Quantitative Comparison Between Two Photosynthesis Models

BioCro Development Team

October 15, 2022

```
This document was generated from the version of BioCro specified as follows:

Commit Hash: 525117c
Date: Fri, 14 Oct 2022 22:30:38 -0500
Branch: change_pkgdown_theme
```

Contents

1 Overview

This vignette is a guided walkthrough of the code used to produce the figures in Section 3 of Lochocki *et al.* (2022) [doi: 10.1093/insilicoplants/diac003] Briefly, this section of the paper compares two different models for C_3 photosynthesis: the mechanistic Farquhar-von-Cammerer-Berry (FvCB) model and an empirical radiation use efficiency (RUE) model. Each of these models is available as a BioCro module and can be used as a component in a larger crop model. The FvCB version in particular plays a key role in BioCro's validated soybean model, and it can be considered as the more accurate model for C_3 photosynthesis. On the other hand, the RUE model has several known shortcomings, such as being insensitive to atmospheric [CO_2] levels. The main idea of Section 3 is to compare the behavior of these two models, both on their own and in the context of a larger soybean growth simulation. The analysis proceeds in the following steps:

1. Find a value of the radiation use efficiency `alpha_rue` that optimizes the agreement between total biomass predicted for soybeans in 2002 using the RUE and FvCB models.

2. Compare the sensitivity of each model to several different environmental variables.

3. Determine the level of disagreement between the two models when simulating soybean growth in other years.

For more information about this analysis, including the motivation for each step and a discussion of the results, please see the BioCro II paper. Also note that as time progresses and the BioCro models are improved, the figures produced in this vignette may slightly diverge from those of the BioCro II paper.

The general strategies employed here can be applied to other modules in order to quantitatively compare the behavior of alternate versions of a model component, or even just to illustrate the behavior of a model. This type of analysis is indispensible for understanding and explaining the properties of models, and it is hoped that this vignette can serve as a template for other analyses in the future.

2 Retrieving weather data and creating figures

For brevity, this R vignette does not display the code used for processing weather data, retrieving weather data, or creating figures, although this code is contained in the associated R script. This is to place emphasis on the BioCro functionality related to analyzing modules, since this part of the code will be of more general use. For other analyses, the required weather data will likely be different, and each user may have their own preferences regarding plotting that differ from the strategies used here.

In this section, we provide a short overview of the hidden code that is required to complete the analysis and produce the figures, but isn't explicitly included in this vignette.

This code creates two named lists to help with retrieving processed weather data:

- `catm`: A list of the atmospheric [CO_2] levels for 1980 - 2020. Essentially, this list contains globally averaged annual atmospheric [CO_2] levels from `catm_data`, a dataset obtained from NOAA and included as part of the BioCro package.

- `soy_weather`: A list of weather data where each named element corresponds to the growing season of one particular year. Essentially, this list contains growing seasons of weather data from Champaign, Illinois that are appropriate to use for the soybean simulations in this analysis. Full years of weather data are included in the BioCro package as the `weather` dataset. To create `soy_weather`, photoperiod length as determined from the soybean circadian clock model is added to the original weather data, and then the full yearly set is truncated to days 152–288.

Most plots in this vignette are created using the `xyplot` function from the `lattice` package, and PDF versions of the plots are saved using another helping function: `pdf_print`. To produce the figures used in the paper, these raw figures were finalized using Adobe Illustrator to modify colors, combine panels, format text, and perform other operations designed to improve readability.

3 Determining the optimal value for `alpha_rue` in 2002

We want to find the value of `alpha_rue` that creates output most similar to the FvCB model. To do this, we will use an optimizer to find the value of `alpha_rue` that minimizes the squared difference in end-of-season total mass between the two models for one year. Our example problem can be considered a simple case of least-squares fitting, where the number of model parameters being considered is one.

The first step is to run the FvCB soybean model for the year 2002. Afterwards, we can extract the final biomass for 2002 as predicted by the FvCB model. Since we will be extracting the final biomass from the results of multiple simulations during this analysis, we will create a reusable function for doing this. Note that we also define a soybean model specialized for this analysis that includes an additional module (`BioCro:total_biomass`) that is not part of the standard soybean model; as its name suggests, this module simply calculates the total biomass by adding the separate tissues together, and including it in the model simplifies some of the later calculations.

```
cmi_soybean <- within(soybean, {
  direct_modules = append(direct_modules, 'BioCro:total_biomass')
})

fvcb_result_2002 <- with(cmi_soybean, {run_biocro(
  initial_values,
  parameters,
  soy_weather[['2002']],
  direct_modules,
  differential_modules,
  ode_solver
)})

final_biomass <- function(df) {
  df[nrow(df), 'total_biomass']
}

final_biomass_fvcb_2002 <- final_biomass(fvcb_result_2002)
```

