Modelling Peak Electricity Demand in Great Britain

Group 30, Logan Hinkley (s2146776), Matthew McCrea (s2157060), Tadhg Jones (s2236397)

Word Count: 2985

Introduction

In this report, we develop a linear regression model to generate realistic forecasts of peak daily electricity demand. Our goal is to provide NESO with a tool to better anticipate high-demand periods and avoid potential supply shortfalls. The analysis is based on historical demand and weather data such as temperature, wind and solar capacity throughout the winter period.

We begin by identifying significant variables in the provided datasets, of hourly temperature data in `SCS_hourly_temp.csv` and other environmental and demand data `SCS_demand_modelling.csv` through visualisations and linear regressions. We also improve on NESO's choice temperature variable by creating our own temperature variable which has more exploratory power.

We conducted a series of statistical tests for heteroskedasticity (variance changes across the model) and autocorrelation to improve and refine our model for greater parsimony before comparing it to historic data and cross validation, using multiple methods across months and weekend/weekday splits. Robustness is further tested by simulating the model predictions across different years in the winter of 2013. We pay particular care in its accuracy for the top 5% of peak daily demand observations to ensure that future planning is less likely to create shortfalls.

Our model provided a marked increase in performance over the current specification used by NESO, where we found an increase in the adjusted R^2 and a decrease in the the RMSE highlighting that our model was better suited to the data. Our model was able to explain 93.60% of the variation in the data highlighting the strong fit as well as massively reducing the RMSE when predicting the top 5% of peak demand which is the area NESO are looking to investigate. But we also discovered limitations within the model for its long term prediction due to our model only considering historical data as well as not considering certain external factors outwith the dataset which need to be taken into consideration as well.

Data description and exploratory analysis

We are provided with two datasets, (`SCS_demand_modelling.csv`) and (`SCS_hourly_temp.csv`). `SCS_demand_modelling.csv` includes demand and environmental data for the winter months (November to March) from 1991 to 2014. `SCS_hourly_temp.csv` contains the population weighted average hourly temperature for all dates in the same time period.

Do Solar or Wind Impact Demand?

The variable Solar S represents the estimated capacity factor of solar generation based on solar outputs on the given date at 6pm, with installed solar generation as at 1 January 2015. It is reasonable to assume that there is a negative correlation between the solar levels and electricity demand as people use less heat and light when the sun is out. The wind variable is the estimated capacity of wind generation at 6pm on a given day based on windspeed and historic data. We plot both of these variables against demand and regress them in a univariate regression as a formal test of significance below.

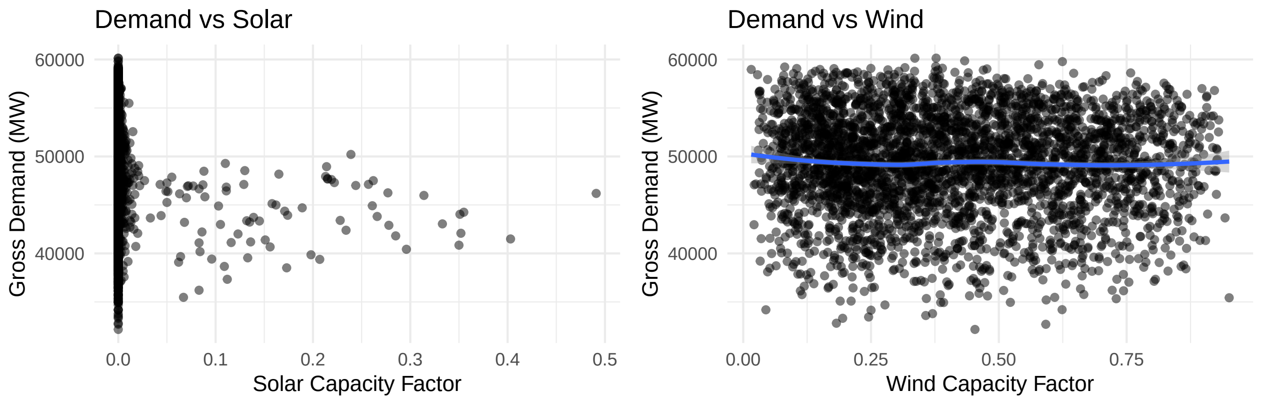


Table 1: Univariate Regression Results for Solar and Wind

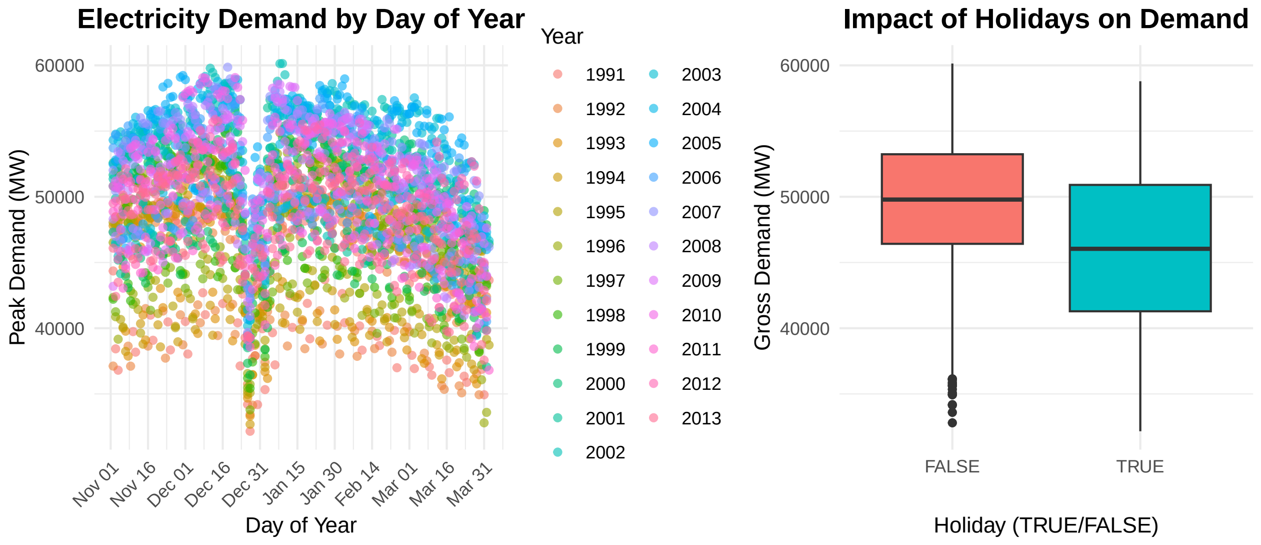
term	estimate	std.error	statistic	p.value	Model
(Intercept)	49394.2145	86.8334	568.8386	0.0000	Solar
solar_S	-24189.6123	2988.0722	-8.0954	0.0000	Solar
(Intercept)	49518.1512	181.8960	272.2334	0.0000	Wind
wind	-538.2084	384.4493	-1.3999	0.1616	Wind

We find that Solar_S is statistically significant at the 1% level with a negative correlation. We will include it in our preliminary model as a result. The wind variable is not significant at the 10% level, and as such will not be included.

How Demand Changes Over Time

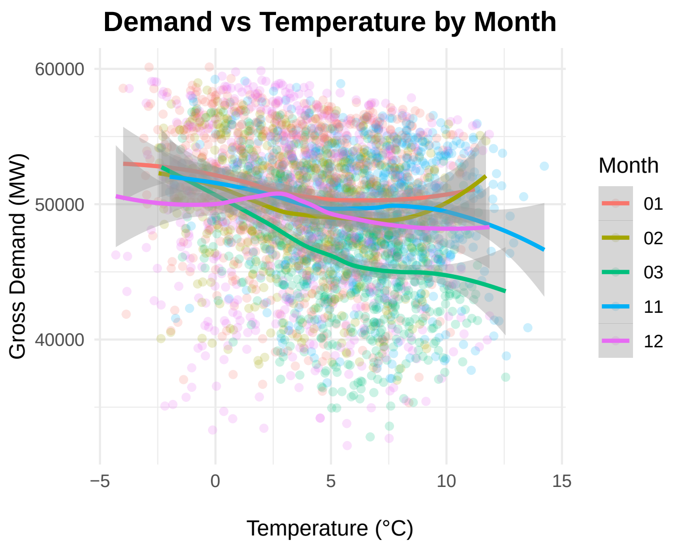
To understand how electricity demand evolves through time, we consider the variable DSN, which counts the number of days since November 1st. This provides a simple way to track time consistently across years.

