Predictability Analysis using nmecr

Mrinalini Sharma* David Jump† Devan Johnson‡

15 January, 2021

The `nmecr` package was written to integrate the past efforts of the energy efficiency industry to streamline meter-based Measurement & Verification of EE projects. It brings together simple linear regression with outside air temperature, the change point models (from ASHRAE 1050-RP), and the Time-of-Week and Temperature model (developed by the Lawrence Berkeley National Laboratory) and builds enhancements upon these by accurately predicting the energy use profiles of facilities with multiple operating modes.

Normalized Metered Energy Consumption (NMEC)

Meter-based energy efficiency programs are proliferating across the globe. This proliferation is influenced by the widespread availability of high frequency energy use measurements from new metering technology as well as advancements in statistical regression and other empirical energy data modeling methodologies. Program administrators may report savings as the overall reduction in normalized metered energy consumption (NMEC). This method to determine savings is based on analysis of meter data collected prior to and after energy efficiency and conservation measures have been installed.

Necessity of Predictability Analysis

Prior to engaging in a meter-based savings project, a good practice from a modeling perspective is to determine whether a sufficiently accurate energy model may be developed. Using a year of pre-installation period energy use data, an energy model is developed, and its goodness of fit metrics are calculated and compared to industry accepted criteria. Adherence to the assumptions of regression modeling may also be checked. The four main assumptions of linear regression are:

- Linearity of data: The relationship between the predictor and the predicted is assumed to be linear
- Normality of residuals: The residuals are assumed to be normally distributed. Residuals are the difference between the actual and predicted values.
- Homoscedasticity (homogeneity of residual variance): The residuals are assumed to have a constant variance.
- Independence of residuals: The residuals are assumed to be independent of each other.

Data

`nmecr` includes two datasets: `temp` and `eload`. These contain three years (03/01/2012 - 02/28/2015) of energy use and outside temperature data from a commercial building in North America. We will use the first year (03/01/2012 - 02/28/2013) of energy use and temperature data for this predictability analysis, as at the start of any project, we generally can collect a year of data.

*msharma@kw-engineering.com
†djump@kw-engineering.com
‡johnson@kw-engineering.com

```
# Load data into an R session:

data(eload)
data(temp)
```

`eload` and `temp` are data frames with the two variables each. When using `nmecr` functions, ensure that the column headers of your datasets are the same as those shown below.

Eload:

```
#> # A tibble: 5 x 2
#>   time                eload
#>   <dttm>              <dbl>
#> 1 2012-03-01 00:00:00 21505.
#> 2 2012-03-02 00:00:00 20892.
#> 3 2012-03-03 00:00:00 20435.
#> 4 2012-03-04 00:00:00 15660.
#> 5 2012-03-05 00:00:00 15864.
```

Temp:

```
#> # A tibble: 5 x 2
#>   time                temp
#>   <dttm>             <dbl>
#> 1 2012-03-01 00:00:00  38.4
#> 2 2012-03-02 00:00:00  39.9
#> 3 2012-03-03 00:00:00  43.0
#> 4 2012-03-04 00:00:00  49.7
#> 5 2012-03-05 00:00:00  48.3
```

Baseline Dataframe for Modeling

`create_dataframe()` combines the `eload` and `temp` dataframes into one, filters by the specified start and end dates, and aggregates to an hourly, daily, or a monthly data interval. It lines up all data such that each timestamp represents data up until that point, e.g. an eload value corresponding to 03/02/2012 represents the energy consumption up until then - the energy consumption of 03/01/2012. If operating mode data is supplied, this information is added to the dataframe created by `create_dataframe()`.

```
# Baseline Dataframe

baseline_df <- create_dataframe(eload_data = eload, temp_data = temp,
                                start_date = "03/01/2012 00:00",
                                end_date = "02/28/2013 23:59",
                                convert_to_data_interval = "Daily")

head(baseline_df, 5)
#> # A tibble: 5 x 5
#>   time                eload  temp   HDD   CDD
#>   <dttm>              <dbl> <dbl> <dbl> <dbl>
#> 1 2012-03-02 00:00:00 21505.  38.4  26.6     0
#> 2 2012-03-03 00:00:00 20892.  39.9  25.1     0
```

```
#> 3 2012-03-04 00:00:00 20435.  43.0  22.0     0
#> 4 2012-03-05 00:00:00 15660.  49.7  15.3     0
#> 5 2012-03-06 00:00:00 15864.  48.3  16.7     0
```

