Introduction to nmecr

Mrinalini Sharma* David Jump† Devan Johnson‡

03 December, 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. Referred to as advanced measurement and verification (M&V) by the industry, this time-granular data and updated modeling methods provide several advantages over other methods used to quantify the benefits of energy efficiency:

- It reliably determines the actual savings achieved at the meter
- It provides fast feedback on the facility's energy performance and savings progress
- It enables identification and troubleshooting of issues that prevent savings realization

The `nmecr` package streamlines the application of the NMEC approach by enabling the:

- Management of high frequency, high volume data.
- Execution of advanced, state-of-the-art time-series data modeling algorithms.
- Comprehensive assessment of the validity of energy data models in specific applications.
- Quantification of uncertainties and risks associated with the energy savings projections in accordance with ASHRAE Guideline 14.

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

Data

`nmecr` includes two datasets: `temp` and `eload`. These contain three years (03/01/2012 - 02/28/2015) of outside air temperature data and energy use data from a commercial building in North America. Energy conservation measures were installed between 03/01/2013 and 02/28/2014 in this building and so, we will use the first year (03/01/2012 - 02/01/2013) as the pre-implementation dataset and the third year (03/01/2014 - 02/28/2015) as the post-implementation dataset.

```
# 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 and Performance Period Dataframes 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 attributes 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")
```

```
#> Registered S3 method overwritten by 'tune':
#>   method                   from
#>   required_pkgs.model_spec parsnip

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
```

```
# Performance Period Dataframe

performance_df <- create_dataframe(eload_data = eload, temp_data = temp,
                                   start_date = "03/01/2014 00:00",
                                   end_date = "02/28/2015 23:59",
                                   convert_to_data_interval = "Daily")

head(performance_df, 5)
#> # A tibble: 5 x 5
#>   time                eload  temp   HDD   CDD
#>   <dttm>              <dbl> <dbl> <dbl> <dbl>
#> 1 2014-03-01 00:00:00 16744.  49.5  15.5     0
#> 2 2014-03-02 00:00:00 15989.  50.0  15.0     0
#> 3 2014-03-03 00:00:00 16033.  43.5  21.5     0
#> 4 2014-03-04 00:00:00 16938   41.8  23.2     0
#> 5 2014-03-05 00:00:00 17325.  51.8  13.2     0
```

Energy Data Modeling

The baseline and performance period dataframes can then 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:

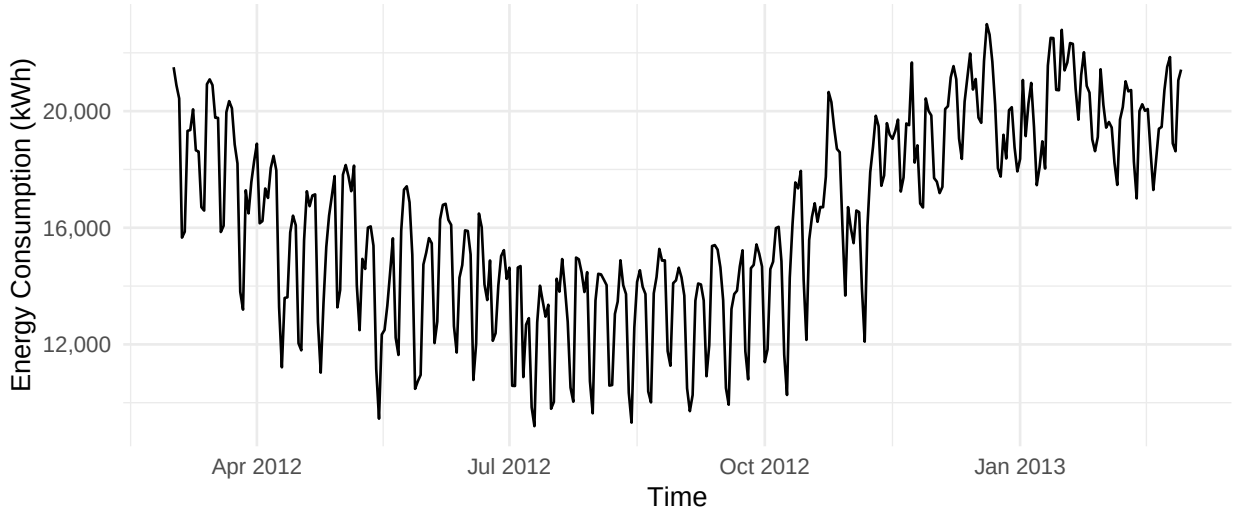

Figure 1: Baseline Energy Use over Time

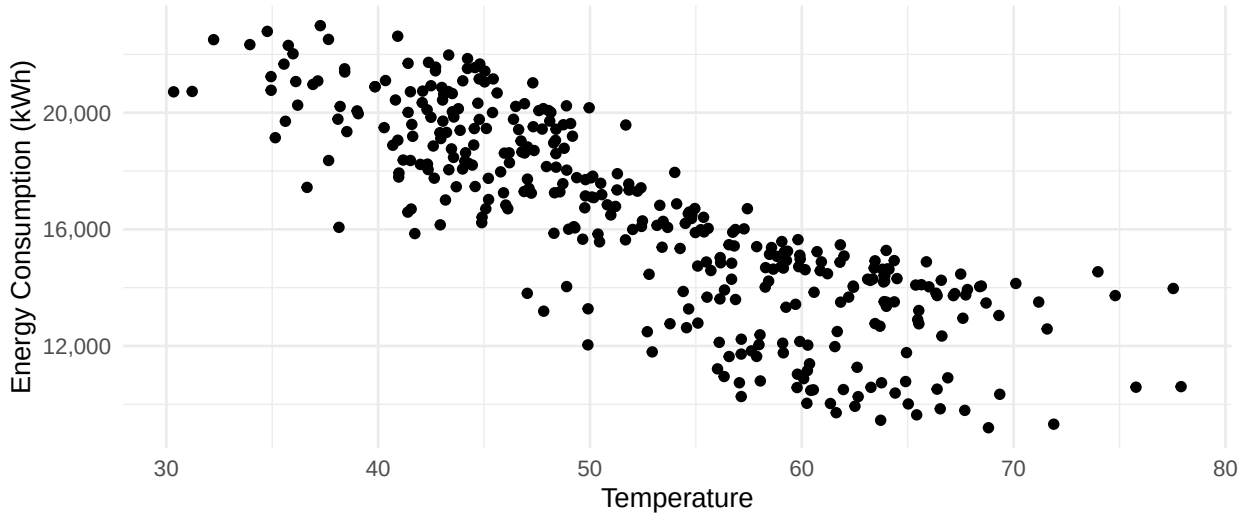

Figure 2: Baseline Energy Use over Temperature

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"))
```

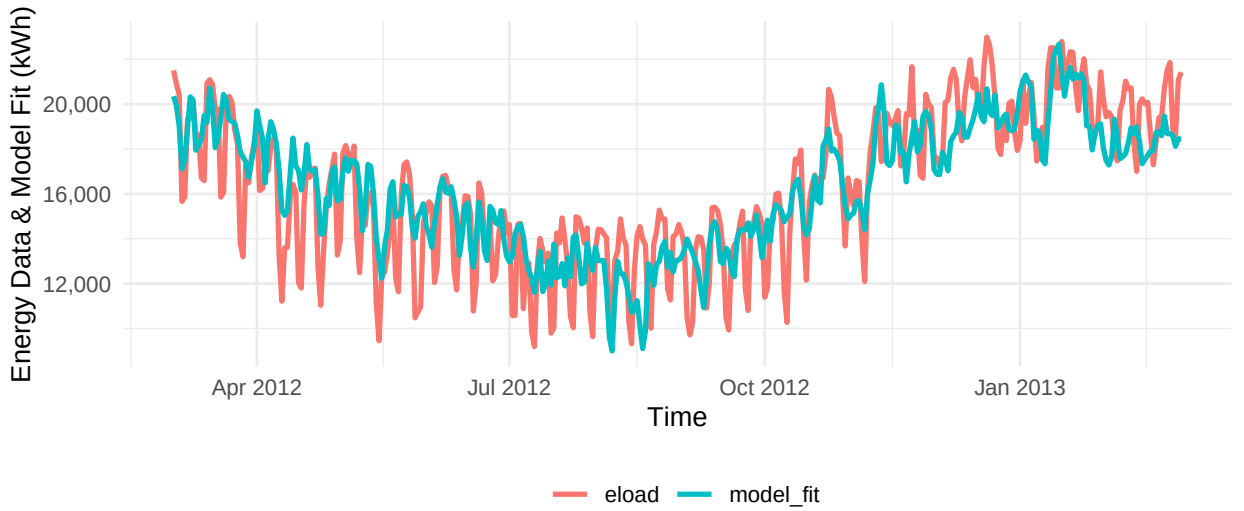