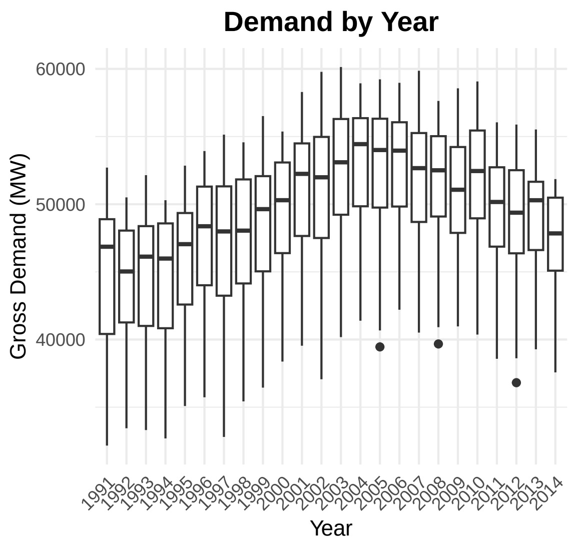
In the figure below on the left, we plot gross electricity demand against DSN, overlaying data from all years in our dataset. This visualisation shows a sharp drop in demand around the holiday period, consistent across years. This fall begins in late December and recovers shortly after the New Year. There are also systematic differences in demand between years, with each winter showing distinct demand levels.



To isolate the holiday effect, we define a holiday window from December 23 to January 3 and exclude these dates from certain parts of our analysis. The above box plot of demand inside vs outside holidays reinforce this conclusion. Excluding these days helps prevent the temporary dip in demand from distorting our understanding of longer-term trends. As we are most concerned with avoiding shortfalls excluding these low demand days does not majorly inhibit our predictions in our use case. Since temperature and demand change together over time, and the holiday period disrupts this trend, we explore whether each month has a distinct effect on demand beyond just temperature.

Below we plot temperature against daily peak demand grouped by month as well as the year against peak daily demand. March differs most from the other months with a steeper decline in demand for temperatures above 0ºC, and we see a gradual increase in peak demand over the years until a peak around 2004 where we then start to see another decline.





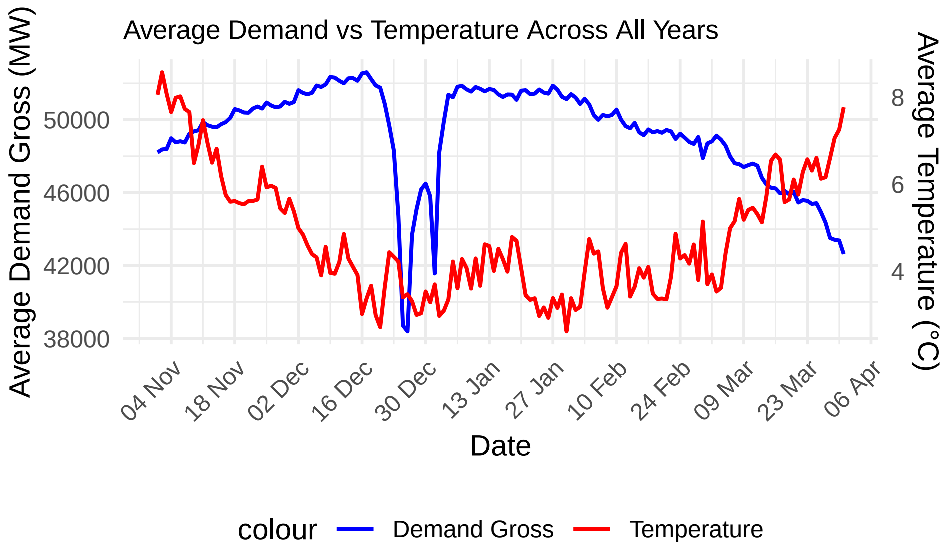
This variation suggests the relationship between temperature and demand is not consistent across months. We will include interaction terms between month and temperature in our model to capture this effect. We can also see that the historic data from the year also impacts out data so we need to introduce a start year variable such that the model takes this into account.

A possible explanation for March's deviation is seasonal behaviour as spring begins people may reduce heating use regardless of actual temperature.

To account for the gradual changes over time and differences across years, we include DSN and its square to allow for nonlinear seasonal effects, the month of each observation and its interaction with temperature to further capture time variant effects, and a dummy for each year to capture larger changes due to population or climate change.

How does peak daily demand depend on temperature?

To investigate the relationship between temperature and electricity demand, we plot average daily demand and temperature across all years. As shown in the figure below, both follow a seasonal pattern—temperature declines into mid-winter while demand rises, except for a sharp drop during the holiday period.



As shown in above, both variables follow a seasonal pattern as temperature declines into mid-winter while demand rises, except for the holiday period.

Table 2: Comparison of Univariate Temperature Models

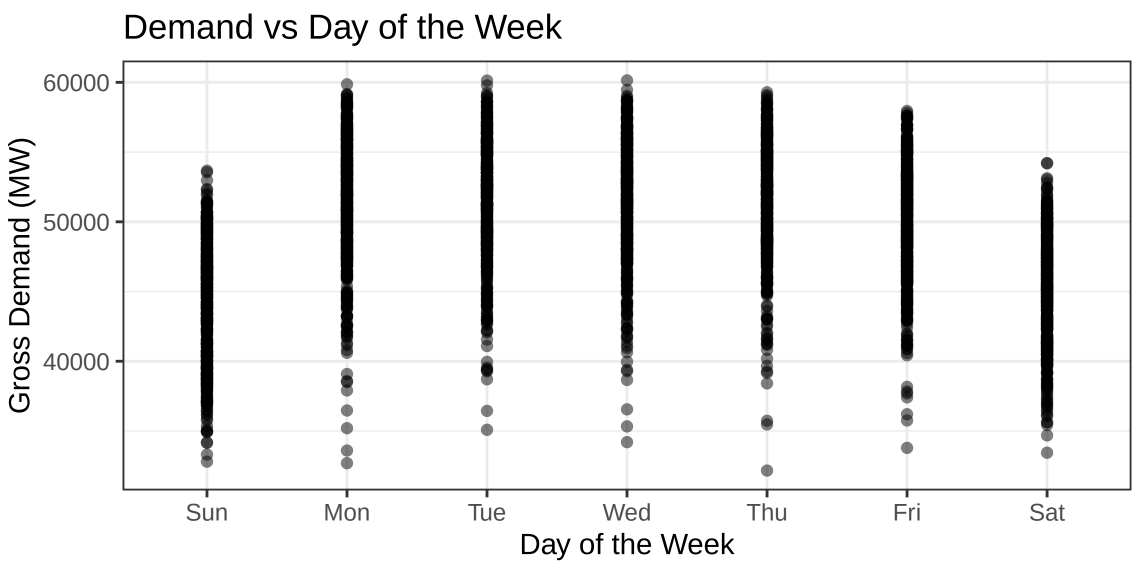
Model	estimate	std.error	statistic	p.value	R2
TE	-476.1891	27.92510	-17.05237	0	0.07718
TO	-394.1328	25.11460	-15.69337	0	0.06615
Temp	-329.0379	25.83818	-12.73456	0	0.04456
TA	-924.4153	50.19628	-18.41601	0	0.08887

To quantify this relationship, we regress the temperature metrics from the dataset of temp, TO, TE on peak daily demand. All yield large, significant negative coefficients ($p < 0.01$), supporting a strong inverse relationship between temperature and demand. Results are summarised in Table 1.

To improve on NESO's current metric (TE), we tested various temperature windows and combinations from day of and day before temperature using the hourly temperature data. After iterating over all combinations TA, a weighted average of today's and yesterday's 16:00 temperature readings (weights 0.4 and 0.6 respectively) performed best (see Table 1). This suggests that NESO's current approach can be improved. As TA relies only on reliable hourly data, we adopt it as our temperature variable going forward.

Do days of the week matter?

The weekday/weekends may exhibit different peak daily demands that must be accounted for due to working patterns. We plot the peak daily demand on each day of the week below.



