Package: OptirrigCORE (via r-universe)

November 1, 2026

Type Package

Title Optirrig Core: Simulate and Optimize Irrigation Scenarios

Version 0.8.0

Description The OptirrigCORE package in R is based on the Optirrig model. It facilitates the extraction and formatting of observation data used by the model, as well as the integration of plant, soil and yield descriptions. It supports various analytical methods, such as Ex-Post and Ex-Ante analyses, with a view to drawing up an optimised irrigation schedule based on the criteria. The package includes all the calculation tools needed to describe biophysical processes with a generalist approach. It can be used as a dependency or complement to modify and reinterpret biophysical descriptions. What's more, it automates the modeling process, the description of biophysical processes and the analysis of results and performance, guaranteeing a reproducible approach. It also makes it easy to compile results in automated reports.

Encoding UTF-8

License AGPL (>= 3)

URL https://forge.inrae.fr/OptirrigHIVE/OptirrigCORE, https://optirrighive.pages-forge.inrae.fr/OptirrigCORE/dev, https://inrae.r-universe.dev/OptirrigCORE

LazyData true

Depends R (>= 4.1.0)

Imports ini, dplyr, utils, readr, lubridate, logger, readxl, readODS, future, future.apply

Suggests OptirrigTOOLS, OptirrigVIEW, yaml, callr, pkgload, ggplot2, patchwork, knitr, kableExtra, testthat (>= 3.0.0), rmarkdown

Additional_repositories https://inrae.r-universe.dev

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev libx11-dev

Repository https://inrae.r-universe.dev

Date/Publication 2026-07-27 09:22:21 UTC

RemoteUrl https://forge.inrae.fr/OptirrigHIVE/OptirrigCORE

RemoteRef main

RemoteSha cd5d66febe91c56aa6acf2ab25532f7b96aa0add

Contents

add_missing_params . . . 4
calc_alpha . . . 5
calc_B . . . 7
calc_B_root . . . 9
calc_Bp . . . 11
calc_Bp_root . . . 11
calc_case . . . 12
calc_Cp . . . 13
calc_crac . . . 14
calc_d . . . 16
calc_degred . . . 17
calc_dose . . . 18
calc_dtheta_RU . . . 20
calc_dzroot . . . 21
calc_dzrootp . . . 23
calc_ES . . . 24
calc_ES0 . . . 26
calc_ET0 . . . 28
calc_ET0_mulch_effect . . . 29
calc_ETM . . . 30
calc_ETR . . . 32
calc_HI . . . 33
calc_HUg . . . 35
calc_IR . . . 36
calc_Kc . . . 38
calc_Kd . . . 39
calc_KES . . . 41
calc_Ks . . . 43
calc_ks_root . . . 45
calc_KTP . . . 46
calc_LAI . . . 47
calc_LAIp . . . 51
calc_LT . . . 53
calc_PAR . . . 55

calc_quota . . . 57
calc_R . . . 58
calc_Rgravity . . . 60
calc_RU . . . 62
calc_RUmax . . . 64
calc_stress . . . 66
calc_stress_noci_B . . . 67
calc_stress_noci_LAI . . . 69
calc_stress_T . . . 70
calc_stress_water . . . 72
calc_sugar_accumulation . . . 73
calc_tau . . . 74
calc_taum . . . 74
calc_Tcrit . . . 76
calc_theta . . . 76
calc_theta_fc . . . 77
calc_theta_r . . . 78
calc_theta_rel . . . 79
calc_theta_sat . . . 80
calc_theta_wp . . . 81
calc_threshold . . . 82
calc_TP . . . 84
calc_TP0 . . . 85
calc_TT . . . 86
calc_TT_norm . . . 87
calc_TT_rel . . . 89
calc_TT_sowing . . . 90
calc_Y_grain . . . 91
calc_Y_hi . . . 94
calc_Y_sugar . . . 95
calc_zroot . . . 96
calc_zrootp . . . 97
create_model_inputs . . . 99
create_model_inputs_climate . . . 100
create_model_inputs_crop . . . 101
create_model_inputs_irrigation . . . 102
create_model_inputs_run . . . 103
create_model_inputs_soil . . . 104
get_data_path . . . 104
get_iday . . . 106
get_model_output_path . . . 107
get_vars_data . . . 108
init_model . . . 109
init_model_climate . . . 110
init_model_crop_dev . . . 111
init_model_irrigation . . . 112
init_model_vars . . . 113
inputs_meta . . . 113

load_config . . . 114
MAIZE_21LAVALETTE_B . . . 115
MAIZE_21LAVALETTE_clim . . . 115
MAIZE_21LAVALETTE_irrig . . . 116
MAIZE_21LAVALETTE_LAI . . . 117
MAIZE_21LAVALETTE_RES . . . 117
parse_date_time . . . 118
read . . . 119
run_model . . . 120
run_model_crop_dev . . . 123
run_model_crop_dev_B . . . 124
run_model_crop_dev_B_root . . . 125
run_model_crop_dev_LAI . . . 126
run_model_crop_dev_stress_water . . . 127
run_model_crop_dev_Y . . . 128
run_model_crop_dev_Y_grain . . . 128
run_model_crop_dev_Y_sugar . . . 129
run_model_crop_dev_zroot . . . 130
run_model_finance . . . 131
run_model_irrig_strat . . . 132
run_model_irrig_strat_dose . . . 132
run_model_water_balance . . . 135
run_model_water_balance_drainage . . . 136
run_model_water_balance_ES . . . 137
run_model_water_balance_ETM . . . 137
run_model_water_balance_init_states . . . 138
run_model_water_balance_others . . . 139
run_model_water_balance_routing_water . . . 140
run_model_water_balance_TP . . . 141
run_model_water_balance_update_R . . . 141
save_model_outputs . . . 142
select_output_vars . . . 143

Index **144**

`add_missing_params` *Add Missing Parameters from Reference INI Files*

Description

This function adds missing parameters to a given list or data frame `x` by referencing parameter definitions from INI files associated with a specified type (e.g., crop, soil, irrigation, run). The function loads the appropriate INI file(s) based on the `types` argument and fills in any missing parameters in `x` with default values and types as specified in the INI files.

Usage

```
add_missing_params(x, types, ..., cfg = load_config())
```

Arguments

x	A list or data frame containing parameters. Missing parameters will be added based on the reference.
types	A character string specifying the type of parameters to reference. Must be one of: "crop", "soil", "irrigation", "run".
...	Additional arguments (currently unused).
cfg	A configuration object, typically loaded via `load_config()`, containing project/package information.

Details

The function searches for INI files using the per-type sources declared in `cfg$ini_sources`. Each source can be a package name or a directory path, and sources are searched in order. It selects the relevant INI file(s) based on the `types` argument and the contents of `x` (e.g., `x$crop` or `x$soil`). Parameter types are inferred from the INI file and used to cast values appropriately (e.g., integer, numeric, logical, date).

Value

A list with the same contents as `x`, but with any missing parameters added and cast to the appropriate types.

`calc_alpha` *Calculate temperature response alpha parameter*

Description

Generic and method-specific functions to assign the temperature response parameter `alpha` based on thermal time (`TT`) and a threshold `TTmax`. For each `TT` value, `alpha1` is used when `TT <= TTmax` and `alpha2` otherwise. Methods are provided for numeric vectors, data.frames and matrices.

Usage

```
calc_alpha(x, ...)

## S3 method for class 'numeric'
calc_alpha(x, TTmax, alpha1, alpha2, TT = x, ...)

## S3 method for class 'data.frame'
calc_alpha(
  x,
  TT = x[iday, TT_name, drop = TRUE],
  TTmax,
  alpha1,
  alpha2,
  iday = seq.int(nrow(x)),
```

```
  TT_name = "TT_sowing",
  alpha_name = "alpha",
  ...
)

## S3 method for class 'matrix'
calc_alpha(
  x,
  TT = x[iday, TT_name],
  TTmax,
  alpha1,
  alpha2,
  iday = seq.int(nrow(x)),
  alpha_name = "alpha",
  TT_name = "TT_sowing",
  ...
)
```

Arguments

x	Input object. Can be numeric, matrix or data.frame.
...	Additional arguments passed to methods.
TTmax	numeric Thermal-time threshold used to switch between alpha1 and alpha2 (same units as TT).
alpha1	numeric Alpha value used when TT <= TTmax. Must be scalar.
alpha2	numeric Alpha value used when TT > TTmax. Must be scalar.
TT	numeric Thermal time (e.g. degree-days, °C.days). For the numeric method, defaults to x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
TT_name	character Layer or column name containing TT values.
alpha_name	character Layer or column name used to store calculated alpha values.

Details

The numeric method applies a simple piecewise-constant mapping: alpha1 is used when TT <= TTmax and alpha2 otherwise. Methods for matrix and data.frame inputs extract TT values from the specified column or layer, compute alpha for the selected time indices (iday), and store the result in alpha_name. If the target column or layer does not exist, it is created and initialised with NA_real_.

Value

For numeric input: a numeric vector of alpha values. For matrix/data.frame input: an object of the same class as x with a column alpha_name created or updated for the specified indices (iday).

calc_B *Calculate Biomass Accumulation (B)*

Description

Generic function and methods to calculate biomass accumulation (B) based on photosynthetically active radiation (PAR), intercepted radiation (IR), radiation use efficiency (RUE), and stress factors. Supports various input types: numeric, matrix and data.frame.

Usage

```
calc_B(x = NULL, ...)

## S3 method for class '`NULL`'
calc_B(x = NULL, B0, ...)

## S3 method for class 'numeric'
calc_B(
  x,
  PAR,
  IR,
  RUE,
  stress_B,
  stress_noci_B,
  iday_crop_dev = rep(1, ifelse(length(x) == 1, length(PAR), length(x))),
  B0 = x,
  ...
)

## S3 method for class 'matrix'
calc_B(
  x,
  B0,
  PAR = x[iday, PAR_name],
  IR = x[iday, IR_name],
  RUE,
  stress_B = x[iday, stress_B_name],
  stress_noci_B = x[iday, stress_noci_B_name],
  iday = seq.int(nrow(x)),
  stress_B_name = "stress_B",
  stress_noci_B_name = "stress_noci_B",
  PAR_name = "PAR",
  IR_name = "IR",
  B_name = "B",
  ...
)
```

```
## S3 method for class 'data.frame'
calc_B(
  x,
  PAR = x$PAR,
  B0,
  IR = x$IR,
  RUE = x$RUE,
  stress_B = x$stress_B,
  stress_noci_B = x$stress_noci_B,
  iday = seq.int(nrow(x)),
  B_name = "B",
  ...
)
```

Arguments

x	Input object. Can be NULL, numeric, matrix, or data.frame.
...	Additional arguments passed to methods.
B0	numeric Initial biomass value. If NULL or NA, defaults to 0 ($t\ ha^{-1}$).
PAR	numeric vector or matrix of photosynthetically active radiation values ($J\ cm^{-2}\ day^{-1}$).
IR	numeric vector or matrix of intercepted radiation fractions (dimensionless, $0 - 1$).
RUE	numeric Radiation use efficiency ($g\ MJ^{-1}$ of PAR).
stress_B	numeric vector or matrix of stress coefficients affecting biomass accumulation.
stress_noci_B	numeric vector or matrix of exponents for stress coefficients.
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
stress_B_name	character Name of the column or array for stress_B (default: "stress_B").
stress_noci_B_name	character Name of the column or array for stress_noci_B (default: "stress_noci_B").
PAR_name	character Name of the column or array for PAR (default: "PAR").
IR_name	character Name of the column or array for IR (default: "IR").
B_name	character Name of the column or array for output biomass (default: "B").

