

# Package ‘csmbuilder’

May 8, 2026

**Type** Package

**Title** A Collection of Tools for Building Cropping System Models

**Version** 0.1.0

**Description** A collection of tools for designing, implementing, testing, documenting and visualizing dynamic simulation cropping system models. Models are specified as a combination of state variables, parameters, intermediate factors and input data that define a system of ordinary differential equations. Specified models can be used to simulate dynamic processes using numerical integration algorithms.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Depends** R (>= 4.2.0)

**Suggests** deSolve, Rcpp, tinytest

**NeedsCompilation** no

**Author** Phillip D Alderman [aut, cre],  
Pratishtha Poudel [aut]

**Maintainer** Phillip D Alderman <phillip.alderman@okstate.edu>

**Repository** CRAN

**Date/Publication** 2026-04-03 08:20:15 UTC

## Contents

csm_create_data_structure . . . . .	2
csm_create_model . . . . .	3
csm_create_parameter . . . . .	4
csm_create_state . . . . .	5
csm_create_transform . . . . .	6
csm_create_variable . . . . .	7
csm_get_at_t . . . . .	8

csm_mod_arr . . . . .	8
csm_render_model . . . . .	9
csm_run_sim . . . . .	10
csm_time_vector . . . . .	12
is_data_structure . . . . .	12
is_input . . . . .	13
is_parameter . . . . .	13
is_state_variable . . . . .	14
is_transform . . . . .	14

## Index 15

csm\_create\_data\_structure

*Create a cropping systems model (CSM) data structure*

### Description

Create a cropping systems model (CSM) data structure

### Usage

```
csm_create_data_structure(name, definition, variables)
```

### Arguments

name	a length-one character vector name of a variable
definition	the definition of the data structure
variables	a list of the CSM variables (as defined with [csmbuilder::csm_create_variable()]) that are contained within the data structure

### Value

a list of 'csm\_data\_structure' objects

### Examples

```
# Create variables:
wth_variables <- c(
  csm_create_variable("Tair",
    "air temperature",
    "Celsius"),
  csm_create_variable("SRAD",
    "solar radiation",
    "MJ/m2/d"))

# Create weather data structure:
weather <- csm_create_data_structure("weather",
  "weather data",
```

wth\_variables)

---

csm\_create\_model      *Create a model definition for a Cropping System Model (CSM)*

---

## Description

Create a model definition for a Cropping System Model (CSM)

## Usage

```
csm_create_model(state, ..., name = "model")
```

## Arguments

state	a list vector containing CSM state variables defined using <code>csm_create_state()</code> in the intended order
...	optional arguments of list vectors containing CSM parameters defined using <code>csm_create_parameter()</code> or CSM variables defined using <code>csm_create_variable()</code>
name	a character string containing a name for the models

## Value

a list which defines all components of a model including state variables, input variables, parameters, transformed variables and data structures.

## Examples

```
# Define state variables
lv_state <- csm_create_state(
  c("x", "y"),
  definition = c("prey", "predator"),
  units = c("rabbits per square km", "foxes per square km"),
  expression(~alpha*x-beta*x*y, ~delta*x*y-gamma*y))

# Define parameters
lv_parameters <- csm_create_parameter(
  c("alpha", "beta", "gamma", "delta"),
  definition = c("maximum prey per capita growth rate",
    "effect of predator population on prey death rate",
    "predator per capita death rate",
    "effect of prey population on predator growth rate"),
  units = c("rabbits per rabbit", "per fox",
    "foxes per fox", "foxes per rabbit"))

# Define model
lotka_volterra_model <-
  csm_create_model(
```

```
state = lv_state,
parms = lv_parameters)
```

---

csm\_create\_parameter *Create a cropping systems model (CSM) parameter variable*