Figure 3: Baseline Energy Use modeled with Simple Linear Regression over Time

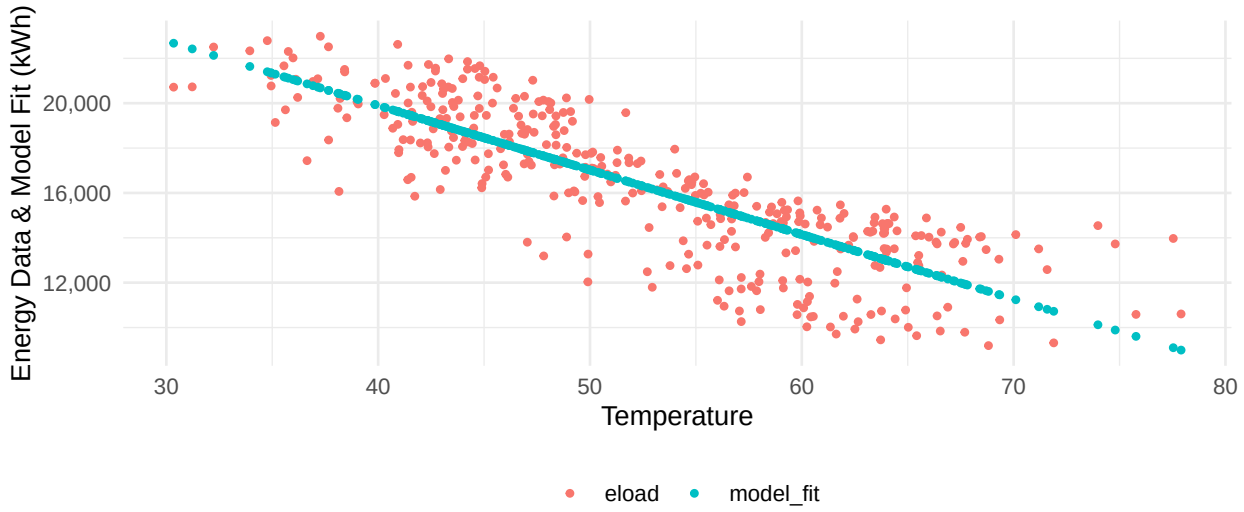

Figure 4: Baseline Energy Use modeled with Simple Linear Regression over Temperature

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"))
```