During the optimization, we will need to run the RUE soybean model many times with different values of `alpha_rue`, keeping the values of all other parameters and drivers constant. In this situation, the technique of *partial application* can be very useful; partial application refers to a process where some arguments to a function are fixed, creating a function with fewer arguments. In BioCro, a convenience function for using partial application with `run_biocro` is provided: the `partial_run_biocro` function. Here we use it to fix all inputs to `run_biocro` except `alpha_rue`. Note that we also replace the default canopy photosynthesis module. The return value from `partial_run_biocro` is a function that we have called `rue_2002`; this function takes a value of `alpha_rue` as an input, runs a simulation using that value and the other arguments specified in the original call to `partial_run_biocro`, and returns the resulting data frame.

```
# The first six arguments are the same as for `run_biocro`
rue_2002 <- with(cmi_soybean, {partial_run_biocro(
  initial_values,
  within(parameters, {alpha_rue = NA}),
  soy_weather[['2002']],
  within(direct_modules, {canopy_photosynthesis = 'BioCro:ten_layer_rue_canopy'}),
  differential_modules,
  ode_solver,
  'alpha_rue'  # here we specify the names of any quantities whose values
)})            # should not be fixed
```

The optimizer we will use requires a function that accepts one argument (the value of the parameter being optimized) and returns one value (the value to minimize). Here, we create a function that accepts a value of `alpha_rue`, passes it to `rue_2002` to run a soybean simulation, extracts the end-of-season biomass, and calculates the squared difference between that result and the FvCB result. This function provides a means for associating an error metric with each value of `alpha_rue`, and it is suitable for passing to the optimizer.

```
rue_fvcb_square_difference = function(alpha_rue) {
    (final_biomass(rue_2002(alpha_rue)) - final_biomass_fvcb_2002)^2
}
```

Here we use the `optim` function, which is included with every R installation; it accepts `f(x)` and finds the value of `x` that minimizes `f(x)`. It requires a starting value for `x`, and for the `Brent` method, upper and lower bounds of `x`. It returns a list with information about the optimization (including the parameters it

found, which are included in the `par` list element). Here we pass the `rue_fvcb_square_difference` function to `optim` in order to find the best value for `alpha_rue`.

```
opt_par = optim(
    0.035,
    rue_fvcb_square_difference,
    method='Brent',
    lower=0.03,
    upper=0.04
)

best_alpha_rue = opt_par$par
```

The optimal `alpha_rue` value can be compared to the square difference curve to ensure that a minimum was truly obtained (Figure 1; this is Figure S1 in the BioCro II paper).

Here we used `partial_run_biocro` to specify one parameter for optimization (`alpha_rue`), used the output from `partial_run_biocro` to define a function that associates a measure of error with a value of `alpha_rue` (`rue_fvcb_square_difference`), and passed that function to an optimizer (`optim`).

This is a general strategy for parameter optimization in BioCro, allowing the user to choose which parameters should be optimized, define an error metric, and choose an optimization algorithm. In this example the error function is simply based on end-of-season biomass, but more detailed error metrics that compare the simulation output to experimentally measured data can easily be created. For the optimization, we use the `optim` function, but this design allows any optimizer available in R to be used. For example, one could easily adapt this to search for parameters using simulated annealing or evolutionary algorithms. This is in contrast to some dynamical system software programs, in particular GUI-driven programs, which offer a limited and nonextendable number of choices for these algorithms.

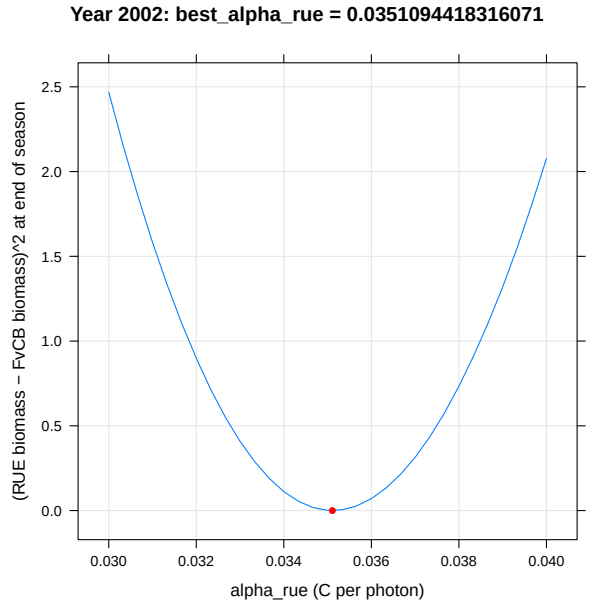

Figure 1: Squared end-of-season biomass difference plotted against `alpha_rue` (blue line); the difference at the optimal value of `alpha_rue` is shown as a red circle.