---

## Description

Create a cropping systems model (CSM) parameter variable

## Usage

```
csm_create_parameter(
  name,
  definition,
  units,
  lower_bound = NULL,
  upper_bound = NULL
)
```

## Arguments

name	a length-one character vector name of a variable
definition	a length-one character vector that defines the CSM variable
units	a length-one character vector of the units of the CSM variable
lower_bound	a numerical value providing the lower bound for the parameter
upper_bound	a numerical value providing the upper bound for the parameter

## Value

a list of csm\_parameter objects

## Examples

```
# Define Lotka-Voterra parameters with single call

lv_parameters <- csm_create_parameter(
  name = c("alpha", "beta", "gamma", "delta"),
  definition = c("maximum prey per capita growth rate",
                "effect of predator population on prey death rate",
                "predator per capita death rate",
                "effect of prey population on predator growth rate"),
  units = c("rabbits per rabbit", "per fox",
            "foxes per fox", "foxes per rabbit"))

# Define Lotka-Volterra parameters with multiple calls
```

```
lv_parameters <-  
  c(  
    csm_create_parameter(  
      name = "alpha",  
      definition = "maximum prey per capita growth rate",  
      units = "rabbits per rabbit"),  
    csm_create_parameter(  
      name = "beta",  
      definition = "effect of predator population on prey death rate",  
      units = "per fox"),  
    csm_create_parameter(  
      name = "gamma",  
      definition = "predator per capita death rate",  
      units = "foxes per fox"),  
    csm_create_parameter(  
      name = "delta",  
      definition = "effect of prey population on predator growth rate",  
      units = "foxes per rabbit"))
```

---

csm\_create\_state

*Create a cropping systems model (CSM) state variable*

---

## Description

Create a cropping systems model (CSM) state variable

## Usage

```
csm_create_state(name, definition, units, equation)
```

## Arguments

name	a length-one character vector name of a variable
definition	a length-one character vector that defines the CSM variable
units	a length-one character vector of the units of the CSM variable
equation	an R expression with the equation for the rate of change of the CSM state variable

## Value

a list of csm\_state objects

**Examples**

```
# Define state variables with single call

lv_state <- csm_create_state(
  c("x", "y"),
  definition = c("prey", "predator"),
  units = c("rabbits per square km", "foxes per square km"),
  expression(~alpha*x-beta*x*y, ~delta*x*y-gamma*y))

# Define state variables with multiple calls

lv_state <-
c(
  csm_create_state(
    name = "x",
    definition = "prey",
    units = "rabbits per square km",
    equation = ~alpha*x-beta*x*y),
  csm_create_state(
    name = "y",
    definition = "predator",
    units = "foxes per square km",
    equation = ~delta*x*y-gamma*y)
)
```

---

csm\_create\_transform *Create a cropping systems model (CSM) transformed variable*

---

**Description**

Create a cropping systems model (CSM) transformed variable

**Usage**

```
csm_create_transform(name, definition, units, equation)
```

**Arguments**

name	a length-one character vector name of a variable
definition	a length-one character vector that defines the CSM variable
units	a length-one character vector of the units of the CSM variable
equation	an R expression with the equation for the value of the transformed CSM state variable

**Value**

a list of csm\_transform objects

## Examples

```
# Define intermediate factor

sp_factors <- csmbuilder::csm_create_transform(
  name = "fv",
  definition = "vernalization factor",
  units = "relative progress towards complete vernalization (0-1)",
  equation = ~min(c(cum_vrn/vreq, 1)))
```

---

`csm_create_variable` *Create a cropping systems model (CSM) variable*

---

## Description

Create a cropping systems model (CSM) variable

## Usage

```
csm_create_variable(name, definition, units)
```

## Arguments

<code>name</code>	a length-one character vector name of a variable
<code>definition</code>	a length-one character vector that defines the CSM variable
<code>units</code>	a length-one character vector of the units of the CSM variable

## Value

a list of `csm_variable` objects

## Examples

```
Tair <- csm_create_variable(name = "Tair",
  definition = "air temperature",
  units = "Celsius")
```

---

`csm_get_at_t`*Linearly interpolate a time-varying variable at specific time point*