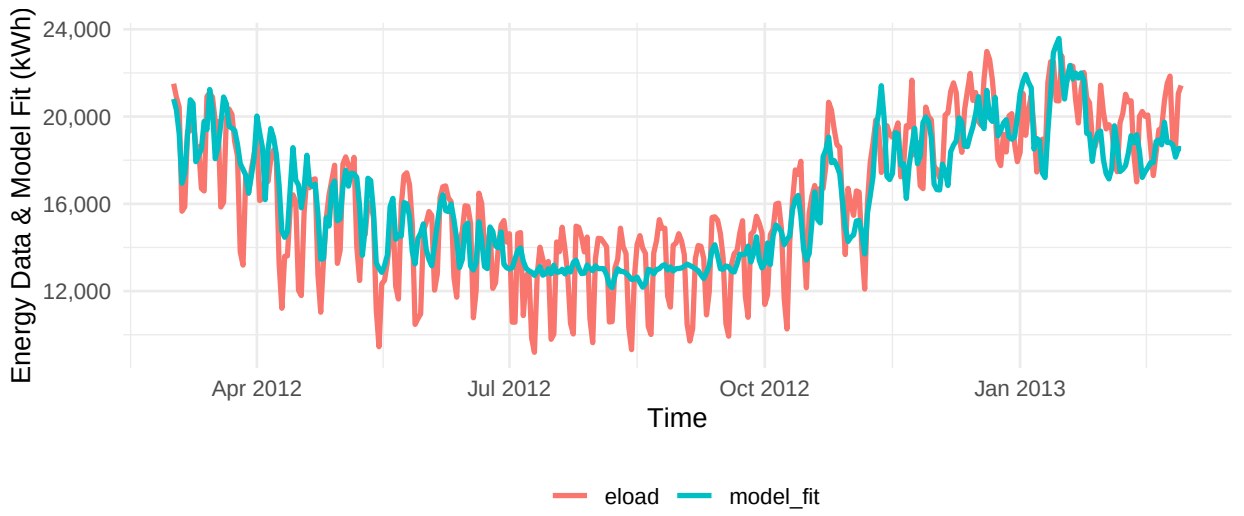

Figure 5: Baseline Energy Use modeled with Three Parameter Heating Regression over Time

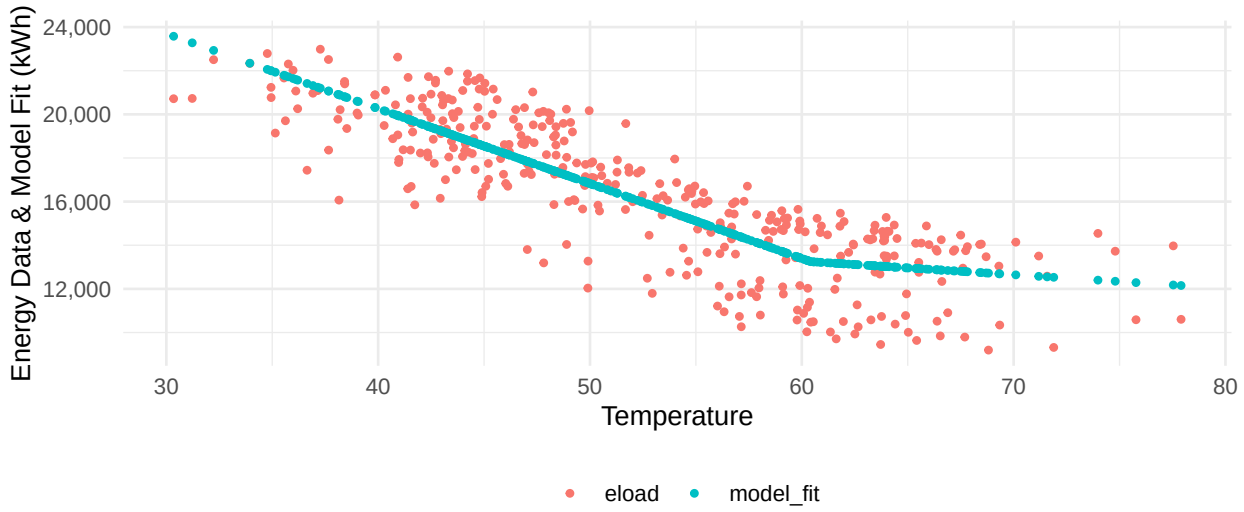

Figure 6: Baseline Energy Use modeled with Three Parameter Heating Regression over Temperature