4 Comparing soybean biomass in 2002

Having found a value of `alpha_rue` that minimizes the end-of-year biomass difference between the two models, we can now run the RUE model with this optimized value and check to see how well the models agree at other points during the growing season (Figure 2; this is Figure 4a in the BioCro II paper).

```
optimal_rue_result_2002 <- with(cmi_soybean, {run_biocro(
  initial_values,
  within(parameters, {alpha_rue = best_alpha_rue}),
  soy_weather[['2002']],
  within(direct_modules, {canopy_photosynthesis = 'BioCro:ten_layer_rue_canopy'}),
  differential_modules,
  ode_solver
)})
```

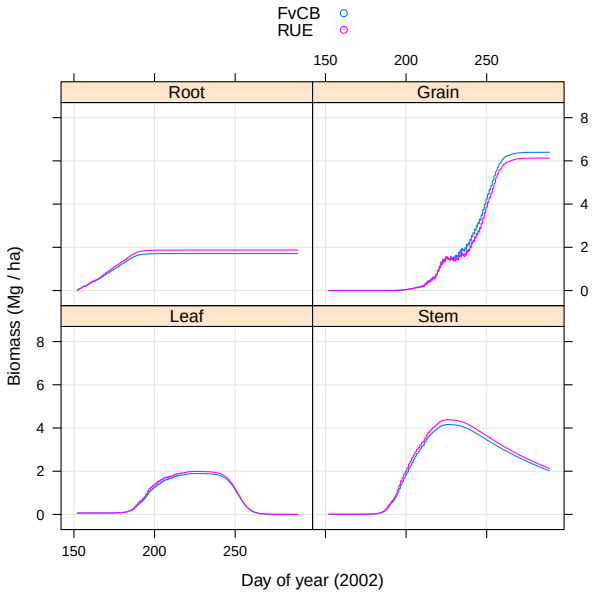

Figure 2: Soybean root, grain, leaf, and stem biomass calculated during 2002 using the RUE and FvCB photosynthesis models

5 Comparing assimilation values in 2002

Additional insight into the agreement between the models can be gained by examining the relationship between gross CO_2 assimilation rates (A_g) and incident light intensity (Q) at the leaf level throughout the entire growing season. The BioCro multilayer canopy photosynthesis modules store these values for each layer (0-9) and leaf class (sunlit and shaded) in quantities with names that specify layer number as a suffix and class as a prefix added to the base names of `incident_ppfd` (Q) and `GrossAssim` (A_g), for example `sunlit_incident_ppfd_layer_0` and `shaded_GrossAssim_layer_9`. To help with plotting, it is beneficial to first extract all the (Q, A_g) pairs from each layer and leaf class in the output of a BioCro simulation; here we make a helping function to accomplish this.

```
extract_aq_scatter <- function(biocro_output) {
  light_column_names <- grep(
    '(sunlit|shaded)_incident_ppfd_layer_[0-9]',
    names(biocro_output),
    value = TRUE
  )

  assim_column_names <- grep(
    '(sunlit|shaded)_GrossAssim_layer_[0-9]',
    names(biocro_output),
    value=TRUE
  )

  aq_scatter <- data.frame(
    incident_ppfd = unlist(biocro_output[light_column_names]),
    gross_assimilation = unlist(biocro_output[assim_column_names]),
    row.names = NULL
  )

  return(aq_scatter)
}
```

Now it is possible to produce scatter plots of (Q, A_g) values for each model during 2002 (Figures 3 and 4; these two figures are combined to form Figure 4b in the BioCro II paper). Note: because the canopy model uses ten layers and two leaf classes, there are twenty (Q, A_g) pairs per time point; because the crop's state is stored at hourly intervals over a growing season that lasts for 137 days, there are 3,288 time points, yielding 65,760 (Q, A_g) pairs in total for each simulation. Plotting all of these points results in very large files, so, to reduce the size of the vignette PDF, here we only plot a subset of the points.

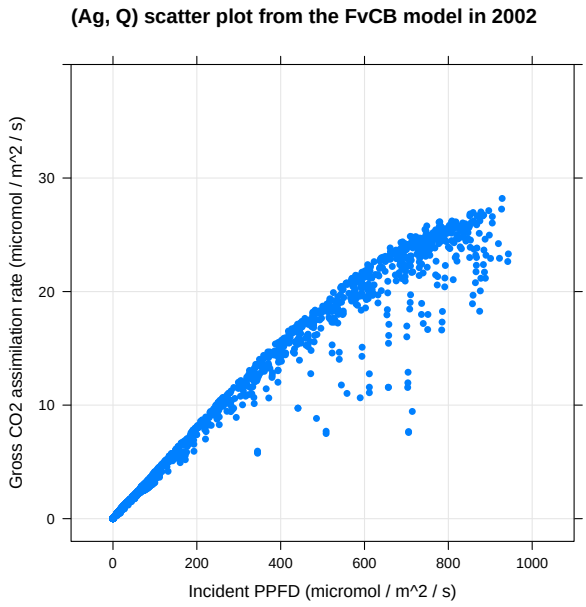

Figure 3: A_g versus Q for soybean leaves in each canopy layer (0-9) and class (sunlit or shaded) as calculated during 2002 using the FvCB photosynthesis model.