Details

The function computes biomass incrementally for each time step, using the formula:

$$B_i = B0_i + 0.0001 \times PAR_i \times IR_i \times RUE_i \times (stress_{B,i})^{stress_noci_B}$$

Here PAR is expected in $J\ cm^{-2}\ day^{-1}$ and IR is a dimensionless intercepted fraction. The factor 0.0001 converts $J\ cm^{-2}$ to $MJ\ m^{-2}$ and then $g\ m^{-2}$ to $t\ ha^{-1}$, so B remains expressed in $t\ ha^{-1}$. If i == 0, the output is set to NA.

For matrix, data.frame methods, the function iterates over time steps, updating the biomass column or array.

Value

Returns a numeric vector, matrix or data.frame with the calculated biomass values.

See Also

- `calc_B_root()`,
- `calc_TT_sowing()`

`calc_B_root` *Calculate biomass root accumulation (B_root)*

Description

Generic and method-specific functions to compute root biomass using thermal-time thresholds. The numeric method updates root biomass for a single time step. Matrix and data.frame methods apply the same rule sequentially over a time series.

Usage

```
calc_B_root(x = NULL, ...)

## S3 method for class 'numeric'
calc_B_root(x, B0, B_root0, TT_sowing, TT_B_root5, TT_B_root70, ...)

## S3 method for class 'matrix'
calc_B_root(
  x,
  B0,
  B_root0,
  TT_sowing = x[iday, TT_sowing_name],
  TT_B_root5,
  TT_B_root70,
  iday = seq.int(nrow(x)),
  B_name = "B",
  B_root_name = "B_root",
  TT_sowing_name = "TT_sowing",
  ...
)

## S3 method for class 'data.frame'
calc_B_root(
  x,
  B0,
  B_root0,
  TT_sowing = x[iday, TT_sowing_name],
  TT_B_root5,
```

```
  TT_B_root70,
  iday = seq.int(nrow(x)),
  B_name = "B",
  B_root_name = "B_root",
  TT_sowing_name = "TT_sowing",
  ...
)
```

Arguments

x	Input object (numeric value, matrix or data.frame).
...	Additional arguments passed to methods.
B0	numeric Previous total biomass used for sequential updates ($t\ ha^{-1}$).
B_root0	numeric Previous root biomass used for sequential updates ($t\ ha^{-1}$).
TT_sowing	numeric Thermal time since sowing (scalar or per-time values) (°C.days).
TT_B_root5	numeric Thermal-time threshold for 5% root fraction (°C.days).
TT_B_root70	numeric Thermal-time threshold for 70% root fraction (°C.days).
iday	integer Indices of time steps to process (for time-series methods).
B_name	character Name of the total biomass column or layer (default "B").
B_root_name	character Name of the root biomass column or layer (default "B_root").
TT_sowing_name	character Name of the thermal-time column or layer (default "TT_sowing").

Details

Root biomass (B_root) is updated from total biomass (B) and thermal time since sowing (TT_sowing) using two thresholds TT_B_root5 and TT_B_root70. A piecewise linear rule controls the fraction of the biomass increment allocated to roots:

- if $TT_{sowing} < TT_{B_root5}$: B_{root} increases by 5\
- if $TT_{B_root5} \leq TT_{sowing} \leq TT_{B_root70}$: B_{root} increases linearly from 5\ increment in B;
- if $TT_{sowing} > TT_{B_root70}$: B_{root} increases by 70\

The update rule can be written as:

$$B_{root} = B_{root0} + f(TT_{sowing}) \times (B - B_0)$$

where $f(TT_{sowing})$ is the allocation fraction defined above.

For matrix and data.frame methods, B0 and B_root0 are taken from the previous time step in the series, so that root biomass is updated sequentially along the time dimension.

Value

For numeric input: a numeric scalar (B_root). For matrix/data.frame input: the object x with a column or layer named B_root_name created or updated for the selected indices (iday).

See Also

- calc_B()
- calc_TT_sowing()

calc_Bp *Calculate potential biomass production (Bp)*

Description

Generic and method-specific functions to calculate potential biomass production (Bp).

Usage

```
calc_Bp(x, ...)

## S3 method for class 'model_init'
calc_Bp(x, ...)

## S3 method for class 'matrix'
calc_Bp(x, ...)
```

Arguments

x	Input object. Can be `model_init`.
...	Additional arguments passed to methods.

Value

Updated `model_init` object with potential biomass production (Bp) values.

See Also

- `calc_IR()`
- `calc_PAR()`
- `calc_B()`

calc_Bp_root *Calculate potential biomass root accumulation (Bp_root)*

Description

Generic function and methods to compute the root component of the biomass associated with the "Bp" pool.

Usage

```
calc_Bp_root(x = NULL, ...)

## S3 method for class 'matrix'
calc_Bp_root(x, TT_B_root5, TT_B_root70, ...)
```

Arguments

x	Input data. For the matrix method, a numeric matrix (rows/columns as required by calc_B_root).
...	Additional arguments passed to calc_B_root (for example other control parameters or names). These are forwarded by the methods.
TT_B_root5	integer Temperature threshold (or other threshold) used to determine the 5% reference for root allocation; passed to calc_B_root (°C days).
TT_B_root70	integer Temperature threshold (or other threshold) used to determine the 70% reference for root allocation; passed to calc_B_root (°C days).

Details

The generic calc_Bp_root dispatches on the class of x. Methods call calc_B_root(...) with B_name = "Bp" and B_root_name = "Bp_root" so that the computed root variable is named "Bp_root" in the returned object.

Value

An object of the same class as x with the computed root variable ("Bp_root") added or returned according to the underlying calc_B_root behavior.

See Also

- `calc_B_root()`

---

calc_case *Calculate model water balance case*

Description

Generic and method-specific functions to calculate model water balance cases.

Usage

```
calc_case(x, ...)

## S3 method for class 'numeric'
calc_case(x, zES, zroot = x, ...)

## S3 method for class 'matrix'
calc_case(
  x,
  zroot = x[iday, zroot_name],
  zES,
  iday = seq.int(nrow(x)),
  zroot_name = "zroot",
```

```
  case_name = "case",
  ...
)
```

Arguments

x	Input object. Can be numeric or matrix.
...	Additional arguments passed to methods.
zES	numeric Evaporation reservoir depth.
zroot	numeric Crop root depth (default: x for numeric method, or column for matrix method) (m).
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
zroot_name	character Layer or column name containing root depth values (default: "zroot").
case_name	character Layer or column name to store calculated case.

---

calc_Cp *Calculate canopy partition coefficient (Cp)*

Description

Generic function to calculate the Cp value, which is used in crop coefficient calculations. This function dispatches methods based on the class of the input x.

Usage

```
calc_Cp(x, ...)

## S3 method for class 'numeric'
calc_Cp(
  x,
  c1,
  c2,
  Kc,
  Ksol = NULL,
  iday_crop_dev = rep(1, length(x)),
  LAI = x,
  ...
)
```

Arguments

x	Input object.
...	Additional arguments passed to methods.
c1	numeric Coefficient related to the crop.

c2	numeric Coefficient related to the crop.
Kc	numeric Crop coefficient.
Ksol	numeric Soil coefficient, optional.
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
LAI	numeric Leaf Area Index, defaults to x.

Details

The Cp value is calculated based on the provided parameters. For numeric inputs, the calculation uses the formula :

$$Cp = c_1 - e^{-c_2 \times LAI}$$

If Ksol is provided and Cp > 0 and Kc < Ksol, then Cp is adjusted as:

$$Cp = \frac{Kc \times Cp}{Ksol}$$

Value

Numeric value representing the Cp calculation.

See Also

- calc_Kc()
- calc_Ksol()
- calc_LAI()

Examples

```
calc_Cp(3, c1 = 1.2, c2 = 0.5, Kc = 0.8, Ksol = 1.0)
```

---

calc_crac *Calculate root growth parameter (crac)*

Description

Generic and method-specific functions to calculate root growth crac parameters.

Usage

```
calc_crac(x, ...)

## S3 method for class 'numeric'
calc_crac(x, ks_root, seuil, iday_crop_dev = rep(1, length(x)), taum = x, ...)

## S3 method for class 'matrix'
calc_crac(
  x,
  taum = x[iday, taum_name],
  ks_root = x[iday - 1, "ks_root"],
  seuil,
  iday = seq.int(nrow(x)),
  taum_name = "taum",
  crac_name = "crac",
  ...
)

## S3 method for class 'model_output'
calc_crac(x, crop, iday, ...)
```

Arguments

x	Input object. Can be numeric, matrix, or model_output.
...	Additional arguments passed to methods.
ks_root	numeric Root growth parameter.
seuil	numeric Root growth parameter.
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
taum	numeric Root growth parameter.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
taum_name	character Layer or column name containing taum values.
crac_name	character Layer or column name to store calculated crac values.
crop	list Crop parameters

Details

The calc_crac function calculates the root growth "crac" parameter. Different methods handle numeric vectors, matrices, and model outputs. For numeric inputs, crac is computed as:

$$crac = taum * ks_root + seuil$$

where crac is set to 0 if there is no crop development on the given day/layer (i.e., when iday_crop_dev is 0).

Value

An object of the same class as x, with the calculated "crac" values added.

See Also

- calc_ks_root()

---

calc_d *Calculate drainage (d)*

Description

Generic and method-specific functions to calculate drainage from multiple possible inputs (gravity water reserve, water content, etc.)

Usage

```
calc_d(x, from = "Rgravity", ...)

calc_d_from_Rgravity(x, ...)

## S3 method for class 'numeric'
calc_d_from_Rgravity(x, Kd, Rgravity = x, ...)

## S3 method for class 'matrix'
calc_d_from_Rgravity(
  x,
  Rgravity = x[iday, Rgravity_name],
  Kd,
  iday = seq.int(nrow(x)),
  Rgravity_name = "Rgravity",
  d_name = "d",
  ...
)
```

Arguments

x	Input object. Can be numeric, matrix, or data.frame.
from	character Input type. Currently only "Rgravity" is implemented.
...	Additional arguments passed to methods.
Kd	numeric Hydraulic conductivity.
Rgravity	numeric Gravity water reserve. Required if from is "Rgravity".
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
Rgravity_name	character Column or layer name for gravity water reserve in x.
d_name	character Column or layer name to store calculated drainage.

Details

Currently, only drainage calculation from gravity water reserve is implemented. Drainage is computed as the minimum between available gravity water reserve and the maximum possible outflow (hydraulic conductivity Kd in mm/h multiplied by 24 h).

Drainage values are compute by the next formula:

$$d = \min(Rgravity, Kd \times 24)$$

Value

An object of the same class as `x` with an additional column or layer for calculated drainage.

See Also

- `calc_Rgravity()`

`calc_degred` *Calculate "degred" root growth calculation input*

Description

Generic and method-specific functions to calculate "degred" root growth calculation input.