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

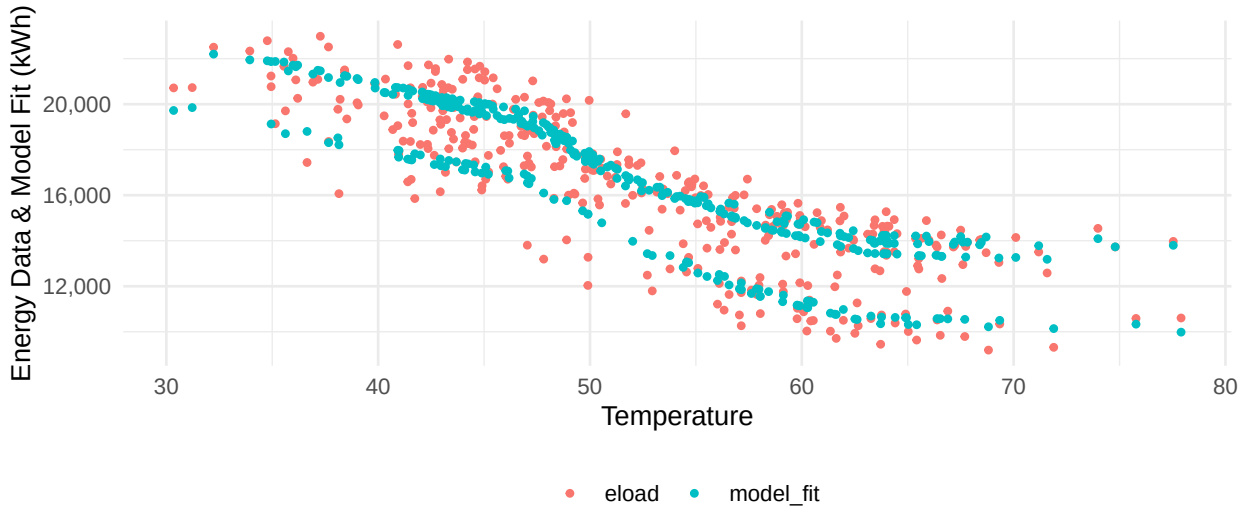

Figure 8: Baseline Energy Use modeled with Time-of-week and Temperature over Temperature

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 0.6908902                0.69    11.46 -8.282761e-16
#> 2 Four Parameter 0.7197438                0.72    10.93  9.203067e-16
#> 3           TOWT 0.8964727                0.89     6.79 -7.521974e-14
#>          NMBE % #Parameters deg_of_freedom
#> 1 -8.328522e-16           2            362
#> 2  9.279547e-16           3            361
#> 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. CVRMSE: 6.79% (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. It 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 $< 50\%$).

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).

Energy Load Prediction

The energy use models can be used to predict energy use during a different time period. Predictions in the reporting period are called the Adjusted Baseline Energy Use. The difference between the Adjusted Baseline Energy Use and reporting period actual energy use is called the Avoided Energy Use.

```
TOWT_predictions <- calculate_model_predictions(training_data = baseline_df,
                                                prediction_data = performance_df,
                                                modeled_object = TOWT_model)
```

