2022 Residential Building Stock Assessment

Data User Guide

Published April 2024

Table of Contents

User Guide Purpose

This document serves to provide instructions for data users on how to prepare the data tables and summary data tables for analysis. It provides background information on what questions the Study was and was not designed to answer. The user guide also includes instructions and code snippets[1] in both R and Python for applying weights, joining tables, and quantifying uncertainty, as well as an entity relationship diagram and general information regarding quantifying uncertainty.

Study Background

The Northwest Energy Efficiency Alliance (NEEA) completed the third Residential Building Stock Assessment (RBSA) in 2024. The study covers both single-family homes (including manufactured homes) and multi-family buildings and dwelling units.

> **The main objective of the 2022 RBSA database is to provide a current, robust, and representative characterization of the existing residential single-family and multi-family building stock in the Northwest.**

The focus of the data collection effort was to document existing building characteristics and equipment that relate to building energy consumption. The 2022 RBSA remains mostly consistent with prior RBSA studies. The study team worked to balance maintaining the integrity of prior approaches (including consistency of collected data for time series analysis) while taking the opportunity to make improvements and updates.

What type of analyses do the 2022 RBSA datasets support?

The 2022 RBSA datasets support estimation of regional and subregional building stock characteristics and the characteristics of energy using equipment in residential spaces across the Northwest (i.e., lighting, appliances, electronics, and mechanical equipment). This includes

[1] Note that the provided code snippets are illustrative, and analysts should adjust them as necessary to work within their production environment.

the ability to estimate the prevalence of equipment types and subtypes, as well as understand their characteristics (e.g., fuel types, vintages, efficiencies, sizes). The datasets also support analysis of building shell characteristics (e.g., windows, walls, insulation) and provides information regarding energy use and building/duct tightness for a subset of participant homes.

The datasets are intended to provide a “snapshot” of residential building characteristics across the Northwest.

There are numerous potential analytical uses for these datasets. For example:

- Distribution of refrigerator types by state
- Prevalence of cooling equipment by cooling zone
- Distribution of common area heating equipment type by height class for multifamily buildings

When analyzed with prior and future RBSA studies, the datasets also support analysis of trends over time. This type of analysis may reveal market transformation successes and remaining potential. The research team developed a mapping of data dictionaries across RBSA studies that provide associations of common (or near common) variables from each study to support trends over time analysis (see the **Data Dictionary.xlsx** file in the downloadable database).

What type of analyses do the 2022 RBSA datasets not support?

Importantly, while the 2022 RBSA collected participant demographic data, the study did not explicitly sample or control for demographics, but rather the geographic and other characteristics of the homes. As a result, the 2022 RBSA does not support demographic-based analyses, such as the saturation of equipment types by highest level of education attained within a household.

Additionally, the sample of multi-family buildings in Idaho and Montana is small and therefore, the 2022 RBSA datasets do not support Idaho- or Montana-specific multi-family building or tenant unit analysis.

Relatedly, we do not recommend relying on analysis with especially small sample sizes (where the sample of homes is less than 10 homes). This is usually the result of overly specific analysis objectives. While analysis of this nature is possible, uncertainty in the results is too high to confidently represent the population. There are numerous analyses which may lead to low sample sizes, here are a few examples:

- Average water heater size (gallons) for homes under 2,000 square feet built in Oregon since 2020 (n = 8)
- Percent of single family attached homes in Heating Zone 3 with a furnace controlled by programmable thermostat (n = 5)
- Average size (square inches) of glass curtain windows in high-rise multifamily buildings (n=3)

The 2022 RBSA includes manufactured homes as single-family homes, and they were not sampled as one or a series of separate strata. As a result, when making comparisons between the 2022 RBSA and prior RBSA studies, it is important to consider their presence in the current definition of single-family home (in the past they were studied separately from single-family homes).

The 2022 RBSA also does not support analysis by manufacturer to estimate a manufacturer's recent sales. The 2022 RBSA does not consider regional availability of equipment by manufacturer (or retailers, distributors, or installers that only sell specific brands of equipment).