Usage

```
calc_degred(x, ...)

## S3 method for class 'Date'
calc_degred(x, degred, vroot, dates = x, ...)

## S3 method for class 'matrix'
calc_degred(
  x,
  dates = x[iday, dates_name],
  degred,
  vroot,
  iday = seq.int(nrow(x)),
  dates_name = "dates",
  degred_name = "degred",
  ...
)

## S3 method for class 'model_init'
calc_degred(x, crop, dates, iday_sowing, iday_harvest, ...)
```

Arguments

x	Input object. Can be numeric, matrix, or model_init
...	Additional arguments passed to methods.
degred	numeric Base degred value
vroot	numeric Root growth parameter
dates	base::Date Simulation dates
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
dates_name	character Column or layer name containing dates
degred_name	character Column or layer name to store calculated degred
crop	list Crop parameters
iday_sowing	integer Day of year corresponding to sowing date
iday_harvest	integer Day of year corresponding to harvest date

Value

An object of the same class as x, with the calculated "degred" values added.

---

calc_dose *Generic function to calculate irrigation (dose)*

Description

Generic dispatcher to calculate an irrigation dose based on soil water content, soil properties (field capacity and wilting point) and a target soil moisture level. The default numeric method assumes a two-layer soil profile and uses water stored in each layer (R1, R2) and their depths (z1, z2).

Usage

```
calc_dose(x = NULL, ...)

## S3 method for class '`NULL`'
calc_dose(x, R1, ...)

## S3 method for class 'numeric'
calc_dose(
  x,
  R1 = x,
  R2,
  z1,
  z2,
  theta_fc,
  theta_wp,
```

```
  RU_target = NA,
  RU_units = 2,
  THETA_target = NA,
  ...
)
```

Arguments

x	An object for which the irrigation dose should be calculated. The method dispatched depends on the class of x.
...	Additional arguments passed to specific methods.
R1	numeric Soil water content in the first layer or initial value (mm).
R2	numeric Soil water content in the second layer (mm).
z1	numeric Depth of the first soil layer (m).
z2	numeric Depth of the second soil layer (m).
theta_fc	numeric Soil water content at field capacity ($m^3\ m^{-3}$).
theta_wp	numeric Soil water content at wilting point ($m^3\ m^{-3}$).
RU_target	numeric Target readily available water (RU), either in mm or as a fraction/percent of the maximum RU, depending on RU_units.
RU_units	integer Units for RU_target: 1 for mm, 2 for fraction/percent (0-1 or 0-100; default).
THETA_target	numeric Target volumetric soil moisture ($m^3\ m^{-3}$).

Details

The numeric method converts water stored in the two soil layers (R1, R2, in mm) and their depths (z1, z2, in m) into an average volumetric soil water content over the total depth (Hz = z1 + z2). Readily available water (RU) and its maximum (RUmax) are computed relative to the wilting point and field capacity.

The target can be specified either as a volumetric soil moisture (THETA_target) or as a target readily available water (RU_target), but not both at the same time. If both are provided, the function throws an error. If neither is provided, a dose of 0 is returned.

When RU_target is given in mm (RU_units = 1), it is clamped to the range [0, RUmax]. When RU_units = 2, RU_target is interpreted as a fraction or percent of RUmax (values in [0,1] or [0,100] are accepted and normalized to [0,1]).

The computed dose is the water depth (mm) required to raise the current soil moisture to the target, limited by optional minimum and maximum dose constraints (dose_min, dose_max) and never negative.

Value

A numeric value representing the irrigation dose (mm), as determined by the method for the class of x.

See Also

- calc_theta_fc()
- calc_theta_wp()
- calc_RU()

---

calc_dtheta_RU *Calculate relative soil moisture above field capacity (dtheta_RU)*

Description

Generic and method-specific functions to calculate a relative soil moisture term (dtheta_RU) based on volumetric water content (theta), field capacity (theta_fc) and wilting point (theta_wp). In the numeric method, dtheta_RU is the amount by which theta exceeds field capacity, capped by the total range between field capacity and wilting point.

Usage

```
calc_dtheta_RU(x, ...)

## S3 method for class 'numeric'
calc_dtheta_RU(x, theta_fc, theta_wp, theta = x, ...)

## S3 method for class 'matrix'
calc_dtheta_RU(
  x,
  theta = x[iday, theta_name],
  theta_fc,
  theta_wp,
  iday = seq.int(nrow(x)),
  theta_name = "theta",
  dtheta_RU_name = "dtheta_RU",
  ...
)
```

Arguments

x	Input object. Can be a numeric vector, matrix, or data.frame (for from = "theta").
...	Additional arguments passed to specific methods.
theta_fc	numeric Volumetric soil water content at field capacity ($m^3\ m^{-3}$).
theta_wp	numeric Volumetric soil water content at wilting point ($m^3\ m^{-3}$).
theta	numeric Volumetric soil water content ($m^3\ m^{-3}$). For the numeric method, defaults to x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).

`theta_name` character Column or layer name containing `theta` values (default `"theta"`).

`dtheta_RU_name` character Column or layer name used to store the calculated relative soil moisture values.

Details

The numeric implementation computes:

$$d\theta_{RU} = \max\left(\min\left(\theta - \theta_{FC}, \theta_{FC} - \theta_{WP}\right), 0\right)$$

i.e. the excess water content above field capacity, limited to the maximum available water between field capacity and wilting point and not allowed to be negative.

If you instead want the classical available water between wilting point and field capacity, you can replace the core formula with:

$$\max\left(\min\left(\theta - \theta_{WP}, \theta_{FC} - \theta_{WP}\right), 0\right)$$

Value

For numeric input: a numeric vector of `dtheta_RU` values. For matrix input: a matrix with a column `dtheta_RU_name` created or updated for the specified indices (`iday`).

See Also

- `calc_RU()`
- `calc_dtheta_RU_WP()`
- `calc_theta()`

`calc_dzroot` *Calculate root depth increment (dzroot)*

Description

Generic and method-specific functions to calculate the increment in root depth (`dzroot`) for a given time step, based on crop state and empirical parameters. Typically used in crop growth models to simulate root development over time.

Usage

```
calc_dzroot(x, ...)

## S3 method for class 'numeric'
calc_dzroot(
  x,
  dcrac,
  dzrootp,
  iday_crop_dev = rep(1, length(x)),
```

```
  taum = x,
  ...
)

## S3 method for class 'matrix'
calc_dzroot(
  x,
  taum = x[iday, "taum"],
  dcrac = x[iday, "dcrac"],
  dzrootp = x[iday, "dzrootp"],
  iday = seq.int(nrow(x)),
  dzroot_name = "dzroot",
  ...
)

## S3 method for class 'model_output'
calc_dzroot(x, crop, iday, ...)
```

Arguments

x	Input object. Can be numeric, a matrix or a model_output object.
...	Additional arguments passed to methods.
dcrac	numeric Base increment of root depth (same units as dzroot).
dzrootp	numeric Potential increment of root depth (same units as dzroot).
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
taum	numeric Dimensionless development variable (e.g. relative thermal time or phenological index). For the numeric method, defaults to x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
dzroot_name	character Column or layer name used to store calculated dzroot values (default "dzroot").
crop	list Crop parameter list, used by the model_output method.

Details

The numeric method applies a simple rule to each time step:

- if iday_crop_dev[i] == 0, no root growth is computed and NA_real_ is returned for that step;
- if taum[i] <= 0.25, the root depth increment is taken as dcrac[i];
- otherwise, the increment is limited by the potential value dzrootp[i] and never exceeds dcrac[i].

In practice, this is implemented as a piecewise expression using taum, dcrac and dzrootp, with the result returned as a numeric vector. Matrix and model_output methods wrap this numeric logic and write dzroot into the appropriate column or layer.

Value

For numeric input: a numeric vector of `dzroot` values. For matrix and model_output input: an object of the same class as `x` with a column or layer named `dzroot_name` (default `"dzroot"`) created or updated for the specified indices.

See Also

- `calc_taum()`
- `calc_crac()`
- `calc_dzrootp()`

`calc_dzrootp` *Calculate the change in root depth (dzrootp)*

Description

Generic and method-specific functions to compute the change in root depth (`dzrootp`) for different input types. The generic dispatcher selects a method based on the class of `x`.

Usage

```
calc_dzrootp(x, ...)

## S3 method for class 'numeric'
calc_dzrootp(x, zrootp_init, zrootp = x, ...)

## S3 method for class 'matrix'
calc_dzrootp(
  x,
  zrootp = x[iday, "zrootp"],
  zrootp_init,
  iday = seq.int(nrow(x)),
  dzrootp_name = "dzrootp",
  ...
)
```

Arguments

`x`	Input object. Can be a numeric value or matrix.
`...`	Additional arguments passed to methods.
`zrootp_init`	numeric Initial or previous root depth (same units as `zrootp`). For matrix inputs, a single value is interpreted as the root depth at the previous time step and used to derive a lagged series.
`zrootp`	numeric Current root depth (same units as `zrootp_init`). For numeric input, defaults to `x`.

iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
dzrootp_name	character Name of the column used to store dzrootp values (default "dzrootp").

Details

Methods:

- numeric: computes dzrootp as the simple difference zrootp - zrootp_init.
- matrix: computes dzrootp row-wise for the indices iday, ensuring that a column named dzrootp_name exists (it is created if missing). When zrootp_init is a single value, it is treated as the root depth at the previous time step and expanded to a lagged vector using c(zrootp_init, zrootp[-length(zrootp)]).

This helper is typically used in root growth routines to express root extension as a per-step increment rather than an absolute depth.

Value

For numeric input: a numeric value (or vector) of dzrootp. For matrix input: the matrix x with a column dzrootp_name created or updated for the selected indices iday.

See Also

- calc_taum()
- calc_crac()
- calc_dzroot()

Examples

```
calc_dzrootp(5, zrootp_init = 2)

mat <- matrix(c(1, 2, 3, 4), ncol = 2)
colnames(mat) <- c("zrootp", "other")
calc_dzrootp(mat, zrootp_init = 1)
```

---

calc_ES *Calculate soil evaporation (ES)*

Description

Generic S3 interface to compute soil evaporation (ES). This function dispatches to class-specific implementations (for example calc_ES.numeric) that perform the actual calculation.

Usage

```
calc_ES(x, ...)

## S3 method for class 'numeric'
calc_ES(x, ES0, KES, Rmin = NULL, R = x, ...)

## S3 method for class 'matrix'
calc_ES(
  x,
  R = x[iday, R_name],
  ES0 = x[iday, ES0_name],
  KES = x[iday, KES_name],
  Rmin = x[iday, Rmin_name],
  iday = seq.int(nrow(x)),
  R_name = "R",
  ES0_name = "ES0",
  KES_name = "KES",
  Rmin_name = "Rmin",
  ES_name = "ES",
  ...
)
```

Arguments

x	Object for method dispatch. Currently only numeric vectors and matrices are supported.
...	Additional arguments passed to specific methods.
ES0	numeric Potential soil evaporation rate ($mm\ day^{-1}$).
KES	numeric Soil evaporation coefficient.
Rmin	numeric Minimum residual soil water (mm) that must remain after evaporation. Can be scalar or vector; if NULL, no residual constraint is applied.
R	numeric Available water for evaporation (mm). For the numeric method, defaults to x.
iday	integer Indices of days (rows) to process. For matrix input, defaults to all rows.
R_name	character Name of the column used for R (default "R").
ES0_name	character Name of the column used for ES0 (default "ES0").
KES_name	character Name of the column used for KES (default "KES").
Rmin_name	character Name of the column used for Rmin (default "Rmin").
ES_name	character Name of the output column for evaporation ("ES" by default).