Energy Data Modeling

The baseline period dataframe can be used for energy data modeling using one of the four modeling algorithms available in nmecr:

- `model_with_TOWT()`: Time-of-Week & Temperature and Time-Only algorithms
- `model_with_CP()`: 3-Parameter Heating, 3-Parameter Cooling, 4-Parameter, and 5-Parameter algorithms
- `model_with_SLR()`: Simple Linear Regression algorithm
- `model_with_HDD_CDD()`: Heating Degree Day only, Cooling Degree Day only, and a combination of Heating Degree Day and Cooling Degree Day algorithms

The common arguments for these four algorithms are:

1. training_data (output from `create_dataframe()`)
2. model_input_options (see below)

`model_with_TOWT()` has two additional arguments: prediction_data and occupancy_info. prediction_data (output from create_dataframe()) provides the independent variable data over which the model can be applied to compute predictions. occupancy_info is an optional parameter to manually define the occupancy for the time-of-week aspect of the algorithm.

`model_with_HDD_CDD()` has two additional optional arguments: HDD_balancepoint and CDD_balancepoint. As the name suggests, these define the balancepoints for heating and cooling degree days. If left undefined, HDD and CDD values are calculated using a balancepoint of 65.

model_input_options()

The four modeling algorithms have many specifications in common that can be specified using the `assign_model_inputs()` function. The following code chunk shows the default values for these model inputs

```
model_input_options <- assign_model_inputs(timescale_days = NULL,
                                           has_temp_knots_defined = FALSE,
                                           equal_temp_segment_points = TRUE,
                                           temp_segments_numeric = 6,
                                           temp_knots_value = c(40, 45, 50, 60, 65, 90),
                                           initial_breakpoints = c(50,65),
                                           regression_type = "TOWT",
                                           occupancy_threshold = 0.65,
                                           day_normalized = FALSE)
```

Before creating data models, it is a good practice to visualize the energy use profiles:

