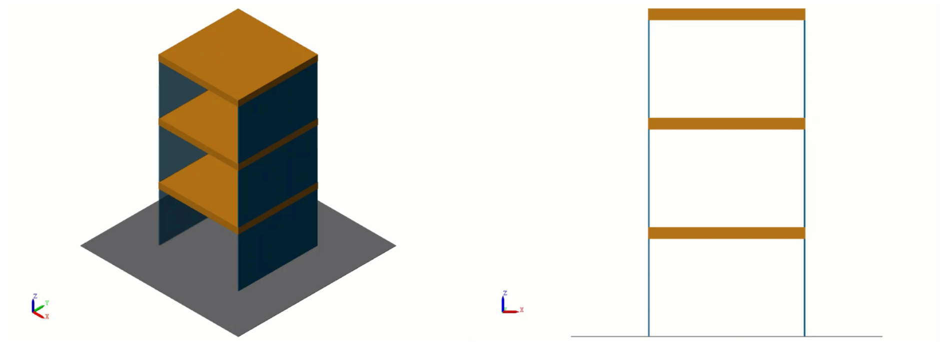
Preprocessing Earthquake Data

So far, you've seen how to perform a few basic data preprocessing tasks. Preprocessing is an important step to prepare data for visualization, analysis, and interpretation. Choosing the necessary preprocessing tasks depends on the characteristics of the data and your analysis goals. Some examples of preprocessing tasks are below.

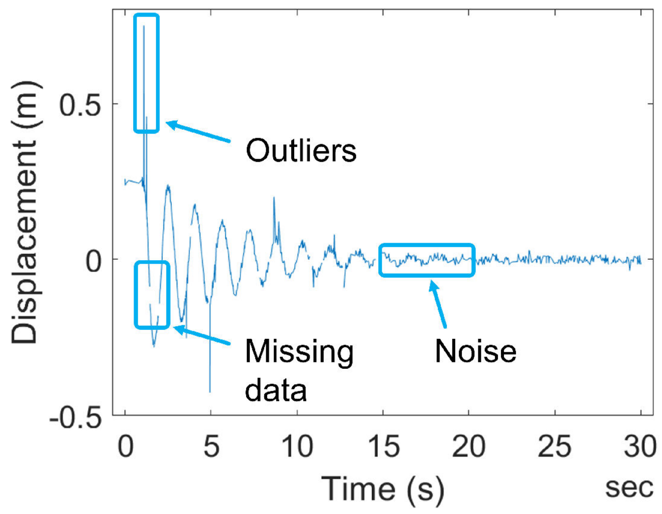
- **Dealing with outlier data** - A dataset may contain points that are statistically inconsistent with the desired signal
- **Dealing with missing data** - A dataset may contain missing points that are undefined (for example, NaN values)
- **Detrending** - A dataset may contain unwanted trends (for example, offset or linear slope)
- **Smoothing noisy data** - A dataset may contain noise in addition to the desired signal
- **Normalizing data** - A dataset may require rescaling to a new range

In this reading, you'll preprocess data measured at the top of a building during an earthquake. The data contains a time signal `t` (in seconds) and a displacement signal `x` (in meters) that show the building swaying back and forth during the earthquake. Run the code below to load and plot the measured data.

```
load earthquakeData.mat
plot(t,x)
xlabel("Time (s)")
ylabel("Displacement (m)")
```



You see that the data contains outliers, missing data, and noise that should be addressed before moving forward with analysis.



In this reading, you will use Live Tasks to perform the following preprocessing tasks:

1. Find all outlier points and fill them using linear interpolation.
2. Find all missing data and fill them using linear interpolation.
3. Smooth the data to remove noise.

After preprocessing the data, you'll determine the building's range of motion and the number of building oscillations recorded during the earthquake.

Preprocessing Task 1 - Clean outliers

Outliers are data points that are statistically inconsistent with the desired signal. Leaving these points in the dataset can lead to incorrect analysis results. Complete the steps below to clean outliers from the earthquake data.

- In the **Live Editor** tab, select **Task > Clean Outlier Data** Live Task.
- Set the **Input data** to `x`.
- Set the **X-axis** to `t`.
- Set the output variable to `x2`.

By default, the Live Task will find outlier data points and replace them using linear interpolation.

After cleaning the data, uncomment the code below to plot the `x2` signal.

```
% plot(t,x2)
% xlabel("Time (s)")
% ylabel("Displacement (m)")
```

Preprocessing Task 2 - Fill missing data

Datasets can contain missing data points for various reasons, such as data acquisition systems that log NaN values for invalid measurements. Missing data can cause unexpected analysis results and discontinuous data visualizations. Complete the steps below to fill all missing data points in the earthquake data.

- In the **Live Editor** tab, select **Task > Clean Missing Data**.
- Set the **Input data** to `x2`.
- Set the **X-axis** to `t`.
- Set the output variable to `x3`.

By default, the Live Task will find missing data points and fill them using linear interpolation.

```

```

After cleaning the data, uncomment the code below to plot the `x3` signal.

```
% plot(t,x3)
% xlabel("Time (s)")
% ylabel("Displacement (m)")
```

Preprocessing Task 3 - Smooth Noisy Data

Noise is common in signals and can be caused by many factors (for example, power supplies, vibrations, sensor errors, random sampling). Noisy data can obscure trends that you are trying to analyze and visualize. Complete the steps below to smooth the noisy earthquake data.

- In the **Live Editor** tab, select **Task > Smooth Data**.
- Set the **Input data** to `x3`.
- Set the **X-axis** to `t`.
- Change the **Smoothing factor** value to `0.025`. This decreases the level of smoothing performed.
- Set the output variable to `x4`.

By default, the Live Task applies a moving average filter to smooth the noisy data.

```

```

After cleaning the data, uncomment the code below to plot the `x4` signal.

```
% plot(t,x4)
% xlabel("Time (s)")
% ylabel("Displacement (m)")
```

Analysis

Now, the data is preprocessed and ready for analysis. There are many things to learn from this data, such as how much the building moved during the earthquake or how many times the building oscillated. First, use the `range` function on the `x4` vector to find the full range of motion (in meters) of the building during the earthquake.

Then, count the number of oscillations that the building encountered during the earthquake. This can be accomplished by finding all local maxima in the preprocessed earthquake data. Follow the steps below to do this.

- In the **Live Editor** tab, select **Task > Find Local Extrema**.
- Set the **Input data** to `x4`.
- Set the **X-axis** to `t`.
- Set the **Min. prominence** to 0.02. This specifies a minimum relative height to identify the maxima.

Based on the results of the analysis, scientists can determine if the building would withstand a stronger quake and decide if something needs to change in the building's structure.

Summary

In this reading, you started with raw earthquake data that was messy and difficult to analyze. To prepare the data for analysis, you completed three preprocessing tasks:

1. Find all outlier points and fill them using linear interpolation.
2. Find all missing data and fill them using linear interpolation.
3. Smooth the data to remove noise.

After completing these tasks, the earthquake data was much cleaner and suitable for analysis. Remember that different datasets may require different preprocessing tasks. Check the documentation for more information about these and other preprocessing tasks.