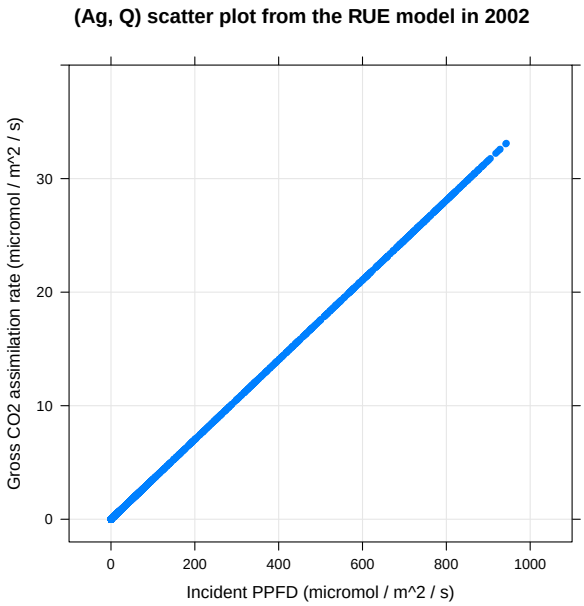

Figure 4: A_g versus Q for soybean leaves in each canopy layer (0-9) and class (sunlit or shaded) as calculated during 2002 using the RUE photosynthesis model.

6 FvCB light response curve sensitivity analysis

Here we calculate the sensitivity of the FvCB model to various inputs known to influence the photosynthetic rate. To do this, we will use the `BioCro:c3_leaf_photosynthesis` module to produce light curves. This module requires incident PPFD and absorbed shortwave energy as inputs, so we create a series of each.

```
# Choose a set of incident PPFD values to use (micromol / m^2 / s)
incident_ppfd <- seq(0, 1000, length.out = 501)

# Determine corresponding incident PAR values (J / m^2 / s) using the average
# energy per micromole of photosynthetically active photons in sunlight
incident_par <- incident_ppfd * cmi_soybean$parameters$par_energy_content

# Determine the corresponding incident shorwave values using the fraction of
# solar energy that lies in the PAR band (J / m^2 / s)
incident_shortwave <- incident_par / cmi_soybean$parameters$par_energy_fraction

# Determine the corresponding absorbed shortwave energy values using the
# shortwave reflectance and transmittance of the leaf (J / m^2 / s)
average_absorbed_shortwave <-
    incident_shortwave *
    (1 - cmi_soybean$parameters$leaf_reflectance -
        cmi_soybean$parameters$leaf_transmittance) /
    (1 - cmi_soybean$parameters$leaf_transmittance)

# Make a data frame with the incident PPFD and absorbed shortwave values, where
# we also include values of a few other required parameters
light_curve_inputs <- data.frame(
    incident_ppfd = incident_ppfd,
```

```
    average_absorbed_shortwave = average_absorbed_shortwave,
    rh = 0.75,
    temp = 25,
    windspeed = 3.28,
    Catm = catm[['2002']],
    StomataWS = 0.99,
    height = 0.75
)
```

Now we need to create a function that calculates assimilation sensitivity coefficients for one independent variable using the `light_curve_inputs`. Here we numerically calculate a value for the derivative df/dx of a function $f(x)$ at x_0 according to

$$\frac{df}{dx} = \frac{f(x_0 + \delta) - f(x_0 - \delta)}{2\delta}, \tag{1}$$

where δ is a small perturbation away from the central value x_0. Then the normalized sensitivity coefficient is given by

$$c = \frac{df/dx}{f(x_0)/x_0} \tag{2}$$

In this function, the `evaluate_module` BioCro function is used to calculate the gross assimilation rate for different values of the independent variable specified by `varname`; in other words, `varname` is x and the `BioCro:c3_leaf_photosynthesis` module defines $f(x)$.

```
assim_sensitivity <- function(
    varname,
    base_inputs,
    relative_perturbation = 1e-6
)
{
  module <- 'BioCro:c3_leaf_photosynthesis'

  var_center <- base_inputs[[varname]]
  gross_assim_center <- evaluate_module(module, base_inputs)$GrossAssim

  neg_inputs <- base_inputs
  neg_var <- base_inputs[[varname]] * (1 - relative_perturbation)
  neg_inputs[[varname]] <- neg_var
  gross_assim_neg <- evaluate_module(module, neg_inputs)$GrossAssim

  pos_inputs <- base_inputs
  pos_var <- base_inputs[[varname]] * (1 + relative_perturbation)
  pos_inputs[[varname]] <- pos_var
  gross_assim_pos <- evaluate_module(module, pos_inputs)$GrossAssim

  dadx = (gross_assim_pos - gross_assim_neg) / (pos_var - neg_var)
  return(dadx / (gross_assim_center / var_center))
}
```

