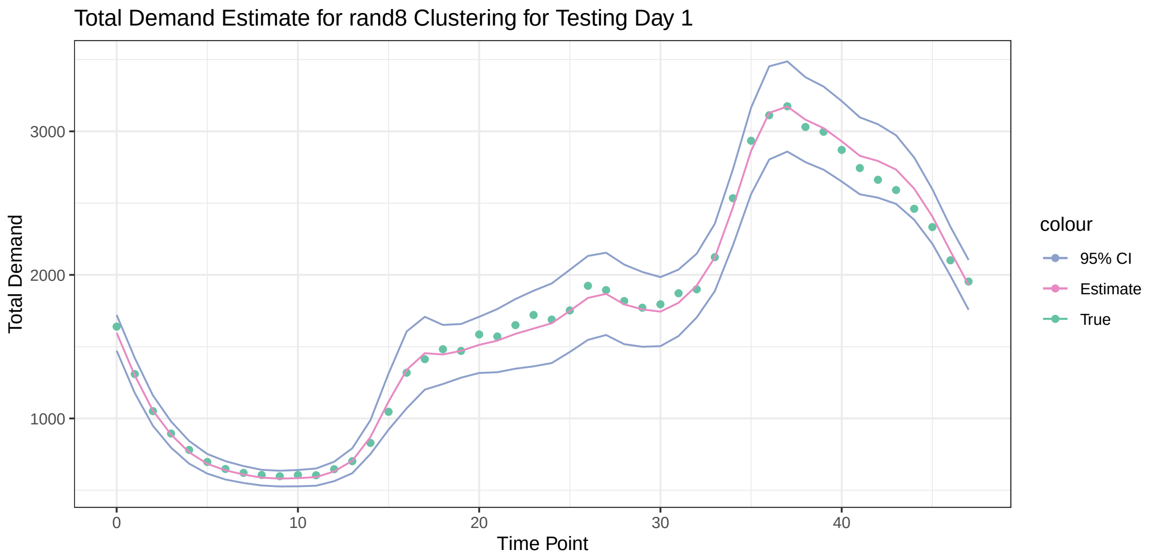
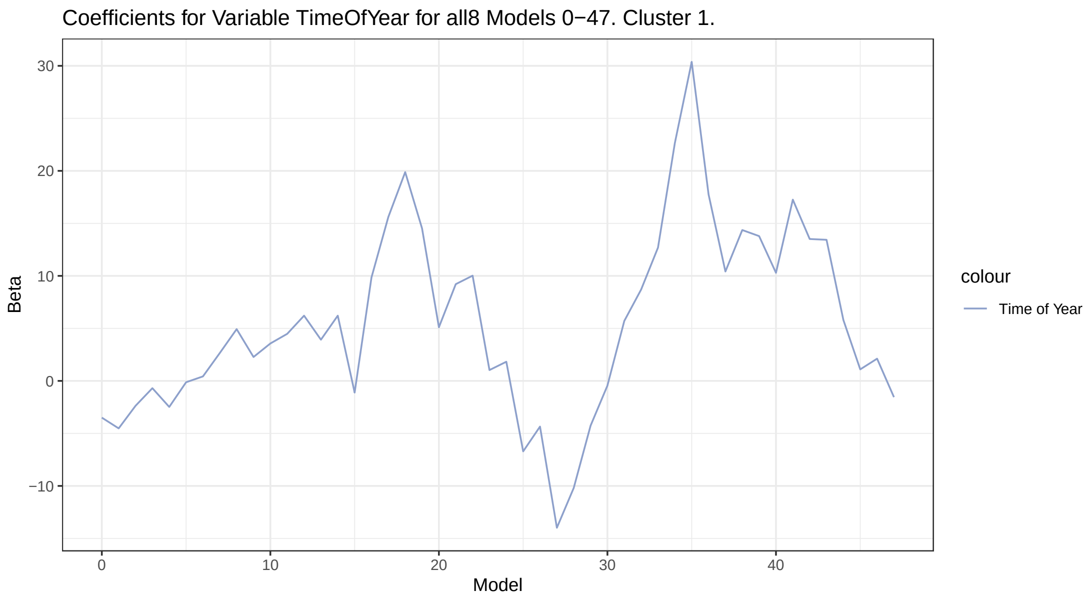


Figure 6: Plot comparing total demand estimate and confidence intervals for random clustering and hierarchical (all data) for 8 cluster methods.