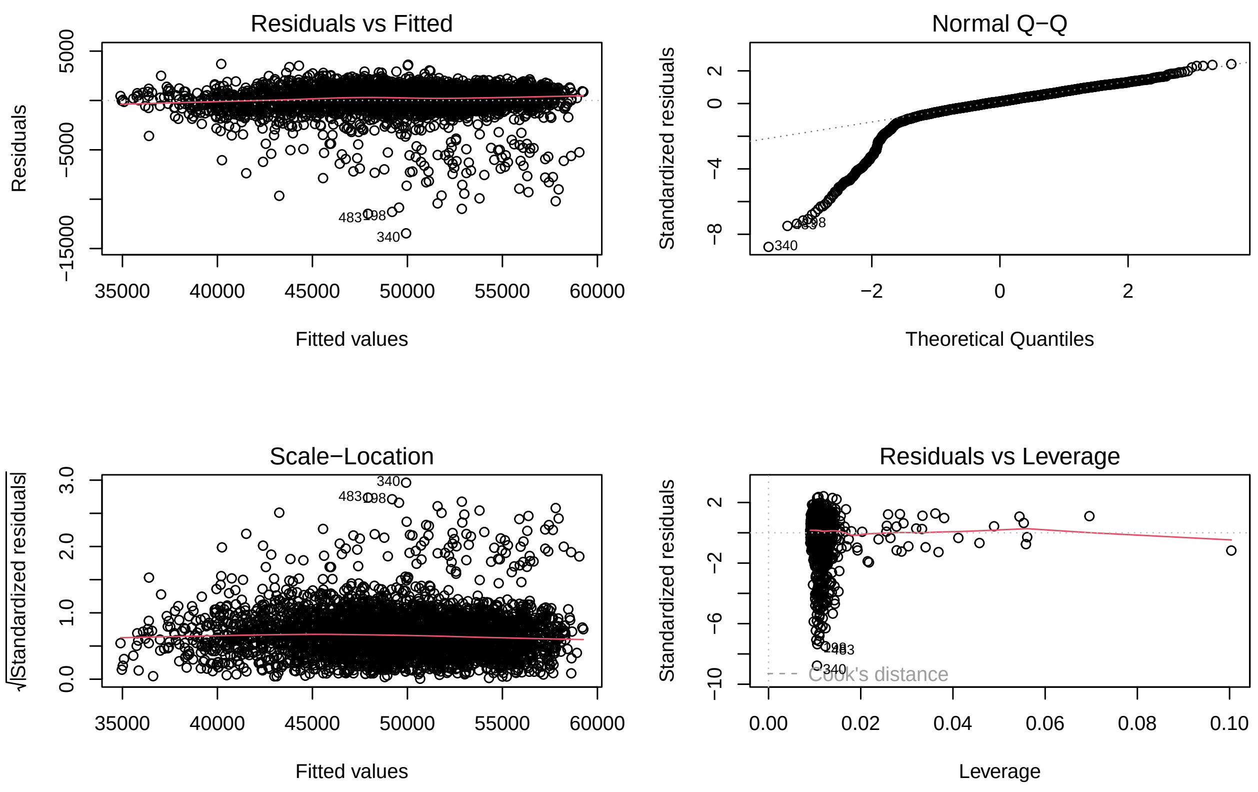
This figure displays a marked difference in the peak daily demand of weekdays and weekends with weekdays consistently higher than weekends. This is not a formal test but is compelling enough to include a weekday variable in our initial model.

Model Fitting

The exploration of the data leads us to construct a model different from the current NESO model. We define our our model as the following

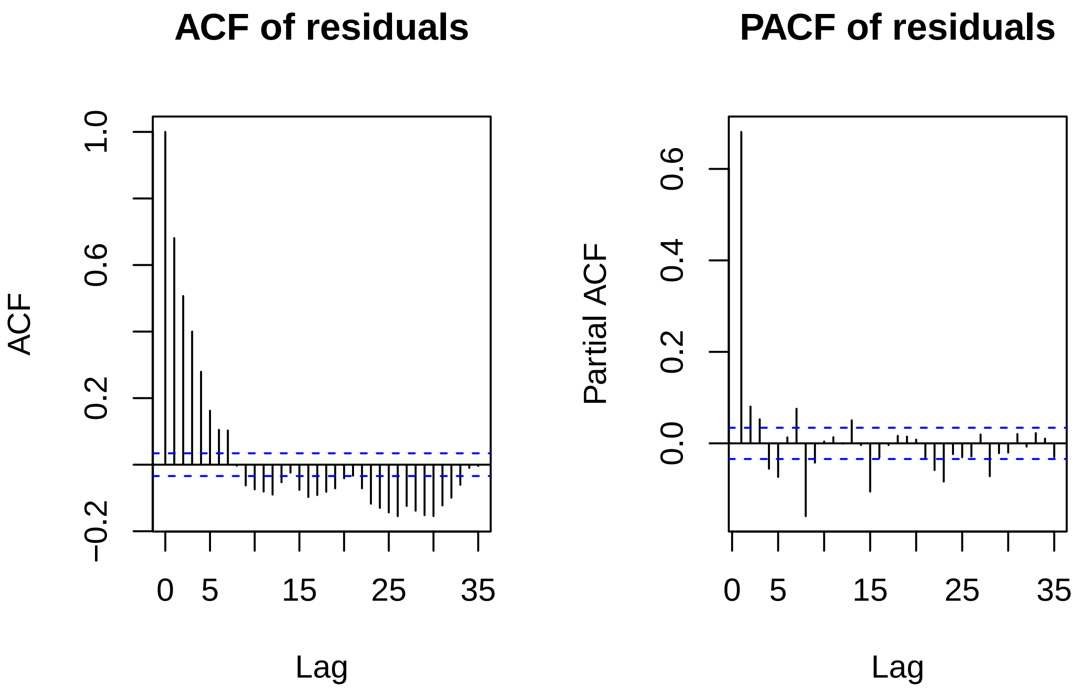
$$D_t = \beta_0 + \beta_1 TA + \beta_2 Solar + \sum_{i=1}^{6} \gamma_i \, Weekday_i + \beta_3 \, Month + \sum_{j=1992}^{2013} \omega_j \, Start\,Year_j + \beta_4 \, DSN + \beta_5 \, DSN^2 + \beta_6 \, TA * Month + \epsilon, \quad \epsilon$$

Where D_t is the daily peak demand, TA is the custom temperature variable, and $Solar$ is the solar capacity variable. The variables of $Weekday$, and $Start\ Year$ are dummies indicating their value. The baselines are Sunday and 1991 respectively. $Month$ is a discrete numerical variable ranging from 1 to 5 where November is 1 and March is 5 and the $TA * Month_k$ term is an interaction between the temperature and month, while ϵ represents the residuals of our model, assuming normal distribution. We examine residual plots for heteroscedasticity, autocorrelation, and normality below.



The Residuals vs Fitted and Scale-Location plots may show increasing variance with higher predicted demand which biases standard errors. A Breusch-Pagan test for non homogeneous variance (heteroscedasticity) confirms this (BP = 238.57, df = 34, p < 0.0001). The Q-Q plot shows non-normality in the lower tail while no influential points are flagged via Cook's Distance.

We plot ACF and PACF to test for autocorrelation below. This checks if the residuals of a model are correlated. This can indicate persistence in shocks or signal ommitted variables which can bias standard errors:



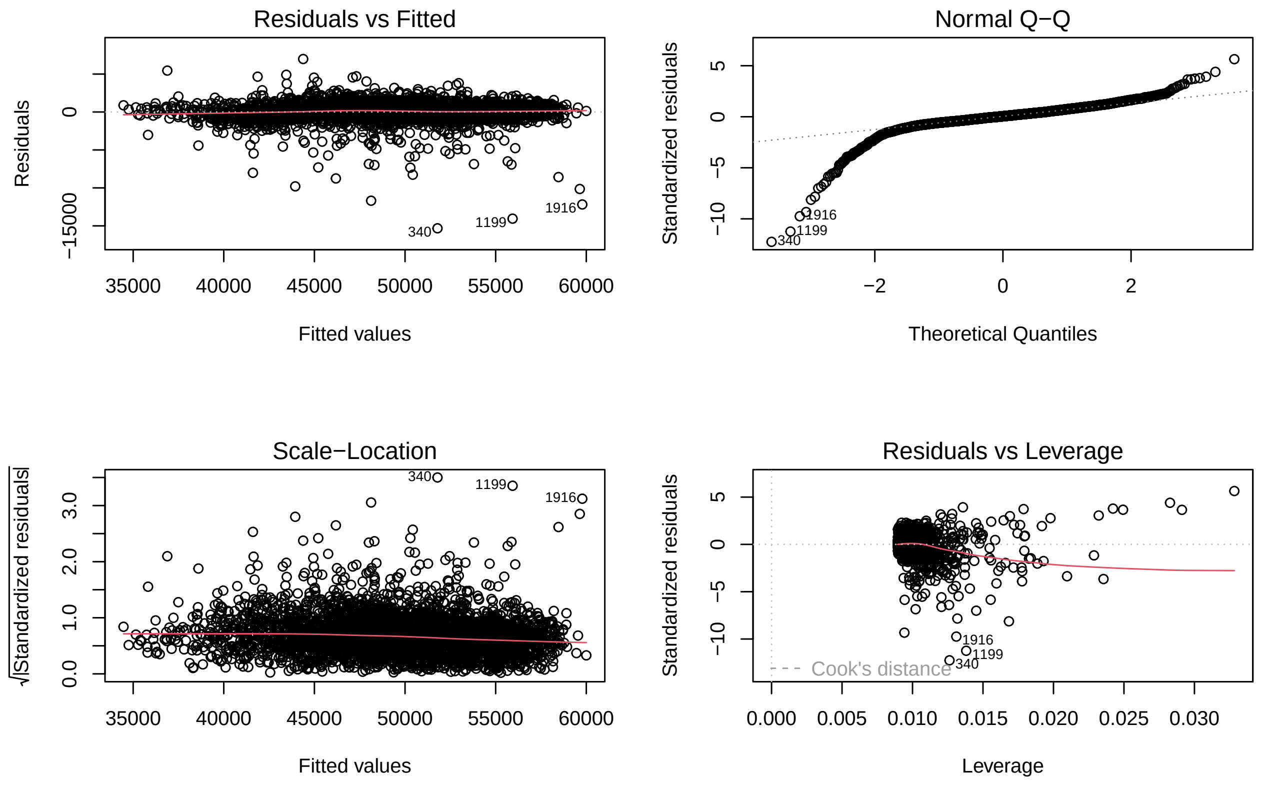
The ACF/PACF plots reveal clear autocorrelation. This biases standard errors, risking spurious significance.

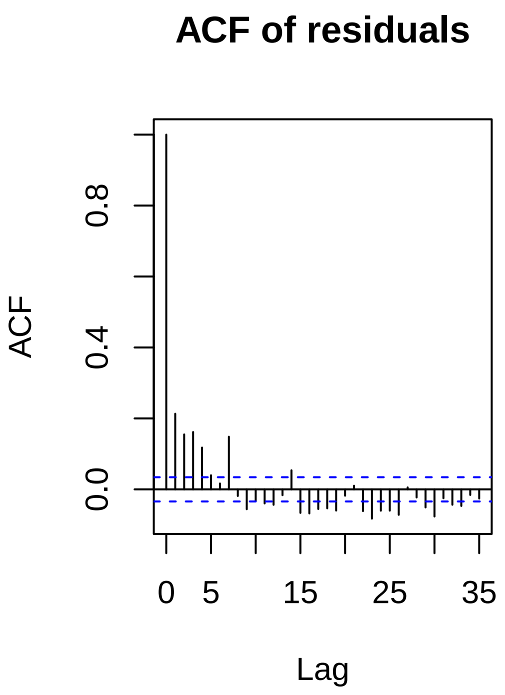
To address this, we add a lag of peak demand which lag significantly reduces the autocorrelation, however, additional lags (up to 20) did not substantially improve the plot. We apply Newey–West robust standard errors which adjust the error for both heteroscedasticity and autocorrelation so that bias is removed.

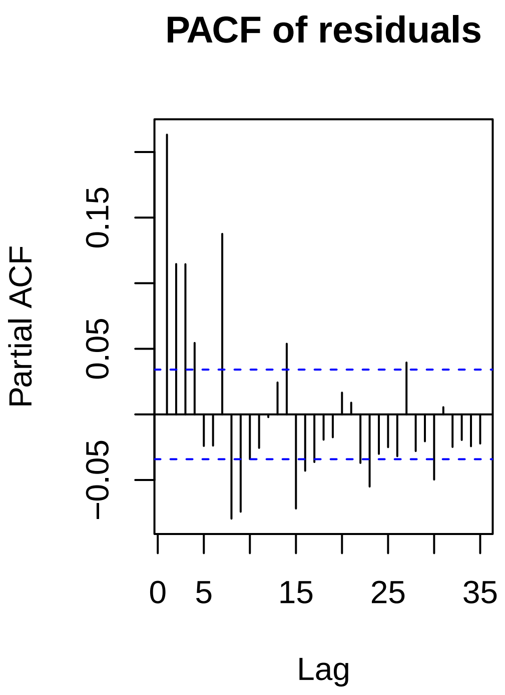
We recalculate the confidence intervals for all coefficients using Newey–West robust standard errors according to the default lag specification in R. We aim to remove any only marginally significant variables for parsimony, the balance of predictive ability and simplicity in a model. The results show that Solar, the TA–Month interaction, and DSN are only marginally significant ($p > 0.05$), so we remove them.

We specify the following model and plot its residuals and autocorrelation plots below:

$$D_t = \beta_0 + \beta_1 TA + \sum_{i=1}^{6} \gamma_i \, Weekday_i + \beta_2 \, Month + \sum_{j=1992}^{2013} \omega_j \, Start\,Year_j + + \beta_3 \, DSN^2 + \beta_4 \, D_{t-1} + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$







This updated model shows significantly reduced autocorrelation, though heteroscedasticity remains, with a Breusch-Pagan statistic of 177.28 on 35 degrees of freedom ($p < 0.01$). The Q-Q plot also reveals increased deviation from normality, now affecting the upper quartiles as well. Despite the violations of the OLS assumptions on heteroscedasticity, autocorrelation, and normal residuals our point forecases are unaffected and Newey–West robust standard errors account for heteroscedasticity and autocorrelation in standard errors.

We adopt this as our final model for forecasting peak daily demand.

The coefficients for this estimation are below:

Table 3: Model Estimates

Greek...1	Estimate...2	Std. Error...3	t value...4	Pr(>|t|)...5	Greek...6	Estimate...7	Std. Error...8	t value...9
β_0	23522.700	1127.853	20.856	0.000	ω_{1999}	2521.995	284.804	8.855
β_1	-444.160	35.072	-12.664	0.000	ω_{2000}	3291.406	240.901	13.663
β_4	0.471	0.024	19.484	0.000	ω_{2001}	3788.888	206.049	18.388
γ_1	7556.547	143.358	52.711	0.000	ω_{2002}	4336.154	252.296	17.187
γ_2	4487.540	180.418	24.873	0.000	ω_{2003}	4624.190	288.613	16.022
γ_3	4327.171	180.660	23.952	0.000	ω_{2004}	4518.260	323.925	13.948
γ_4	4175.942	184.597	22.622	0.000	ω_{2005}	4582.762	302.498	15.150
γ_5	2706.242	180.524	14.991	0.000	ω_{2006}	4106.277	231.970	17.702
γ_6	-1386.430	126.313	-10.976	0.000	ω_{2007}	4388.136	252.319	17.391
β_3	-0.153	0.008	-19.099	0.000	ω_{2008}	3387.225	232.001	14.600
ω_{1992}	-111.117	235.062	-0.473	0.636	ω_{2009}	3365.944	223.711	15.046
ω_{1993}	-26.240	248.007	-0.106	0.916	ω_{2010}	3242.146	261.777	12.385
ω_{1994}	580.300	227.841	2.547	0.011	ω_{2011}	2587.132	219.255	11.800
ω_{1995}	1168.020	194.223	6.014	0.000	ω_{2012}	2707.490	203.999	13.272
ω_{1996}	1406.562	200.799	7.005	0.000	ω_{2013}	1956.127	161.366	12.122
ω_{1997}	1876.749	212.755	8.821	0.000	β_2	-101.303	9.583	-10.571
ω_{1998}	2087.641	246.064	8.484	0.000	NA	NA	NA	NA

All parameters now included are significant with p-values $< 1\%$ with the exception of ω_{1992}, ω_{1993}.

We compute the values of R^2, adjusted R^2, RMSE, RSE, and the RMSE for the top 5% of the demand to compare our model's fit to the data. These results are in the table below for the baseline and a new model:

Table 4: Model Comparison

Model	R_squared	Adjusted_R_squared	RMSE	RSE	RMSE_Top_5_Percent
M0	0.5461	0.5443	3334.038	3341.143	4371.3176
New Model	0.9360	0.9354	1252.007	1258.324	823.6638

The R^2 value measures the proportion of variance in the dependent variable that is explained by the model, it is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where n is the number of observations, y_i is the actual value, $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean of the actual values. An R^2 value of 1 implies a perfect model fit, while 0 means the model performs no better than the mean.

