Introduction to nmecr

NMEC, or Normalized Metered Energy Consumption, enables energy efficiency program stakeholders to determine energy and cost savings after measures are installed, based on the analysis of pre- and post-installation meter data. Traditionally applied with billing data, this approach has been updated with advanced modeling methods applied to high frequency data, enabling a granular savings analysis.

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.

Data

'nmecr' includes two datasets: 'temp' and 'eload'. These contain two years (03/01/2012 - 03/01/2014) of outside air temperature data and energy use data from a commercial facility in North America. We will use the first year as the pre-implementation dataset and the second year as the post-implementation dataset.

'temp' and 'eload' 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.

```
#> # 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
```

```
#> # 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.
```

Aggregating Energy Use and Temperature data

nmecr contains three main data manipulation functions:

- `agg_eload_temp_df_to_hourly()`
- `agg_eload_temp_df_to_daily()`

- `agg_eload_temp_df_to_monthly()`

These functions perform two tasks:

1. Aggregating data from small time intervals to large time intervals.
2. Combining eload and temp datasets into one data frame.

Note that the temp and eload do not need to necessarily have the same time interval. For example, you can input 15-min electric consumption data and hourly temperature data to `agg_eload_temp_df_to_daily()` to generate aggregated electric consumption and temperature data at the daily level.

```
daily_energy_use <- agg_eload_temp_df_to_daily(eload_data = eload, temp_data = temp,
                                               data_is_quantity = T,
                                               balancepoint_temp = 65)
```

```
#>         time    eload     temp      HDD CDD
#> 1 2012-03-01 21505.44 38.41722 26.58278   0
#> 2 2012-03-02 20892.24 39.85819 25.14181   0
#> 3 2012-03-03 20434.56 43.04194 21.95806   0
#> 4 2012-03-04 15660.48 49.65604 15.34396   0
#> 5 2012-03-05 15863.52 48.29951 16.70049   0
```

Separating pre-implementation and post-implementation datasets

Once the eload and temp datasets are combined into one data frame, the next step is to create the pre- and post-implementation data frames. You can choose any method to do so. We recommend using the dplyr and the lubridate package.

```
daily_pre <- daily_energy_use %>%
  filter(time < lubridate::mdy("3/1/2013"))

daily_post <- daily_energy_use %>%
  filter(time > lubridate::mdy("2/28/2013"))
```

Data Modeling

nmecr contains four data modeling algorithms. These create the data models, report on the models' goodness-of-fit metrics, provide details on the model residuals' skewness and kurtosis, and predict using the developed model when given a prediction dataset.

- `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

`model_with_TOWT()` and `model_with_SLR()` have three additional arguments: has_operating_modes, train_operating_mode_data, pred_operating_mode_data. We use these arguments for facilities that have more than one operating mode. Schools, for example, have a distinct operating mode in the summer. For usage of these arguments, please refer to the function documentation (type ?model_with_SLR or ?model_with_TOWT in the console.)

Modeling with Time of Week and Temperature

Simple Linear Regression and its derivatives (changepoint models and HDD/CDD models) are the most popular data modeling algorithms in the energy efficiency community. The Time-of-Week and Temperature model, developed by Lawrence Berkeley National Laboratory, has improved upon the predictive capabilties of these algorithms by adding an additional variable to regress upon: Time-of-Week. While primitive implementations of TOWT required the occupancy schedule of the facility to be an input to the algorithm, the implementation below gleans this information from the energy use profile itself. As a result, the only independent variable input required by the algorithm is temperature, which is made available in the daily_pre and the daily_post data frames already.

