carbon_model.rb

Automobile trip carbon model

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

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 trip occurred. For example, if the `timeframe` is January 2010, a trip that occurred on January 5, 2010 will have emissions but a trip 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.

Standard compliance

Each method lists any established calculation standards with which it **complies**. When compliance with a standard is requested, all methods that

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

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.

Emission calculation

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

Emission from CO_2 emission, CH_4 emission, N_2O emission, and HFC emission

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

Sums the non-biogenic emissions to give $kg\ CO_2e$.

```
          committee :emission do


            quorum 'from co2 emission, ch4 emission, n2o emission, and hfc emission',
              :needs => [:co2_emission, :ch4_emission, :n2o_emission, :hfc_emission],


              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

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

CO_2 emission calculation

Returns the `co2 emission` (kg).

CO_2 emission from fuel use, CO_2 emission factor, date, and timeframe

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

```
          committee :co2_emission do


            quorum 'from fuel use, co2 emission factor, date, and timeframe',
              :needs => [:fuel_use, :co2_emission_factor, :date],


              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics, timeframe|
```

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

Multiplies `fuel use` (l) by the `co2 emission factor` (kg / l) to give kg.

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

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

                characteristics[:fuel_use] * characteristics[:co2_emission_factor]
              else

                0
              end
            end
          end
```

CO_2 biogenic emission calculation

Returns the `co2 biogenic emission` (kg).

CO_2 biogenic emission from fuel use, CO_2 biogenic emission factor, date, and timeframe

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

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

Multiplies `fuel use` (l) by the `co2 biogenic emission factor` (kg / l) to give kg.

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

```
        committee :co2_biogenic_emission do


          quorum 'from fuel use, co2 biogenic emission factor, date, and timeframe',
            :needs => [:fuel_use, :co2_biogenic_emission_factor, :date],


            :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics, timeframe|

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

                characteristics[:fuel_use] *
characteristics[:co2_biogenic_emission_factor]
              else

                0
              end
            end
          end


        committee :ch4_emission do
```

CH_4 emission calculation

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

CH_4 emission from fuel use, CH_4 emission factor, date, and timeframe

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

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

Multiplies `fuel use` (l) by the `ch4 emission factor` ($kg\ CO_2e\ /\ l$) to give $kg\ CO_2e$.

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

```
          quorum 'from fuel use, ch4 emission factor, date, and timeframe',
            :needs => [:fuel_use, :ch4_emission_factor, :date],

            :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics, timeframe|

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

                characteristics[:fuel_use] * characteristics[:ch4_emission_factor]
              else

                0
              end
            end
          end
```

N_2O emission calculation

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

N_2O emission from fuel use, N_2O emission factor, date, and timeframe

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

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

Multiplies `fuel use` (l) by the `n2o emission factor` ($kg\ CO_2e\ /\ l$) to give $kg\ CO_2e$.

```
          committee :n2o_emission do

          quorum 'from fuel use, n2o emission factor, date, and timeframe',
            :needs => [:fuel_use, :n2o_emission_factor, :date],

            :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics, timeframe|

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

                characteristics[:fuel_use] * characteristics[:n2o_emission_factor]
              else
```

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

HFC emission calculation

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

HFC emission from fuel use, HFC emission factor, date, and timeframe

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

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

Multiplies `fuel use` (l) by the `hfc emission factor` ($kg\ CO_2e\ /\ l$) to give $kg\ CO_2e$.

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

CO_2 emission factor calculation

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

CO_2 emission factor from automobile fuel

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

```
              0
            end
          end
        end


        committee :hfc_emission do


          quorum 'from fuel use, hfc emission factor, date, and timeframe',
            :needs => [:fuel_use, :hfc_emission_factor, :date],


            :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics, timeframe|


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


                characteristics[:fuel_use] * characteristics[:hfc_emission_factor]
              else


              0
            end
          end
        end


        committee :co2_emission_factor do


          quorum 'from automobile fuel',
            :needs => :automobile_fuel,


            :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|
```

Looks up the fuel `co2 emission factor` (kg / l).

CO_2 biogenic emission factor calculation

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

CO_2 biogenic emission factor from automobile fuel

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

Looks up the fuel `co2 biogenic emission factor` (kg / l).

CH_4 emission factor calculation

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

CH_4 emission factor from automobile fuel

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

Looks up the fuel `ch4 emission factor` ($kg\ CO2e / l$).