---

**Description**

Linearly interpolate a time-varying variable at specific time point

**Usage**

```
csm_get_at_t(  
  x,  
  t_ind,  
  t,  
  method = "linear",  
  search = c("interpolation", "bisection", "bruteforce", "i=t+1")  
)
```

**Arguments**

<code>x</code>	a vector of a time-varying values for which to interpolate
<code>t_ind</code>	a vector of times corresponding to the values in <code>x</code>
<code>t</code>	a single time point at which to return a value
<code>method</code>	a string indicating what method to use for interpolation. One of: "linear"
<code>search</code>	a string indicating what method to use for finding the correct indices within <code>t_ind</code> . One of: "bisection", "interpolation"

**Value**

the numeric value of the time-varying variable interpolated at time `t`

---

`csm_mod_arr`*Modified Arrhenius function*

---

**Description**

Modified Arrhenius function

**Usage**

```
csm_mod_arr(Tt, ko, H, E, To)
```

**Arguments**

Tt	temperature in Celsius
ko	reaction rate at the optimum temperature (To)
H	deactivation energy parameter
E	activation energy parameter
To	optimum temperature in Celsius

**Value**

a numeric value of the reaction rate at temperature Tt

---

csm_render_model	<i>Render a defined Cropping System Model (CSM)</i>
------------------	---

---

**Description**

Render a defined Cropping System Model (CSM)

**Usage**

```
csm_render_model(
  model,
  name = "dy_dt",
  output_type = c("function", "code"),
  language = c("R", "Rcpp"),
  arg_alias = NULL,
  insert_functions = NULL
)
```

**Arguments**

model	a list vector containing a CSM as created by <a href="#">csm_create_model()</a>
name	name of the resulting function
output_type	a character value indicating the type of output to produce; one of: "function" (a callable function) or "code" (computer code for the model)
language	a character value indicating which programming language into which to render the model
arg_alias	an optional named character vector whose names indicate variables for which to use an alias within the generated function and whose elements provide the corresponding alias
insert_functions	an optional list of functions to add to rendered code

**Value**

Either an R function object (if `output_type="function"`) or a character vector of model code (if `output_type="code"`) in the programming language specified by `language`.

**Examples**

```
# Define state variables
lv_state <- csm_create_state(
  c("x", "y"),
  definition = c("prey", "predator"),
  units = c("rabbits per square km", "foxes per square km"),
  expression(~alpha*x-beta*x*y, ~delta*x*y-gamma*y))

# Define parameters
lv_parameters <- csm_create_parameter(
  c("alpha", "beta", "gamma", "delta"),
  definition = c("maximum prey per capita growth rate",
    "effect of predator population on prey death rate",
    "predator per capita death rate",
    "effect of prey population on predator growth rate"),
  units = c("rabbits per rabbit", "per fox",
    "foxes per fox", "foxes per rabbit"))

# Define model
lotka_volterra_model <-
  csm_create_model(
    state = lv_state,
    parms = lv_parameters)

# Render model into raw R code
lotka_volterra_code <-
  csm_render_model(lotka_volterra_model,
    output_type = "code",
    language = "R")

# Render model into a callable R function
lotka_volterra_fun <-
  csm_render_model(lotka_volterra_model,
    output_type = "function",
    language = "R")
```

---

csm\_run\_sim

*Run a Cropping System Model (CSM) simulation*


---

**Description**

Run a Cropping System Model (CSM) simulation

**Usage**

```
csm_run_sim(model_function, y_init, t, ..., method = "euler")
```

**Arguments**

`model_function` a rendered model produced by `csm_render_model()`

`y_init` a vector of initial values for the model state variables

`t` an optional vector of time points for which simulated model outputs are desired

`...` additional arguments to pass to `model_function` for simulation

`method` numerical integration method to be used. See `deSolve::ode()` for more details

