# carbon_model.rb



# Computation carbon model

This model is used by Brighter Planet's carbon emission web service to estimate the **greenhouse gas emissions of server use**.

**Timeframe and activity period**

The model estimates the emissions that occur during a particular `timeframe`. To do this it needs to know the `date` on which the computations occurred. For example, if the `timeframe` is January 2010, a computation that occurred on January 5, 2010 will have emissions but a computation that occurred on February 1, 2010 will not.

**Calculations**

The final estimate is the result of the **calculations** detailed below. These calculations are performed in reverse order, starting with the last calculation listed and finishing with the `emission` calculation. Each calculation is named according to the value it returns.

**Methods**

To accomodate varying client input, each calculation may have one or more **methods**. These are listed under each calculation in order from most to least preferred. Each method is named according to the values it requires. If any of these values is not available the method will be ignored. If all the methods for a calculation are ignored, the calculation will not return a value. "Default" methods do not require any values, and so a calculation with a default method will always return a value.

**Collaboration**

```
module BrighterPlanet
  module Computation
    module CarbonModel
      def self.included(base)
        base.decide :emission, :with => :characteristics do
```

Contributions to this carbon model are actively encouraged and warmly welcomed. This library includes a comprehensive test suite to ensure that your changes do not cause regressions. All changes should include test coverage for new functionality. Please see sniff, our emitter testing framework, for more information.

## Emission calculation

Returns the `emission` ($kg\ CO_2e$).

### Emission from $CO_2$ emission, $CH_4$ emission, and $N_2O$ emission

Adds `co2 emission` ($kg$), `ch4 emission` ($kg\ CO_2e$), and `n2o emission` ($kg\ CO_2e$) to give ($kg\ CO_2e$).

## $CO_2$ emission calculation

Returns the `co2 emission` ($kg$).

### $CO_2$ emission from electricity use, $CO_2$ emission factor, date, and timeframe

Checks whether the computation `date` falls within the `timeframe`.

Multiplies `electricity use` ($kWh$) by `co2 emission factor` ($kg / kWh$) to give $kg$.

If the `date` does not fall within the `timeframe`, `co2 emission` is zero.

```
          committee :emission do


            quorum 'from co2 emission, ch4 emission, and n2o emission',
:needs => [:co2_emission, :ch4_emission, :n2o_emission] do
|characteristics|

              characteristics[:co2_emission] +
characteristics[:ch4_emission] + characteristics[:n2o_emission]
            end
          end


          committee :co2_emission do


            quorum 'from electricity use, co2 emission factor, date, and
timeframe', :needs => [:electricity_use, :co2_emission_factor, :date] do
|characteristics, timeframe|

              if timeframe.include?
Date.parse(characteristics[:date].to_s)

                characteristics[:electricity_use] *
characteristics[:co2_emission_factor]
              else

                0
```

## $CO_2$ biogenic emission calculation

Returns the `co2 biogenic emission` ($kg$).

### $CO_2$ biogenic emission from electricity use, $CO_2$ biogenic emission factor, date, and timeframe

Checks whether the computation `date` falls within the `timeframe`.

Multiplies `electricity use` ($kWh$) by `co2 biogenic emission factor` ($kg / kWh$) to give $kg$.

If the `date` does not fall within the `timeframe`, `co2 biogenic emission` is zero.

## $CH_4$ emission calculation

Returns the `ch4 emission` ($kg\ CO_2e$).

### $CH_4$ emission from electricity use, $CH_4$ emission factor, date, and timeframe

Checks whether the computation `date` falls within the `timeframe`.

```
            end
          end
        end


        committee :co2_biogenic_emission do



          quorum 'from electricity use, co2 biogenic emission factor,
date, and timeframe', :needs => [:electricity_use,
:co2_biogenic_emission_factor, :date] do |characteristics, timeframe|

            if timeframe.include?
Date.parse(characteristics[:date].to_s)

              characteristics[:electricity_use] *
characteristics[:co2_biogenic_emission_factor]
            else

              0
            end
          end
        end


        committee :ch4_emission do



          quorum 'from electricity use, ch4 emission factor, date, and
timeframe', :needs => [:electricity_use, :ch4_emission_factor, :date] do
|characteristics, timeframe|

            if timeframe.include?
Date.parse(characteristics[:date].to_s)
```

Multiplies `electricity use` ($kWh$) by `ch4 emission factor` ($kg\ CO_2e\ /\ kWh$) to give $kg\ CO_2e$.