N_2O emission factor calculation

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

```
            characteristics[:automobile_fuel].co2_emission_factor
          end
        end


        committee :co2_biogenic_emission_factor do


          quorum 'from automobile fuel',
            :needs => :automobile_fuel,

            :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

            characteristics[:automobile_fuel].co2_biogenic_emission_factor
          end
        end


        committee :ch4_emission_factor do


          quorum 'from automobile fuel',
            :needs => :automobile_fuel,

            :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

            characteristics[:automobile_fuel].ch4_emission_factor
          end
        end


        committee :n2o_emission_factor do
```

N_2O emission factor from automobile fuel

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

Looks up the fuel `n2o emission factor` ($kg\ CO2e\ /\ l$).

```
            quorum 'from automobile fuel',
              :needs => :automobile_fuel,

              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

                characteristics[:automobile_fuel].n2o_emission_factor
              end
            end
```

HFC emission factor calculation

Returns the `hfc emission factor` ($kg\ CO_2e\ /\ l$).

HFC emission factor from automobile fuel

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

Looks up the fuel `hfc emission factor` ($kg\ CO2e\ /\ l$).

```
            quorum 'from automobile fuel',
              :needs => :automobile_fuel,

              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

                characteristics[:automobile_fuel].hfc_emission_factor
              end
            end
```

Automobile fuel calculation

Returns the type of `automobile fuel` used.

Automobile fuel from client input

Complies: All

Uses the client-input automobile fuel.

Automobile fuel from make model year variant

```
            quorum 'from make model year variant',
              :needs => :make_model_year_variant,
```

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

Looks up the variant `automobile fuel`.

Default automobile fuel

Complies: GHG Protocol Scope 3, ISO 14064-1

Looks up the default automobile fuel.

Fuel use calculation

Returns the trip `fuel use` (*l*).

Fuel use from fuel efficiency and distance

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

Divides the `distance` (*km*) by the `fuel efficiency` (*km / l*) to give *l*.

Distance calculation

Returns the trip `distance` (*km*).

Distance from client input

Complies: All

Uses the client-input `distance` (*km*).

```
              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

                characteristics[:make_model_year_variant].fuel
            end


            quorum 'default',

              :complies => [:ghg_protocol_scope_3, :iso] do

                AutomobileFuel.fallback
            end
          end


          committee :fuel_use do


            quorum 'from fuel efficiency and distance',
              :needs => [:fuel_efficiency, :distance],

              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

                characteristics[:distance] / characteristics[:fuel_efficiency]
            end
          end


          committee :distance do
```

Distance from origin and destination locations

Complies: GHG Protocol Scope 3, ISO 14064-1

Uses the Mapquest directions API to calculate distance by road between the origin and destination locations.

```
            quorum 'from origin and destination locations',
              :needs => [:origin_location, :destination_location],

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

                mapquest = ::MapQuestDirections.new characteristics[:origin_location],
characteristics[:destination_location]
                begin
                  mapquest.distance_in_kilometres
                rescue
                  nil
                end
              end
```

Distance from duration and speed

Complies: GHG Protocol Scope 3, ISO 14064-1

Divides the `duration` (*seconds*) by 3600 and multiplies by the `speed` (*km / hour*) to give *km*.

```
            quorum 'from duration and speed',
              :needs => [:duration, :speed],

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

                (characteristics[:duration] / 60.0 / 60.0) * characteristics[:speed]
              end
```

Distance from country

Complies: GHG Protocol Scope 3, ISO 14064-1

Looks up the country `automobile trip distance`.

```
            quorum 'from country',
              :needs => :country,

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

                characteristics[:country].automobile_trip_distance
              end
            end
```

Destination location calculation

Returns the `destination location` (*lat / lng*).

```
          committee :destination_location do
```

Destination location from destination

```
            quorum 'from destination',
```

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

Uses the Geokit geocoder to determine the destination location (*lat / lng*).

Origin location committee

Returns the `origin location` (*lat / lng*).

Destination location from destination

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

Uses the Geokit geocoder to determine the origin location (*lat / lng*).

Destination calculation

Returns the client-input `destination`.

Origin calculation

Returns the client-input `origin`.

Duration calculation

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

```
              :needs => :destination,

              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

                code = ::Geokit::Geocoders::MultiGeocoder.geocode
characteristics[:destination].to_s
                code.ll == ',' ? nil : code.ll
              end
            end



            committee :origin_location do



              quorum 'from origin',
                :needs => :origin,

              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

                code = ::Geokit::Geocoders::MultiGeocoder.geocode
characteristics[:origin].to_s
                code.ll == ',' ? nil : code.ll
              end
            end
```

Speed calculation

Returns the average `speed` at which the automobile travels ($km/hour$).

Speed from client input

Complies: All

Uses the client-input `speed` ($km/hour$).