**Value**

a data frame with one row for each time point specified by `t`

**Examples**

```
# Define state variables
lv_state <- csm_create_state(
  c("x", "y"),
  definition = c("prey", "predator"),
  units = c("rabbits per square km", "foxes per square km"),
  expression(~alpha*x-beta*x*y, ~delta*x*y-gamma*y))

# Define parameters
lv_parameters <- csm_create_parameter(
  c("alpha", "beta", "gamma", "delta"),
  definition = c("maximum prey per capita growth rate",
    "effect of predator population on prey death rate",
    "predator per capita death rate",
    "effect of prey population on predator growth rate"),
  units = c("rabbits per rabbit", "per fox",
    "foxes per fox", "foxes per rabbit"))

# Define model
lotka_volterra_model <-
  csm_create_model(
    state = lv_state,
    parms = lv_parameters)

# Render model into a callable R function
lotka_volterra_fun <-
  csm_render_model(lotka_volterra_model,
    output_type = "function",
    language = "R")

# Run model simulation
lotka_volterra_out <-
  csm_run_sim(model_function = lotka_volterra_fun,
    y_init = c(x = 10,
```

```

        y = 10),
t = csm_time_vector(0, 100, 0.01),
parms = c(alpha = 1.1,
          beta = 0.4,
          gamma = 0.1,
          delta = 0.4))

```

---

csm_time_vector	<i>Generate vector of times for simulation</i>
-----------------	--

---

### Description

Generate vector of times for simulation

### Usage

```
csm_time_vector(t_init, t_max, dt = 1)
```

### Arguments

t_init	a numerical value providing the initial time to use for simulation
t_max	a numerical value providing the final time to use for simulation
dt	a numerical value providing the size of the time step to use for simulation

### Value

a numeric vector

### Examples

```
csm_time_vector(0, 100, dt = 0.01)
```

---

is_data_structure	<i>Determine if object is a data structure</i>
-------------------	--

---

### Description

Determine if object is a data structure

### Usage

```
is_data_structure(x)
```

**Arguments**

x                    any R object

**Value**

a logical value indicating whether or not x is of class csm\_data\_structure

---

is\_input                    *Determine if object is a model input*

---

**Description**

Determine if object is a model input

**Usage**

is\_input(x)

**Arguments**

x                    any R object

**Value**

a logical value indicating whether or not x is an input variable

---

is\_parameter                    *Determine if object is a model parameter*

---

**Description**

Determine if object is a model parameter

**Usage**

is\_parameter(x)

**Arguments**

x                    any R object

**Value**

a logical value indicating whether or not x is of class csm\_parameter

---

is\_state\_variable      *Determine if object is state variable*

---

**Description**

Determine if object is state variable

**Usage**

```
is_state_variable(x)
```

**Arguments**

x                      any R object

**Value**

a logical value indicating whether or not x is of class csm\_state

---

is\_transform            *Determine if object is a transformed variable*

---

**Description**

Determine if object is a transformed variable

**Usage**

```
is_transform(x)
```

**Arguments**

x                      any R object

**Value**

a logical value indicating whether or not x is of class csm\_transform

# Index

`csm_create_data_structure`, [2](#)  
`csm_create_model`, [3](#)  
`csm_create_model()`, [9](#)  
`csm_create_parameter`, [4](#)  
`csm_create_parameter()`, [3](#)  
`csm_create_state`, [5](#)  
`csm_create_state()`, [3](#)  
`csm_create_transform`, [6](#)  
`csm_create_variable`, [7](#)  
`csm_create_variable()`, [3](#)  
`csm_get_at_t`, [8](#)  
`csm_mod_arr`, [8](#)  
`csm_render_model`, [9](#)  
`csm_render_model()`, [11](#)  
`csm_run_sim`, [10](#)  
`csm_time_vector`, [12](#)

`deSolve::ode()`, [11](#)

`is_data_structure`, [12](#)  
`is_input`, [13](#)  
`is_parameter`, [13](#)  
`is_state_variable`, [14](#)  
`is_transform`, [14](#)