Notes

2024-05-15

Import the data

We are using a dataset of Irish household electricity demand available from the `electBook` package. We have three datasets:

- `indCons`: 16799 x 2672 matrix of individual household electricity consumption. Each column corresponds to a household and each row to a time point. Demand is observed every half hour, so there are 48 observations per day per household.
- `survey`: 2672 row dataframe of household survey data. This dataset contains household level data on variables such as social class, renting vs. owning, appliances, etc.
- `extra`: 16799 row dataframe of time-related variables. This dataset contains the date-time of each demand observation, time of year, day of week, time of day, whether the day was a holiday, and external temperature.

```
# devtools::install_github("mfasiolo/electBook")
library(electBook)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
data(Irish)
```

There are 48 observations per day per customer.

Exploratory data analysis

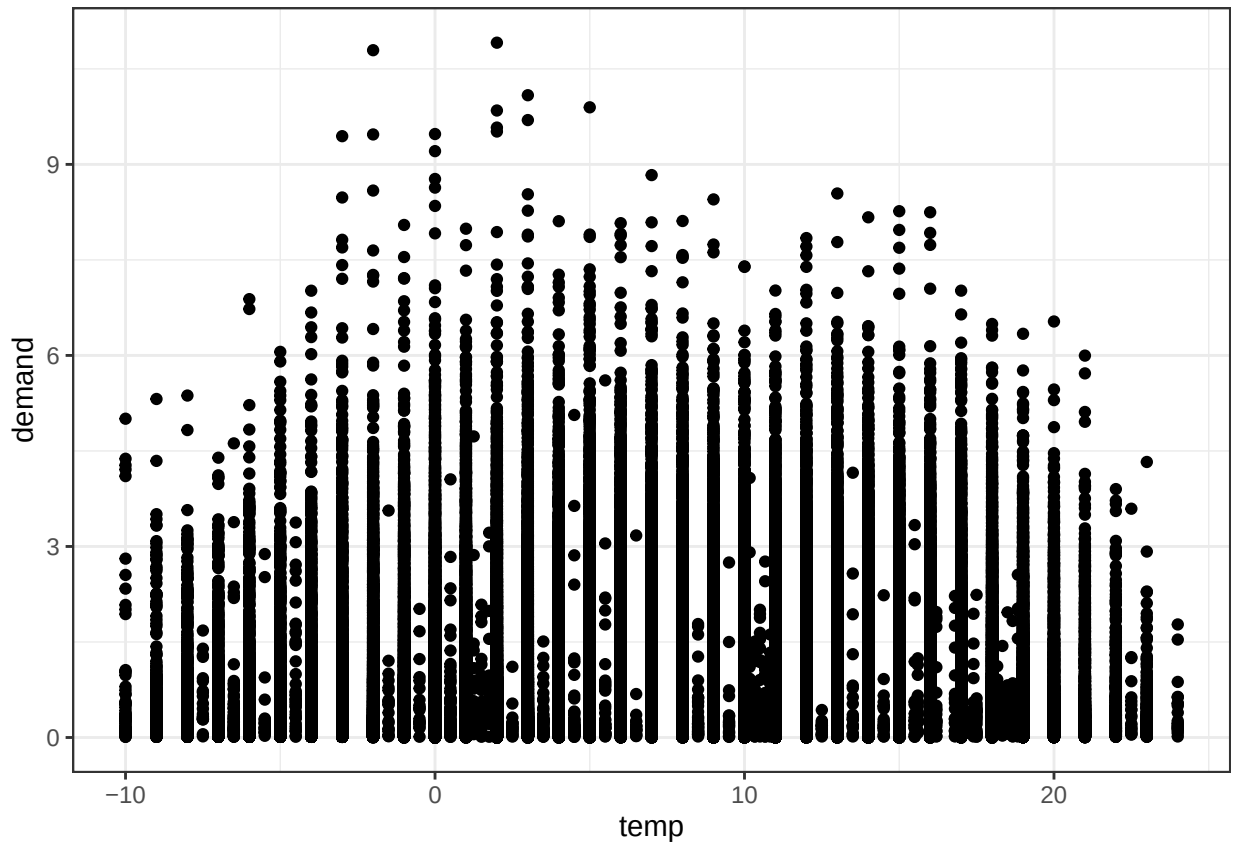
Let's check any relationship between temperature and demand.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# Data cleaning
# Sample a subset of households for plotting
house_sample <- sample(colnames(Irish$indCons), 20)
irish_demand_sample <- Irish$indCons[,house_sample] %>%
  as.data.frame() %>%
```

```r
  # Row numbers to column
  mutate(time = 1:nrow(.)) %>%
  pivot_longer(-time, names_to = "household", values_to = "demand") %>%
  left_join(Irish$extra %>% select(time, temp), by = "time")