We can see that the fit for the earlier time points is very accurate, this is likely to do with it being estimated by time points closer to it (as it uses all day before time points), with the tighter confidence intervals agreeing. From the plots we can't see significant difference based on the clustering methods. We will explore the effectiveness of these clustering methods later in this report.

Viewing the effects of ‘Time of Year’, ‘Weekday’ and ‘Weekend’ on models for time points 0-47, we can see that their effects are dependent on the time of day. Implying that our choice of fitting the models separately for each time of day has merit.



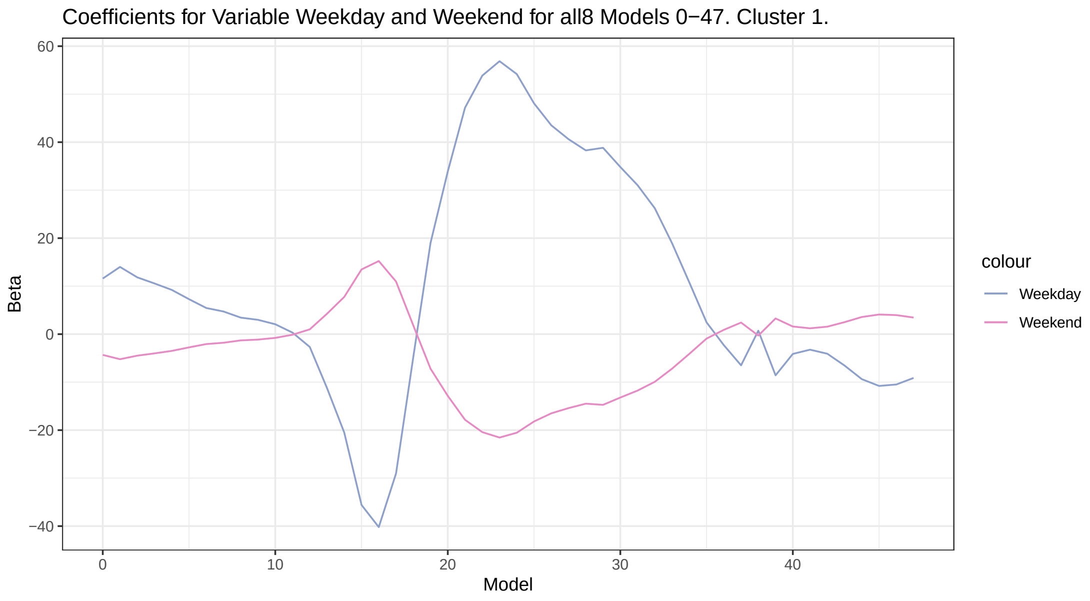


Figure 7: Plots showing the coefficients for ‘TimeOfYear’ and ‘Weekday’ ‘Weekend’ for each model in cluster 1 (using 8 clusters and all the data).

We calculate the RMSE by calculating the RMSE for the total demand for each day (for each clustering type) and taking the mean over them. While it is hard to tell from the plots, there is a noticeable improvement over all testing data for all clustering over random clustering in the 8 cluster case, as we will see below in Table 1. We note, however, that further analysis should be done with a larger testing data set.

	ALL	**DEM**	**FIX**	**RAN**
4 Clusters	131.39	126.05	125.81	131.09
8 Clusters	120.61	127.57	126.69	139.51

Table 1: Table showing the average RMSEs for each of the different clusterings investigated.