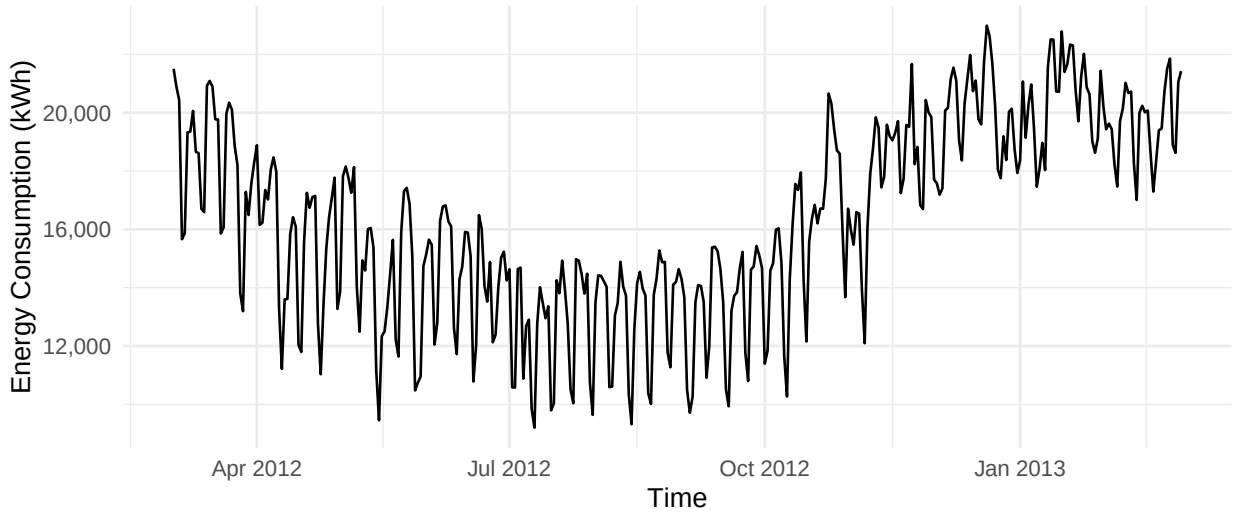

Figure 1: Baseline Energy Use over Time

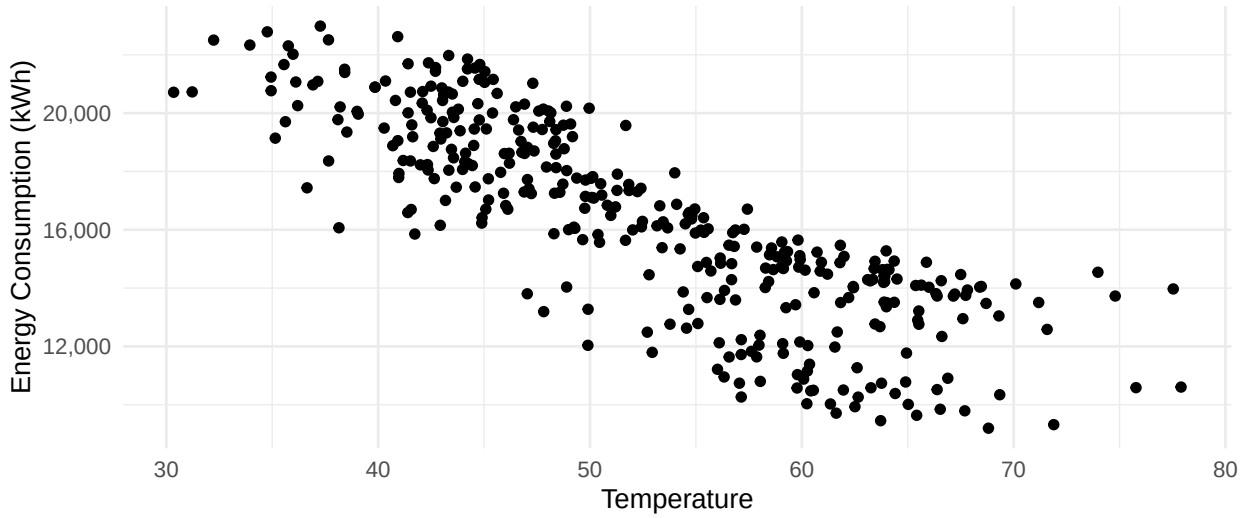

Figure 2: Baseline Energy Use over Temperature

Due to the high dependence on temperature, we can begin the data modeling using Simple Linear Regression and compare that to other modeling algorithms

Model Creation

Simple Linear Regression:

```
SLR_model <- model_with_SLR(training_data = baseline_df,
                            model_input_options =
                              assign_model_inputs(regression_type = "SLR"))
```