Speed from urbanity and country

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

Looks up the country average city and highway driving speeds and calculates the harmonic mean of those speeds weighted by `urbanity`.

```
          committee :speed do

            quorum 'from urbanity and country',
              :needs => [:urbanity, :country],

              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

                1 / (characteristics[:urbanity] /
characteristics[:country].automobile_city_speed + (1 - characteristics[:urbanity]) /
characteristics[:country].automobile_highway_speed)
            end
          end
```

Fuel efficiency calculation

Returns the `fuel efficiency` (km/l).

Fuel efficiency from client input

Complies: All

Uses the client-input `fuel efficiency` (km/l).

Fuel efficiency from make model year variant and urbanity

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

Looks up the city and highway fuel efficiencies of the automobile make

```
          committee :fuel_efficiency do

            quorum 'from make model year variant and urbanity',
              :needs => [:make_model_year_variant, :urbanity],

              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

                fuel_efficiency_city =
```

model year variant (km/l).

```
characteristics[:make_model_year_variant].fuel_efficiency_city
                fuel_efficiency_highway =
characteristics[:make_model_year_variant].fuel_efficiency_highway
                urbanity = characteristics[:urbanity]
                if fuel_efficiency_city.present? and fuel_efficiency_highway.present?
```

Calculates the harmonic mean of those fuel efficiencies, weighted by `urbanity`.

```
                  1.0 / ((urbanity / fuel_efficiency_city) + ((1.0 - urbanity) /
fuel_efficiency_highway))
                end
            end
```

Fuel efficiency from make model year and urbanity

```
            quorum 'from make model year and urbanity',
              :needs => [:make_model_year, :urbanity],
```

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

```
              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|
```

Looks up the city and highway fuel efficiencies of the automobile make model year (km/l).

```
                fuel_efficiency_city =
characteristics[:make_model_year].fuel_efficiency_city
                fuel_efficiency_highway =
characteristics[:make_model_year].fuel_efficiency_highway
                urbanity = characteristics[:urbanity]
                if fuel_efficiency_city.present? and fuel_efficiency_highway.present?
```

Calculates the harmonic mean of those fuel efficiencies, weighted by `urbanity`.

```
                  1.0 / ((urbanity / fuel_efficiency_city) + ((1.0 - urbanity) /
fuel_efficiency_highway))
                end
            end
```

Fuel efficiency from make model and urbanity

```
            quorum 'from make model and urbanity',
              :needs => [:make_model, :urbanity],
```

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

```
              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|
```

Looks up the city and highway fuel efficiencies of the automobile make model (km/l).

```
                fuel_efficiency_city =
characteristics[:make_model].fuel_efficiency_city
                fuel_efficiency_highway =
characteristics[:make_model].fuel_efficiency_highway
                urbanity = characteristics[:urbanity]
```

Calculates the harmonic mean of those fuel efficiencies, weighted by `urbanity`.

```
                if fuel_efficiency_city.present? and fuel_efficiency_highway.present?

                  1.0 / ((urbanity / fuel_efficiency_city) + ((1.0 - urbanity) /
fuel_efficiency_highway))
                end
            end
```

Fuel efficiency from size class, hybridity multiplier, and urbanity

Complies: GHG Protocol Scope 3, ISO 14064-1

Looks up the automobile size class city and highway fuel efficiency (km / l).

Calculates the harmonic mean of those fuel efficiencies, weighted by `urbanity`, and multiplies the result by the `hybridity multiplier`.

```
            quorum 'from size class, hybridity multiplier, and urbanity',
              :needs => [:size_class, :hybridity_multiplier, :urbanity],

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

                fuel_efficiency_city =
characteristics[:size_class].fuel_efficiency_city
                fuel_efficiency_highway =
characteristics[:size_class].fuel_efficiency_highway
                urbanity = characteristics[:urbanity]
                if fuel_efficiency_city.present? and fuel_efficiency_highway.present?

                  (1.0 / ((urbanity / fuel_efficiency_city) + ((1.0 - urbanity) /
fuel_efficiency_highway))) * characteristics[:hybridity_multiplier]
                end
            end
```

Fuel efficiency from make year and hybridity multiplier

Complies: GHG Protocol Scope 3, ISO 14064-1

Looks up the automobile make year combined fuel efficiency (km / l) and multiplies it by the `hybridity multiplier`.

```
            quorum 'from make year and hybridity multiplier',
              :needs => [:make_year, :hybridity_multiplier],

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

                characteristics[:make_year].fuel_efficiency.try :*,
characteristics[:hybridity_multiplier]
            end
```

Fuel efficiency from make and hybridity multiplier

Complies: GHG Protocol Scope 3, ISO 14064-1

```
            quorum 'from make and hybridity multiplier',
              :needs => [:make, :hybridity_multiplier],

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

Looks up the automobile make combined fuel efficiency (km/l) and multiplies it by the `hybridity multiplier`.

Fuel efficiency from hybridity multiplier and country

Complies: GHG Protocol Scope 3, ISO 14064-1

Looks up the country `automobile fuel efficiency` and multiplies it by the `hybridity multiplier`.

Hybridity multiplier calculation

Returns the `hybridity multiplier`. This value may be used to adjust the fuel efficiency based on whether the automobile is a hybrid or conventional vehicle.

Hybridity multiplier from size class, hybridity, and urbanity

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

Looks up the appropriate city and highway hybridity multipliers for the automobile size class.

Calculates the harmonic mean of those multipliers, weighted by `urbanity`.