# Plot demand vs. temperature
ggplot(irish_demand_sample, aes(x = temp, y = demand)) +
  geom_point() +
  theme_bw()
```



Might be a quadratic relationship? (very noisy)

Ridge regression

Theory

Ridge regression is a method for penalized regression. Consider the model

$$Y_i^0 = \alpha + \beta x_i^0 + \epsilon_i, \quad i = 1, ..., n$$

where $\beta \in \mathbb{R}^p$, $\alpha \in \mathbb{R}$, and for all $i, l \in \{1, ..., n\}$, $\mathbb{E}[\epsilon_i] = 0$ and $\mathbb{E}[\epsilon_i \epsilon_l] = \sigma^2 \delta_{il}$ for some $\sigma^2 > 0$. Then the ridge regression estimator is defined as the minimizer of the following objective function:

$$(\hat{\alpha}_\lambda, \hat{\beta}_\lambda) = \underset{\alpha \in \mathbb{R}, \beta \in \mathbb{R}^p}{\arg\min} \|y^0 - \alpha - \boldsymbol{X}^0 \beta\|_2^2 + \lambda \|\beta\|_2^2$$

where $\lambda > 0$ is a tuning parameter and $\|\cdot\|_2$ denotes the Euclidean norm. Ridge regression is thus imposing a penalty on the size of β, with the strength of that penalty determined by the choice of λ. The coefficients will be shrunk towards zero, but will not be set to zero (as opposed to in lasso regression).

Note: we are going to need to scale the input variables

Resources

List of random links that might be helpful:

- Let's Use Ridge Regression to Predict Time Series: application of ridge regression to time series of stock prices in Python

Application

We can try running an individual ridge regression model for each household. We will use the `indCons` dataset as the response variable and the `survey` and `extra` datasets as the predictors. We will use the `glmnet` package to fit the ridge regression models.

Data cleaning

Matteo seemed to recommend testing on the final month of the dataset (training on the rest).

The `extra` dataset contains a `time` variable that is a progressive time counter – this isn't so great because there are dates in the dataset that are missing, e.g. 2009-12-31, but the time counter just increases by 1. I went ahead and dropped this variable in favor of the `dateTime` variable.

The variable `holy` is supposed to indicate whether the day is a holiday or not, but it's always `FALSE`, so I just drop it. I actually think holidays are excluded from the dataset itself, like New Years Eve and Easter Sunday are both missing.

Not sure how to deal with the day of the week `dow`: it's a factor variable, but not sure whether to treat it as a continuous 1-7 or to one-hot encode it. (I ended up one-hot encoding later.)

For ridge regression, we're supposed to be scaling the data, but we have a factor (`dow`) and a date-time variable (`dateTime`) that we need to keep as is. I'm not scaling anything for now (but I do so in the next section).

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

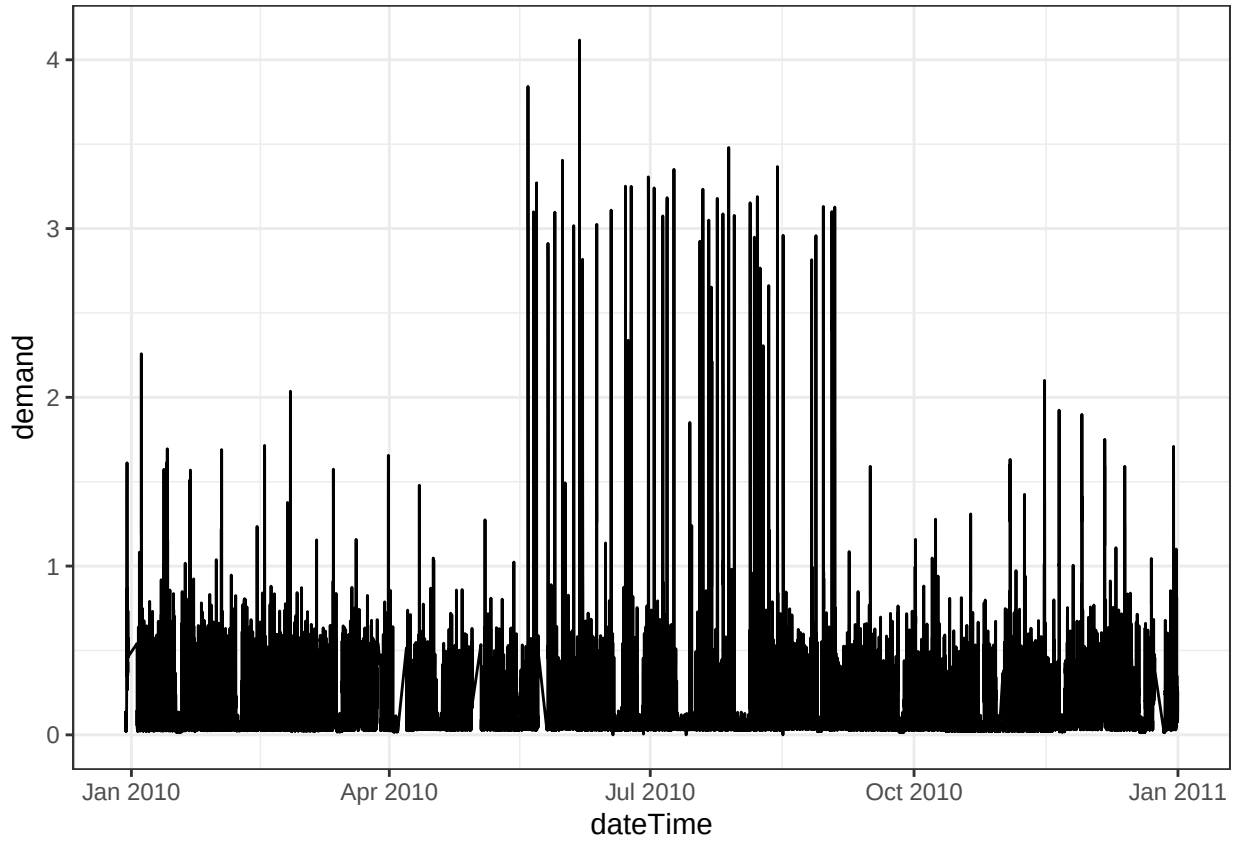
```
## Loaded glmnet 4.1-8
```