The baseline model has an R^2 of 0.5461. The new model has an R^2 of 0.9360 suggesting a substantial improvement in model fit and variance explanation. This is not a foolproof metric for evaluating model quality so we also consider the adjusted R^2 which is adjusted for the number of predictors in the model. It is given by:

$$R^2_{adj} = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1}$$

where p is the number of predictors in the model. The adjusted R^2 rises from 0.5443 in the baseline model to 0.9354 in the new model, confirming a meaningful improvement in fit after accounting for model complexity.

We use RMSE to measure the average prediction error as it is in the same units as the variable and harshly punishes large errors and is given by:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

RMSE drops from 3334.04 MW in the baseline model to 1252.01 MW in the new model, indicating much more accurate predictions.

The RSE or Residual Standard Error, is a measure of how far off the predictions are from the actual values, adjusted for the number of predictors in the model as below:

$$RSE = \sqrt{\frac{1}{n-p}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

The RSE also improves, decreasing from 3341.14 MW to 1258.32 MW and again indicates improved power without overfitting.

Motivated by our focus on helping forward planning avoid energy shortfalls we evaluate performance using RMSE for the top 5% of demand. This value drops from 4371.32 MW in the baseline to 823.66 MW in the new model and supports the hypothesis that we capture peak demand accurately. This error is less than 2% of the average peak daily demand which indicates that it is highly accurate.

Cross Validation

To assess the robustness of our model across different periods, we implement a cross-validation scheme that evaluates prediction accuracy separately for each month and for weekdays/weekends.

For the monthly cross validation we use two methods. These are leave-one-month-out (LOMO) where an entire month is excluded from training and used for testing, and a random 80:20 split within each month, using 80% of the data for fitting and 20% for testing. We report RMSE to penalize large deviations and interval scores to assess predictive uncertainty. Results are presented in the table below.

Table 5: Comparison of Leave-One-Month-Out and 80:20 Cross-Validation Results by Month

month	RMSE Score LOMO	Interval Score LOMO	RMSE Score 80:20 Split	Interval Score 80:20 Split
Nov	3158.47	11727.05	1051.33	4175.78
Dec	3113.55	11208.85	2212.94	8272.71
Jan	2900.44	10469.46	1209.60	4413.63
Feb	2718.51	9732.22	831.84	3771.57
Mar	2596.09	9098.44	1132.01	4299.18

Model performance varies by month. February and March consistently show the lowest RMSE and interval scores, indicating reliable performance. December exhibits the highest error, likely due to residual holiday effects left over despite our exclusion of some days. Performance is notably worse under LOMO than 80:20 which is expected since predicting an unseen month is more difficult than sampling within known months.

For weekdays vs weekends we perform an 80:20 cross-validation in each group. Training on only one group is highly innacurate due to their different baselines so we only use this method. The result of this scheme is below:

Table 6: Cross-Validation Results by Day Type (Weekday vs Weekend)

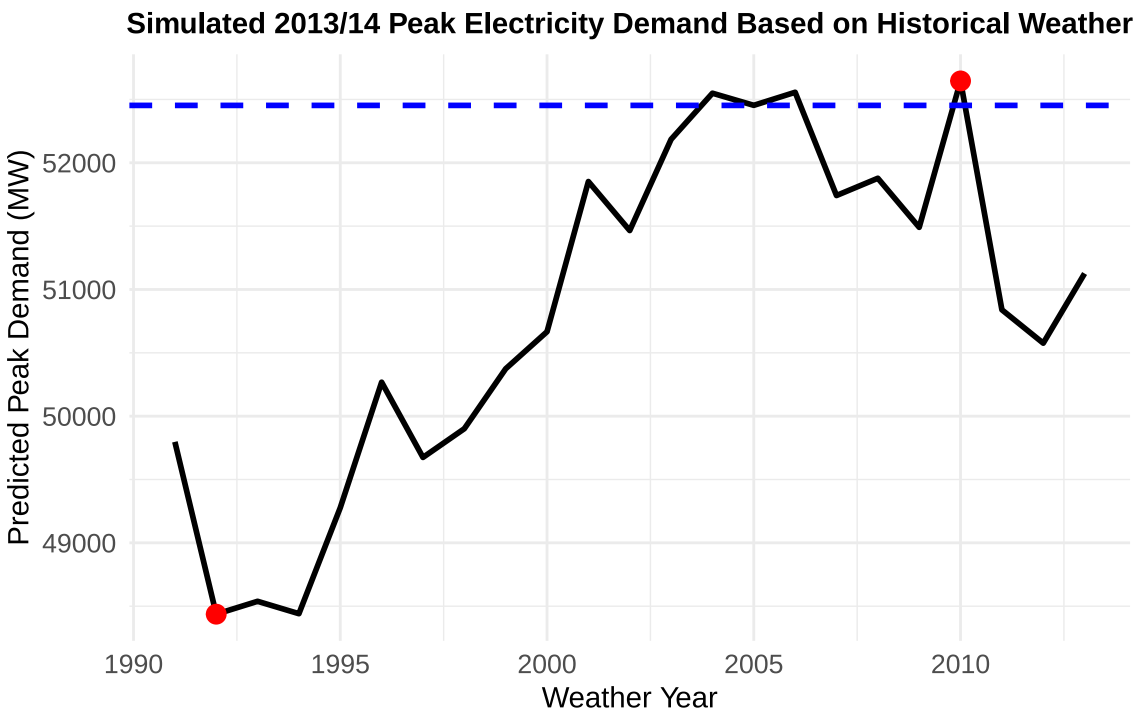
Day Type	Average RMSE	Average Interval Score
Weekday	1262.10	4502.44
Weekend	985.31	3542.62

Our results show weekend predictions to be more accurate, with lower RMSE and interval scores. Weekdays may show greater variability due to irregular work schedules, holidays which would not affect demand on the weekend, or time invariant demand in workplaces (offices, factories) which often have automated heating, cooling, and lighting systems which vary less with external factors and make prediction less accurate.

Across all schemes, our model achieves strong predictive performance. No RMSE exceeds 7% of mean daily peak demand which is promising. However, high December errors require caution as they could reflect either holiday influenced low-demand days or missed high-demand events. Fortunately our prior reporting on the RMSE of the top 5% of observations supports the explanation that this is due to low demand days not high ones.

How could the maximum annual demand have varied in the 2013-14 winter season if different weather conditions had occurred?

To evaluate how peak electricity demand during the 2013/14 winter might have changed under different weather conditions, we trained our model on historical data from years prior to 2013/14. We used this model to predict 2013/14 peak demand using weather data from earlier years (1991/92–2012/13) to observe the 2013/14 year effect, which was present under the full model.



Red dots denote highest and lowest simulated peaks; dashed line shows actual 2013 peak demand.

Table 7: Simulated vs Actual Peak Electricity Demand (2013/14)

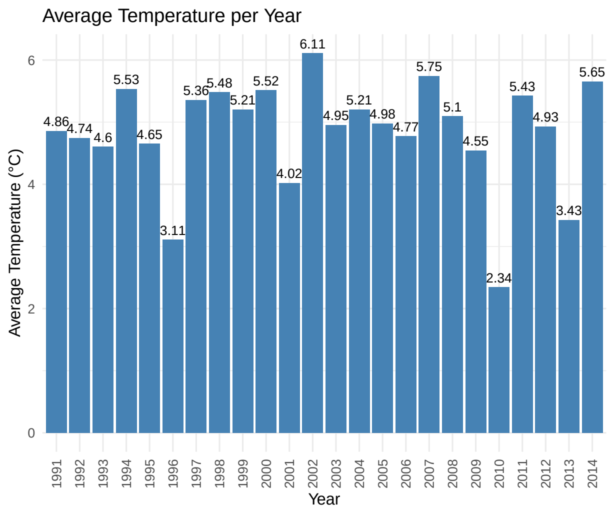
Demand Type	Value (MW)
Highest simulated peak	52646
Lowest simulated peak	48437
Actual peak (2013/14)	**52453**

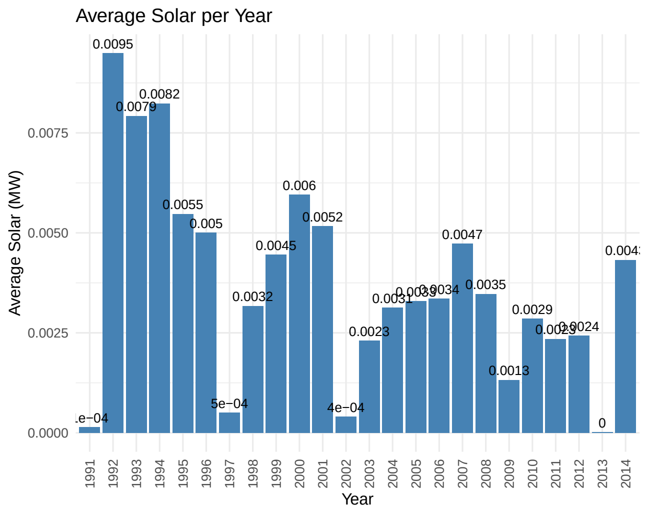
The plot shows substantial variation in predicted peak demand depending on the weather year. Red dots indicate the maximum and minimum simulated peaks, while the blue dashed line marks the actual 2013/14 peak of 52,453 MW.