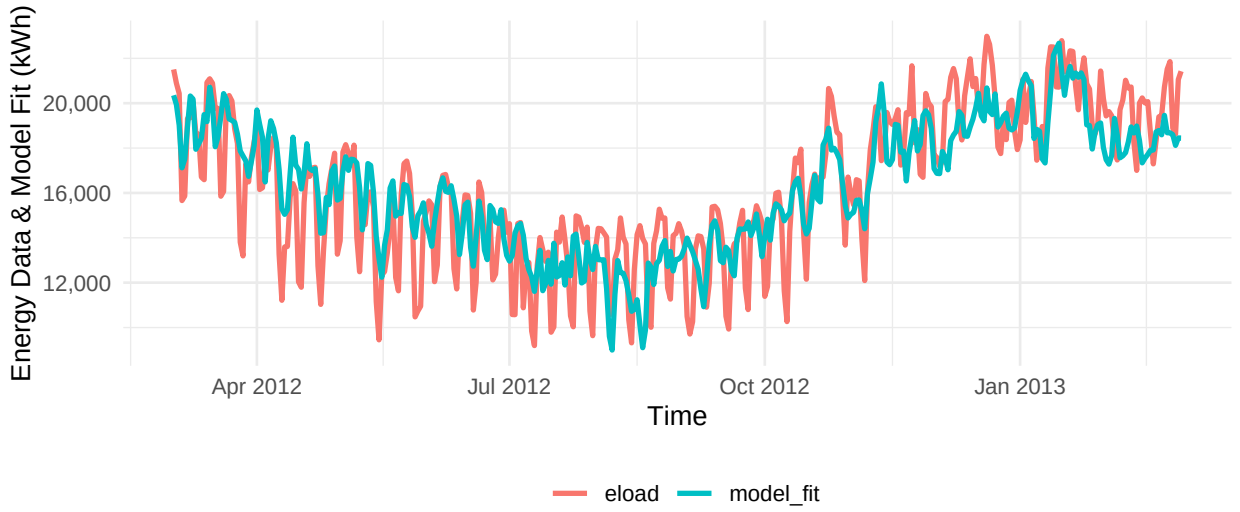

Figure 3: Baseline Energy Use modeled with Simple Linear Regression over Time

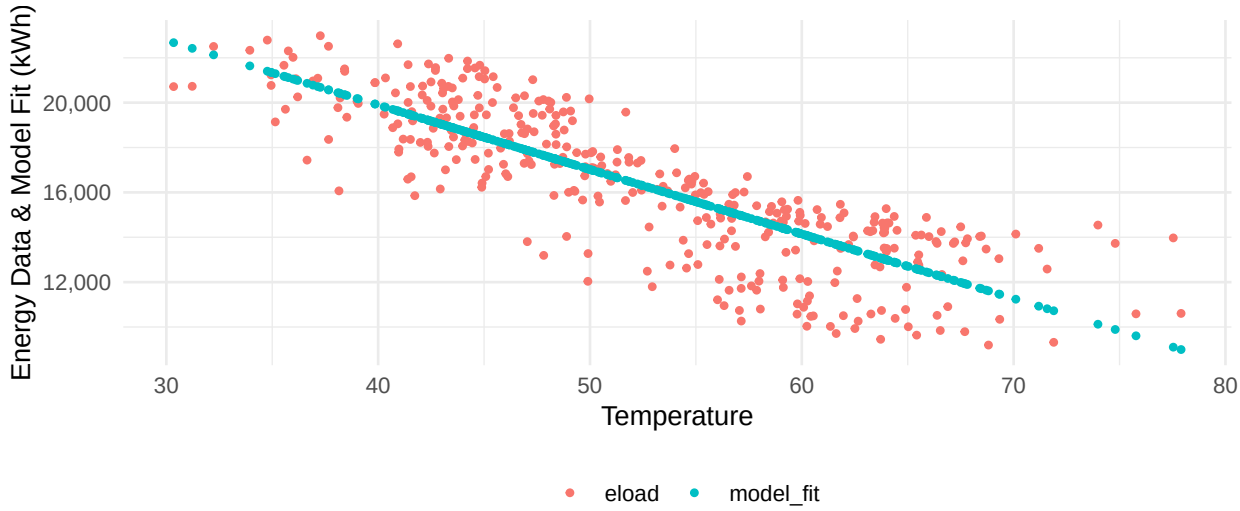

Figure 4: Baseline Energy Use modeled with Simple Linear Regression over Temperature

Four Parameter Heating:

```
Four_P_model <- model_with_CP(training_data = baseline_df,
                              model_input_options =
                                assign_model_inputs(regression_type =
                                                      "Four Parameter Linear Model"))
```

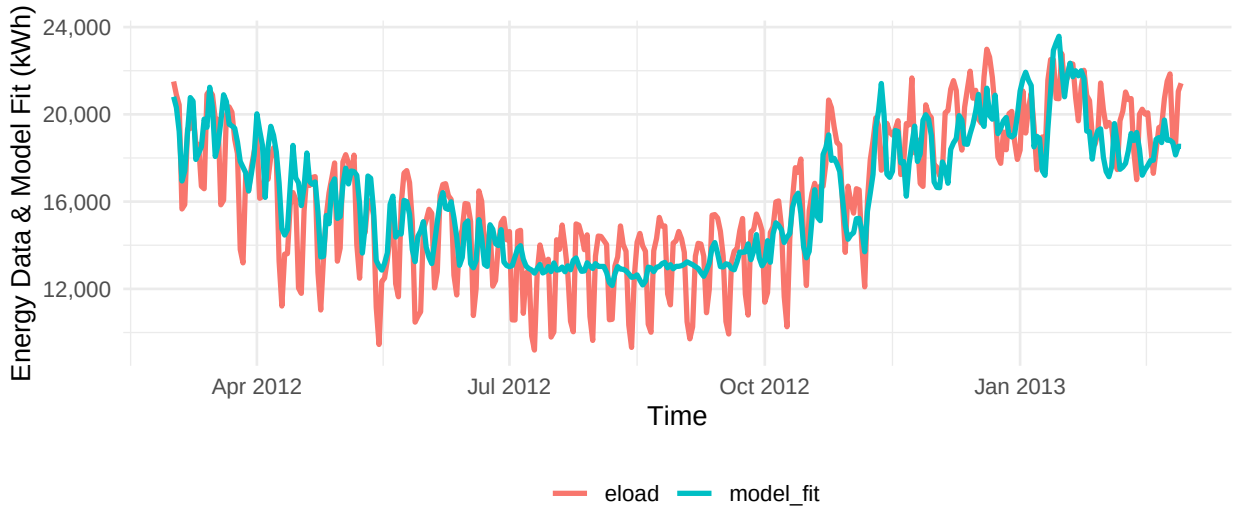

Figure 5: Baseline Energy Use modeled with Three Parameter Heating Regression over Time