```
# Model for household I1002
# We need to one-hot encode the day of the week
I1002_data <- data.frame(demand = Irish$indCons[, "I1002"]) %>%
  bind_cols(Irish$extra) %>% # add time-related variables
  # Data cleaning
  select(-time, -holy) %>%
  # Feature engineering
  mutate(
    hour = hour(dateTime),
    month = month(dateTime),
    weekend = ifelse(dow %in% c("Sat", "Sun"), 1, 0)
  ) %>%
  mutate(temp_sq = temp^2) %>%  # quadratic term for temperature
```

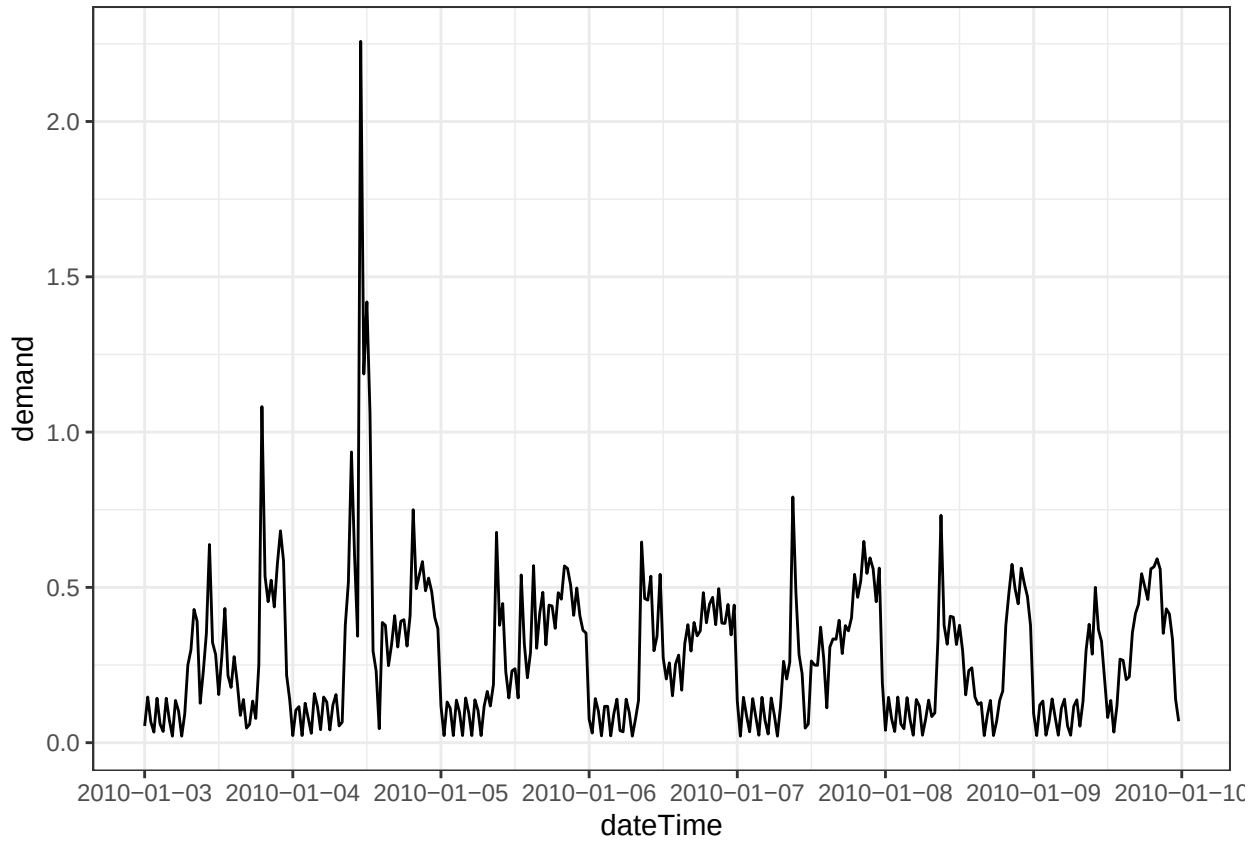
```
  # One-hot encode the day of the week
  bind_cols(model.matrix(~ dow - 1, data = .)) %>%
  select(-dow)