Accessing and Preparing 2022 RBSA Data

The 2022 RBSA datasets consist of a series of comma-separated values (CSV) files on NEEA's website, as well as an Excel file containing the data dictionary. These files can be downloaded here: **neea.org/rbsa.**

Once downloaded, users can view and analyze the data using Microsoft Excel, Google Sheets, Numbers (on MacOS), or any number of statistical analysis software packages (e.g., R, SPSS, Python, SAS). The use of statistical analysis software packages is advisable for more complicated analysis tasks and to ensure replicability of analysis results.

Joining Tables

The database CSV files relate across datasets based on certain key identifier variables. Participant sites have site-level characteristics, room- or space-level characteristics, and equipment-level characteristics. The datasets are organized to facilitate the connection of equipment in spaces to the rooms and sites via the key identifier variables.

For examples, all files have a Site ID variable so that they can be easily joined to the site datasets and relevant information obtained (e.g., site geography, weights). Most files have a Room ID variable that links the characteristic or piece of equipment to a specific room in the rooms.csv data file for any rooms-based analysis.

See the Database Schema section for a map of how datasets can be linked via joins.

Below are examples in R and Python for joining datasets. The examples show how to add the state to the CSV file containing information about televisions (appliance_television.csv).

```
#### R Code Demonstrating Joining ####
# Example: Adding information on site state to the Appliance_Television table
library(tidyverse)

example_join <- left_join(
  Appliance_Television,
  SiteDetail %>% select(SiteID, State), # Select relevant information from SiteDetail table
  by = " SiteID" # Key to join tables (see Entity Relationship Diagram)
)
```

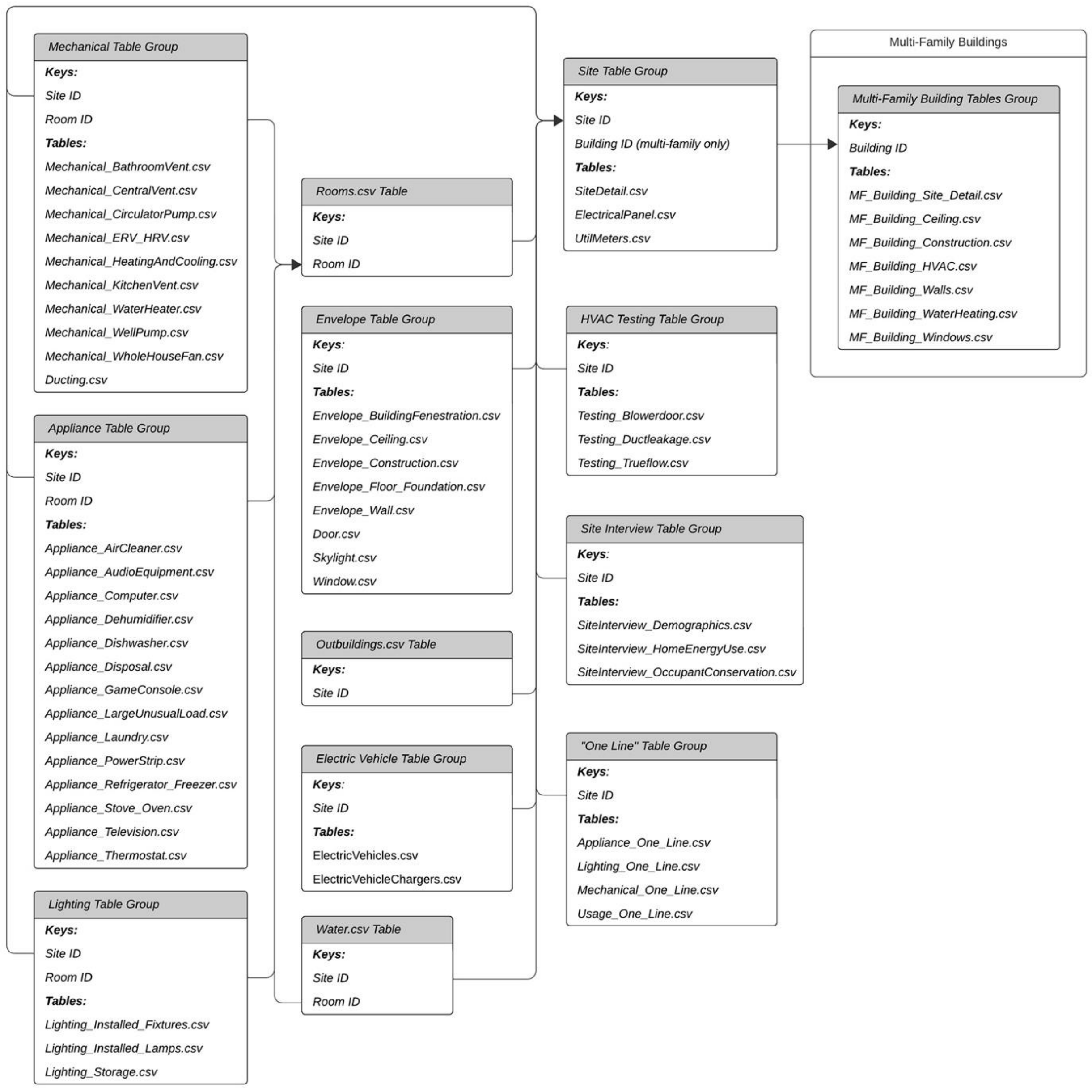
```
# Python Code Demonstrating Joining
# Example: Adding information on site state to the Appliance_Television table

import pandas as pd

# Performing a left join to add state information from SiteDetail to Appliance_Television
example_join = pd.merge(
    Appliance_Television,
    SiteDetail[['SiteID', 'State']],  # SiteID and State columns from SiteDetail
    on='SiteID',
    how='left'
)
```