Now we can use this function to calculate sensitivity coefficients for atmospheric [CO_2], relative humidity, air temperature, water stress, and wind speed (Figure 5; this is Figure 5a in the BioCro II paper).

```
fvcb_light_curve_sensitivity_variables <-
  c('Catm', 'rh', 'temp', 'StomataWS', 'windspeed')
```

```
fvcb_sensitivity_light_curve_result <- data.frame(
  incident_ppfd = light_curve_inputs[['incident_ppfd']]
)

# For each variable of interest, calculate sensitivity at each of the light
# intensities in `light_curve_inputs`
for (varname in fvcb_light_curve_sensitivity_variables) {
  fvcb_sensitivity_light_curve_result[[varname]] <-
    apply(
      light_curve_inputs,
      1,
      function(x) {assim_sensitivity(
        varname,
        c(within(cmi_soybean$parameters, {rm(Catm)}), as.list(x))
      )}
    )
}
```

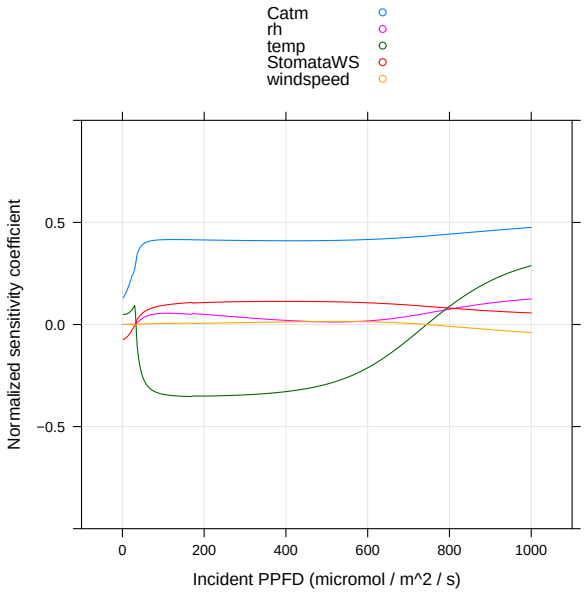

Figure 5: Normalized sensitivity coeffficients characterizing the response of A_g to several inputs in the FvCB photosynthesis model.

7 Biomass sensitivity analysis

Here we wish to use the two photosynthesis models in a full crop model and compare the sensitivities of the total biomass to various inputs. The approach is slightly different depending on whether the input is a driver or a parameter.

7.1 Sensitivity to a driver: air temperature

Several routes are possible when analyzing the sensitivity of a model's output to a driver. Here we choose the instantaneous biomass as the dependent variable, and we perturb the driver at all time points. For example,

in the case of air temperature, we perturb the air temperature at every time point, run the simulation, and calculate the resulting change in biomass at every time point.

First we will define a function that calculates biomass sensitivity coefficients for one independent driver value using weather data from 2002. Here the independent variable is specified by the `varname` input. To make the perturbation required to numerically calculate a derivative, the value of the driver is increased or decreased at every time point.

Special care must be taken when `varname` is `temp`, since the air temperature drops below zero near the end of the year, causing strange behavior in the sensitivity analysis. To avoid this, we convert from °C to K, then perform the perturbation, and finally convert back to °C to run the simulation.

```
biomass_driver_sensitivity <- function(
    varname,
    sens_parameters,
    canopy_photosynthesis_module,
    relative_perturbation = 1e-5
)
{
  c_to_k <- 273.15

  default_drivers <- within(soy_weather[['2002']], {temp = temp + c_to_k})

  default_result <- with(cmi_soybean, {run_biocro(
    initial_values,
    sens_parameters,
    within(default_drivers, {temp = temp - c_to_k}),
    within(direct_modules, {canopy_photosynthesis = canopy_photosynthesis_module}),
    differential_modules,
    ode_solver
  )})

  neg_drivers <- default_drivers
  neg_drivers[[varname]] <- default_drivers[[varname]] * (1 - relative_perturbation)
  neg_result <- with(cmi_soybean, {run_biocro(
    initial_values,
    sens_parameters,
    within(neg_drivers, {temp = temp - c_to_k}),
    within(direct_modules, {canopy_photosynthesis = canopy_photosynthesis_module}),
    differential_modules,
    ode_solver
  )})

  pos_drivers <- default_drivers
  pos_drivers[[varname]] <- default_drivers[[varname]] * (1 + relative_perturbation)
  pos_result <- with(cmi_soybean, {run_biocro(
    initial_values,
    sens_parameters,
    within(pos_drivers, {temp = temp - c_to_k}),
    within(direct_modules, {canopy_photosynthesis = canopy_photosynthesis_module}),
    differential_modules,
    ode_solver
  )})

  dMdx <-
    (pos_result[['total_biomass']] - neg_result[['total_biomass']]) /
```

```
    (pos_drivers[[varname]] - neg_drivers[[varname]])

  normalized_sensitivity <-
    dMdx / (default_result[['total_biomass']] / default_drivers[[varname]])

  return(
    data.frame(
      normalized_sensitivity = normalized_sensitivity,
      time = default_result[['time']]
    )
  )
}
```