Details

For numeric input, soil evaporation is computed as:

$$ES = \min(R, ES0 \times KES)$$

i.e. evaporation cannot exceed either the potential rate (ES0 * KES) or the available water R. When a residual Rmin is provided, the result is adjusted so that:

$$R - ES \geq Rmin$$

for each element, i.e. evaporation is reduced where necessary so as not to deplete soil water below Rmin. Matrix input methods extract the required columns by name for the selected rows (iday) and write the result to a column named ES_name, which is created if missing.

Value

For numeric input: a numeric vector of soil evaporation values (ES, $mm\ day^{-1}$). For matrix input: the matrix x with a column ES_name created or updated for the selected rows.

---

calc_ES0 *Calculate potential soil evaporation (ES0)*

Description

Generic and method-specific functions to calculate potential soil evaporation (ES0), representing the atmospheric demand for evaporation from the soil surface given reference evapotranspiration and partitioning coefficients.

Usage

```
calc_ES0(x, ...)

## S3 method for class 'numeric'
calc_ES0(x, Kc, Cp, Ksol, ET0 = x, ...)

## S3 method for class 'matrix'
calc_ES0(
  x,
  ET0 = x[iday, ET0_name],
  Kc = x[iday, Kc_name],
  Cp = x[iday, Cp_name],
  Ksol,
  iday = seq.int(nrow(x)),
  ET0_name = "ET0",
  Kc_name = "Kc",
  Cp_name = "Cp",
  ES0_name = "ES0",
  ...
)
```

Arguments

x	Input object. Can be numeric or matrix.
...	Additional arguments passed to methods.
Kc	numeric Crop coefficient.
Cp	numeric Canopy partition coefficient, representing the fraction of ET0 attributed to the canopy. Values are internally truncated to be at least zero.
Ksol	numeric Soil evaporation scaling or limiting factor, used together with Kc to bound the soil evaporation potential.
ET0	numeric Reference evapotranspiration ($mm\ day^{-1}$).
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
ET0_name	character Column or layer name containing ET0 values (default "ET0").
Kc_name	character Column or layer name containing Kc values (default "Kc").
Cp_name	character Column or layer name containing Cp values (default "Cp").
ES0_name	character Column or layer name used to store calculated ES0 values (default "ES0").

Details

The numeric implementation computes potential soil evaporation as:

$$ES0_i = ET0_i \times \max(K_{c,i}, K_{sol}) \times (1 - \max(C_{p,i}, 0))$$

i.e. the reference evapotranspiration is scaled by the larger of the crop and soil coefficients, and then reduced by the fraction attributed to the canopy (Cp, truncated at 0). The result is a potential soil evaporation rate expressed in $mm\ day^{-1}$.

For matrix inputs, ET0, Kc and Cp are read from the specified columns for rows indexed by iday, and ES0 is written to the column named ES0_name, which is created if it does not exist.

Value

For numeric input: a numeric vector of potential soil evaporation (ES0, $mm\ day^{-1}$). For matrix input: an object of the same class as x with a column ES0_name created or updated where applicable.

See Also

- calc_ES()

`calc_ET0` *Calculate reference evapotranspiration (ET0)*

Description

Generic and method-specific functions to calculate reference evapotranspiration (`ET0`) from potential evapotranspiration (`ETP`) using a specified ratio. Methods are provided for numeric vectors and matrices.

Usage

```
calc_ET0(x, ...)

## S3 method for class 'numeric'
calc_ET0(x, ratio_ET0_ETP, ETP = x, ...)

## S3 method for class 'matrix'
calc_ET0(
  x,
  ETP = x[iday, "ETP"],
  ratio_ET0_ETP = 1,
  iday = seq.int(nrow(x)),
  ET0_name = "ET0",
  ...
)
```

Arguments

`x`	Input data. Can be a numeric vector or matrix.
`...`	Further arguments passed to or from other methods.
`ratio_ET0_ETP`	numeric Scalar ratio applied to `ETP` to obtain `ET0`.
`ETP`	numeric Potential evapotranspiration ($mm\ day^{-1}$). For numeric and matrix methods, defaults to `x` or a column named `"ETP"`.
`iday`	integer Indices of days or time steps to process. Defaults to all rows (for matrix).
`ET0_name`	character Name of the ET0 column or variable to create or overwrite (default `"ET0"`).

Details

For numeric and matrix inputs, the function applies a simple scaling:

$$ET0 = ratio_ET0/ETP \times ETP$$

For matrices, `ETP` is taken from the column named `"ETP"` by default, and the result is written to a column named `ET0_name`, which is created if missing.

For `model_init` objects, the ratio `ratio_ET0_ETP` is first computed via `calc_ET0_mulch_effect` using soil and irrigation parameters (e.g. mulch and irrigation method), then the appropriate next method is called to perform the actual ET0 computation.

Value

For numeric input: a numeric vector of ET0 values ($mm\ day^{-1}$). For matrix input: a matrix with an added or updated ET0 column. For model_init input: a `model_init` object with ET0 calculated.

Examples

```
# Numeric method
etp <- c(2.1, 2.5, 2.3)
calc_ET0(etp, ratio_ET0_ETP = 0.95)

# Matrix method
m <- matrix(c(1:6, rep(NA, 3)), ncol = 3)
colnames(m) <- c("ETP", "foo", "bar")
calc_ET0(m, ratio_ET0_ETP = 1.0, ET0_name = "ET0")
```

`calc_ET0_mulch_effect` *Calculate mulch effect on reference evapotranspiration (ET0)*

Description

Generic S3 function to compute a multiplicative factor representing the effect of mulch on reference evapotranspiration (`ET0`). The returned value is intended to multiply `ET0` to account for mulch presence or specific simulation conditions.

Usage

```
calc_ET0_mulch_effect(x = NULL, ...)

## S3 method for class '`NULL`'
calc_ET0_mulch_effect(x = NULL, mulch, ...)

## S3 method for class 'numeric'
calc_ET0_mulch_effect(x, gge, mulch = x, ...)
```

Arguments

`x`	Input object. Used for method dispatch. Currently only `numeric` and `NULL` are supported. For the numeric method, `x` can act as the mulch level when `mulch` is not explicitly provided.
`...`	Additional arguments passed to or from other methods.
`mulch`	numeric Mulch level or mulching factor (dimensionless or in arbitrary model units). For the numeric method, defaults to `x`.
`gge`	numeric Indicator of the irrigation / simulation method. Commonly: • `999`: special simulation condition (GGS); • `1`: regular method with a proportional mulch effect; • `0`: ASP method with no mulch effect.

Details

The numeric method returns a scalar multiplicative factor for ET0 based on gge and mulch:

- if gge == 999 and mulch == 1000, a fixed factor 0.1 is returned (special GGS simulation case);
- if gge == 1, a proportional reduction is used:

$$f = \frac{1}{1 + 0.1 \times mulch/1000}$$

- if gge == 0, mulch has no effect and the factor is 1.

Any other value of gge triggers an error. This function is intended to be applied at the time-step level; both gge and mulch are treated as scalar values.

Value

A numeric scalar representing the multiplicative factor to apply to ET0 to account for mulch effects.

seealso:

- calc_ET0()

Examples

```
# Using mulch as an explicit argument
calc_ET0_mulch_effect(5, gge = 1, mulch = 50)

# Using x as mulch when mulch is omitted
calc_ET0_mulch_effect(50, gge = 1)
```

---

calc_ETM *Calculate maximum crop evapotranspiration (ETM)*

Description

Generic and method-specific functions to calculate maximum evapotranspiration (ETM), representing the crop water demand as the sum of soil evaporation (ES0) and potential transpiration (TP0).

Usage

```
calc_ETM(x = NULL, ...)

## S3 method for class '`NULL`'
calc_ETM(x, ES0, ...)

## S3 method for class 'numeric'
calc_ETM(x = NULL, TP0, ES0 = x, ...)
```

```
## S3 method for class 'matrix'
calc_ETM(
  x,
  ES0 = x[iday, ES0_name],
  TP0 = x[iday, TP0_name],
  iday = seq.int(nrow(x)),
  ES0_name = "ES0",
  TP0_name = "TP0",
  ETM_name = "ETM",
  ...
)
```

Arguments

x	Input object. Can be NULL, numeric or matrix.
...	Additional arguments passed to methods.
ES0	numeric Reference or potential soil evaporation ($mm\ day^{-1}$). For the numeric method, defaults to x.
TP0	numeric Potential transpiration ($mm\ day^{-1}$).
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
ES0_name	character Name of the column in the matrix containing ES0 (default "ES0").
TP0_name	character Name of the column in the matrix containing TP0 (default "TP0").
ETM_name	character Name of the column in the matrix used to store ETM (default "ETM").

Details

The calc_ETM function is a generic S3 interface with methods for different input types:

- NULL: delegates to the next method using the ES0 argument as input;
- numeric: computes ETM as the sum $ES_{0,i} + TP_{0,i}$;
- matrix: ensures an ETM column exists, then fills it for the selected rows as $ES_{0,i} + TP_{0,i}$.

In all cases, ETM_{i} is expressed in $mm\ day^{-1}$, consistent with ES0_{i} and TP0_{i}.

Value

For numeric input: a numeric vector of ETM values ($mm\ day^{-1}$). For matrix input: the matrix x with a column ETM_name created or updated. For NULL input: dispatches to the numeric method using ES0.

See Also

- calc_ET0()
- calc_TP0()

Examples

```
# Numeric example
calc_ETM(ES0 = 2, TP0 = 1)

# Matrix example
mat <- matrix(c(2, 1, 3, 2), ncol = 2)
colnames(mat) <- c("ES0", "TP0")
calc_ETM(mat)
```

`calc_ETR` *Calculate actual evapotranspiration (ETR)*

Description

Generic and method-specific functions to calculate actual evapotranspiration (`ETR`) as the sum of soil evaporation (`ES`) and crop transpiration (`TP`).

Usage

```
calc_ETR(x, ...)

## S3 method for class 'numeric'
calc_ETR(x, TP, ES = x, ...)

## S3 method for class 'matrix'
calc_ETR(
  x,
  ES = x[iday, ES_name],
  TP = x[iday, TP_name],
  iday = seq.int(nrow(x)),
  ES_name = "ES",
  TP_name = "TP",
  ETR_name = "ETR",
  ...
)
```

Arguments

`x`	Input object. Can be `numeric` or `matrix`.
`...`	Additional arguments passed to methods.
`TP`	numeric Crop transpiration ($mm\ day^{-1}$).
`ES`	numeric Soil evaporation ($mm\ day^{-1}$). For the numeric method, defaults to `x`.
`iday`	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
`ES_name`	character Layer or column name containing ES values (default `"ES"`).

TP_name	character Layer or column name containing TP values (default "TP").
ETR_name	character Layer or column name used to store ETR values (default "ETR").

Details

The numeric implementation combines soil evaporation and crop transpiration as:

$$ETR_i = ES_i + \max(TP_i, 0)$$

i.e. negative transpiration values, if present, are treated as zero. For matrix inputs, ES and TP are read from the specified columns for rows indexed by iday, and ETR is written to a column named ETR_name, which is created if missing.