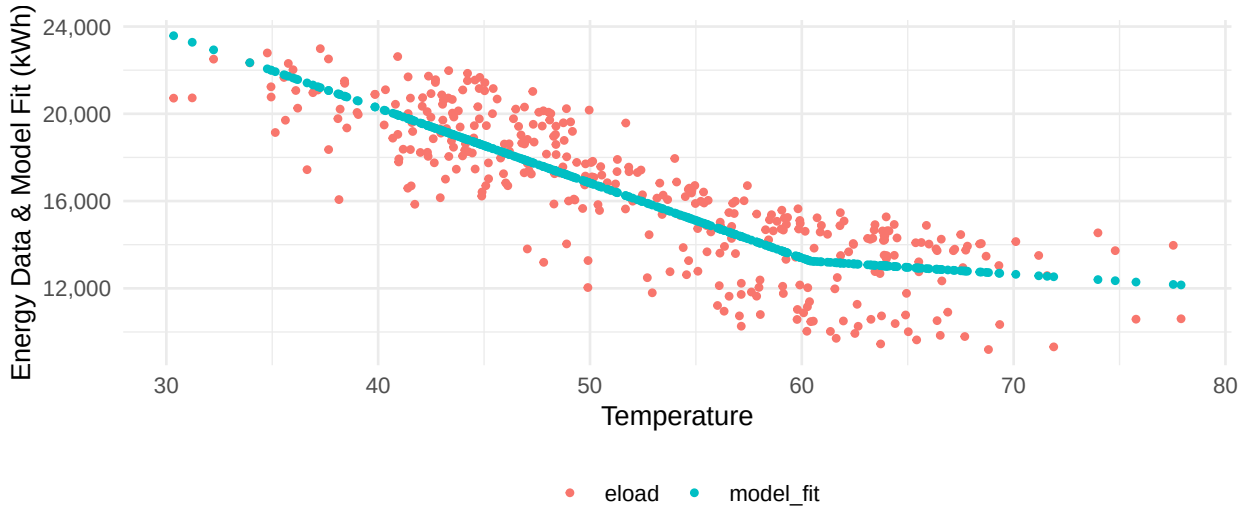

Figure 6: Baseline Energy Use modeled with Three Parameter Heating Regression over Temperature

Time of Week and Temperature:

```
TOWT_model <- model_with_TOWT(training_data = baseline_df,
                              model_input_options =
                                assign_model_inputs(regression_type = "TOWT"))
```

Figure 7: Baseline Energy Use modeled with Time-of-week and Temperature over Time

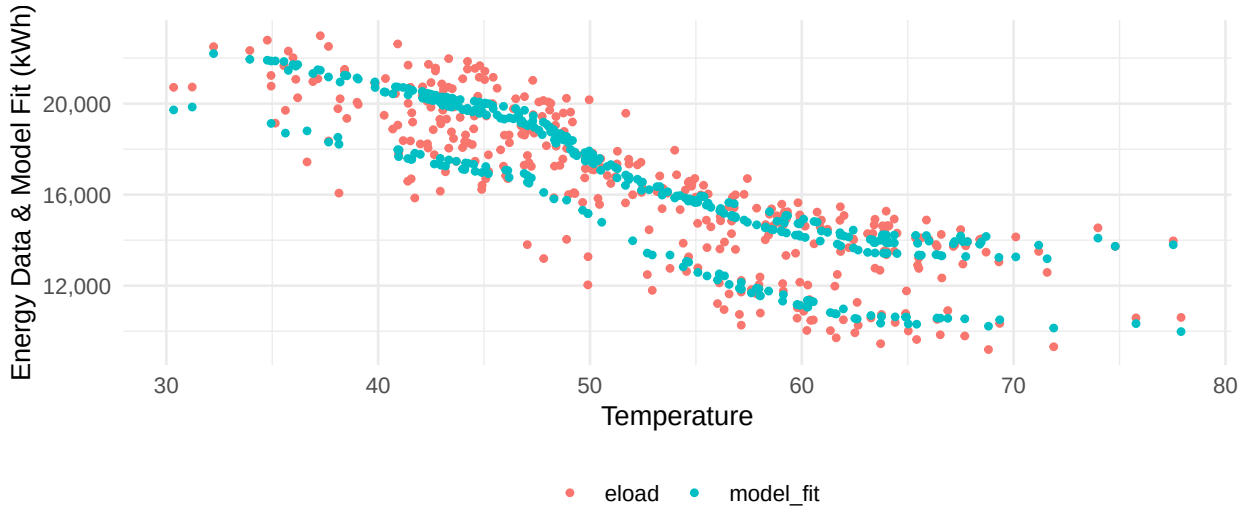

Figure 8: Baseline Energy Use modeled with Time-of-week and Temperature over Temperature

The plots show that the Time-of-Week and Temperature algorithm models the energy use profile better than the other two.