The highest simulated peak, 52,646 MW, occurred using 2010 weather, which had the lowest average temperature of 2.34°C which makes sense as more heating was required. The lowest simulated peak was 48,437 MW from 1992 which had a mild winter and high solar capacity which may have reduced demand.

Below, we show the average temperature and solar capacity for each year. These match our prediction pattern with high demand in 1991, a dip in 1992, and by a steady rise until 2010. Our simulation also captures the post-2010 decline, consistent with actual demand trends.

This result suggests our model effectively reflects the relationship between weather and peak demand and highlights the importance of stress-testing future demand scenarios under varying weather conditions, especially in the face of climate change. The difference between mild and severe winters spans over 4,000 MW or >10% of average demand which is vital for NESO's planning.





Limitations

The model we created to predict peak electricity demand in Great Britain has some limitations. While it is accurate, several assumptions and exclusions may affect its reliability.

The exclusion of holidays to better predict peak demand is the most striking limitation, especially as they are known to significantly alter demand. As observed, periods like Christmas and New Year show sharp deviations in usage and omitting these days could lead to inaccurate predictions.

The model contains only time and weather variables. Further variables, such as economic growth, energy prices, demographics, and government regulations can have substantial impacts on demand but are not quantified here. Exogenous shocks due to geopolitical events or pandemics are inherently unpredictable and can drastically change usage patterns.

Climate change introduces further uncertainty. As extreme weather becomes more frequent and domestic/industrial appliances adjust, historical weather-demand relationships may no longer hold.

The reliance on historical data presumes that past patterns will continue into the future, which may not be true due to the effects of global warming, technological advancements, or shifts in consumer behavior. The model's predictive performance may be compromised if future demand deviates significantly from historical trends.

Conclusion

The final model we developed outperforms the standard model, M_0, in predicting peak electricity demand. Notable improvements were observed across various performance metrics including R^2, adjusted R^2, RMSE, RSE, and RMSE for the top 5% of demand. The selected model explains a significant portion of the variance in the data and is able to predict peak demand with a high degree of accuracy, making it useful for NESO's long-term planning goals.

One of the key factors contributing to the model's success was the inclusion of lagged demand, which captured autocorrelation effects. This adjustment led to improved residuals and performance, as it better captured the true relationship between weather patterns and peak demand.

It is important to note that the model then under performed in certain areas. The exclusion of holidays and using only weather-related factors could introduce bias, particularly when forecasting peak demand during holiday periods or across major economic/social changes. The model relies heavily on historical data, which may not accurately predict future demand and changes in behavior, such as rapid increases in electricity demand caused by government policy. While the inclusion of lagged demand improved predictive power, it also introduced challenges, such as further deviations from normality in the residuals. This was accounted for with Newey West standard errors but nonetheless limits our modelling.

Despite these limitations, the model provides valuable insights for NESO's long-term planning. It allows NESO to make informed decisions about future energy demand, including investments in renewable energy sources and necessary infrastructure upgrades to model the very peak of demand, which this model is accurate at according to RMSE. However, NESO should remain cautious of the model's assumptions and the potential impact of external factors. Further investigation to provide electricity during the holiday period is required in order to avoid oversupply during that time.