Database Schema

The figure below provides a visual representation of the 2022 RBSA database. All detailed datasets are linked to sites, and some equipment types are mapped to rooms within sites. Multi-family sites (i.e., tenant units) are further linked to buildings.



Applying Weights

Case weights are available for single family sites and multifamily tenant units (Site_Case_Weight in SiteDetail.csv), and multifamily buildings (Building_Case_Weight in MF_Building_Site_Detail.csv). Analysis should be done using these weights. The 2022 RBSA used a stratified sample design, and the sample weight for site i in stratum h is calculated as:

$$w_i = \frac{N_h}{n_h},$$

Where N_h is the (estimated) total number of sites in the population that meet the requirements of stratum h, and n_h is the number of stratum h sites included in the final sample. The weights represent the estimated number of sites that are similar to the given site based on strata characteristics. To use the provided weights:

- First, identify the correct weighting variable
- Second, merge the weighting variable into the dataset of interest
- Third, conduct analysis while applying weight

For example, the weighted average of known television sizes by state could be calculated as:

$$TV_{ave} = \frac{\sum TV_{size,\,i} \times w_i}{\sum w_i}$$

Here, the summations in the numerator and denominator are over all 2022 RBSA television records that have valid size entries. If a site has multiple televisions, each television's size is multiplied by the site's weight and included in the numerator, and that site's weight is repeated multiple times in the denominator. A site that has zero televisions does not enter the calculation at all.

This calculation can be done using the following code:

```
#### R Code Demonstrating Weighted Analysis ####
# Example: Weighted Average of Known Television Sizes by State
library(tidyverse)


television_data_with_state_and_weight <- left_join(
  Appliance_Television,
  SiteDetail %>% select(SiteID, State, Site_Case_Weight),
  by = "SiteID"
)

television_data_with_state_and_weight %>%
  filter(Size!="Unknown") %>% # Remove unknown size TVs
  mutate(Size=as.numeric(Size)) %>% # Ensure that size is a numeric variable
  group_by(State) %>%
  summarize(
    average_television_size = weighted.mean(Size, w=Site_Case_Weight) # Calculated weighted
average of TV size using Site_Case_Weight as weight
  )
```

# Python Code Demonstrating Weighted Analysis

```
# Example: Weighted Average of Known Television Sizes by State
import pandas as pd
import numpy as np

television_data_with_state_and_weight = pd.merge(
    Appliance_Television,
    SiteDetail[['SiteID', 'State', 'Site_Case_Weight']],
    on='SiteID',
    how='left'
)

# Filtering out rows where Size is 'Unknown', converting Size to numeric
television_data_grouped = television_data_with_state_and_weight[
    television_data_with_state_and_weight['Size'] != 'Unknown'
].copy()
television_data_grouped['Size'] = pd.to_numeric(television_data_grouped['Size'])
```

```
# Calculating the weighted average of TV size by state
average_television_size = television_data_grouped.groupby('State').apply(
    lambda x: np.average(x['Size'], weights=x['Site_Case_Weight'])
).reset_index(name='average_television_size')
```