Table 1 and Figure 8 show the mean values and density estimates of the RMSE of the aggregated models, fitted using our different clustering strategies. All of our HAC based models outperformed the random clustering models, with the exception of that fitted using all of the data with only 4 clusters. This had the largest average RMSE of all of the HAC models. This suggests that generally, our global model performs better when we form sub-populations based on characteristics rather than random allocation. We see that the model fitted using 8 clusters, taking into account all of the household information, has the smallest average RMSE. Density plots have thinner tails moving from 4 to 8 clusters, suggesting improvement in our model, however further investigation into different cluster numbers and larger test sets would be required to validate this conclusion as differences are small.

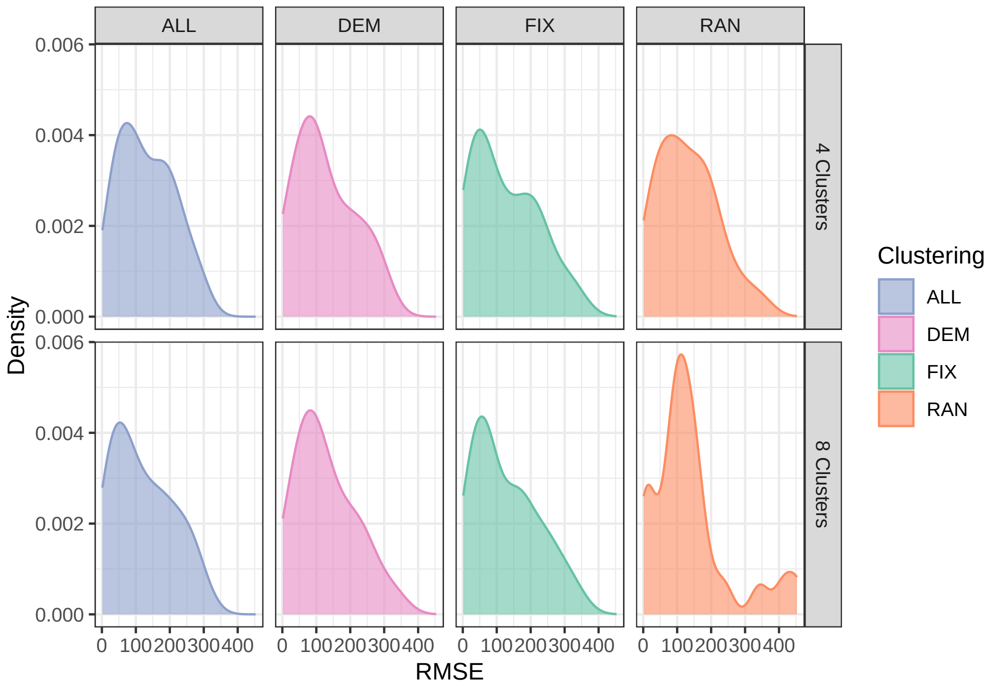


Figure 8: Plots showing the kernel density estimates of the RMSE values of each of the different clustering based models considered.

Cluster	1	2	3	4	5	6	7	8
Count	914	401	105	1196	48	4	3	1

Table 2: Table showing the number of households in each cluster, produced using all household information and partitioning with 8 clusters.

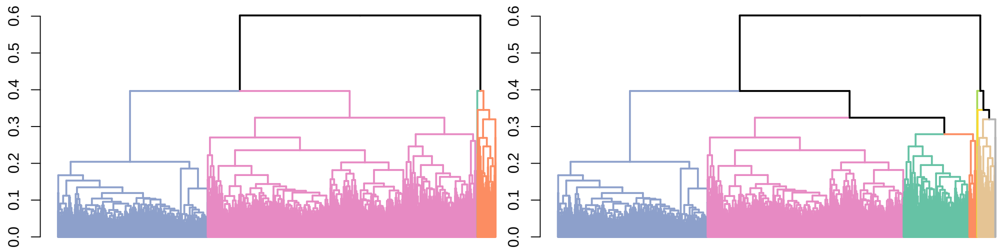


Figure 9: Dendrograms showing the clustering results using all extra information. Left, with 4 clusters. Right, with 8 clusters.