Code Appendix

Include here all the code used in your analysis. Ensure that the code is well-commented so it is easy to follow and reuse.

```r
# Here is all the code to include in the porject

#-----------------------------------------------
#              Load Libraries
#-----------------------------------------------
library(tidyverse)
library(lubridate)
library(estimatr)
library(sandwich)
library(lmtest)
library(rsample)
library(purrr)
library(broom)

#-----------------------------------------------
#               Data Cleaning
#-----------------------------------------------
# Load data
```

```r
demand_data <- read_csv("SCS_demand_modelling.csv")
temp_hourly <- read_csv("SCS_hourly_temp.csv", col_names = FALSE)

# Clean temp_hourly
temp_hourly <- temp_hourly[-1, ]  # remove header row if present again
colnames(temp_hourly) <- c("timestamp", "temp")

temp_hourly <- temp_hourly %>%
  mutate(
    timestamp = parse_date_time(timestamp, orders = "dmy HM"),
    date = as_date(timestamp),
    hour = hour(timestamp),
    temp = as.numeric(temp)
  )

# Clean demand_data
demand_data <- demand_data %>%
  mutate(
    Date = as_date(Date),
    date = Date)

# Define fixed holiday list
holiday_dates <- c("23-12", "24-12", "25-12", "26-12",
                   "31-12", "01-01", "01-02", "01-03")

# Mark holidays
demand_data <- demand_data %>%
  mutate(
    holiday_code = format(Date, "%d-%m"),
    is_holiday = holiday_code %in% holiday_dates
  )

# Exclude holidays
demand_data <- demand_data %>% filter(!is_holiday)

# Extract temp at 16:00
temp_16 <- temp_hourly %>%
  filter(hour == 16) %>%
  select(date, temp) %>%
  rename(temp_16 = temp)

# Merge in today & previous day temp
demand_data <- demand_data %>%
  left_join(temp_16, by = "date") %>%
  left_join(
    temp_16 %>%
      rename(temp_16_prior = temp_16) %>%
      mutate(date = date + 1),
    by = "date"
  )

# Create weighted temp variable
demand_data <- demand_data %>%
  mutate(TA = (0.4 * temp_16 + 0.6 * temp_16_prior) / 2)
# Convert Date and extract components
demand_data <- demand_data %>%
  mutate(
    winter_month = case_when(
      month(Date) == 11 ~ 1,  # November
      month(Date) == 12 ~ 2,  # December
      month(Date) == 1  ~ 3,  # January
      month(Date) == 2  ~ 4,  # February
      month(Date) == 3  ~ 5,  # March
      TRUE ~ NA_integer_
    )
  )

# Create a grouping variable for month and weekday/weekend
demand_data <- demand_data %>%
  mutate(
    is_weekend = if_else(wday(Date) %in% c(1, 7), "Weekend", "Weekday"),
    month = month(Date)
  )
demand_data <- demand_data %>%
  mutate(
    month = month(Date),
    is_weekend = if_else(wday(Date) %in% c(1, 7), "Weekend", "Weekday")
  )
demand_data <- demand_data %>%
  arrange(Date) %>%
  mutate(lag_demand = lag(demand_gross, default = first(demand_gross)))





#---------------------------------------------
#              Our Model
#---------------------------------------------

# Base Model
m0 <- lm(demand_gross ~ wind + solar_S + temp + as.factor(wdayindex) + as.factor(monthindex), data = de

# Initial model
initial_model <-lm(demand_gross ~ TA+
                     as.factor(wdayindex) + I(DSN^2)+
                     as.factor(start_year) + month(date),
                   data = demand_data)

# Final Model
model_formula <-lm(demand_gross ~ TA+ lag_demand +
                     as.factor(wdayindex) + I(DSN^2)+
                     as.factor(start_year) + month(date),
                   data = demand_data)
```

```r
#----------------------------------------------------
#               Heteroskedasticity Test
#----------------------------------------------------

bptest(good_model)

#----------------------------------------------------
#                Autocorrelation Analysis
#----------------------------------------------------
residuals_model <- residuals(good_model)
par(mfrow=c(1,2))

acf(residuals_model, main = "ACF of residuals")
pacf(residuals_model, main = "PACF of residuals")


#----------------------------------------------------
#             Finding the best Temperature variable
#----------------------------------------------------
# generate_temp_variable() creates a weighted average temperature feature using specified hour ranges f
# evaluate_model() fits a linear model using the generated temperature feature and returns performance 
# search_best_combination() iterates over different hour windows and weights to find the temperature fe

temp_data <- read_csv("SCS_hourly_temp.csv", col_names = FALSE)
temp_data <- temp_data[-1, ]

colnames(temp_data) <- c("timestamp", "temp")

temp_data <- temp_data %>%
  rename(datetime = timestamp, temp = temp) %>%
  mutate(
    timestamp = parse_date_time(datetime, orders = "dmy HM"),
    date = as_date(timestamp),
    hour = hour(timestamp),
    temp = as.numeric(temp)
  )

demand_data <- read_csv("SCS_demand_modelling.csv")
demand_data <- demand_data %>%
  rename(date = Date) %>%
  mutate(date = as_date(date))


generate_temp_variable <- function(today_range, prev_range, weight_today = 0.5, weight_prev = 0.5) {
  # Average temp for today window
  today_avg <- temp_data %>%
    filter(hour >= today_range[1], hour <= today_range[2]) %>%
    group_by(date) %>%
    summarise(today_temp = mean(temp, na.rm = TRUE), .groups = "drop")

  # Average temp for previous day window, shifted forward by 1 day
  prev_avg <- temp_data %>%
    filter(hour >= prev_range[1], hour <= prev_range[2]) %>%
    mutate(date = date + 1) %>%
    group_by(date) %>%
    summarise(prev_temp = mean(temp, na.rm = TRUE), .groups = "drop")

  # Combine and weight
  combined <- today_avg %>%
    inner_join(prev_avg, by = "date") %>%
    mutate(
      custom_temp = weight_today * today_temp + weight_prev * prev_temp
    ) %>%
    select(date, custom_temp)

  return(combined)
}
evaluate_model <- function(today_range, prev_range, weight_today = 0.5) {
  weight_prev <- 1 - weight_today
  temp_feat <- generate_temp_variable(today_range, prev_range, weight_today, weight_prev)

  model_data <- demand_data %>%
    inner_join(temp_feat, by = "date") %>%
    filter(!is.na(demand_gross), !is.na(custom_temp))

  if (nrow(model_data) < 10) return(NULL)

  model <- lm(demand_gross ~ custom_temp, data = model_data)
  preds <- predict(model, newdata = model_data)
  rmse_val <- sqrt(mean((model_data$demand_gross - preds)^2))

  r_squared <- summary(model)$r.squared

  return(list(
    today_start = today_range[1],
    today_end = today_range[2],
    prev_start = prev_range[1],
    prev_end = prev_range[2],
    weight_today = weight_today,
    rmse = rmse_val,
    r_squared = r_squared,
    model = list(model)
  ))
}

search_best_combination <- function(window_size = 3) {
  # All valid start times between 13 and (20 - window_size + 1)
  start_hours <- 13:(20 - window_size + 1)
  today_windows <- map(start_hours, ~c(.x, .x + window_size - 1))
  prev_windows <- today_windows
  weights <- seq(0.1, 0.9, by = 0.1)

  combos <- crossing(
    today_range = today_windows,
    prev_range = prev_windows,
    weight_today = weights
  )

  # Evaluate
  results <- map_dfr(1:nrow(combos), function(i) {
    row <- combos[i, ]
    res <- evaluate_model(row$today_range[[1]], row$prev_range[[1]], row$weight_today)
    if (!is.null(res)) return(as_tibble(res))
    return(NULL)
  })

  # Return best
  best <- results %>% arrange(rmse) %>% slice(1)

  cat("Best combo:\n")
  cat("Today hours:", best$today_start, "-", best$today_end, "\n")
  cat("Prev day hours:", best$prev_start, "-", best$prev_end, "\n")
  cat("Weight (today):", best$weight_today, "\n")
  cat("Lowest RMSE:", round(best$rmse, 5), "\n")

  return(best)
}

best_result <- search_best_combination(window_size = 1)
summary(best_result$model[[1]])


#-------------------------------------------------
#             Newey -West Standard errors to determine significance
#-------------------------------------------------

# Reload all data so that it works nicely
temp_hourly <- read_csv("SCS_hourly_temp.csv")
demand_data <- read_csv("SCS_demand_modelling.csv")

temp_hourly <- temp_hourly %>%
  mutate(datetime = dmy_hm(Date),
         date = as_date(datetime),
         hour = hour(datetime))
# construct the specification selected by our window tool
temp_16 <- temp_hourly %>%
  filter(hour == 16) %>%
  select(date, temp) %>%
  rename(temp_16 = temp)
demand_data <- demand_data %>%
  mutate(
    Date = as_date(Date),
    date = Date)
demand_data$Date <- as.Date(demand_data$Date)

demand_data <- demand_data %>%
  mutate(date = as_date(Date)) %>%
  left_join(temp_16, by = "date") %>%
```

```r
  left_join(
    temp_16 %>%
      rename(temp_16_prior = temp_16) %>%
      mutate(date = date + 1),
    by = "date"
  )
demand_data <- demand_data %>%
  mutate(TA = (0.4 * temp_16 + 0.6 * temp_16_prior) / 2)

demand_data <- demand_data %>%
  arrange(Date) %>%
  mutate(lag_demand = lag(demand_gross, default = first(demand_gross)))
# define the initial model before cutting variables
model_formula <-lm(demand_gross ~ TA+ lag_demand + solar_S+
                     as.factor(wdayindex) + I(DSN^2)+ DSN+
                     as.factor(start_year) + month(date),
                   data = demand_data)

# Compute Newey-West robust standard errors and run the coefficient test
# we take the ourput of this by printing "print(nw_test)" and get rid of variables not significant at t
nw_se <- NeweyWest(model_formula, prewhite = TRUE)