Note in particular how the numerator, calculated as the sum-product size-weights, tracks with the intention of estimating the regional total of television sizes, since each site weight estimates the number of similar sites within the region. Similarly, the denominator counts the total number of regional televisions included in the numerator. This kind of reasoning—using weights to estimate one regional total, and then dividing by another estimated regional total—is a very common approach to constructing valid and efficient estimates from weighted sample data. For example:

- To estimate the average number of televisions per household, take the sum-product (over all 2022 RBSA sites that observed number of televisions, including sites with zero televisions) of the site television count and site weight, and divide by the sum of all included sites' weights.
- To estimate the average gas furnace capacity per conditioned square foot in heating zone 1, take the sum-product (over all heating zone 1 gas furnaces homes for which furnace capacity and conditioned square footage were both recorded) of furnace capacity and site weight, and divide by the sum-product (over the same set of sites) of conditioned square footage and site weight.

Recalculating Weights

Certain situations may require recalculating weights. For example, due to the limited number of sites with available energy usage data, the energy usage analysis needed to be weighted using recalculated weights. In most cases, you should not need to recalculate weights. When only a subset of the sample was included in the data collection (e.g., homes for which usage data was available), analysts should carefully consider whether re-weighting is appropriate.

Recalculating weights can be done by dividing the strata population estimate (found in the SiteDetail.csv and MF_Building_Site_Detail.csv tables) by the number of available sites/buildings to recalculate new weights. Example code is provided next.

```
#### R Code For Recalculating Weights ####
# Example: Recalculating Weights for User Determined sites_included_in_analysis
library(tidyverse)

sites_included_in_analysis %>% # subset of SiteDetail table with only sites included in
analysis
  group_by(Strata_Territory) %>%
  mutate(
    New_Site_Case_Weight = Strata_Population_Estimate / n() # New weight is equal to strata
population estimate divided by sample size of strata
         )
```

```
# Python Code For Recalculating Weights
# Example: Recalculating Weights for User Determined sites_included_in_analysis
import pandas as pd

# Grouping by 'Strata_Territory', then calculating the new site case weight
sites_included_in_analysis['New_Site_Case_Weight'] =
sites_included_in_analysis.groupby('Strata_Territory')['Strata_Population_Estimate'].transform(
    lambda x: x / x.count()
)
```

Uncertainty

When analyzing 2022 RBSA data, we quantify the uncertainty in the empirical data with **smoothed bootstrap** confidence intervals. This technique is more accurate than traditional percentile-based methods for metrics that are skewed or fail any of the other assumptions of a parametric distribution. Bootstrapping accomplishes this by estimating uncertainty from the observed data and portrays the most complete picture available to us of any skewness or bias that may be present. Bootstrapping is the best available technique to estimate uncertainty of metrics in the 2022 RBSA, ensuring that findings are both robust and credible.[2]

Statistical uncertainty estimates generally seek to answer this question: *Based on what the sample tells us about population characteristics, what can we infer about the range of values we would obtain if we repeatedly gathered new data samples and recalculated the same estimator for each.* Many traditional methods deal with this question by looking at how much the sample data varies, which is called the sample variance. Then, they rely on the central limit theorem, which says that as you gather more data, the sample averages tend to behave like a normal distribution, even if the original data isn't normally distributed. The bootstrap resampling method addresses the basic question much more directly by selecting a series of random samples from the original dataset with replacement. Each of these resamples is then used to calculate the estimate of interest, such as the mean or proportion. By repeating this process many times, we develop a new distribution of estimates rather than a single estimate for the full sample, as you would when creating a sampling distribution. We then analyze the distribution of sample estimates to determine a reasonable confidence for the population value (e.g., population mean or population proportion).

Any confidence interval estimated from a **small sample** has the potential to **overstate confidence** (i.e., estimating unrealistically tight error bounds) if the sample measurements are similar by random chance (if none of the natural outliers made it into the sample). Larger sample sizes (in terms of the number of sites or number of pieces of equipment of a certain type) will improve the accuracy of the estimated characteristics and accuracy of the confidence interval—the confidence intervals do not necessarily get tighter (if more outliers are observed), but they will provide a more realistic estimate of the true variability across sites.