# Split the data into training and test sets
I1002_train <- I1002_data %>%
  filter(dateTime < "2010-12-01") # Training set: all data before December 2010
# Test on the last month of the dataset
I1002_test <- I1002_data %>%
  filter(dateTime >= "2010-12-01")
```

We can visualize the demand for household I1002 over time:

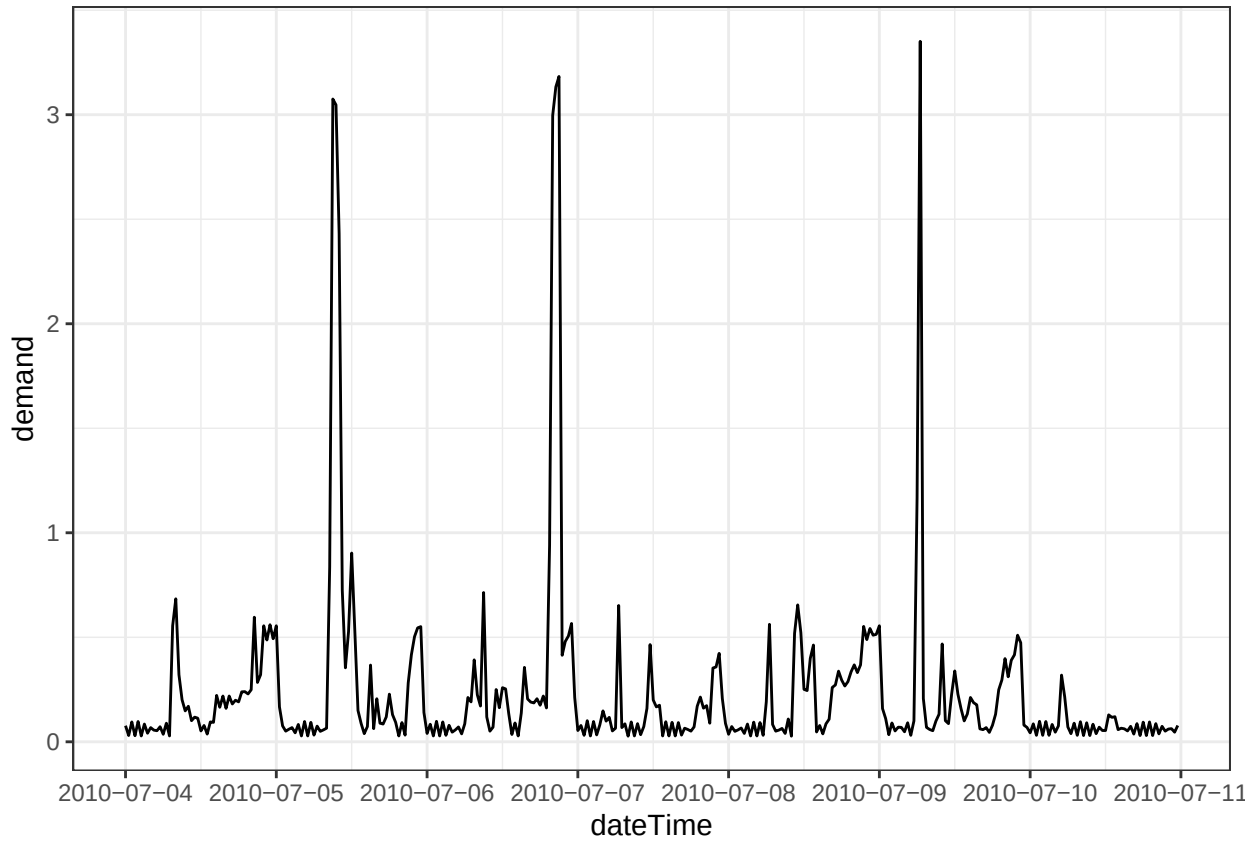
```
# All time
ggplot(I1002_data, aes(x = dateTime, y = demand)) +
  geom_line() +
  theme_bw()
```



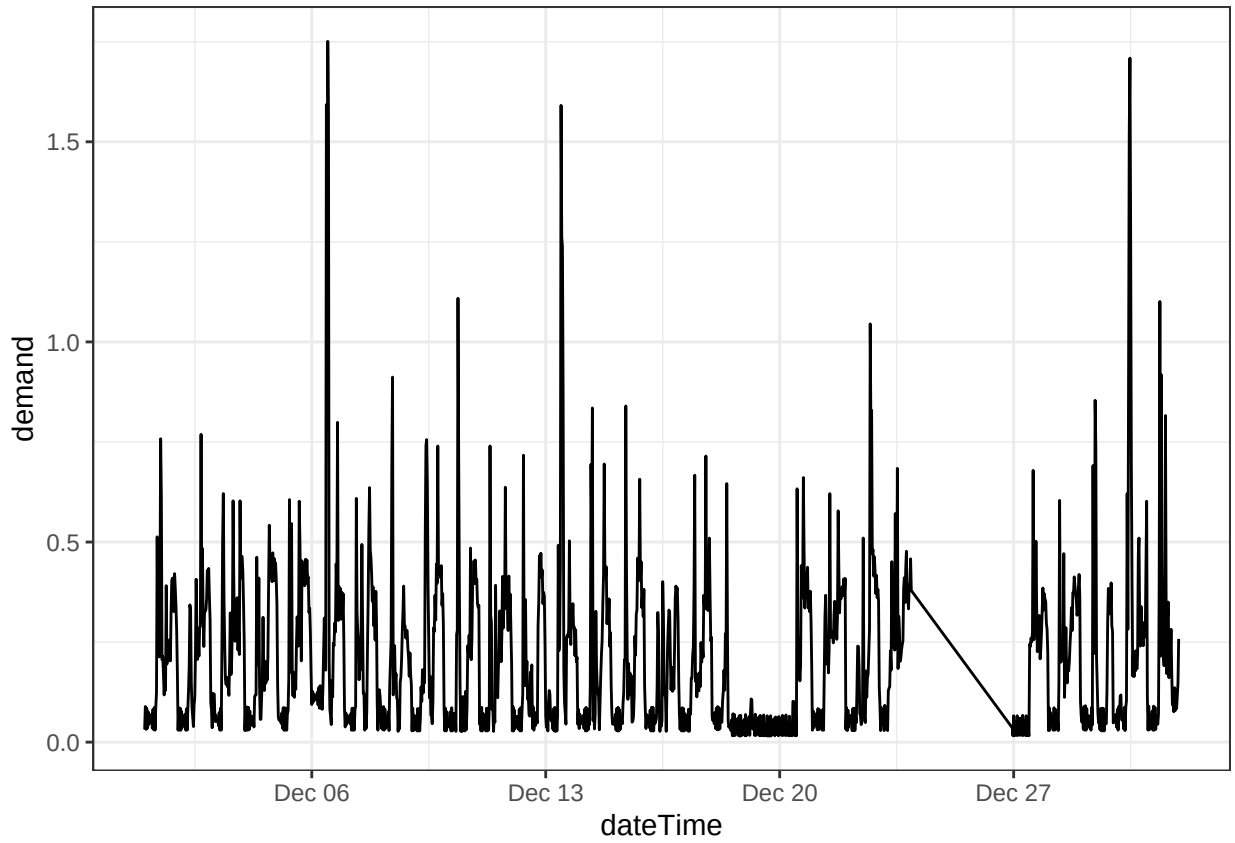

```
# A single week in January 2010
ggplot(I1002_data %>%
         # Monday - Sunday
         filter(dateTime >= "2010-01-03" & dateTime < "2010-01-10"),
       aes(x = dateTime, y = demand)) +
  geom_line() +
  scale_x_datetime(date_breaks = "1 day") +
  theme_bw()
```