If the `date` does not fall within the `timeframe`, `ch4 emission` is zero.

## $N_2O$ emission calculation

Returns the `n2o emission` ($kg\ CO_2e$).

### $N_2O$ emission from electricity use, $N_2O$ emission factor, date, and timeframe

Checks whether the computation `date` falls within the `timeframe`.

Multiplies `electricity use` ($kWh$) by `n2o emission factor` ($kg\ CO_2e\ /\ kWh$) to give $kg\ CO_2e$.

If the `date` does not fall within the `timeframe`, `n2o emission` is zero.

## $CO_2$ emission factor calculation

Returns the `co2 emission factor` ($kg\ /\ kWh$).

### $CO_2$ emission factor from eGRID subregion

```
                characteristics[:electricity_use] *
characteristics[:ch4_emission_factor]
              else


                0
              end
            end
          end



          committee :n2o_emission do




            quorum 'from electricity use, n2o emission factor, date, and
timeframe', :needs => [:electricity_use, :n2o_emission_factor, :date] do
|characteristics, timeframe|


              if timeframe.include?
Date.parse(characteristics[:date].to_s)


                characteristics[:electricity_use] *
characteristics[:n2o_emission_factor]
              else


                0
              end
            end
          end



          committee :co2_emission_factor do




            quorum 'from eGRID subregion', :needs => :egrid_subregion do
|characteristics|
```

Looks up the eGRID subregion `co2 emission factor` ($kg / kWh$).

## $CO_2$ biogenic emission factor calculation

Returns the `co2 biogenic emission factor` ($kg / kWh$).

### $CO_2$ biogenic emission factor from eGRID subregion

Looks up the eGRID subregion `co2 biogenic emission factor` ($kg / kWh$).

## $CH_4$ emission factor calculation

Returns the `ch4 emission factor` ($kg\ CO_2e / kWh$).

### $CH_4$ emission factor from eGRID subregion

Looks up the eGRID subregion `ch4 emission factor` ($kg\ CO_2e / kWh$).

## $N_2O$ emission factor calculation

Returns the `n2o emission factor` ($kg\ CO_2e / kWh$).

```
characteristics[:egrid_subregion].electricity_co2_emission_factor
            end
          end


          committee :co2_biogenic_emission_factor do


            quorum 'from eGRID subregion', :needs => :egrid_subregion do
|characteristics|


characteristics[:egrid_subregion].electricity_co2_biogenic_emission_factor
            end
          end


          committee :ch4_emission_factor do


            quorum 'from eGRID subregion', :needs => :egrid_subregion do
|characteristics|


characteristics[:egrid_subregion].electricity_ch4_emission_factor
            end
          end


          committee :n2o_emission_factor do
```

### $N_2O$ emission factor from eGRID subregion

Looks up the eGRID subregion `n2o emission factor` ($kg\ CO_2e\ /\ kWh$).

```
            quorum 'from eGRID subregion', :needs => :egrid_subregion do
|characteristics|


characteristics[:egrid_subregion].electricity_n2o_emission_factor
            end
          end
```

## Electricity use calculation

Returns `electricity use` ($kWh$) including distribution losses.

### Electricity use from duration, electricity intensity, PUE, and electricity loss factor

Divides `duration` ($seconds$) by 3,600 $seconds\ /\ hour$, multiplies by `electricity intensity` ($kW$) and `PUE`, and divides by (1 − `electricity loss factor`) to give $kWh$.

```
          committee :electricity_use do



            quorum 'from duration, electricity intensity, PUE, and
electricity loss factor', :needs => [:duration, :electricity_intensity,
:power_usage_effectiveness, :electricity_loss_factor] do |characteristics|


              (characteristics[:duration] / 3600.0 *
characteristics[:electricity_intensity] *
characteristics[:power_usage_effectiveness]) / (1 -
characteristics[:electricity_loss_factor])
            end
          end
```

## Electricity loss factor calculation

Returns the `electricity loss factor`. This is the percentage of electricity lost during transmission and distribution.