This insight can be confirmed through model statistics by using the `calculate_summary_statistics()` function. The following table summarizes the results from this function for each of the models assessed above:

```
SLR_stats <- calculate_summary_statistics(SLR_model)

Four_P_stats <- calculate_summary_statistics(Four_P_model)

TOWT_stats <- calculate_summary_statistics(TOWT_model)

all_stats <- bind_rows(SLR_stats, Four_P_stats, TOWT_stats)

model_names <- c("SLR", "Four Parameter", "TOWT")
```

```
all_stats <- bind_cols("Model Name" = model_names, all_stats)

all_stats
#>       Model Name R_squared Adjusted_R_squared CVRMSE %        NDBE %
#> 1            SLR      0.69                0.69    11.46 -8.282761e-16
#> 2 Four Parameter      0.72                0.72    10.94  2.984862e-14
#> 3           TOWT      0.90                0.89     6.79 -7.521974e-14
#>          NMBE % #Parameters deg_of_freedom
#> 1 -8.328522e-16           2            362
#> 2  3.018027e-14           4            360
#> 3 -7.936227e-14          19            345
```

- CVRMSE: Coefficient of Variation of Root Mean Squared Error
- NDBE: Net Determination Bias Error
- MBE: Mean Bias Error
- R_squared: Coefficient of Determination

Assessing project fit for NMEC

Using the modeling statistics and the savings uncertainity for 10%, we can determine the validity of the NMEC approach for a certain project. For the building decribed here, following are the key values to be used for this assessment

1. CV(RMSE): 6.7% (should be $< 25\%$)
2. NMBE: ~0.00% (should be $< 0.5\%$ and $> -0.5\%$)

The California Public Utilities Commission requires savings to be detectable above model variations. `nmecr` interprets this using ASHRAE Guideline 14 - 2014's formulation for savings uncertainty, which relates the savings uncertainty to the model goodness of fit metric CV(RMSE), the confidence level, the amount of savings, the amount of data used to develop the model, and the amount of data required to report savings. The formulation includes the correction polynomial developed by Sun and Balthazar[1] for daily and monthly models. It also includes a correction when autocorrelation is present (which occurs mainly in models developed from daily and hourly data). LBNL has shown this uncertainty formulation with correction for autocorrelation underestimates the savings uncertainty. More work on this issue is needed. Until a better formulation is available, `nmecr` uses ASHRAE's method only as an estimation.

```
TOWT_savings_10 <- calculate_savings_and_uncertainty(prediction_df = NULL,
                                                     savings_fraction = 0.1,
                                                   modeled_object = TOWT_model,
                                                   model_summary_statistics = TOWT_stats,
                                                   confidence_level = 90)

TOWT_savings_10$savings_summary_df
#>   savings_fraction savings_uncertainty savings_frac_for_50pct_uncertainty
#> 1              0.1           0.1863996                          0.03727992
#>   confidence_level
#> 1               90
```

3. Savings Uncertainty for 10% savings (at 90% confidence level): 18.6% (shoule be $< 50\%$).

[1] Sun, Y. and Baltazar, J.C., "Analysis and Improvement of the Estimation of Building Energy Savings Uncertainty," ASHRAE Transactions, vol. 119, May 2013

The savings percentage required to meet the threshold of 50% uncertainty, at the 90% confidence level, is 4% (as shown by the output above, see: savings_frac_for_50pct_uncertainty).

In addition to evaluating the model metrics, it is essential to ensure that the modeled profile follows the actual energy use closely (see Figure 8 above), and that the four assumptions of linear regression are met by the model.

a. Linearity of Data Linearity of data is checked by plotting the predicted values (predicted energy use) against the predictor values (time, temperature). This method is straightforward when we have only one predictor variable; however, it gets increasingly tedious as the number of predictor variables increase. Another method to check for linearity in data is to plot the residuals against the predicted values.