```
# A single week in July 2010
ggplot(I1002_data %>%
         # Monday - Sunday
         filter(dateTime >= "2010-07-04" & dateTime < "2010-07-11"),
       aes(x = dateTime, y = demand)) +
  geom_line() +
  scale_x_datetime(date_breaks = "1 day") +
  theme_bw()
```



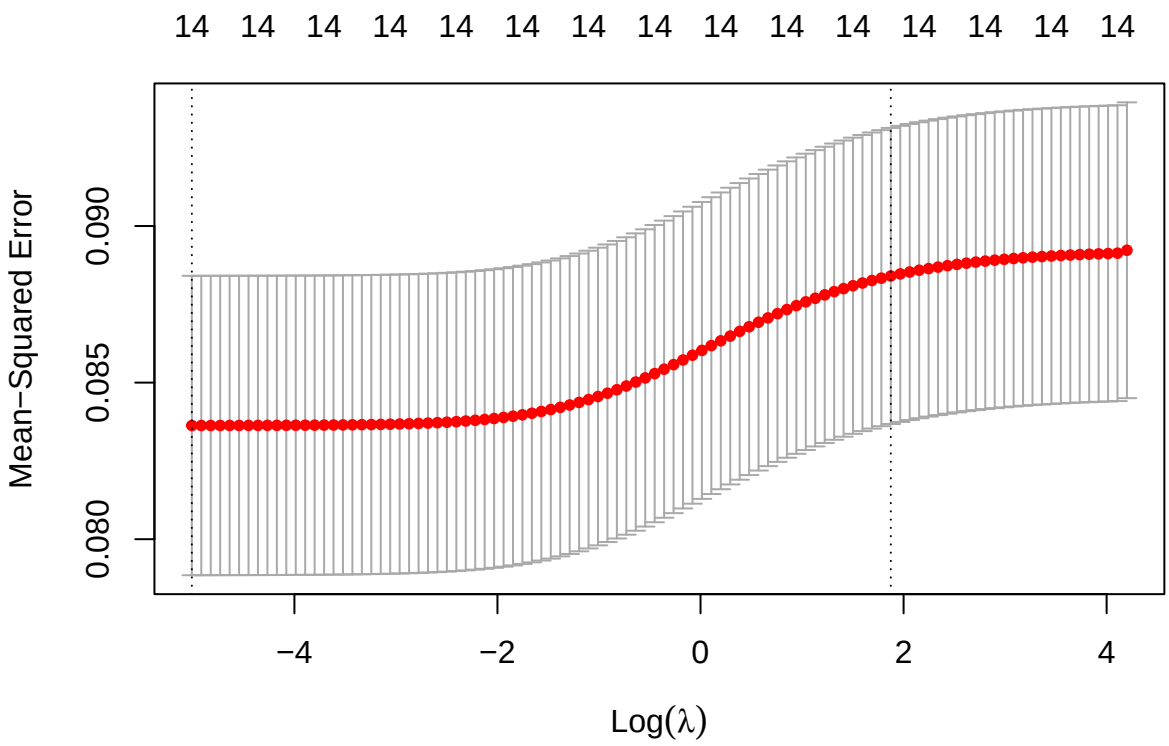

```
# Testing set
ggplot(I1002_test, aes(x = dateTime, y = demand)) +
  geom_line() +
  theme_bw()
```



Model fitting

We now fit the ridge regression model using 10-fold cross-validation to select our λ parameter.

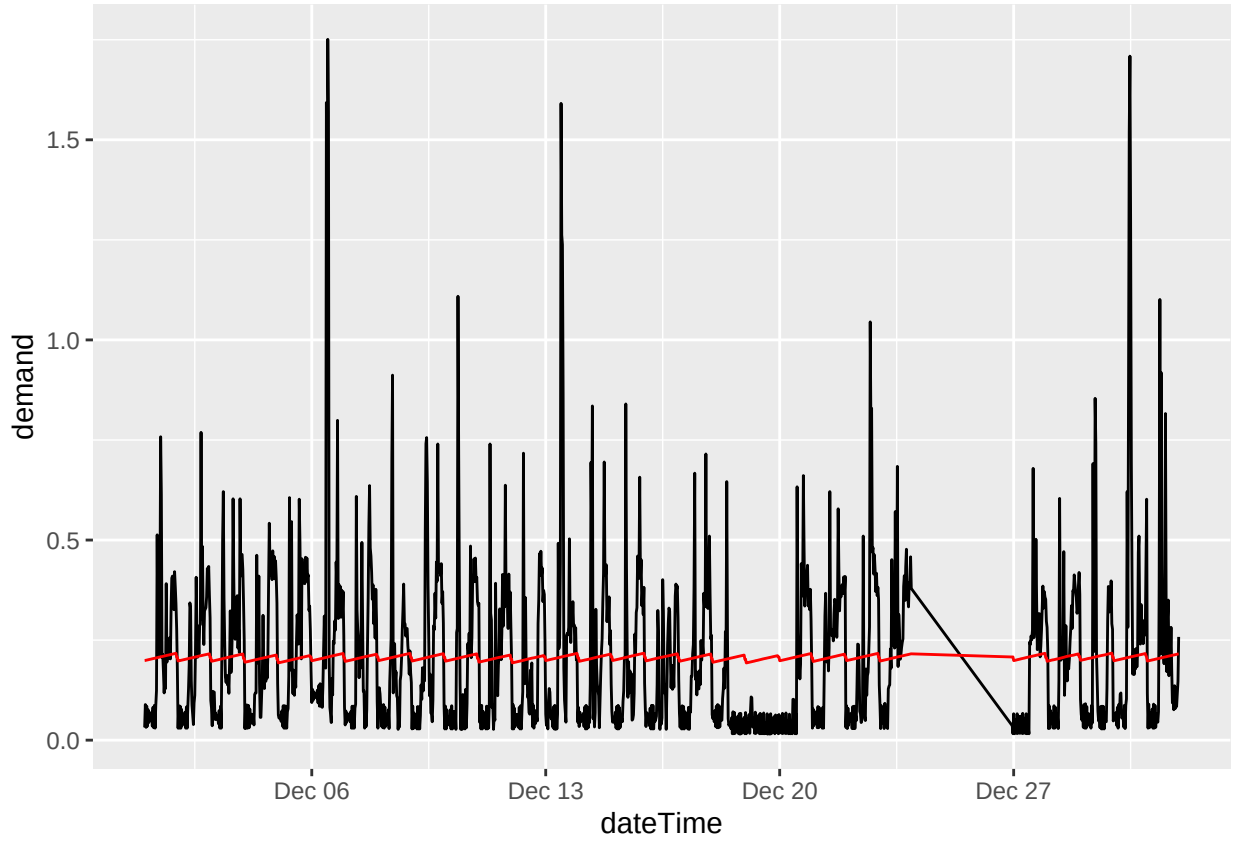
```
# Fit the ridge regression model
I1002_ridge <- cv.glmnet(as.matrix(I1002_train %>% select(-demand, -dateTime)),
                         I1002_train$demand,
                         family = "gaussian",
                         alpha = 0, # Ridge regression
                         type.measure = "mse",
                         nfolds = 10)
plot(I1002_ridge)
```