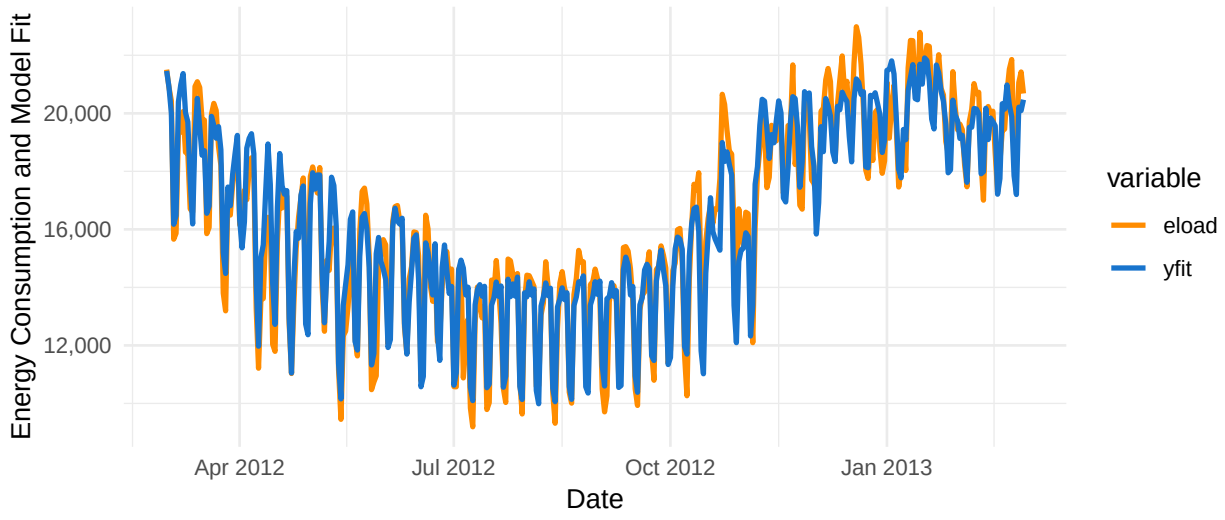
```
TOWT_results <- model_with_TOWT(training_data = daily_pre, data_interval = "Daily",
                                data_units = "kWh")
```

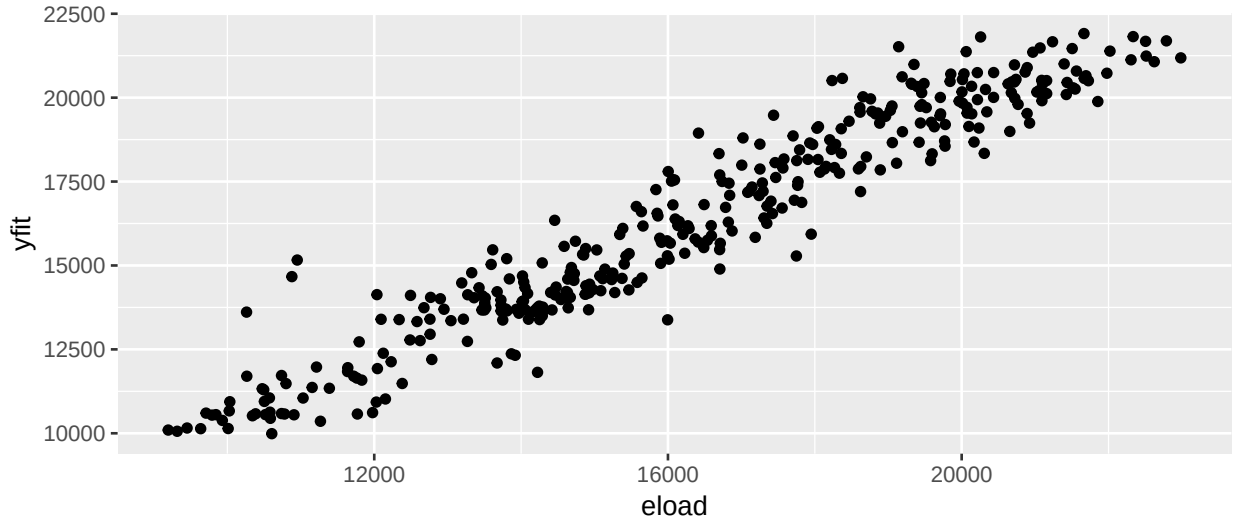
The `TOWT_results` object contains all information pertaining to the developed model, some of which is shown below:

Model Goodness of Fit:

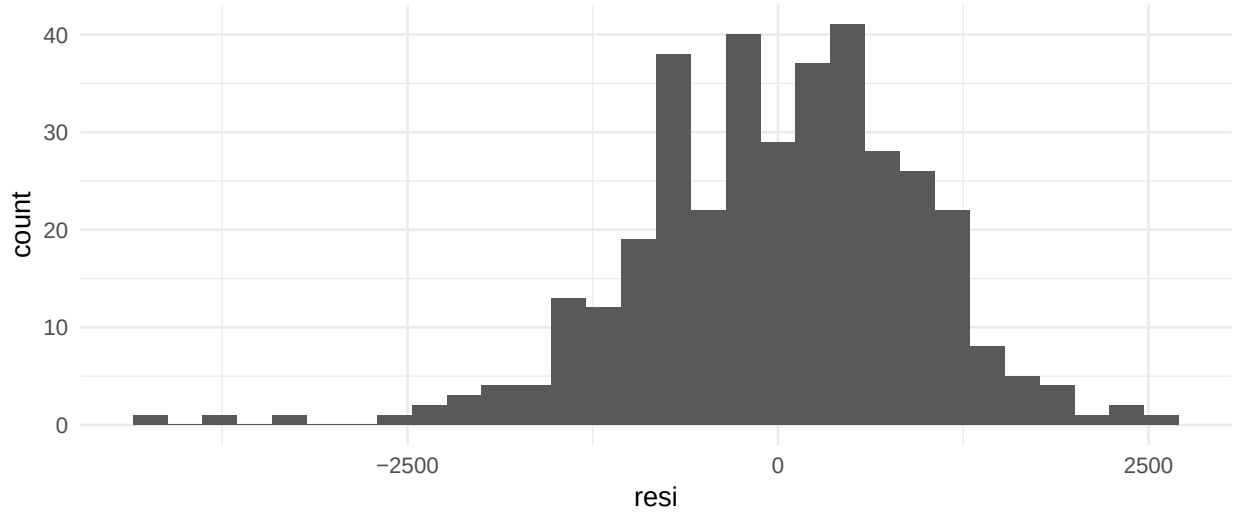
```
#>      fit_R2 fit_CVRMSE  fit_NDBE #Parameters
#> 1 0.9207413 0.06014035 1.39e-14%          26
```

Model Results:





Model Residuals:




```
#>            Skewness       Excess Kurtosis
#> 1 Moderately Skewed Profusion of outliers
```

The California Public Utility Commission requires savings uncertainty for 10% savings at the 90% confidence level to be reported to qualify a model as 'good' and fit for use in energy efficieny programs. This information can be obtained using the `calculate_savings_uncertainty()` function:

```
savings_uncertainty_10 <- calculate_savings_uncertainty(modeled_data_obj = TOWT_results,
                                                         savings_percent = 10)
```

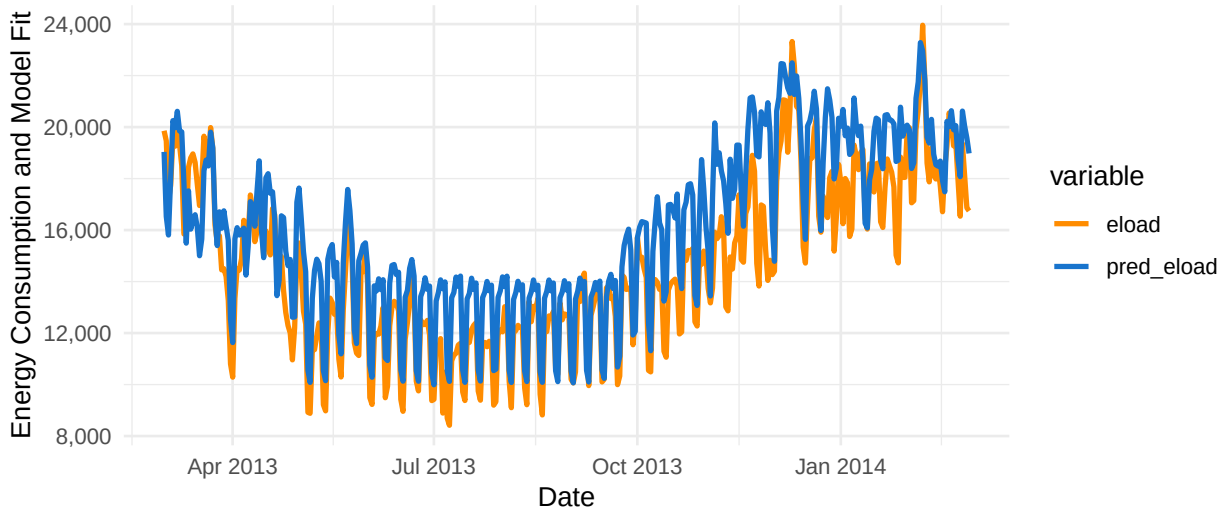
```
#>   Savings Uncertainty Min. Savings needed at 50% uncertainty
#> 2              14.24%                                169,406
```