The data fulfills this linearity assumption if the residuals are scatted randomly around the zero line.

```
TOWT_res <- TOWT_model$training_data$eload - TOWT_model$training_data$model_fit

TOWT_res_df <- data.frame("Time" = TOWT_model$training_data$time, "Residuals" = TOWT_res,
                          "Modeled_eload" = TOWT_model$training_data$model_fit)


ggplot2::ggplot(TOWT_res_df, aes(x = Modeled_eload, y = Residuals)) +
  geom_point() +
  xlab("Modeled Data") +
  scale_y_continuous(name = "Residuals", labels = scales::comma)+
  theme_minimal()
```

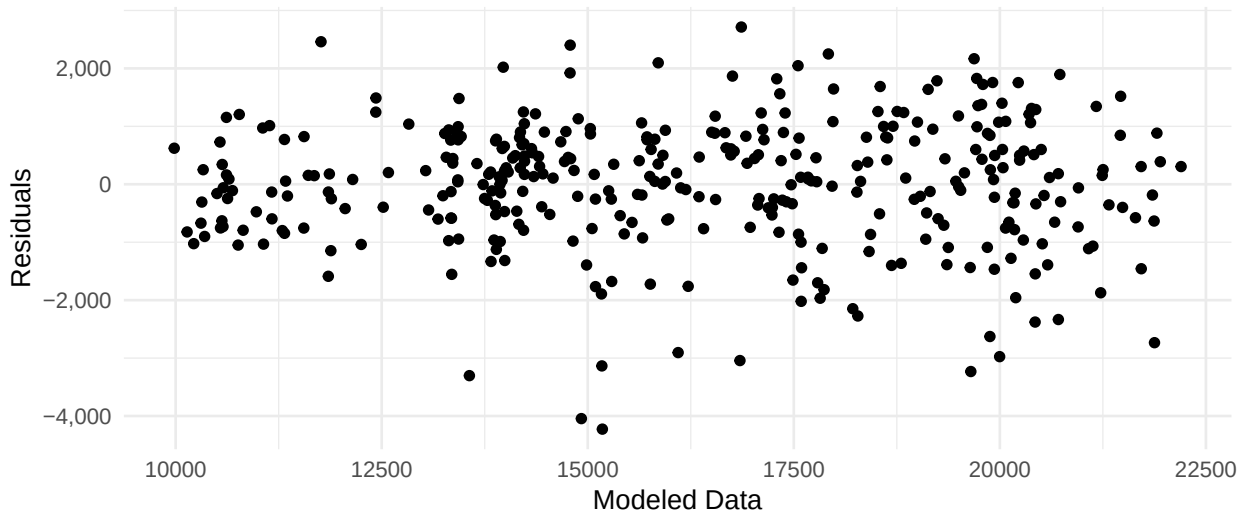

Figure 9: Linearity of Data

b. Normality of Residuals Normality of residuals is tested by plotting their histograms.

```
ggplot2::ggplot(TOWT_res_df, aes(x = Residuals)) +
  geom_histogram(binwidth = 40) +
  theme_minimal()
```

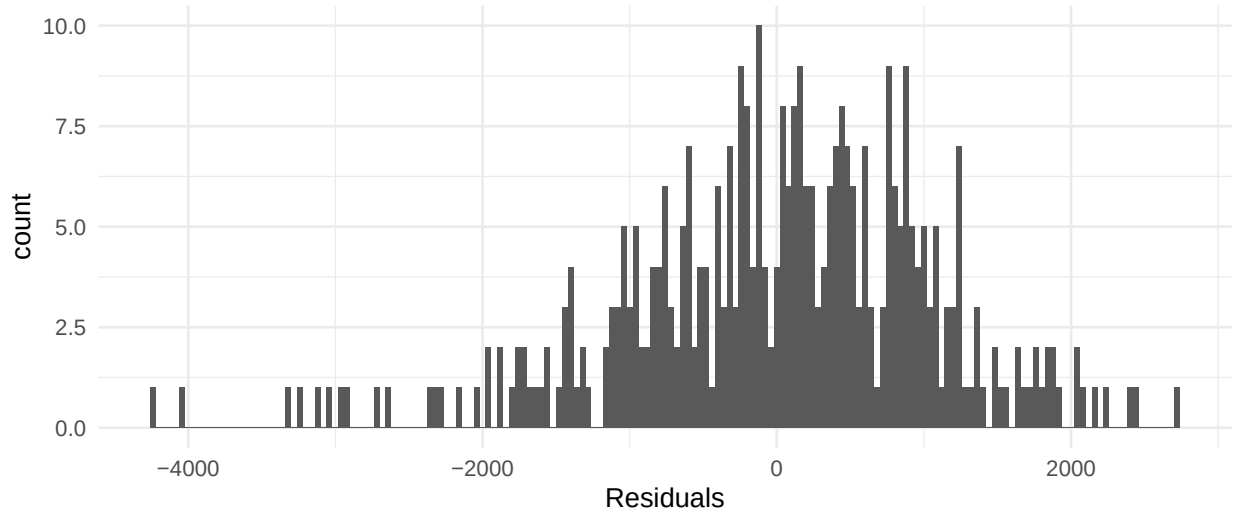

Figure 10: Normality of Residuals

Residuals, of the Time-of-Week and Temperature model, are normally distributed with a slight left skew.