```r
# Predict on the test set
I1002_pred <- predict(I1002_ridge, newx = as.matrix(I1002_test %>% select(-demand, -dateTime)))
# Calculate MSE
I1002_mse <- mean((I1002_pred - I1002_test$demand)^2)
print(paste("Ridge regression test error:", I1002_mse))
```

```
## [1] "Ridge regression test error: 0.0413155953961133"
```

```r
# Plot the predictions
ggplot(I1002_test, aes(x = dateTime, y = demand)) +
  geom_line() +
  geom_line(aes(y = I1002_pred), color = "red")
```



This is a terrible fit.

First MSE: 0.04217874 MSE after date feature engineering: 0.04084892

Fourier terms

We can try adding Fourier terms to the model to capture the seasonality in the data. Fourier terms are sine and cosine functions with different frequencies that can capture periodic patterns in the data. For a given period P, the Fourier terms are defined as follows:

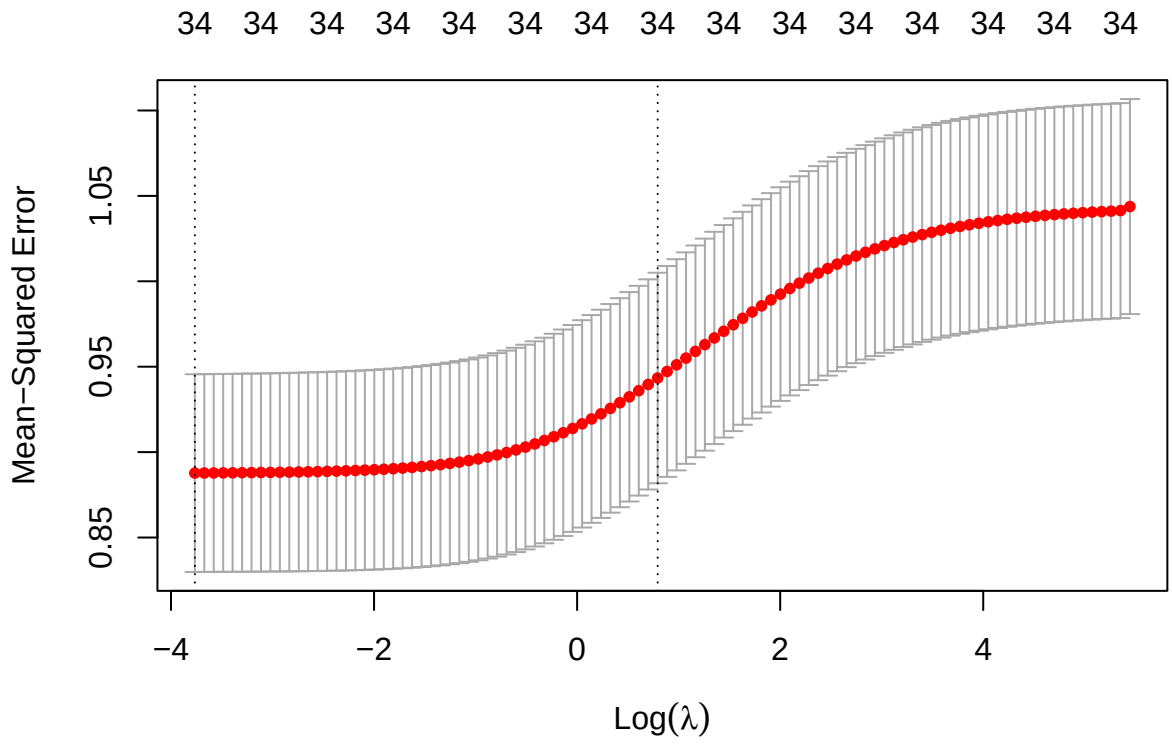
$$\sin_k(t) = \sin\left(\frac{2\pi kt}{P}\right), \quad \cos_k(t) = \cos\left(\frac{2\pi kt}{P}\right)$$

where k is the frequency and t is the time.

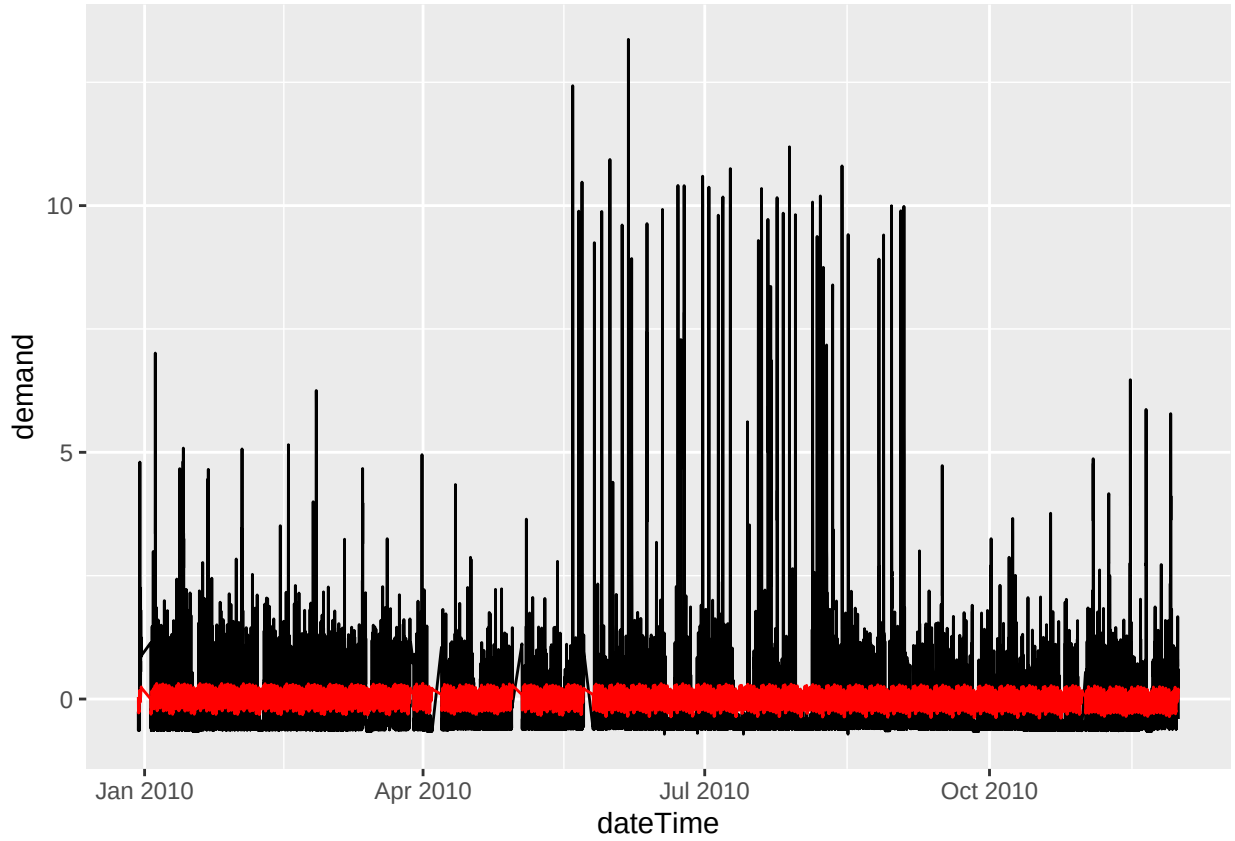
We will add a Fourier term for each of the 24 hours in a day and for each of the 12 months in a year.