```
              characteristics[:make].fuel_efficiency.try :*,
characteristics[:hybridity_multiplier]
            end



            quorum 'from hybridity multiplier and country',
              :needs => [:hybridity_multiplier, :country],

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

              characteristics[:country].automobile_fuel_efficiency.try :*,
characteristics[:hybridity_multiplier]
            end
          end



          committee :hybridity_multiplier do









            quorum 'from size class, hybridity, and urbanity',
              :needs => [:size_class, :hybridity, :urbanity],

              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

              drivetrain = characteristics[:hybridity] ? :hybrid : :conventional
              city_fuel_efficiency_multiplier =
characteristics[:size_class].send(:"#{drivetrain}_fuel_efficiency_city_multiplier")
              highway_fuel_efficiency_multiplier =
characteristics[:size_class].send(:"#{drivetrain}_fuel_efficiency_highway_multiplier")

              if city_fuel_efficiency_multiplier or
highway_fuel_efficiency_multiplier

                1.0 / ((characteristics[:urbanity] /
city_fuel_efficiency_multiplier) + ((1.0 - characteristics[:urbanity]) /
highway_fuel_efficiency_multiplier))
```

Hybridity multiplier from hybridity and urbanity

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

Looks up the appropriate default city and highway hybridity multipliers.

Calculates the harmonic mean of those multipliers, weighted by `urbanity`.

```
            end
          end


          quorum 'from hybridity and urbanity',
            :needs => [:hybridity, :urbanity],

            :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
|characteristics|

            drivetrain = characteristics[:hybridity] ? :hybrid : :conventional
            city_fuel_efficiency_multiplier =
AutomobileSizeClass.fallback.send(:"#{drivetrain}_fuel_efficiency_city_multiplier")
            highway_fuel_efficiency_multiplier =
AutomobileSizeClass.fallback.send(:"#{drivetrain}_fuel_efficiency_highway_multiplier")

            1.0 / ((characteristics[:urbanity] / city_fuel_efficiency_multiplier)
+ ((1.0 - characteristics[:urbanity]) / highway_fuel_efficiency_multiplier))
          end
```

Default hybridity multiplier

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

Uses a default `hybridity multiplier` of 1.

```
          quorum 'default',

            :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do

            base.fallback.hybridity_multiplier
          end
        end


        committee :urbanity do
```

Urbanity calculation

Returns the `urbanity`. This is the fraction of the total distance driven that occurs on towns and city streets as opposed to highways (defined using a 45 miles per hour “speed cutpoint”).

Urbanity from country

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

```
          quorum 'from country',
            :needs => :country,

            :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
```

Looks up the country `automobile urbanity`.

```
|characteristics|

                characteristics[:country].automobile_urbanity
              end
            end
```

Hybridity calculation

Returns the client-input `hybridity`. This indicates whether the automobile is a hybrid electric vehicle or a conventional vehicle.

Size class calculation

Returns the client-input automobile size class.

Make model year variant calculation

Returns the client-input automobile make model year variant.

Make model year calculation

Returns the client-input automobile make model year.

Make model calculation

Returns the client-input automobile make model.

Make year calculation

Returns the client-input automobile make year.

Make calculation

Returns the client-input automobile make.

Country calculation

Returns the `country` in which the trip occurred.

```
            committee :country do
```

Country of [illegible]

Country from client input

Complies: All

Uses the client-input `country`.

Default country

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

Uses an artificial country that contains global averages.

Date calculation

Returns the `date` on which the trip occurred.

Date from client input

Complies: All

Uses the client-input `date`.

Date from timeframe

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

Assumes the trip occurred on the first day of the `timeframe`.

Timeframe calculation

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

```
            quorum 'default',
              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso] do
                Country.fallback
            end
          end

          committee :date do
            quorum 'from timeframe',
              :complies => [:ghg_protocol_scope_1, :ghg_protocol_scope_3, :iso, :tcr]
do |characteristics, timeframe|
                timeframe.from
            end
          end
```

Timeframe from client input

Complies: All

Uses the client-input `timeframe`.

Default timeframe

Complies: All

Uses the current calendar year.

```
        end
      end
    end
  end
end
```