All fluxes are assumed to be expressed in $mm\ day^{-1}$ and consistent across ES, TP and ETR.

Value

For numeric input: a numeric vector of actual evapotranspiration (ETR, $mm\ day^{-1}$). For matrix input: the matrix x with a column ETR_name created or updated.

See Also

- calc_ES()
- calc_TP()

---

calc_HI *Calculate harvest index (HI)*

Description

Generic and method-specific functions to calculate harvest index (HI), defined as the ratio between harvested (economic) yield and total above-ground biomass.

Usage

```
calc_HI(x, ...)

## S3 method for class 'numeric'
calc_HI(x, Bp, HIp, iday_crop_dev = rep(1, length(x)), B = x, ...)

## S3 method for class 'matrix'
calc_HI(
  x,
  B = x[iday, B_name],
  Bp = x[iday, Bp_name],
  HIp = x[iday, HIp_name],
  iday = seq.int(nrow(x)),
  B_name = "B",
```

```
  Bp_name = "Bp",
  HIp_name = "HIp",
  HI_name = "HI",
  ...
)
```

Arguments

x	Input object. Can be numeric or matrix.
...	Additional arguments passed to methods.
Bp	numeric Potential or reference biomass ($t\ ha^{-1}$).
HIp	numeric Potential harvest index. Can be scalar or time-varying.
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
B	numeric Total above-ground biomass ($t\ ha^{-1}$). For the numeric method, defaults to x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
B_name	character Column or layer name containing biomass values (default "B").
Bp_name	character Column or layer name containing potential biomass values (default "Bp").
HIp_name	character Column or layer name containing potential harvest index values (default "HIp").
HI_name	character Column or layer name used to store calculated harvest index values (default "HI").

Details

The numeric method computes harvest index for each time step or element where the crop is in development:

$$HI_i = HIp_i \times \frac{B_i}{Bp_i}$$

where HIp_i is either a scalar potential harvest index or a time-varying value. For elements where iday_crop_dev[i] == 0, NA_real_ is returned.

Matrix methods extract biomass and potential biomass (and HIp) from the specified columns for rows indexed by iday, compute the corresponding harvest index values using the numeric method, and store them in a column named HI_name, which is created if it does not exist.

Value

For numeric input: a numeric vector of harvest index values. For matrix input: the matrix x with a column HI_name created or updated.

See Also

- calc_Bp()

Examples

```
# Numeric example
calc_HI(1:100, Bp = seq(1, 200, 2), HIp = 1)

# Matrix example
calc_HI(cbind(B = 1:100, Bp = seq(1, 200, 2), HIp = 1))
```

`calc_HUg` *Calculate grain moisture*

Description

Calculates grain moisture on the standing crop from flowering (R1) using an exponential decrease constrained at physiological maturity (R6).

Usage

```
calc_HUg(x, ...)

## S3 method for class 'numeric'
calc_HUg(x, TT_R1, TT_R6, HUg_R1, HUg_R6, HUg_min, ...)

## S3 method for class 'matrix'
calc_HUg(
  x,
  TT_R1,
  TT_R6,
  HUg_R1,
  HUg_R6,
  HUg_min,
  TT = x[iday, TT_name],
  iday = seq_len(nrow(x)),
  TT_name = "TT_sowing",
  HUg_name = "HUg",
  ...
)
```

Arguments

x	A numeric vector of thermal times, or a matrix containing thermal time.
...	Additional arguments passed to methods.
TT_R1, TT_R6	numeric Thermal times at flowering and physiological maturity (degree-days).
HUg_R1, HUg_R6	numeric Grain moisture at R1 and at R6 (percent, wet basis).
HUg_min	numeric Minimum grain moisture that can be reached while the crop remains standing (percent, wet basis). It is the lower asymptote of field drying, not a harvest target or a post-harvest drying target.

TT numeric Thermal time values used by the matrix method.

iday integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).

TT_name, HUg_name character Input and output column names for matrix inputs.

Details

Grain moisture follows:

$$HUg(TT) = HUg_{min} + (HUg_{R1} - HUg_{min}) \exp(-(TT - TT_{R1})/\tau_{HUg})$$

with:

$$\tau_{HUg} = (TT_{R6} - TT_{R1}) / \log((HUg_{R1} - HUg_{min})/(HUg_{R6} - HUg_{min})).$$

Values before R1 are undefined and returned as NA_real_.

Value

A numeric vector of grain-moisture values, or the input matrix with its grain-moisture column updated.

Examples

```
calc_HUg(
  c(1000, 1200, 1600, 2000),
  TT_R1 = 1200, TT_R6 = 2000,
  HUg_R1 = 85, HUg_R6 = 32, HUg_min = 15
)
```

---

calc_IR *Calculate intercepted radiation fraction (IR)*

Description

Generic and method-specific functions to calculate the fraction of intercepted radiation (IR) based on leaf area index (LAI) and the Beer-Lambert extinction coefficient (k_BL).

Usage

```
calc_IR(x = NULL, ...)

## S3 method for class '`NULL`'
calc_IR(x, LAI, ...)

## S3 method for class 'numeric'
calc_IR(x, k_BL, iday_crop_dev = rep(1, length(x)), LAI = x, ...)
```

```
## S3 method for class 'matrix'
calc_IR(
  x,
  LAI = x[iday, LAI_name],
  iday = seq.int(nrow(x)),
  k_BL,
  LAI_name = "LAI",
  IR_name = "IR",
  ...
)
```

Arguments

x	Input object. Can be NULL, a numeric vector or a matrix.
...	Additional arguments passed to methods.
LAI	numeric Leaf area index (m^2 m^{-2}). For the numeric method, defaults to x. For matrix methods, values are taken from the column or layer named LAI_name.
k_BL	numeric Beer-Lambert extinction coefficient.
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
LAI_name	character Name of the LAI column or variable in matrix objects (default "LAI").
IR_name	character Name of the IR column or variable to create or update (default "IR").

Details

The intercepted fraction is computed from the Beer-Lambert law as:

$$IR_i = 1 - \exp(-k_{BL} \cdot LAI_i)$$

For the numeric method, this expression is applied element-wise. For indices where iday_crop_dev[i] == 0, the result is set to NA_real_ to indicate no crop development.

Matrix methods extract LAI values for the rows or time indices specified by iday, compute intercepted radiation using the numeric method, and store the results in a column or layer named IR_name, which is created if it does not exist.

Value

For numeric input: a numeric vector of intercepted radiation values (dimensionless, between 0 and 1). For matrix input: the object x with a column variable named IR_name created or updated.

See Also

- calc_LAI()

Examples

```
# Numeric usage
LAI <- c(0.5, 1.5, 3.0)
k <- 0.5
calc_IR(LAI, k_BL = k)

# Matrix usage
mat <- matrix(c(0.5, 1.5, 3.0), ncol = 1)
colnames(mat) <- "LAI"
calc_IR(mat, k_BL = k)
```

`calc_Kc` *Calculate crop coefficient (Kc) from leaf area index (LAI)*

Description

Generic and method-specific functions to calculate the crop coefficient (`Kc`) using leaf area index (`LAI`), a maximum crop coefficient (`Kc_max`) and an extinction coefficient (`k`). Methods are provided for numeric vectors and matrices.

Usage

```
calc_Kc(x, ...)

## S3 method for class 'numeric'
calc_Kc(x, Kc_max, k, iday_crop_dev = rep(1, length(x)), LAI = x, ...)

## S3 method for class 'matrix'
calc_Kc(x, LAI = x[iday, "LAI"], Kc_max, k, iday = seq.int(nrow(x)), ...)
```

Arguments

`x`	Input object. Can be a numeric vector or a matrix.
`...`	Further arguments passed to methods.
`Kc_max`	numeric Maximum crop coefficient.
`k`	numeric Extinction coefficient in the Kc-LAI relationship.
`iday_crop_dev`	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as `x`.
`LAI`	numeric Leaf area index (m^2 m^{-2}). For the numeric method, defaults to `x`. For the matrix method, values are taken from the subset of rows specified by `iday`.
`iday`	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).

Details

For the numeric method, the crop coefficient is computed for each element as:

$$K_{c,i} = K_{c_max} \times (1 - \exp(-k \cdot LAI_i))$$

for time steps where `iday_crop_dev[i] == 1`. For indices where `iday_crop_dev[i] == 0`, the crop is considered not developed and `K_{c,i}` is set to 0.

For the matrix method, LAI values are extracted from the rows specified by `iday`, converted to a numeric vector, and passed to the numeric method. The result is returned as a numeric vector of `K_{c}` values aligned with the rows of the input matrix (with `NA_real_` for rows not in `iday`).

Value

For numeric input: a numeric vector of crop coefficient values . For matrix input: a numeric vector of Kc values aligned with the rows of `x`.

Examples

```
# Numeric vector example
calc_Kc(c(1, 2, 3), Kc_max = 1.2, k = 0.5)

# Matrix example (returns a vector of Kc values)
mat <- cbind(LAI = c(1, 2, 3))
calc_Kc(mat, Kc_max = 1.2, k = 0.5)
```

`calc_Kd` *Calculate hydraulic conductivity (Kd) using various methods*

Description

Generic interface to compute hydraulic conductivity (`Kd`) for a given soil water content (`theta`) using different empirical methods. The default method `"saxton"` implements the Saxton-Rawls formulation.

Usage

```
calc_Kd(x, from = "saxton", ...)

calc_Kd_from_saxton(x, ...)

## S3 method for class 'numeric'
calc_Kd_from_saxton(x, Ks, theta_fc, theta_wp, theta_sat, ...)

## S3 method for class 'matrix'
calc_Kd_from_saxton(x, Ks, theta_fc, theta_wp, theta_sat, ...)
```

Arguments

x	Numeric value, numeric vector or matrix representing volumetric soil water content ($m^3\ m^{-3}$).
from	character Name of the method used for the calculation. Default is "saxton". The method name is appended to "calc_Kd_from_" to locate the implementation.
...	Additional arguments passed to method-specific functions.
Ks	numeric Saturated hydraulic conductivity ($mm\ day^{-1}$). Can be scalar or conformable with x.
theta_fc	numeric Volumetric water content at field capacity ($m^3\ m^{-3}$).
theta_wp	numeric Volumetric water content at permanent wilting point ($m^3\ m^{-3}$).
theta_sat	numeric Volumetric water content at saturation ($m^3\ m^{-3}$).

Details

The Saxton-Rawls method (from = "saxton") estimates hydraulic conductivity as a function of soil water content and soil hydraulic properties. It uses a Campbell-type relationship:

$$B_{Campbell} = \frac{\log(1500) - \log(33)}{\log(\theta_{fc}) - \log(\theta_{wp})}$$

$$s = \max\left(\min\left(\frac{\theta_i}{\max(\theta_{sat}, 10^{-9})}, 1\right), 0\right)$$

$$e = 3 + 2B_{Campbell}$$

$$K_{d,i} = \max(K_s, 10^{-9}) \times s^e$$

where:

- θ_i: current volumetric water content (x);
- K_s: saturated hydraulic conductivity;
- θ_{fc}: water content at field capacity;
- θ_{wp}: water content at permanent wilting point;
- θ_{sat}: water content at saturation.