Figure 9 shows the dendrograms associated with the HAC methods that had the highest and lowest average RMSE values. Additional dendrograms of the other clusterings considered can be seen in Appendix A.3

In Table 2 and Figure 9 we observe that, as mentioned previously in Section 2.1.3, the clusters for this model are quite imbalanced in size. This may have lead to models with large errors being produced for some of the clusters, and inflated the overall errors in our aggregrated global model. We discuss ways to address this issue in Section 5

5 Conclusions

In this work, we aimed to create a framework for electricity forecasting, based on clustering by household demand profiles in order to improve model performance. We have seen that there are clear advantages to fitting with clustered sets of data, both in terms of capturing more individual level behaviour and in terms of computational efficiency. Grouping the households by their weekly demand profile as well as their characteristics proved to be a simple but effective way of clustering the data set, which outperformed simple random grouping in almost all cases.

Our investigations were limited in their scope, however. In order to mitigate the limitations of HAC in relation to cluster sizes, seen in Section 4, different clustering methods such as k-means clustering and spectral clustering for demand data could also be investigated. Post-processing steps may involve splitting larger clusters or merging smaller ones to achieve more balanced cluster sizes, and refitting our models according to those.

Methods such as principal component analysis and multiple correspondance analysis could also be explored in order to reduce the dimensionality of the demand and survey data, however initial exploration of these methods found they lacked the ability to capture the variance in our data sets and so we elected not to use them here.

Further work may also wish to extend our hierarchical clustering approach. For example with increased computational power we could, for each of the sets of variables, generate a sequence of partitions of the data into $k \in \{2, \ldots, 10\}$ clusters, or $k \in \{2^p : p = 0, \ldots, 10\}$ clusters, using HAC. We could then compare the RMSE across these different partitions. We could also elect to compare different linkage methods within HAC, such as single and average linkage.

In terms of the models used for forecasting, we could of experimented further with different priors and prior structures. For example, a hyperprior could of been used on the variance parameter of the coefficients β. However, training further models would of taken a considerate amount of time.

In addition, further work could include carrying out feature selection by using the model evidence, that can be computed in the Bayesian setting. Another idea would be to fit a global model, and incorporate characteristics of this model as features in the individual models on the clusters. The idea is that the global model would include the shared information between the households. Then for a given model, we could sample from the predictive posterior to compute our confidence intervals.

In conclusion, we have demonstrated the analysis steps required to forecast electricity demand using a combination of clustering methods and Bayesian modelling, and have produced a documented and tested R package utilising Stan and parallel programming to aid in this analysis. We conclude that, whilst further exploration is required to determine optimal practises, there is clear benefit to utilising additional individual-level information for global demand forecasting.

References

[1] A. Antoniadis, J. Cugliari, M. Fasiolo, Y. Goude, and J.-M. Poggi. *Statistical Learning Tools for Electricity Load Forecasting*. Not yet published, 2022.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, 2006. ISBN 978-0-387-31073-2.

[3] Commission for Energy Regulation (CER). CER Smart Metering Project - Electricity Customer Behaviour Trial, 2009-2010 [dataset]. Irish Social Science Data Archive, 2012. URL `https://www.ucd.ie/issda/data/commissionforenergyregulationcer/`. SN: 0012-00.

[4] J. C. Gower. A General Coefficient of Similarity and Some of Its Properties. *Biometrics*, 27(4):857–871, 1971. ISSN 0006-341X. doi: 10.2307/2528823.

[5] Stan Development Team. RStan: the R interface to Stan, 2023. URL `https://mc-stan.org/`. R package version 2.21.8.