c. Homoscedasticity Homoscedasticity, or homogenous variance of residuals, is tested by plotting the residuals over time. Ideally, we would like to see a constant amount of scatter around the zero line. If the scatter increases or decreases over time, the dataset is in violation of this linearity assumption.

```
ggplot2::ggplot(TOWT_res_df, aes(x = Time, y = Residuals)) +
  geom_point() +
  xlab("Time") +
  scale_y_continuous(name = "Residuals", labels = scales::comma)+
  theme_minimal()
```

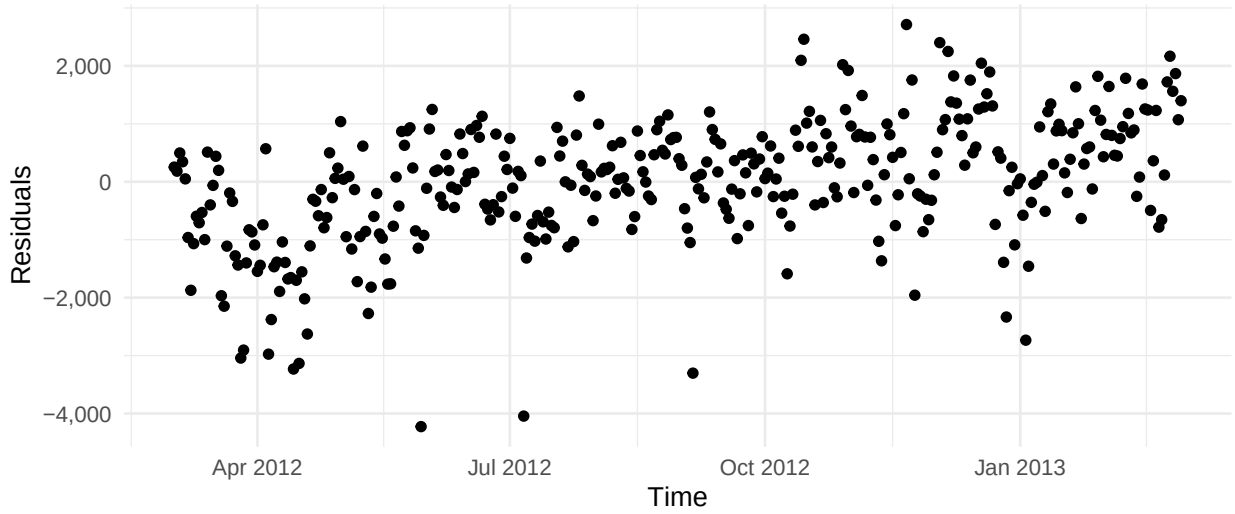

Figure 11: Homoscedasticity of Residuals

The residuals exhibit a positive slope which means that the model underpredicts during the later half of the baseline period. Additional information can be added to this model to correct for this behavior.

d. Independence of Residuals When the residuals from different time periods (usually adjacent) are correlated, we say that the residuals are autocorrelated. It occurs in time series data when the errors (residuals) associated with one time period carry over into future time periods. While this does not affect the unbiasedness of the ordinary least-squares (OLS) estimators, it causes their associated standard errors to be smaller than the true standard errors, leading to the conclusion that the estimates are more precise than they really are.

The `stats` package in R, provides a function to estimate the lag-1 autocorrelation: `acf()`. The package comes pre-loaded with R.

```
acf(TOWT_res_df$Residuals, lag.max = 1, plot = FALSE)
#>
#> Autocorrelations of series 'TOWT_res_df$Residuals', by lag
#>
#>     0     1
#> 1.000 0.606
```

The autocorrelation in the residuals at lag 1 is 0.606.

As noted above, the Time-of-Week and Temperature models for hourly and daily data have autocorrelation, due to which the associated savings uncertainity estimates are biased. Research is being conducted to create models with reduced autocorrelation. Until then, we present the savings values with the caveat that these may be underestimated.