### Electricity loss factor from eGRID region

Looks up the eGRID region `electricity loss factor`.

```
          committee :electricity_loss_factor do



            quorum 'from eGRID region', :needs => :egrid_region do
|characteristics|


              characteristics[:egrid_region].loss_factor
            end
          end
```

## eGRID region calculation

Returns the `eGRID region` where the data center is located.

### eGRID region from eGRID subregion

Looks up the eGRID subregion `eGRID region`.

## eGRID subregion calculation

Returns the `eGRID subregion` where the data center is located.

### eGRID subregion from zip code

Looks up the zip code `eGRID subregion`.

### eGRID subregion from carrier region

Looks up the carrier region `eGRID subregion`.

### Default eGRID subregion

Uses the fallback eGRID subregion, representing the U.S. average.

```
          committee :egrid_region do



            quorum 'from eGRID subregion', :needs => :egrid_subregion do
|characteristics|

              characteristics[:egrid_subregion].egrid_region
            end
          end



          committee :egrid_subregion do



            quorum 'from zip code', :needs => :zip_code do
|characteristics|

              characteristics[:zip_code].egrid_subregion
            end



            quorum 'from carrier region', :needs => :carrier_region do
|characteristics|

              characteristics[:carrier_region].egrid_subregion
            end



            quorum 'default' do

              EgridSubregion.fallback
            end
          end
```

## Zip code calculation

Returns the client-input `zip code` of the data center.

## Carrier region calculation

Returns the client-input carrier region of the data center.

## Power usage effectiveness calculation

Returns the `power usage effectiveness` (PUE). This is the ratio of total data center power to IT infrastructure power.

### Power usage effectivenss from client input

Uses the client-input `power usage effectiveness`.

### Power usage effectiveness from carrier

Looks up the carrier `power usage effectiveness`.

## Electricity intensity calculation

Returns the `electricity intensity` ($kW$). This is the average load of the data center IT infrastructure.

### Electricity intensity from client input

Uses the client-input `electricity intensity` ($kW$).

### Electricity intensity from carrier instance class

Looks up the carrier instance class `electricity intensity` ($kW$).

```
        committee :power_usage_effectiveness do
          quorum 'from carrier', :needs => :carrier do |characteristics|
            characteristics[:carrier].power_usage_effectiveness
          end
        end

        committee :electricity_intensity do
          quorum 'from carrier instance class', :needs =>
:carrier_instance_class do |characteristics|
```

## Carrier instance class calculation

Returns the computation `carrier instance class`. This is the type of virtual instance.

### Carrier instance class from client input

Uses the client-input carrier instance class.

### Default carrier instance class

Assumes Amazon m1.small.

## Carrier calculation

Returns the computation `carrier`. This is the company that runs the data center.

### Carrier from client input

Uses the client-input carrier.

### Default carrier

Assumes Amazon.

```
characteristics[:carrier_instance_class].electricity_intensity
            end
          end


          committee :carrier_instance_class do









            quorum 'default' do

              ComputationCarrierInstanceClass.fallback
            end
          end


          committee :carrier do









            quorum 'default' do

              ComputationCarrier.fallback
            end
          end
```

## Duration calculation

Returns the computation's `duration` (*seconds*).

### Duration from client input

Uses the client-input `duration`.

### Default duration

Assumes 3,600 *seconds*.

## Date calculation

Returns the `date` on which the computation occurred.

### Date from client input

Uses the client-input `date`.

### Date from timeframe

Assumes the first day of the `timeframe`.

## Timeframe calculation

Returns the `timeframe`. This is the period during which to calculate emissions.

### Timeframe from client input

```
committee :duration do

  quorum 'default' do

    base.fallback.duration
  end
end

committee :date do

  quorum 'from timeframe' do |characteristics, timeframe|

    timeframe.from
  end
end
```

**Complies:** All

Uses the client-input `timeframe`.

## Default timeframe

**Complies:** All

Uses the current calendar year.

```
        end
      end
    end
  end
end
```