For matrix input, the Saxton method is applied element-wise and the result has the same dimensions as x. Parameters Ks, theta_fc, theta_wp and theta_sat may be scalars or arrays conformable with x.

Reference: Saxton, K.E., and Rawls, W.J. (2006). Soil water characteristic estimates by texture and organic matter. *Soil Science Society of America Journal*, 70, 1569-1578.

Value

For numeric input: a numeric vector of hydraulic conductivity values (Kd, same units as Ks). For matrix input: a numeric matrix of Kd values with the same dimensions as x.

See Also

- calc_theta_fc()
- calc_theta_wp()
- calc_theta_sat()

Examples

```
# Example for numeric input
calc_Kd(
  0.25,
  Ks = 0.1,
  theta_fc = 0.3,
  theta_wp = 0.1,
  theta_sat = 0.4
)
```

---

calc_KES *Calculate soil evaporation coefficient (KES)*

Description

Generic and method-specific functions to calculate the soil evaporation coefficient (KES) from soil water content. KES is a simple switch-like coefficient that indicates whether soil moisture is above a residual threshold.

Usage

```
calc_KES(x, ...)

## S3 method for class 'numeric'
calc_KES(x, theta_r, theta = x, ...)

## S3 method for class 'matrix'
calc_KES(
  x,
  theta = x[iday, theta_name],
  theta_r,
  iday = seq.int(nrow(x)),
  theta_name = "theta",
  KES_name = "KES",
  ...
)
```

Arguments

x	Input object. Can be a numeric vector or a matrix.
...	Additional arguments passed to methods.
theta_r	numeric Residual soil water content ($m^3\ m^{-3}$). Can be scalar or the same length as theta.
theta	numeric Volumetric soil water content ($m^3\ m^{-3}$). For the numeric method, defaults to x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
theta_name	character For matrix input, name of the column containing soil water content (default "theta").
KES_name	character For matrix input, name of the column used to store calculated KES values (default "KES").

Details

For numeric input, KES is computed element-wise as:

$$KES_i = \begin{cases} 1 & \text{if } \theta_i > \theta_{r,i} \\ 0 & \text{otherwise} \end{cases}$$

where $\theta_{r,i}$ is either a scalar residual water content or a value specific to each element when theta_r has the same length as theta.

For matrix input, soil water content is taken from the column named theta_name for the rows specified by iday. The computed KES values are stored in a column named KES_name, which is created if it does not exist.

Value

For numeric input: a numeric vector of KES values (0 or 1). For matrix input: the matrix x with a column KES_name created or updated for the selected rows.

See Also

- calc_theta_r()
- calc_Kd()

calc_Ks *Calculate saturated hydraulic conductivity (Ks) from soil texture*

Description

Generic interface and method-specific functions to estimate saturated hydraulic conductivity ($mm\ h^{-1}$) from soil texture (percent sand and percent clay) using different empirical models.

Usage

```
calc_Ks(x, from = "saxton", ...)

calc_Ks_from_saxton(x, ...)

## S3 method for class 'numeric'
calc_Ks_from_saxton(
  x,
  psand,
  pclay = x,
 ref = data.frame(names = c("C", "CL", "L", "LSa", "Sa", "SaC", "SaCL", "SaL", "SiL",
  "SiC", "SiCL"), psand = c(25, 30, 40, 80, 88, 50, 60, 65, 20, 10, 10), pclay = c(50,
   35, 20, 5, 5, 40, 25, 10, 15, 45, 35), Ks_Saxton_mm_h = c(1.1, 4.3, 15.5, 96.7,
    108.1, 1.4, 11.3, 50.3, 16.1, 3.7, 5.7), stringsAsFactors = FALSE),
  ...
)

calc_Ks_from_cosby(x, ...)

## S3 method for class 'numeric'
calc_Ks_from_cosby(x, psand, pclay = x, ...)

calc_Ks_from_ottoni(x, ...)

## S3 method for class 'numeric'
calc_Ks_from_ottoni(x, psand, pclay = x, ...)
```

Arguments

x	Numeric vector. In the numeric methods, x can be used as a placeholder for pclay when pclay is not explicitly provided.
from	character Name of the method to use for calculation. Options are "saxton", "cosby" or "ottoni". Default is "saxton". The method name is appended to "calc_Ks_from_" to locate the implementation.
...	Additional arguments passed to methods.
psand	numeric Percent sand in the soil sample (0-100).

`pclay` numeric Percent clay in the soil sample (0-100). If not provided, defaults to `x` in the numeric methods.

`ref` data.frame Reference table for the Saxton method, containing soil texture classes, percent sand, percent clay and Ks values. Used only in the Saxton method; a default table is provided.

Details

The main function `calc_Ks()` dispatches to the appropriate method based on the `from` argument:

Saxton method Matches the input texture to the closest reference class using Euclidean distance in the (sand, clay) space and returns the corresponding Ks value. Based on Saxton and Rawls (2006).

Cosby method Uses an empirical formula based on percent sand and clay (Cosby et al., 1984).

Ottoni method Uses an empirical formula based on percent sand and clay, with silt estimated as $100 - (sand + clay)$ (Ottoni et al., 2019).

All methods return Ks in $mm\ h^{-1}$. For the Saxton method, a small reference table of texture classes and associated Ks values is used by default, but can be overridden via the `ref` argument.

References:

- Saxton, K.E., and Rawls, W.J. (2006). Soil water characteristic estimates by texture and organic matter. *SSSAJ* 70, 1569-1578.
- Cosby, B.J. et al. (1984). A statistical exploration of the relationships of soil moisture characteristics to the physical properties of soils. *Water Resour. Res.* 20(6), 682-690.
- Ottoni, M.V. et al. (2019). Pedotransfer functions for saturated hydraulic conductivity. *Journal of Hydrology* 575, 1214-1223.

Value

A numeric vector of estimated saturated hydraulic conductivity values (`Ks`, $mm\ h^{-1}$), with one value per input texture.

Examples

```
calc_Ks(20, psand = 40, pclay = 20, from = "saxton")
calc_Ks(20, psand = 40, pclay = 20, from = "cosby")
calc_Ks(20, psand = 40, pclay = 20, from = "ottoni")
```

`calc_ks_root` *Calculate root growth ks_root parameter*

Description

Generic and method-specific functions to calculate root growth ks_root parameters.

Usage

```
calc_ks_root(x, ...)

## S3 method for class 'numeric'
calc_ks_root(
  x,
  ks_init,
  theta_raw,
  iday_crop_dev = rep(1, length(x)),
  theta = x,
  ...
)

## S3 method for class 'matrix'
calc_ks_root(
  x,
  theta = x[iday, "theta"],
  ks_init = NULL,
  iday_crop_dev = x[iday, "iday_crop_dev"],
  theta_raw,
  iday = seq.int(nrow(x)),
  ks_name = "ks_root",
  ...
)
```

Arguments

`x`	Input object. Can be `numeric`, `matrix`, etc.
`...`	Additional arguments passed to methods.
`ks_init`	numeric Initial root growth parameter.
`theta_raw`	numeric Raw soil moisture content.
`iday_crop_dev`	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as `x`.
`theta`	numeric Current soil moisture content.
`iday`	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
`ks_name`	character Layer or column name to store calculated ks_root values.

Value

An object of the same class as x, with the calculated "ks_root" values added.

---

calc_KTP *Calculate transpiration reduction coefficient (KTP)*

Description

Generic and method-specific functions to calculate the transpiration reduction coefficient (KTP) from soil water content. KTP ranges between 0 and 1 and scales potential transpiration according to water availability between wilting point and readily available water.

Usage

```
calc_KTP(x, ...)

## S3 method for class 'numeric'
calc_KTP(x, theta_wp, theta_raw, theta = x, ...)

## S3 method for class 'matrix'
calc_KTP(
  x,
  theta = x[iday, theta_name],
  theta_wp,
  theta_raw,
  iday = seq.int(nrow(x)),
  theta_name = "theta",
  KTP_name = "KTP",
  ...
)
```

Arguments

x	Input object. Can be a numeric vector or a matrix.
...	Additional arguments passed to methods.
theta_wp	numeric Volumetric soil water content at wilting point ($m^3\ m^{-3}$).
theta_raw	numeric Volumetric soil water content defining the upper limit of readily available water ($m^3\ m^{-3}$).
theta	numeric Volumetric soil water content ($m^3\ m^{-3}$). For the numeric method, defaults to x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
theta_name	character For matrix input, name of the column containing soil water content (default "theta").
KTP_name	character For matrix input, name of the column used to store calculated KTP values (default "KTP").

Details

For numeric input, KTP is computed element-wise from `theta` as:

- if $\theta > \theta_{raw}$: $KTP = 1$ (no stress, full transpiration);
- if $\theta_{wp} < \theta \leq \theta_{raw}$: KTP decreases linearly from 1 to 0:

$$KTP = \frac{\theta - \theta_{wp}}{\theta_{raw} - \theta_{wp}}$$

- if $\theta \leq \theta_{wp}$: $KTP = 0$ (no transpiration).

For matrix input, soil water content is taken from the column named `theta_name` for rows specified by `iday`. The calculated KTP values are stored in a column named `KTP_name`, which is created if it does not exist.

Value

For numeric input: a numeric vector of KTP values in `[0, 1]`. For matrix input: the matrix `x` with a column `KTP_name` created or updated for the selected rows.

See Also

- `calc_theta_wp()`
- `calc_theta()`

Examples

```
# Numeric example
theta <- c(0.08, 0.12, 0.18)
calc_KTP(theta, theta_wp = 0.1, theta_raw = 0.16)

# Matrix example
mat <- cbind(theta = c(0.08, 0.12, 0.18))
calc_KTP(mat, theta_wp = 0.1, theta_raw = 0.16)
```

`calc_LAI` *Calculate leaf area index (LAI)*

Description

Generic function and methods to calculate leaf area index (`LAI`) from potential LAI (`LAIp`), its initial value (`LAIp_init`) and stress factors affecting leaf growth or senescence.

Usage

```
calc_LAI(x = NULL, ...)

## S3 method for class '`NULL`'
calc_LAI(x = NULL, LAI0, ...)

## S3 method for class 'numeric'
calc_LAI(
  x,
  LAIp,
  LAIp_init,
  stress_LAI,
  stress_noci_LAI,
  LAImin = NA_real_,
  k = NA_real_,
  iday_crop_dev = rep(1, length(LAIp)),
  LAI0 = x,
  ...
)

## S3 method for class 'data.frame'
calc_LAI(
  x,
  LAI0 = NULL,
  LAIp = x$LAIp[iday],
  stress_LAI = x$stress_LAI[iday],
  stress_noci_LAI = x$stress_noci_LAI[iday],
  LAImin = NA_real_,
  k = NA_real_,
  iday = seq.int(nrow(x)),
  LAI_name = "LAI",
  ...
)

## S3 method for class 'matrix'
calc_LAI(
  x,
  LAI0 = ifelse(iday[1] == 1, 0, x[iday - 1, LAI_name]),
  LAIp = x[iday, LAIp_name],
  LAIp_init = ifelse(iday == 1, 0, x[iday - 1, LAIp_name]),
  stress_LAI = x[iday, "stress_LAI"],
  stress_noci_LAI = x[iday, "stress_noci_LAI"],
  LAImin = NA_real_,
  k = NA_real_,
  iday = seq.int(nrow(x)),
  LAIp_name = "LAIp",
  LAI_name = "LAI",
  ...
```

)

Arguments

x	Input object. Can be NULL, a numeric value, a data.frame or a matrix.
...	Additional arguments passed to methods.
LAI0	numeric Initial LAI value(s) (m^2 m^{-2}). If NULL or NA, defaults to 0 in the numeric and data.frame/matrix methods (see Details).
LAIp	numeric Potential LAI value(s) at each time step (m^2 m^{-2}).
LAIp_init	numeric Initial potential LAI value(s) used to compute daily increments (LAIp - LAIp_init) (m^2 m^{-2}).
stress_LAI	numeric Stress factor(s) for LAI increment (must be ≥ 0).
stress_noci_LAI	numeric Exponent(s) applied to stress_LAI when scaling the LAI increment.
LAImin	numeric Lower asymptote for LAI during senescence. Use NA to disable the damping rule.
k	numeric Senescence damping coefficient applied when LAImin is active. Use NA to disable the damping rule.
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
LAI_name	character Name of the LAI column to use or create in tabular data (default "LAI").
LAIp_name	character Name of the potential LAI column in matrix input (default "LAIp").

