carbon_model.rb

Meeting carbon model

This model is used by Brighter Planet's carbon emission web service to estimate the **greenhouse gas emissions of a meeting** (e.g. a conference).

Timeframe and date

The model estimates the emissions that occur during a particular `timeframe`. To do this it needs to know the meeting's `date`. For example, if the `timeframe` is January 2010, a meeting that occurred on January 11 2010 will have emissions but a meeting that occurred on Febraury 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.

Standard compliance

```ruby
require 'conversions'

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

Each method lists any established calculation standards with which it **complies**. When compliance with a standard is requested, all methods that do not comply with that standard are ignored. This means that any values a particular method requires will have been calculated using a compliant method, because those are the only methods available. If any value did not have a compliant method in its calculation then it would be undefined, and the current method would have been ignored.

Collaboration

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.

```
          committee :emission do
```

Emission calculation

Returns the `emission` estimate ($kg\ CO_2e$). This is the total emission produced by the meeting venue.

Emission from duration, area, and emission factor

```
            quorum 'from duration, area, and emission factor', :needs =>
[:duration, :area, :emission_factor],
              :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
                characteristics[:duration] / 3600.0 *
characteristics[:area] * characteristics[:emission_factor]
            end
```

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

Multiplies `area` ($square\ m$) by `duration` ($seconds$) and the `emission factor` ($kg\ CO_2e\ /\ square\ m\ hour$) to give $kg\ CO_2e$.

Default emission

```
            quorum 'default' do
              raise "The emission committee's default quorum should never
be called"
```

Displays an error if the previous method fails.

```
          end
        end
```

Emission factor calculation

Returns the `emission factor` (*lbs CO_2e / square m hour).

```
        committee :emission_factor do
```

Emission factor from fuel intensities and eGRID

```
          quorum 'from fuel intensities and eGRID', :needs =>
[:natural_gas_intensity, :fuel_oil_intensity, :electricity_intensity,
:district_heat_intensity, :egrid_subregion, :egrid_region],
```

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

```
            :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
```

Calculates an energy-based emission factor for natural gas by dividing its `co2 emission factor` (*kg / cubic m*) by its `energy content` (*MJ / cubic m*) to give *kg* CO_2 */ MJ*

```
              natural_gas = Fuel.find_by_name "Pipeline Natural Gas"
              natural_gas_energy_ef = natural_gas.co2_emission_factor /
natural_gas.energy_content
```

Calculates an energy-based emission factor for fuel oil by dividing its `co2 emission factor` (*kg / l*) by its `energy content` (*MJ / l*) to give *kg* CO_2 */ MJ*

```
              fuel_oil = Fuel.find_by_name "Distillate Fuel Oil No. 2"
              fuel_oil_energy_ef = fuel_oil.co2_emission_factor /
fuel_oil.energy_content
```

Calculates an energy-based emission factor for district heat by dividing the energy-based natural gas emission factor by 0.817 and the energy-based fuel oil emission factor by 0.846 (to account for boiler inefficiencies), averaging the two, and dividing by 0.95 (to account for transmission losses) to give *kg* CO_2 */ MJ*

```
              district_heat_ef = (((natural_gas_energy_ef / 0.817) +
(fuel_oil_energy_ef / 0.846)) / 2) / 0.95 # kg / MJ
```

Calculates an electricity emission factor by dividing the eGRID subregion electricity emission factor by 1 – the eGRID region loss factor (to account for transmission and distribution losses) to give *kg* CO_2 */ kWh*

```
              electricity_ef =
characteristics[:egrid_subregion].electricity_emission_factor / (1 -
characteristics[:egrid_region].loss_factor)
```

Multiplies `natural gas intensity` (*cubic m / room-night*) by the volume-

```
              (characteristics[:natural_gas_intensity] *
```

based natural gas emission factor (*kg* CO_2 / *room-night*), `fuel oil intensity` (*l* / *room-night*) by the volume-based fuel oil emission factor (*kg* CO_2 / *l*), `electricity intensity` (*kWh* / *room-night*) by the electricity emission factor (*kg* CO_2 / *kWh*), and `district heat intensity` (*MJ* / *room-night*) by the energy-based district heat emission factor (*kg* CO_2 / *MJ*), and adds these together to give *kg* CO_2 / *room-night*.

```
natural_gas.co2_emission_factor) +
                (characteristics[:fuel_oil_intensity] *
fuel_oil.co2_emission_factor) +
                (characteristics[:district_heat_intensity] *
district_heat_ef) +
                (characteristics[:electricity_intensity] * electricity_ef)
            end
          end
```

Natural gas intensity calculation

Returns the meeting venue's `natural gas intensity` (*cubic m* / *square m hour*).

```
          committee :natural_gas_intensity do # returns cubic metres per
square metre hour
```

From census division

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

Looks up the census division meeting building `natural gas intensity` (*cubic m* / *square m hour*).

```
            quorum 'from census division', :needs => :census_division,
              :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
characteristics[:census_division].meeting_building_natural_gas_intensity
            end
```

Default natural gas intensity

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

Uses the U.S. average `natural gas intensity` (*cubic m* / *square m hour*).