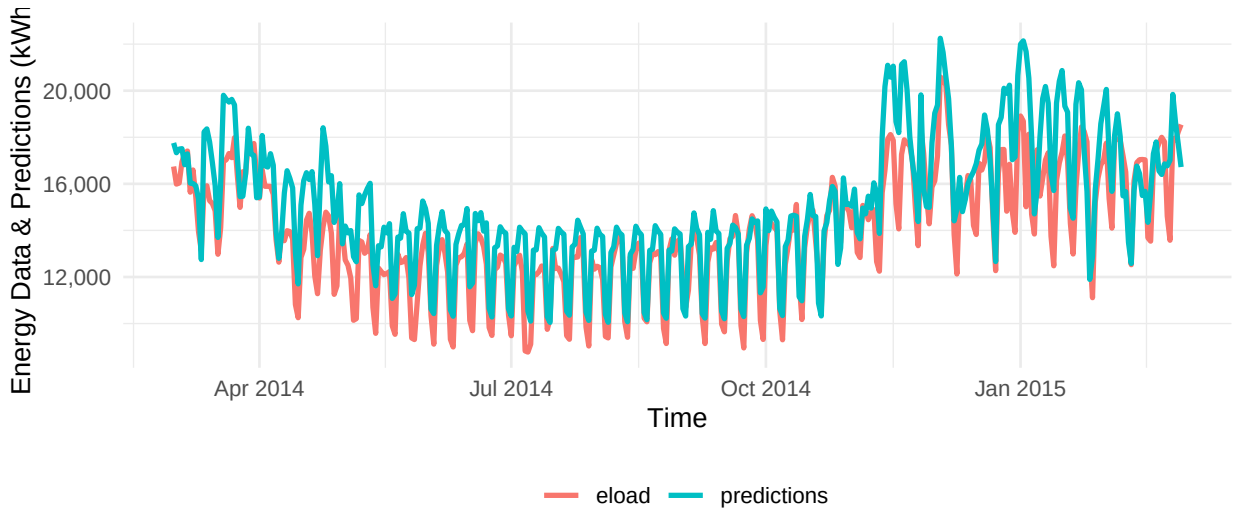

Figure 9: Performance period energy use and TOWT model predictions over Time

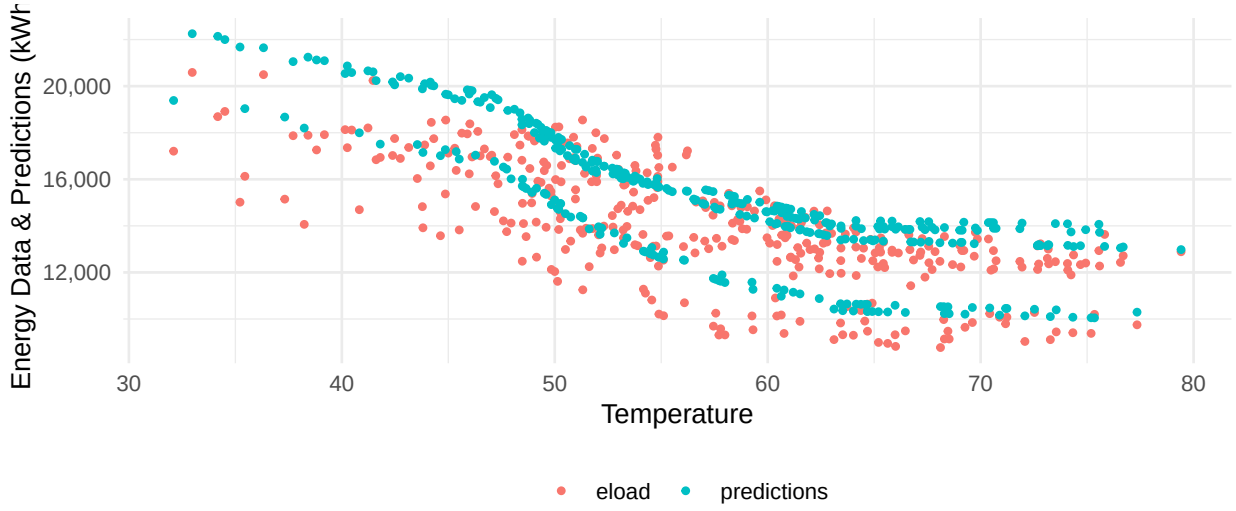

Figure 10: Performance period energy use and TOWT model predictions over Temperature

Energy Savings Achieved and Uncertainity

```
TOWT_savings <- calculate_savings_and_uncertainty(prediction_df = TOWT_predictions,
                                                  modeled_object = TOWT_model,
                                                  model_summary_statistics = TOWT_stats,
                                                  confidence_level = 90)

TOWT_savings$savings_summary_df
#>   performance_period_use adjusted_baseline_use  savings savings_fraction
#> 1                5101762                5509903 408141.6             0.07
#>   savings_uncertainty savings_frac_for_50pct_uncertainty confidence_level
#> 1           0.2659201                        0.03722881               90
```

Savings achieved at the meter for this project amount to ~7% of the Adjusted Baseline Use with an associated savings uncertainty of 26.6%. ASHRAE requires that the savings uncertainty for a project be below 50% at the 90% confidence level. The 26.6% uncertainty is well below this threshold.