Due to the **significant uncertainty** associated with small samples, we have not provided confidence intervals derived from bootstrap resampling when the sample size is smaller than 10, as such intervals may be overstated. The 2022 RBSA Findings Report Appendix omits confidence intervals for sample sizes below this threshold. Furthermore, we advise users to be cautious when interpreting confidence intervals for sample sizes just above this threshold, as the confidence intervals may still be unreliable and could overstate the precision of the estimates. Also, calculated confidence intervals do not account for unquantified factors such as potential self-selection bias, which could skew the sample in ways that can be difficult to detect.

[2] If you are calculating uncertainty, especially using Excel, for many applications it may be practical to use the standard formulas provided in resources like the UMP Sample Design chapter (Ch. 11, Appendix B, here: https://www.nrel.gov/docs/fy17osti/68567.pdf)

Example code for implementing bootstrap resampling is provided next.

```
#### R Code For Bootstrapping ####
# Example: Confidence Interval for Average of Known Television Sizes by State

# Function to perform weighted bootstrap resampling and calculate 95% CI
bootstrap_ci_weighted <- function(variable, weights, n_iterations = 1000) {
  # Check if the number of observations is less than 10
  if (length(variable) < 10) {
    return(c(Lower_CI = NA, Upper_CI = NA))
  }

  means <- numeric(n_iterations)

  for (i in 1:n_iterations) {
    # Sample with replacement, considering weights
    sampled_indices <- sample(seq_along(variable), size = length(variable), replace = TRUE, prob =
weights)
    sample <- variable[sampled_indices]
    # Calculate mean
    means[i] <- mean(sample)
  }

  # Calculate 95% confidence interval
  ci <- quantile(means, probs = c(0.025, 0.975))
  return(ci)
}

# Example of applied function
television_data_with_state_and_weight %>%
  filter(Size!="Unknown") %>% # Remove unknown size TVs
  mutate(Size=as.numeric(Size)) %>% # Ensure that size is a numeric variable
  group_by(State) %>%
  summarise(
    average_television_size = weighted.mean(Size, w=Site_Case_Weight), # Calculated weighted
average of TV size using Site_Case_Weight as weight
    Lower_CI = bootstrap_ci_weighted(Size, Site_Case_Weight, 1000)[1],
    Upper_CI = bootstrap_ci_weighted(Size, Site_Case_Weight, 1000)[2]
  )
```

# Python Code For Bootstrapping

```
# Example: Confidence Interval for Average of Known Television Sizes by State
import pandas as pd
import numpy as np
from sklearn.utils import resample

# Function to perform weighted bootstrap resampling and calculate 95% CI
def bootstrap_ci_weighted(variable, weights, n_iterations=1000):
    # Check if the number of observations is less than 10
    if len(variable) < 10:
        return {'Lower_CI': np.nan, 'Upper_CI': np.nan}

    means = []

    for _ in range(n_iterations):
        # Sample with replacement, considering weights
        sampled_indices = np.random.choice(range(len(variable)), size=len(variable), replace=True,
p=weights/np.sum(weights))
        sample = variable[sampled_indices]
        # Calculate mean
        means.append(np.mean(sample))
    # Calculate 95% confidence interval
    ci = np.percentile(means, [2.5, 97.5])
    return {'Lower_CI': ci[0], 'Upper_CI': ci[1]}

# Filter out TVs with unknown size and convert the Size column to numeric
television_data_with_state_and_weight =
television_data_with_state_and_weight[television_data_with_state_and_weight['Size'] != "Unknown"]
television_data_with_state_and_weight['Size'] =
pd.to_numeric(television_data_with_state_and_weight['Size'])

# Group by state and calculate summary statistics
result = television_data_with_state_and_weight.groupby('State').apply(
    lambda df: pd.Series({
        'average_television_size': np.average(df['Size'], weights=df['Site_Case_Weight']),
        **bootstrap_ci_weighted(df['Size'].values, df['Site_Case_Weight'].values, 1000)
    })
)
```