With this function, we can calculate normalized sensitivity coefficients for biomass in response to air temperature with both the FvCB and RUE models (Figure 6; this is Figure 5b in the BioCro II paper).

```
biomass_temp_sensitivity_fvcb <- biomass_driver_sensitivity(
  'temp',
  cmi_soybean$parameters,
  'BioCro:ten_layer_c3_canopy'
)

biomass_temp_sensitivity_rue <- biomass_driver_sensitivity(
  'temp',
  within(cmi_soybean$parameters, {alpha_rue = best_alpha_rue}),
  'BioCro:ten_layer_rue_canopy'
)
```

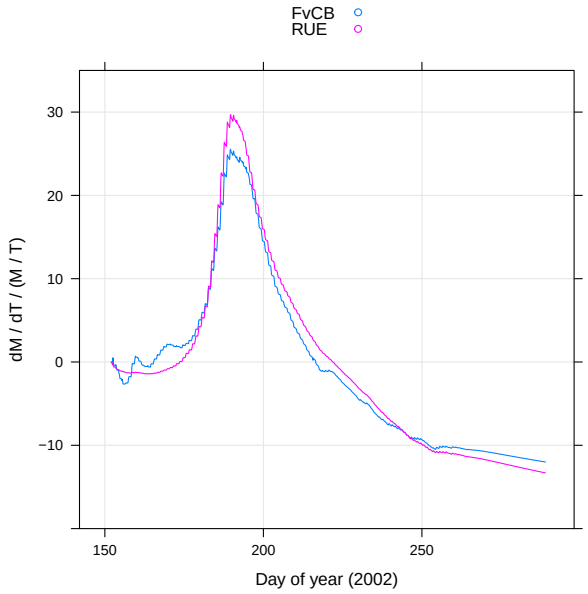

Figure 6: Normalized sensitivity coeffficients characterizing the response of the total biomass M to air temperature when using either the FvCB or RUE photosynthesis model.

7.2 Sensitivity to a parameter: atmospheric [CO_2]

Here we will define a function that calculates biomass sensitivity coefficients for one independent parameter value using weather data from 2002. Here the independent variable is specified by the `varname` input. Note: This function will produce errors for any `varname` whose base value is zero.

```
biomass_parameter_sensitivity <- function(
    varname,
    sens_parameters,
    canopy_photosynthesis_module,
    relative_perturbation = 1e-6
)
{
  default_result <- with(cmi_soybean, {run_biocro(
    initial_values,
    sens_parameters,
    soy_weather[['2002']],
    within(direct_modules, {canopy_photosynthesis = canopy_photosynthesis_module}),
    differential_modules,
    ode_solver
  )})

  neg_parameters <- sens_parameters
  neg_parameters[[varname]] <- sens_parameters[[varname]] * (1 - relative_perturbation)
  neg_result <- with(cmi_soybean, {run_biocro(
    initial_values,
    neg_parameters,
    soy_weather[['2002']],
    within(direct_modules, {canopy_photosynthesis = canopy_photosynthesis_module}),
    differential_modules,
    ode_solver
  )})

  pos_parameters <- sens_parameters
  pos_parameters[[varname]] <- sens_parameters[[varname]] * (1 + relative_perturbation)
  pos_result <- with(cmi_soybean, {run_biocro(
    initial_values,
    pos_parameters,
    soy_weather[['2002']],
    within(direct_modules, {canopy_photosynthesis = canopy_photosynthesis_module}),
    differential_modules,
    ode_solver
  )})

  dMdx <-
    (pos_result[['total_biomass']] - neg_result[['total_biomass']]) /
    (pos_parameters[[varname]] - neg_parameters[[varname]])

  normalized_sensitivity <-
    dMdx / (default_result[['total_biomass']] / sens_parameters[[varname]])

  return(
    data.frame(
      normalized_sensitivity = normalized_sensitivity,
```

```
      time = default_result[['time']]
    )
  )
}
```