Details

The numeric method updates LAI based on changes in potential LAI and a stress-modified increment. Let:

$$\Delta LAI_i = LAIp_i - LAIp_{init,i}$$

For each time step i, the increment is scaled by a stress factor:

$$stress_{LAI,i} = \begin{cases} stress_{LAI,i} & \text{Growth Phase} \\ 2 - stress_{LAI,i} & \text{Senescence Phase (when } \Delta LAI_i < 0) \end{cases}$$

and:

$$LAI_i = LAI0_i + \Delta LAI_i \, stress_{LAI,i}^{stress_noci_{LAI}}$$

When both LAImin and k are provided and $\Delta LAI_i < 0$, the senescence step is damped by limiting the daily loss to:

$$dLAI_i = k_i(LAI0_i - LAImin_i)$$

and the retained LAI becomes:

$$LAI_i = \max\left(LAI_i,\ LAI0_i - dLAI_i,\ LAImin_i\right)$$

so that LAI tends progressively towards LAImin instead of dropping too quickly towards zero.

When LAI0 is a single scalar, the method maintains an internal cumulative value LAI0 across time:

- LAI0 is initialised to the scalar LAI0 (or 0 if LAI0 is NA);
- for each time step, LAI0 is updated by the scaled increment and stored as LAI[i];
- when i == 0, LAI[i] is NA_real_ and the internal LAI0 value is left unchanged.

When LAI0 is a vector, each time step is computed independently using the corresponding LAI0[i] (or 0 if NA).

For data.frame and matrix inputs, LAI is written to a column named LAI_name. For data.frame, LAI0 is taken from the argument for the first row and then from the previous row's LAI for subsequent rows. For matrix, a similar logic is applied using LAIp_name and LAI_name.

Value

For numeric input: a numeric vector of LAI values ($m^2\ m^{-2}$). For data.frame and matrix input: the object x with a column LAI_name created or updated.

See Also

- calc_stress()
- calc_stress_noci_LAI()
- calc_LT()
- calc_LAIp()

Examples

```
# Numeric example
calc_LAI(
  1,
  LAIp = 2,
  LAIp_init = 1,
  stress_LAI = 0.8,
  stress_noci_LAI = 1,
  LAImin = 0.5,
  k = 0.05
)

# Data frame example
df <- data.frame(
  LAIp = c(1, 2, 3),
  stress_LAI = 0.8,
  stress_noci_LAI = 1
)
calc_LAI(df)

# Matrix example
mat <- as.matrix(
  data.frame(
    LAIp = c(1, 2, 3),
```

```
    stress_LAI = 0.8,
    stress_noci_LAI = 1
  )
)
calc_LAI(mat)
```

---

calc_LAIp *Calculate potential leaf area index (LAIp)*

Description

Generic function and methods to calculate potential leaf area index (LAIp) from a normalized/logistic temperature index (LT), maximum LAI (LAImax) and an initial LAI (LAI0). Implemented for numeric vectors, matrices, data.frames and model initialisation objects.

Usage

```
calc_LAIp(x = NULL, ...)

## S3 method for class '`NULL`'
calc_LAIp(x, LT, ...)

## S3 method for class 'numeric'
calc_LAIp(x, LAImax, LAI0, LT = x, ...)

## S3 method for class 'matrix'
calc_LAIp(
  x,
  LT = x[iday, LT_name],
  LAImax,
  LAI0,
  iday = seq.int(nrow(x)),
  LT_name = "LT",
  LAIp_name = "LAIp",
  ...
)

## S3 method for class 'data.frame'
calc_LAIp(
  x,
  LT = x[iday, LT_name, drop = TRUE],
  LAImax = x$LAImax,
  LAI0 = x$LAI0,
  iday = seq.int(nrow(x)),
  LT_name = "LT",
  LAIp_name = "LAIp",
```

```
  ...
)

## S3 method for class 'model_init'
calc_LAIp(x, crop, iday_sowing, iday_harvest, ...)
```

Arguments

x	Input object. Can be numeric, matrix, data.frame or model_init.
...	Additional arguments passed to methods.
LT	numeric Logistic or normalized temperature index ([0, 1]). For most methods, defaults to x or a column/layer of x.
LAImax	numeric Maximum potential leaf area index ($m^2\ m^{-2}$).
LAI0	numeric Initial potential leaf area index ($m^2\ m^{-2}$).
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
LT_name	character Column or layer name containing LT values (default "LT").
LAIp_name	character Column or layer name used to store calculated potential LAI values (default "LAIp").
crop	list Crop parameter list for the model_init method (must contain at least TT_LAImax, alpha1, alpha2, beta, LAImax).
iday_sowing	integer Index (or indices) of sowing day(s) for the model_init method.
iday_harvest	integer Index (or indices) of harvest day(s) for the model_init method.

Details

The numeric method computes potential LAI at each time step by a simple linear interpolation between LAI0 and LAImax using LT ([0, 1]):

$$LAIp_i = LAI0_i + (LAImax_i - LAI0_i) \cdot LT_i$$

with checks enforcing:

- $0 \leq LT \leq 1$,
- $LAI0 \geq 0$,
- $LAImax \geq LAI0$.

For matrix and data.frame inputs, LT values are read from the specified column (LT_name) and LAIp is written to a column named LAIp_name, which is created if missing.

The model_init method typically computes a normalized thermal time and an LT-like index using helper functions (e.g. calc_TT_norm(), calc_alpha(), calc_LT()), then delegates to the appropriate calc_LAIp method (usually the data.frame or matrix method) to fill LAIp between sowing and harvest.

Value

A numeric vector, matrix, data.frame or model_init object (same type as x) with a potential LAI field (LAIp) computed or updated.

See Also

- calc_LT()
- calc_LAI()
- calc_TT_norm()

Examples

```
# Numeric example
calc_LAIp(0.5, LAImax = 6, LAI0 = 0.1)
```

---

calc_LT *Calculate logistic temperature index (LT)*

Description

Generic function and methods to calculate a logistic temperature index (LT) from a normalised thermal-time variable (TT_norm) and shape parameters alpha and beta.

Usage

```
calc_LT(x, ...)

## S3 method for class 'numeric'
calc_LT(x, alpha, beta, TTnorm_LTsat, TT_norm = x, ...)

## S3 method for class 'data.frame'
calc_LT(
  x,
  TT_norm = x[iday, TT_norm_name, drop = TRUE],
  alpha = x[iday, alpha_name, drop = TRUE],
  beta,
  TTnorm_LTsat,
  iday = seq.int(nrow(x)),
  TT_norm_name = "TT_norm",
  alpha_name = "alpha",
  LT_name = "LT",
  ...
)

## S3 method for class 'matrix'
calc_LT(
  x,
  TT_norm = x[iday, TT_norm_name],
  alpha = x[iday, alpha_name],
  beta,
```

```
  TTnorm_LTsat,
  iday = seq.int(nrow(x)),
  TT_norm_name = "TT_norm",
  alpha_name = "alpha",
  LT_name = "LT",
  ...
)
```

Arguments

x	Input object. Can be numeric, data.frame or matrix.
...	Additional arguments passed to methods.
alpha	numeric Alpha parameter(s) controlling curve shape.
beta	numeric Beta parameter(s) controlling curve shape.
TTnorm_LTsat	numeric Normalised thermal-time value at which the senescence progression is considered complete. Values above 1 stretch the senescence branch to the right.
TT_norm	numeric Normalised thermal time [0,1]. For the numeric method, defaults to x. For other methods, values are taken from the column or layer named TT_norm_name.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
TT_norm_name	character Column or layer name containing TT_norm values (default "TT_norm").
alpha_name	character Column or layer name containing alpha values (default "alpha").
LT_name	character Column or layer name used to store calculated logistic temperature values (default "LT").

Details

The numeric method computes LT element-wise from TT_norm, alpha, and beta using:

$$LT_i = \mathrm{TT}_{\mathrm{norm}_i}^{\beta} \exp\left[\frac{\beta}{\alpha_i}\left(1 - \mathrm{TT}_{\mathrm{norm}_i}^{\alpha_i}\right)\right]$$

where the parameter α_i depends on crop phenological phase:

$$\alpha_i = \begin{cases} \alpha_1, & \text{Growth phase} \\ \alpha_2, & \text{Senescence phase} \end{cases}$$

For data.frame and matrix inputs, TT_norm and alpha are read from the specified columns for the rows indexed by iday. The result is written to a column named LT_name, which is created if missing.

Value

For numeric input: a numeric vector of logistic temperature index values. For data.frame and matrix input: the object x with a column LT_name created or updated.

Examples

```
# Numeric example
calc_LT(
  seq(0, 1, length.out = 5),
  alpha = 2,
  beta = 3,
  TTnorm_LTsat = 1
)
```

`calc_PAR` *Calculate photosynthetically active radiation (PAR)*

Description

Generic function and methods to calculate photosynthetically active radiation (PAR) from incoming solar radiation (Rg). Implemented for numeric vectors, matrices and data.frames.

Usage

```
calc_PAR(x, ...)

## S3 method for class 'numeric'
calc_PAR(x, Rg = x, ...)

## S3 method for class 'matrix'
calc_PAR(
  x,
  Rg = x[iday, Rg_name],
  iday = seq.int(nrow(x)),
  Rg_name = "Rg",
  PAR_name = "PAR",
  ...
)

## S3 method for class 'data.frame'
calc_PAR(
  x,
  Rg = x[iday, Rg_name],
  iday = seq.int(nrow(x)),
  Rg_name = "Rg",
  PAR_name = "PAR",
  ...
)
```

Arguments

x	Input object. Can be numeric, matrix or data.frame.
...	Additional arguments passed to methods.
Rg	numeric Incoming global solar radiation ($J \cdot cm^{-2} \cdot day^{-1}$). For the numeric method, defaults to x. For other methods, values are taken from the column or layer named Rg_name.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
Rg_name	character Column or layer name containing incoming solar radiation (default "Rg").
PAR_name	character Column or layer name used to store calculated PAR values (default "PAR").