nw_test <- coeftest(model_formula, vcov = nw_se)


#------------------------------------------------
#        Cross validation schemes (pulled directly from the Rmd)
#------------------------------------------------

# Make both the LOMO and 80:20 for month below

# For an 80% prediction interval
alpha <- 0.2
# making LOMO redefine months to make sure
data_cv1 <- demand_data %>%
  filter(!is_holiday, month(Date) %in% c(11, 12, 1, 2, 3)) %>%
  mutate(monthindex = factor(case_when(
    month(Date) == 11 ~ "Nov",
    month(Date) == 12 ~ "Dec",
    month(Date) == 1  ~ "Jan",
    month(Date) == 2  ~ "Feb",
    month(Date) == 3  ~ "Mar"
  ), levels = c("Nov", "Dec", "Jan", "Feb", "Mar"))) %>%
  arrange(Date)

# Define model formula, follows from tutorial 7
model_formula1 <- demand_gross ~ TA + lag_demand + wday(Date) + month(Date) +
  I(DSN^2) + as.factor(start_year) + DSN

rmse_vec1 <- numeric(length(levels(data_cv1$monthindex)))
int_score_vec1 <- numeric(length(levels(data_cv1$monthindex)))

for (mi in levels(data_cv1$monthindex)) {
  train_set1 <- data_cv1 %>%
    filter(monthindex != mi) %>%
    mutate(monthindex = factor(monthindex, levels = levels(data_cv1$monthindex)))

  test_set1 <- data_cv1 %>%
    filter(monthindex == mi) %>%
    mutate(monthindex = factor(monthindex, levels = levels(data_cv1$monthindex)))

  if(nrow(test_set1) == 0) next

  fit1 <- lm(model_formula1, data = train_set1)
  pred1 <- predict(fit1, newdata = test_set1, interval = "prediction", level = 0.8)
  obs1 <- test_set1$demand_gross

  rmse_vec1[which(levels(data_cv1$monthindex) == mi)] <- sqrt(mean((obs1 - pred1[,"fit"])^2, na.rm = TRU

  int_score_vec1[which(levels(data_cv1$monthindex) == mi)] <- mean(
    (pred1[,"upr"] - pred1[,"lwr"]) + (2/alpha) * (
      ((pred1[,"lwr"] - obs1) * (obs1 < pred1[,"lwr"])) +
        ((obs1 - pred1[,"upr"]) * (obs1 > pred1[,"upr"]))
    ),
    na.rm = TRUE
  )
}

results_method1 <- tibble(
  month = levels(data_cv1$monthindex),
  'RMSE Score LOMO' = rmse_vec1,
  'Interval Score LOMO' = int_score_vec1
)

# 80:20 split
data_cv2 <- demand_data %>%
  filter(!is_holiday, month(Date) %in% c(11, 12, 1, 2, 3)) %>%
  mutate(winter_month = case_when(
    month(Date) == 11 ~ "Nov",
    month(Date) == 12 ~ "Dec",
    month(Date) == 1  ~ "Jan",
    month(Date) == 2  ~ "Feb",
    month(Date) == 3  ~ "Mar",
    TRUE ~ NA_character_
  )) %>%
  mutate(winter_month = factor(winter_month, levels = c("Nov", "Dec", "Jan", "Feb", "Mar"))) %>%
  arrange(Date) %>%
  mutate(lag_demand = lag(demand_gross, default = first(demand_gross)))

# same model again
model_formula2 <- demand_gross ~ TA + lag_demand + as.factor(wdayindex) + month(Date) +
  I(DSN^2) + as.factor(start_year)

rmse_vec2 <- numeric(length(levels(data_cv2$winter_month)))
int_score_vec2 <- numeric(length(levels(data_cv2$winter_month)))

for (mi in levels(data_cv2$winter_month)) {
  train_set2 <- data_cv2 %>%
    filter(winter_month != mi) %>%
    mutate(winter_month = factor(winter_month, levels = levels(data_cv2$winter_month)))

  test_set2 <- data_cv2 %>%
    filter(winter_month == mi) %>%
    mutate(winter_month = factor(winter_month, levels = levels(data_cv2$winter_month)))

  if(nrow(test_set2) == 0) next

  fit2 <- lm(model_formula2, data = train_set2)
  pred2 <- predict(fit2, newdata = test_set2, interval = "prediction", level = 0.8)
  obs2 <- test_set2$demand_gross

  rmse_vec2[which(levels(data_cv2$winter_month) == mi)] <- sqrt(mean((obs2 - pred2[,"fit"])^2, na.rm =

  int_score_vec2[which(levels(data_cv2$winter_month) == mi)] <- mean(
    (pred2[,"upr"] - pred2[,"lwr"]) + (2/alpha) * (
      ((pred2[,"lwr"] - obs2) * (obs2 < pred2[,"lwr"])) +
        ((obs2 - pred2[,"upr"]) * (obs2 > pred2[,"upr"]))
    ),
    na.rm = TRUE
  )
}

results_method2 <- tibble(
  month = levels(data_cv2$winter_month),
  'RMSE Score 80:20 Split' = rmse_vec2,
  'Interval Score 80:20 Split' = int_score_vec2
)

# Combine the results from both methods by month
combined_results <- left_join(results_method1, results_method2, by = "month")

kable(combined_results,
      digits = 2,
      caption = "Comparison of Leave-One-Month-Out and 80:20 Cross-Validation Results by Month")

# Repeat for weekend/weekday split

alpha <- 0.2  # 80% prediction interval

# Ensure day type is defined
demand_data <- demand_data %>%
  mutate(
    is_weekend = if_else(wday(Date) %in% c(1, 7), "Weekend", "Weekday"),
    is_weekend = factor(is_weekend, levels = c("Weekday", "Weekend"))
  )

# CV by day type
cv_results_weekday <- demand_data %>%
  group_by(is_weekend) %>%
  group_modify(~ {
    set.seed(123)
    cv_folds <- vfold_cv(.x, v = 5)

    fold_results <- cv_folds %>%
      mutate(
        model = map(splits, ~ lm(model_formula, data = analysis(.x))),
        preds = map2(model, splits, ~ suppressWarnings(
          predict(.x, newdata = assessment(.y), interval = "prediction", level = 0.8)
        )),
        actuals = map(splits, ~ assessment(.x)$demand_gross),

        rmse = map2_dbl(preds, actuals, ~ sqrt(mean((.y - .x[,"fit"])^2, na.rm = TRUE))),

        interval_score = map2_dbl(preds, actuals, ~ mean(
          (.x[,"upr"] - .x[,"lwr"]) + (2/alpha) * (
            ((.x[,"lwr"] - .y) * (.y < .x[,"lwr"])) +
              ((.y - .x[,"upr"]) * (.y > .x[,"upr"]))
          ),
          na.rm = TRUE
        ))
      )

    tibble(
      avg_rmse = mean(fold_results$rmse),
      avg_interval_score = mean(fold_results$interval_score)
    )
  }) %>%
  ungroup()

# Display results as a kable table
kable(cv_results_weekday,
      col.names = c("Day Type", "Average RMSE", "Average Interval Score"),
      digits = 2,
      caption = "Cross-Validation Results by Day Type (Weekday vs Weekend)") %>%
  kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover", "condensed"))


#-------------------------------------------------
#        Using other winters to predict 2013
#-------------------------------------------------
# Exclude 2013 winter from training (i.e. use data from 1991 to 2012)
train_data <- demand_data %>%
  filter(year(Date) != 2013, month(Date) %in% c(11, 12, 1, 2, 3))

# Fit the model on the training data
model_excl_2013 <- lm(demand_gross ~ TA + lag_demand +
                        as.factor(wdayindex) + I(DSN^2) +
                        as.factor(start_year) + month(date),
                      data = train_data)

## Get unique weather years from training data
sim_years <- sort(unique(train_data$start_year))
```

```r
# store simulated peak predictions
predictions <- tibble(WeatherYear = sim_years, PredictedPeak = NA_real_)

# Loop over each weather year
# For each weather year, compute the average TA and lag_demand for the winter period.
for(i in seq_along(sim_years)) {
  yr <- sim_years[i]
  weather_yr <- demand_data %>%
    filter(year(Date) == yr, month(Date) %in% c(11,12,1,2,3))

  # Compute average values for TA and lag_demand from that winter
  avg_TA <- mean(weather_yr$TA, na.rm = TRUE)
  avg_lag <- mean(weather_yr$lag_demand, na.rm = TRUE)
  # For DSN, we can use the median value as it's in the middle
  med_DSN <- median(weather_yr$DSN, na.rm = TRUE)

  # Create a new data frame for prediction
  new_data <- tibble(
    TA = avg_TA,
    lag_demand = avg_lag,
    wdayindex = 3, # we chose a workday to be baseline as they are higher
    DSN = med_DSN,
    start_year = 2013,  # target winter year effect
    date = as.Date("2013-01-15")
  )

  # Predict peak demand using the fitted model
  pred <- predict(model_excl_2013, newdata = new_data)
  predictions$PredictedPeak[i] <- pred
}

# Obtain the actual peak demand for winter 2013/14 from the full data
actual_2013_14 <- demand_data %>%
  filter((year(date) == 2013 & month(date) %in% c(11,12)) |
           (year(date) == 2014 & month(date) %in% c(1,2,3))) %>%
  pull(demand_gross) %>%
  max(na.rm = TRUE)

# Identify the highest and lowest predicted peaks
max_pred <- max(predictions$PredictedPeak, na.rm = TRUE)
min_pred <- min(predictions$PredictedPeak, na.rm = TRUE)

#plot the results
ggplot(predictions, aes(x = WeatherYear, y = PredictedPeak)) +
  geom_line(size = 1) +
  geom_point(data = filter(predictions, PredictedPeak %in% c(max_pred, min_pred)),
             aes(x = WeatherYear, y = PredictedPeak), color = "red", size = 3) +
  geom_hline(yintercept = actual_2013_14, linetype = "dashed", color = "blue", size = 1) +
  labs(
    title = "Simulated 2013/14 Peak Electricity Demand Based on Historical Weather",
    x = "Weather Year",
    y = "Predicted Peak Demand (MW)",
    caption = "Red dots denote highest and lowest simulated peaks; dashed line shows actual 2013 peak de
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 11, face = "bold"),
    axis.text = element_text(size = 10)
  )

peak_data <- data.frame(
  Type = c("Highest simulated peak", "Lowest simulated peak", "Actual peak (2013/14)"),
  Demand_MW = c(max_pred, min_pred, actual_2013_14)
)

kable(peak_data,
      col.names = c("Demand Type", "Value (MW)"),
      digits = 0,  # Round to whole numbers
      caption = "Simulated vs Actual Peak Electricity Demand (2013/14)") %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "center") %>%
  row_spec(3, bold = TRUE, background = "#E8F4F8")
```