With this function, we can calculate normalized sensitivity coefficients for biomass in response to atmospheric $[CO_2]$ with both the FvCB and RUE models (Figure 7; this is Figure 5c in the BioCro II paper).

```
biomass_catm_sensitivity_fvcb <- biomass_parameter_sensitivity(
  'Catm',
  cmi_soybean$parameters,
  'BioCro:ten_layer_c3_canopy'
)

biomass_catm_sensitivity_rue <- biomass_parameter_sensitivity(
  'Catm',
  within(cmi_soybean$parameters, {alpha_rue = best_alpha_rue}),
  'BioCro:ten_layer_rue_canopy'
)
```

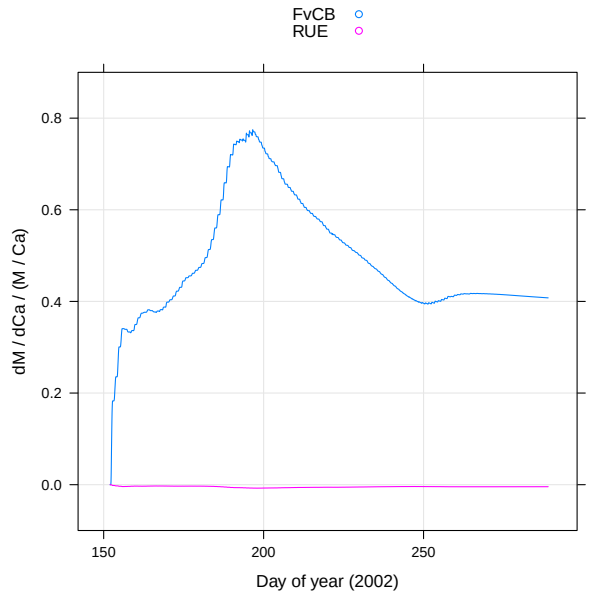

Figure 7: Normalized sensitivity coeffficients characterizing the response of the total biomass M to atmospheric $[CO_2]$ when using either the FvCB or RUE photosynthesis model.

8 Comparing FvCB and RUE photosynthesis models in the year 2006

A simple way to demonstrate that the two photosynthesis models have different responses to changes in environmental variables is to plot biomass values in a different year, such as 2006. To do this, we will first run the FvCB soybean model for the year 2006, ensuring that we're using the correct value for the atmospheric $[CO_2]$. Then we will run the RUE model for 2006, using the value of `alpha_rue` that was

optimized for 2002. A comparison shows a divergence in the predicted mass values (Figure 8; this is Figure 6c in the BioCro II paper).

```
fvcb_result_2006 <- with(cmi_soybean, {run_biocro(
  initial_values,
  within(parameters, {Catm = catm[['2006']]}),
  soy_weather[['2006']],
  direct_modules,
  differential_modules,
  ode_solver
)})

# Run the RUE model with the optimal value for alpha_rue determined for 2002
rue_result_2006 <- with(cmi_soybean, {run_biocro(
  initial_values,
  within(parameters, {alpha_rue = best_alpha_rue; Catm = catm[['2006']]}),
  soy_weather[['2006']],
  within(direct_modules, {canopy_photosynthesis = 'BioCro:ten_layer_rue_canopy'}),
  differential_modules,
  ode_solver
)})
```

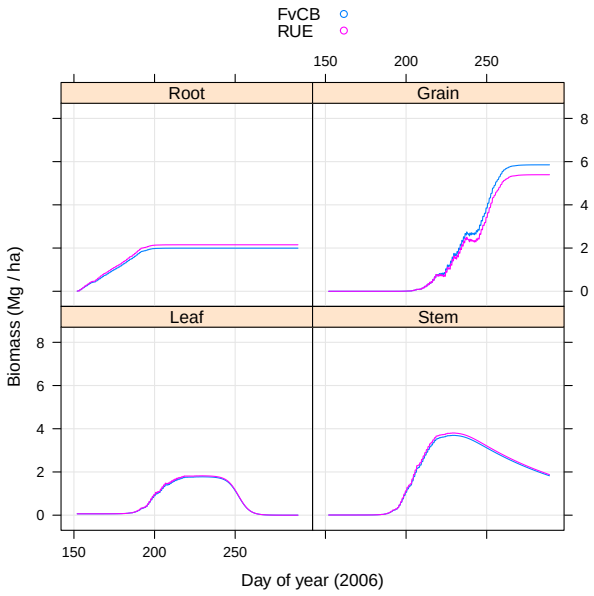

Figure 8: Soybean root, grain, leaf, and stem biomass values calculated during 2006 using either the FvCB or RUE photosynthesis model.