```
            quorum 'default',
              :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
CensusDivision.fallback.meeting_building_natural_gas_intensity
            end
          end
```

Fuel oil intensity calculation

Returns the meeting venue's `fuel oil intensity` (*l* / *square m hour*).

```
          committee :fuel_oil_intensity do
```

Fuel oil intensity from census division

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

Looks up the census division meeting building `fuel oil intensity` (*l / square m hour*).

```
          quorum 'from census division', :needs => :census_division,
            :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
characteristics[:census_division].meeting_building_fuel_oil_intensity
          end
```

Default fuel oil intensity

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

Uses the U.S. average `fuel oil intensity` (*l / square m hour*).

```
          quorum 'default',
            :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
CensusDivision.fallback.meeting_building_fuel_oil_intensity
          end
        end
```

Electricity intensity calculation

Returns the meeting venue's `electricity intensity` (*kWh / square m hour*).

```
        committee :electricity_intensity do
```

Electricity intensity from census division and eGRID region

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

- Looks up the census division meeting building `electricity intensity` (*kWh / square m hour*)
- Looks up the eGRID region loss factor

```
          quorum 'from eGRID region and census division', :needs =>
[:egrid_region, :census_division],
            :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
characteristics[:census_division].meeting_building_electricity_intensity /
(1 - characteristics[:egrid_region].loss_factor)
          end
```

- Divides the `electricity intensity` by 1 – the loss factor to account for electricity transmission and distribution losses

Electricity intensity from eGRID region

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

- Uses the U.S. average meeting building `electricity intensity` (*kWh / square m hour*)
- Looks up the eGRID region loss factor
- Divides the `electricity intensity` by (1 – the loss factor) to account for electricity transmission and distribution losses

```
          quorum 'from eGRID region', :needs => :egrid_region,
            :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
CensusDivision.fallback.meeting_building_electricity_intensity / (1 -
characteristics[:egrid_region].loss_factor)
          end
        end
```

District heat intensity calculation

Returns the meeting venue's `district heat intensity` (*MJ / square m hour*)

```
        committee :district_heat_intensity do
```

District heat intensity from census division

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

Looks up the census division meeting building `district heat intensity`.

```
          quorum 'from census division', :needs => :census_division,
            :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
characteristics[:census_division].meeting_building_district_heat_intensity
          end
```

Default district heat intensity

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

Uses the U.S. average.

```
          quorum 'default',
            :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
```

```
CensusDivision.fallback.meeting_building_district_heat_intensity
            end
          end
```

eGRID region calculation

Returns the meeting venue’s eGRID region.

```
          committee :egrid_region do
```

eGRID region from eGRID subregion

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

Looks up the eGRID subregion `eGRID region`.

```
            quorum 'from eGRID subregion', :needs => :egrid_subregion,
              :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
              characteristics[:egrid_subregion].egrid_region
            end
          end
```

eGRID subregion calculation

Returns the meeting venue’s eGRID subregion.

```
          committee :egrid_subregion do
```

eGRID subregion from zip code

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

Looks up the zip code `eGRID subregion`.

```
            quorum 'from zip code', :needs => :zip_code,
              :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
                characteristics[:zip_code].egrid_subregion
            end
```

Default eGRID subregion

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

```
            quorum 'default',
              :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
```

Uses an artificial eGRID subregion that represents the U.S. average.

```
          EgridSubregion.find_by_abbreviation 'US'
        end
      end
```

Census division calculation

Returns the meeting venue's census division.

```
      committee :census_division do
```

Census division from state

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

```
        quorum 'from state', :needs => :state,
          :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
```

Looks up the state `census division`.

```
          characteristics[:state].census_division
        end
      end
```

State calculation

Returns the meeting venue's state.

```
      committee :state do
```

State from zip code

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

```
        quorum 'from zip code', :needs => :zip_code,
          :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
|characteristics|
```

Looks up the zip code `state`.

```
          characteristics[:zip_code].state
        end
      end
```

Zip code calculation

Returns the meeting venue's zip code.

Zip code from client input

Complies: All

Uses the client-input zip code.

Area calculation

Returns the meeting venue's `area` (*square m*).

```
committee :area do
```

Area from client input

Complies: All

Uses the client-input `area` (*square m*).

Default area

Complies: GHG Protocol Scope 3, ISO 14064-1, Climate Registry Protocol

Uses a default `area` of 1,184.5 *square m*. This is the average size of meeting buildings in the EIA Commercial Building Energy Consumption Survey.

```
  quorum 'default',
    :complies => [:ghg_protocol_scope_3, :iso, :tcr] do
      10_448.square_feet.to(:square_metres)
  end
end
```

Duration calculation

Returns the meeting's `duration` (seconds). This is the number of seconds the meeting venue is in use. For example, a two-day conference that runs 8 hours each day would have a duration of 57600.

```
committee :duration do
```

Duration from client input

Complies: All

Uses the client-input `duration` (*seconds*).

Default duration

Uses a default `duration` of 28800 *seconds* (8 hours).

```
              quorum 'default' do


                28800.0
              end
            end
          end
        end
      end
    end
  end
```