```
library(forecast)
# Add Fourier terms
fourier_daily <- fourier(ts(I1002_data$demand, frequency = 48), K = 5)
fourier_weekly <- fourier(ts(I1002_data$demand, frequency = 48 * 7), K = 5)
I1002_data_fourier <- cbind(I1002_data, fourier_daily, fourier_weekly)

# Scale numeric features
I1002_data_scaled <- I1002_data_fourier %>%
  mutate(across(where(is.numeric), scale))

# Split the data into training and test sets
```

```r
I1002_train_scaled <- I1002_data_scaled %>%
  filter(dateTime < "2010-12-01") # Training set: all data before December 2010
# Test on the last month of the dataset
I1002_test_scaled <- I1002_data_scaled %>%
  filter(dateTime >= "2010-12-01")

# Fit the ridge regression model
I1002_ridge_scaled <- cv.glmnet(as.matrix(I1002_train_scaled %>% select(-demand, -dateTime)),
                                I1002_train_scaled$demand,
                                family = "gaussian",
                                alpha = 0, # Ridge regression
                                type.measure = "mse",
                                nfolds = 10)
plot(I1002_ridge_scaled)
```



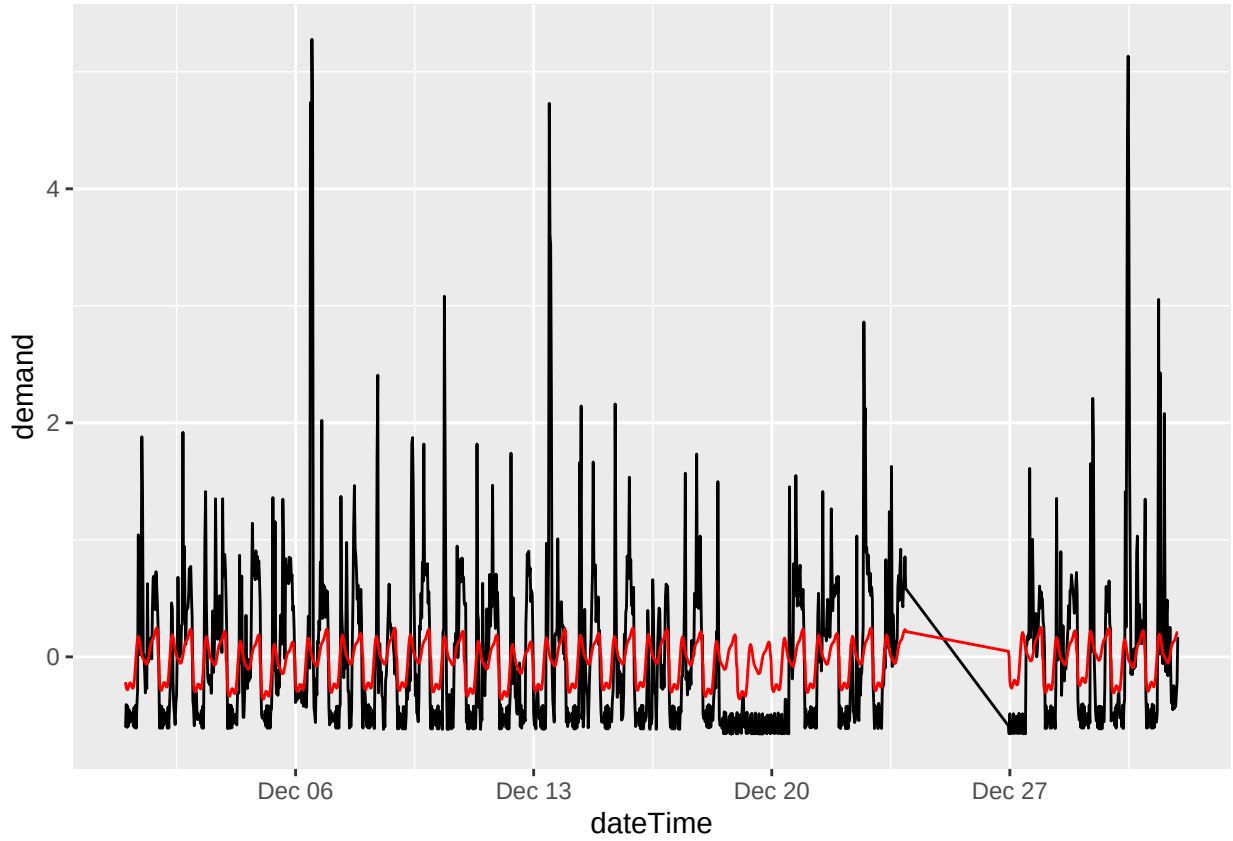