9 Comparing RUE and FvCB models across multiple years

Here we wish to examine the results of the two models over a range of years. In particular, atmospheric [CO_2] has steadily increased over the past several decades, so do the two models diverge with time? To answer this question, we run the soybean model with either the FvCB or RUE photosynthesis equations for

multiple years and compare the end-of-season biomass (Figure 9; this is Figure 6b from the BioCro II paper); from this result, a clear trend in the biomass difference with atmospheric [CO_2] can be observed (Figure 10; this is Figure 6c from the BioCro II paper).

```
# Decide which years to use
years <- as.character(seq(1995, 2020))

# Initialize vectors to store final biomass and atmospheric CO2 values
final_biomass_seq_rue <- numeric(length(years))
final_biomass_seq_fvcb <- numeric(length(years))
catm_seq <- numeric(length(years))

# Get final biomass values for each year in each model
for (i in seq_along(years)) {
  # Run the RUE soybean model for this year, ensuring that we're using the
  # correct value for the atmospheric CO2 concentration
  rue_result <- with(cmi_soybean, {run_biocro(
    initial_values,
    within(parameters, {alpha_rue = best_alpha_rue; Catm = catm[[years[i]]]}),
    soy_weather[[years[i]]],
    within(direct_modules, {canopy_photosynthesis = 'BioCro:ten_layer_rue_canopy'}),
    differential_modules,
    ode_solver
  )})

  # Run the FvCB soybean model for this year, ensuring that we're using the
  # correct value for the atmospheric CO2 concentration
  fvcb_result <- with(cmi_soybean, {run_biocro(
    initial_values,
    within(parameters, {Catm = catm[[years[i]]]}),
    soy_weather[[years[i]]],
    direct_modules,
    differential_modules,
    ode_solver
  )})

  # Store the final biomass and atmospheric CO2 values
  final_biomass_seq_rue[i] <- final_biomass(rue_result)
  final_biomass_seq_fvcb[i] <- final_biomass(fvcb_result)
  catm_seq[i] <- catm[[years[i]]]
}
```

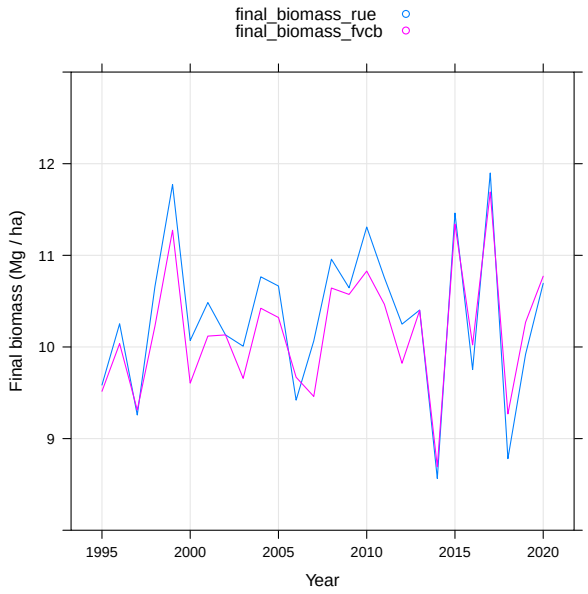

Figure 9: End-of-season soybean biomass values calculated for Champaign, IL using either the FvCB or RUE photosynthesis equations for each year from 1995 - 2020.

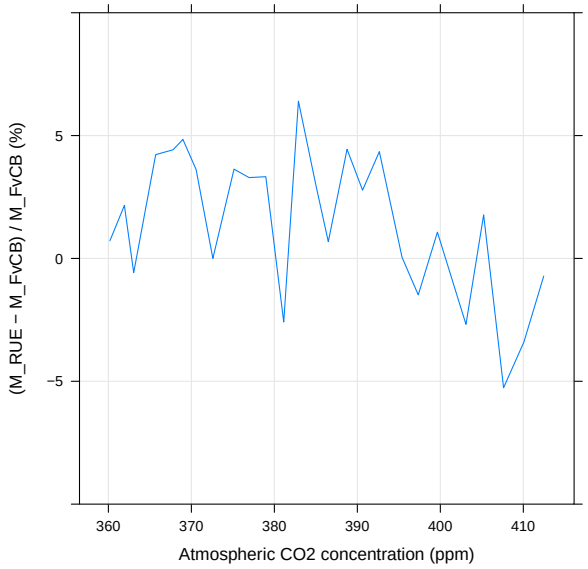

Figure 10: End-of-season biomass differences between the FvCB and RUE soybean models expressed as percentages for each year from 1995 - 2020, plotted against yearly average [CO_2] levels.