The uncertainty for 10% savings at a confidence level of 90% is 14.2% for this model. The corresponding minimum savings needed for this project are 169,406 kWh

Predicting energy use using Time-of-Week and Temperature algorithm

We can create model predictions by simply adding an argument with the prediction dataframe to the `model_with_towt()` function:

```
TOWT_predictions <- model_with_TOWT(training_data = daily_pre, prediction_data = daily_post,
                                    data_interval = "Daily", data_units = "kWh")
```



The predictions can further be summarized to calculate 'Avoided Energy Use' for the project:

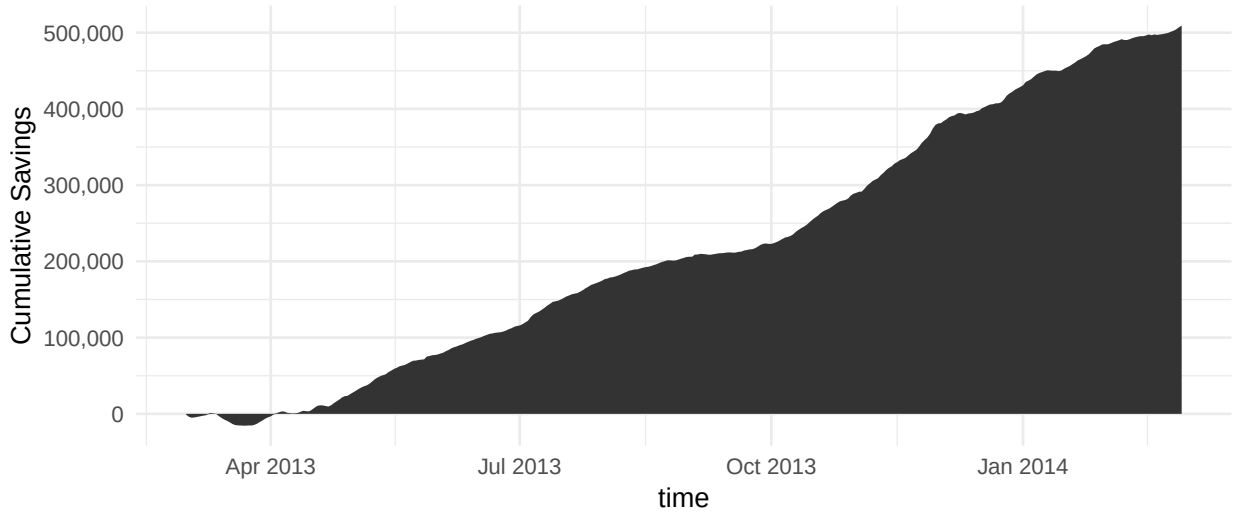
Avoided Energy Use and Savings Uncertainty

```
avoided_energy_use <- calculate_avoided_energy_use(modeled_data_obj = TOWT_predictions)
```

```
#>   Savings Savings (%)
#> 1  509314       8.71%
```

```
actual_savings_uncertainty <- calculate_savings_uncertainty(modeled_data_obj =
                                                              TOWT_predictions,
                                                            savings_percent =
                                                            avoided_energy_use$pct_savings)
```

```
#>   Savings Uncertainty Min. Savings needed at 50% uncertainty
#> 2              16.34%                                169,406
```



Normalized Savings

Normalized Savings for an energy efficiency project can be calculated using the functions described above by assigning a normalized temperature dataset as the prediction dataset. Remember to name the dataframe columns as: 'time' and 'temp'.