```r
# Plot the predictions on the training set
I1002_pred_scaled <- predict(I1002_ridge_scaled, newx = as.matrix(I1002_train_scaled %>% select(-demand
ggplot(I1002_train_scaled, aes(x = dateTime, y = demand)) +
  geom_line() +
  geom_line(aes(y = I1002_pred_scaled), color = "red")
```




```
# Predict on the test set
I1002_pred_scaled <- predict(I1002_ridge_scaled, newx = as.matrix(I1002_test_scaled %>% select(-demand,
# Calculate MSE
I1002_mse_scaled <- mean((I1002_pred_scaled - I1002_test_scaled$demand)^2)
print(paste("Ridge regression test error with Fourier terms:", I1002_mse_scaled))
```

```
## [1] "Ridge regression test error with Fourier terms: 0.404278936415921"
```

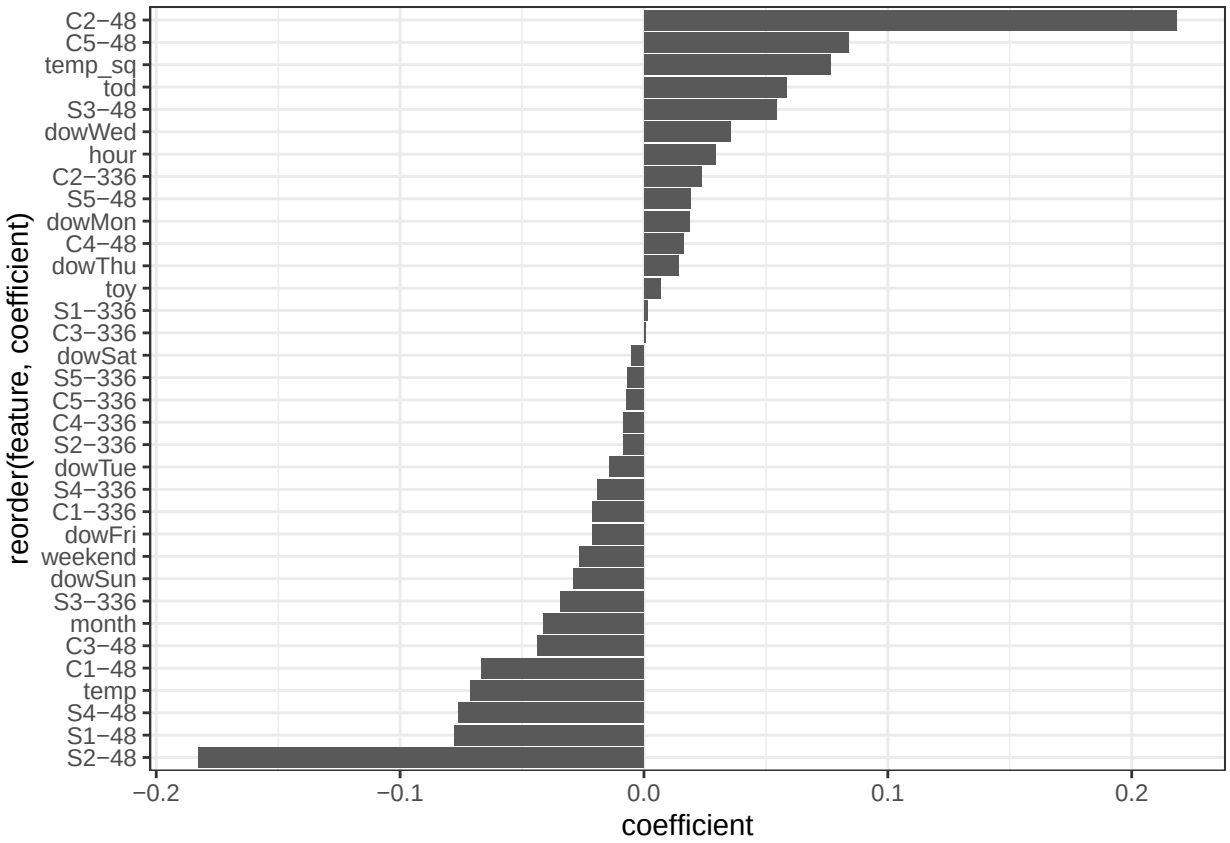
```
# Plot the predictions
ggplot(I1002_test_scaled, aes(x = dateTime, y = demand)) +
  geom_line() +
  geom_line(aes(y = I1002_pred_scaled), color = "red")
```



I1002_mse_scaled with K = 3: 0.3998798 K = 5 and quadratic temp: 0.3945322 K = 5, quadratic temp, and one-hot dow: 0.3905405

Let's see which terms are most important:

```
# Extract coefficients
I1002_coefs <- coef(I1002_ridge_scaled, s = "lambda.min") %>% as.matrix()

# Standardize the coefficients (multiply by the standard deviation of the corresponding feature)
standard_deviations <- apply(as.matrix(I1002_train_scaled %>% select(-demand, -dateTime)), 2, sd)
standardized_coefficients <- I1002_coefs[-1, ] * standard_deviations # Exclude the intercept

# Plot the coefficients
coefficients_df <- data.frame(
  feature = names(standardized_coefficients),
  coefficient = as.numeric(standardized_coefficients)
)
ggplot(coefficients_df, aes(x = reorder(feature, coefficient), y = coefficient)) +
  geom_col() +
  coord_flip() +
  theme_bw()
```


C2−48
C5−48
temp_sq
tod
S3−48
dowWed
hour
C2−336
S5−48
dowMon
C4−48
dowThu
toy
S1−336
C3−336
dowSat
S5−336
C5−336
C4−336
S2−336
dowTue
S4−336
C1−336
dowFri
weekend
dowSun
S3−336
month
C3−48
C1−48
temp
S4−48
S1−48
S2−48
reorder(feature, coefficient)
−0.2
−0.1
0.0
0.1
0.2
coefficient