[6] N. K. Vishnoi. An Introduction to Hamiltonian Monte Carlo Method for Sampling. (arXiv:2108.12107), Aug 2021. doi: 10.48550/arXiv.2108.12107. URL `http://arxiv.org/abs/2108.12107`.

A Appendix

A.1 Bayesian regression predictive distribution

$$p(y^* \,|\beta, \mathbf{y}; \mathbf{x}^*, \mathbf{X}) = \int p(y^* \,|\beta; \mathbf{x}^*, \sigma^2) p(\beta \,|\mathbf{y}; \mathbf{X}, \sigma^2, \sigma_\beta^2) \, d\beta$$

This can the be computed using the Gaussian identity ([2], equation 2.115)

In a fully Bayesian setting it is possible to generate draws from this posterior, can carry this out in Stan, example is shown here.

A.2 Invariant distribution of HMC

To show that $\tilde{\pi}(\theta, v)$ is the invariant distribution, we need to show that if $(\theta, v) \sim \tilde{\pi}$, then $T_s((\theta, v)) = (\theta', v') \sim \tilde{\pi}$. Can show this by using the change of variable formula:

$$\begin{aligned} p\left(\theta', v'\right) &= \left|\det\left(\boldsymbol{J}_{T_s^{-1}}\left(\theta', v'\right)\right)\right| \tilde{\mu}(T_s^{-1}\left(\theta', v'\right)) \\ &= \tilde{\mu}(T_s^{-1}\left(\theta', v'\right)) \\ &= e^{-H(T_s^{-1}(\theta', v'))} \\ &= e^{-H(\theta, v)} \\ &= e^{-H(\theta', v')}. \end{aligned}$$

The reversibility property allows us to take the inverse of T_s, the volume preservation property means that the Jacobian is equal to one. And the final equality is using that the energy is conserved in the Hamiltonian, it does not change with time.

A.3 Additional HAC dendrograms

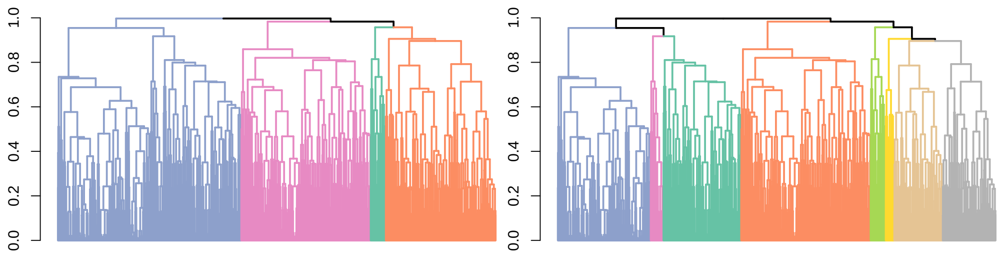


Figure 10: dendrograms showing the clustering results using only fixed household characteristics from the `survey` data. Left, with 4 clusters. Right, with 8 clusters.

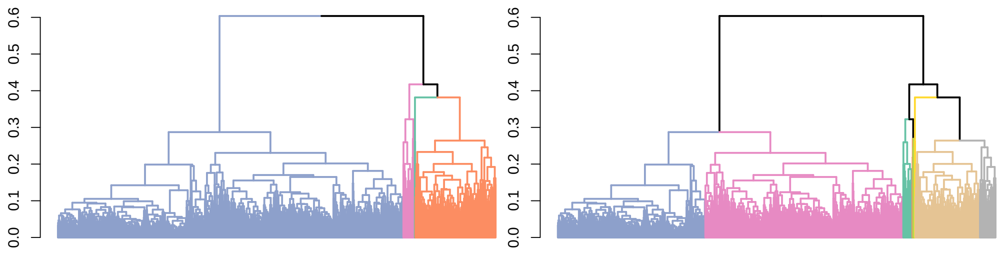


Figure 11: dendrograms showing the clustering results using only demand based information. Left, with 4 clusters. Right, with 8 clusters.