Details

The numeric method computes PAR as a fixed fraction of the incoming solar radiation:

$$PAR_i = \frac{1}{2} Rg_i$$

PAR is returned in the same unit as Rg, i.e. $J \cdot cm^{-2} \cdot day^{-1}$.

For matrix and data.frame inputs, Rg values are read from the column named Rg_name for rows indexed by iday. The result is written to a column named PAR_name, which is created if missing.

Value

For numeric input: a numeric vector of PAR values (same length as Rg). For matrix and data.frame input: the object x with a column PAR_name created or updated.

Examples

```
# Numeric example
calc_PAR(20) # returns 10

# Matrix example
mat <- cbind(Rg = c(10, 20, 30))
calc_PAR(mat)

# Data frame example
df <- data.frame(Rg = c(15, 25, 35))
calc_PAR(df)
```

`calc_quota` *Calculate Applied Irrigation Dose with Quota Management*

Description

This function manages irrigation quota allocation across different strategies and time periods. It supports two modes: a simple global quota cap (mode 0) and an intelligent strategy-based quota distribution (mode 1).

Usage

```
calc_quota(
  x,
  dose,
  quota_max,
  quota_part,
  iday,
  strategy_id,
  dose_min = NA_real_,
  quota_mod = 1L
)
```

Arguments

`x`	A matrix or data.frame containing irrigation history with columns: • `I1`: Applied irrigation dose for each day • `iday_strategy`: Strategy identifier for each day
`dose`	Numeric. The desired irrigation dose (mm) to apply today.
`quota_max`	Numeric. Maximum total irrigation quota (mm) for the season.
`quota_part`	Numeric. Percentage (0-100) or fraction (0-1) of the total quota allocated to the current strategy/period.
`iday`	Integer. Current day index in the irrigation history matrix.
`strategy_id`	Integer. Identifier for the current irrigation strategy.
`dose_min`	Numeric. Minimum irrigation dose (mm). If the calculated dose is below this threshold, no irrigation is applied. Default is `NA_real_`.
`quota_mod`	Integer. Quota management mode: • `0`: Simple global quota cap only • `1`: Intelligent strategy-based quota distribution (default)

Details

The function implements two quota management strategies:

Mode 0 (Legacy): Applies a simple cap based on remaining global quota, without considering strategy-specific allocations or temporal distribution.

Mode 1 (Intelligent): Distributes quota across strategies and time:

- Allocates a fraction of total quota to each strategy
- Distributes strategy quota evenly across days in that strategy
- Allows catch-up if previous days used less than theoretical allocation
- Always respects global quota limits

The minimum dose threshold (`dose_min`) prevents inefficient small irrigations when quota is nearly exhausted.

Value

A list with three elements:

dose_applied Numeric. The actual irrigation dose (mm) to apply, considering quota constraints.

quota_used_strategy Numeric. Total quota (mm) used by the current strategy including today's dose.

quota_used_global Numeric. Total quota (mm) used across all strategies including today's dose.

`calc_R` *Calculate soil water reserve (R)*

Description

Generic function and methods to calculate soil water reserve (`R`) from different input representations. Currently, only computation from volumetric soil water content (`theta`) is implemented via `from = "theta"`.

Usage

```
calc_R(x, ...)

## Default S3 method:
calc_R(x, from, ...)

## S3 method for class 'model_output'
calc_R(x, ...)

calc_R_from_theta(x, ...)

## S3 method for class 'numeric'
calc_R_from_theta(x, z, theta = x, ...)

## S3 method for class 'matrix'
calc_R_from_theta(
  x,
  z,
  theta = x[iday, theta_name],
```

```
  iday = seq.int(nrow(x)),
  theta_name = "theta",
  R_name = "R",
  ...
)

## S3 method for class 'data.frame'
calc_R_from_theta(
  x,
  z,
  theta = x[iday, theta_name],
  iday = seq.int(nrow(x)),
  theta_name = "theta",
  R_name = "R",
  ...
)
```

Arguments

x	Input object. Can be numeric, matrix, data.frame or model_output.
...	Additional arguments passed to methods.
from	character Input type / method selector. Currently only "theta" is supported, which delegates to calc_R_from_theta().
z	numeric Effective soil depth (m). Can be scalar or the same length as theta.
theta	numeric Volumetric soil water content ($m^3\ m^{-3}$) used when from = "theta". For the numeric method, defaults to x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
theta_name	character For matrix and data.frame methods, name of the column containing volumetric water content (default "theta").
R_name	character For matrix and data.frame methods, name of the column used to store calculated soil water reserve (default "R").

Details

When from = "theta", soil water reserve R (mm) is computed from volumetric water content theta ($m^3\ m^{-3}$) and soil depth z (m) as:

$$R_i = \theta_i\, z_i \times 1000$$

where the factor 1000 converts metres of water column to millimetres.

For matrix and data.frame inputs, theta is read from the column theta_name for rows indexed by iday, and the result is written to a column R_name, which is created if missing.

For model_output objects, the current implementation is a placeholder that returns x unchanged; it can be extended to compute and attach R based on model internals.

Value

For numeric input: a numeric vector of soil water reserve values (mm). For `matrix` and `data.frame` input: the object `x` with a column `R_name` created or updated. For `model_output` input: the object `x` unchanged.

Examples

```
# Numeric example: theta = 0.25 (m3/m3), z = 0.6 m -> R = 150 mm
calc_R_from_theta(0.25, z = 0.6)

# Matrix example
mat <- cbind(theta = c(0.2, 0.25, 0.3))
calc_R(mat, from = "theta", z = 0.5)
```

---

calc_Rgravity *Calculate gravitational soil water (Rgravity)*

Description

Generic helpers to calculate gravitational soil water (`Rgravity`) from either total soil water reserve (`from = "R"`) or from volumetric soil water content and field capacity (`from = "theta"`). Currently, only the `"theta"`-based methods are implemented.

Usage

```
calc_Rgravity(x, from, ...)

calc_Rgravity_from_R(x, ...)

calc_Rgravity_from_theta(x, ...)

## S3 method for class 'numeric'
calc_Rgravity_from_theta(x, theta_fc, z, theta = x, ...)

## S3 method for class 'matrix'
calc_Rgravity_from_theta(
  x,
  theta = x[iday, theta_name],
  theta_fc,
  z = x[iday, z_name],
  iday = seq.int(nrow(x)),
  theta_name = "theta",
  z_name = "z",
  Rgravity_name = "Rgravity",
  ...
)
```

Arguments

x	Input object. For from = "theta", can be numeric or matrix. For from = "R", only dispatch stubs are defined (not yet implemented).
from	character Source of the input data, either "R" or "theta". This controls which helper (calc_Rgravity_from_R or calc_Rgravity_from_theta) is used.
...	Additional arguments passed to methods.
theta_fc	numeric Volumetric water content at field capacity ($m^3\ m^{-3}$); used when from = "theta". A single (scalar) value is expected in the numeric method.
z	numeric Soil depth (m); used when from = "theta". In the numeric method a single value is expected; in the matrix method it is taken per row (or can be supplied).
theta	numeric Volumetric soil water content ($m^3\ m^{-3}$); used when from = "theta". For the numeric method, defaults to x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
theta_name	character For matrix inputs with from = "theta", name of the column containing volumetric water content (default "theta").
z_name	character For matrix inputs with from = "theta", name of the column containing soil depth (default "z").
Rgravity_name	character For matrix inputs, name of the column used to store calculated gravitational water (default "Rgravity").

Details

When from = "theta", gravitational soil water for time step i is computed as:

$$R_i^{gravity} = \max\left[(\theta_i - \theta_{fc})\, z_i, 0\right] \times 1000$$

where θ_i is volumetric soil water content ($m^3\ m^{-3}$), θ_{fc} is field capacity ($m^3\ m^{-3}$), z_i is soil depth (m), and the factor 1000 converts metres of water column to millimetres.

For matrix inputs, theta and z are read from the specified columns for rows indexed by iday, and the result is written to a column named Rgravity_name, which is created if missing.

The from = "R" path (calc_Rgravity_from_R) is currently not implemented and will raise an error if used.

Value

For from = "theta" and numeric input: a numeric vector of gravitational water (mm). For from = "theta" and matrix input: the matrix x with a column Rgravity_name created or updated. For from = "R": an error is raised (not implemented).

Examples

```
# Numeric example: theta = 0.35, theta_fc = 0.30, z = 1 m
# -> (0.05 * 1 * 1000) = 50 mm
calc_Rgravity(0.35, from = "theta", theta_fc = 0.3, z = 1)
```

```
# Matrix example
mat <- cbind(theta = c(0.28, 0.35))
calc_Rgravity(mat, from = "theta", theta_fc = 0.3, z = 0.1)
```

`calc_RU` *Calculate soil available water content (RU)*

Description

Generic helpers to calculate soil available water content (`RU`) from volumetric soil water content (`theta`), field capacity (`theta_fc`), wilting point (`theta_wp`) and soil depth (`z`). Currently only the `"theta"`-based method (`from = "theta"`) is implemented.

Usage

```
calc_RU(x, from = "theta", ...)

calc_RU_from_theta(x, ...)

## S3 method for class 'numeric'
calc_RU_from_theta(x, theta_fc, theta_wp, z, theta = x, ...)

## S3 method for class 'matrix'
calc_RU_from_theta(
  x,
  theta = x[iday, theta_name],
  z = x[iday, z_name],
  theta_fc = x[iday, theta_fc_name],
  theta_wp = x[iday, theta_wp_name],
  iday = seq.int(nrow(x)),
  theta_name = "theta",
  z_name = "z",
  theta_fc_name = "theta_fc",
  theta_wp_name = "theta_wp",
  RU_name = "RU",
  ...
)

## S3 method for class 'data.frame'
calc_RU_from_theta(
  x,
  theta = x[iday, theta_name],
  z = x[iday, z_name],
  theta_fc = x[iday, theta_fc_name],
  theta_wp = x[iday, theta_wp_name],
```

```
  iday = seq.int(nrow(x)),
  theta_name = "theta",
  z_name = "z",
  theta_fc_name = "theta_fc",
  theta_wp_name = "theta_wp",
  RU_name = "RU",
  ...
)
```

Arguments

x	Input object. Can be a numeric vector, matrix, or data.frame (for from = "theta").
from	character Method used to calculate RU. Currently only "theta" is supported, which delegates to calc_RU_from_theta().
...	Additional arguments passed to specific methods.
theta_fc	numeric Volumetric soil water content at field capacity ($m^3\ m^{-3}$).
theta_wp	numeric Volumetric soil water content at wilting point ($m^3\ m^{-3}$).
z	numeric Soil depth (m).
theta	numeric Volumetric soil water content ($m^3\ m^{-3}$). For the numeric method, defaults to x.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
theta_name	character Column or layer name containing theta values (default "theta").
z_name	character Column or layer name containing soil depth (z) values (default "z").
theta_fc_name	character Column or layer name containing field capacity (theta_fc) values (default "theta_fc").
theta_wp_name	character Column or layer name containing wilting point (theta_wp) values (default "theta_wp").
RU_name	character Column or layer name used to store available water content (RU) (default "RU").

Details

For from = "theta", available water RU (mm) at time step i is computed from volumetric water contents and depth as:

$$RU_i = \max\left[\min\left(\theta_i - \theta_{wp}, \theta_{fc} - \theta_{wp}\right), 0\right] \times z_i \times 1000$$

where θ_i, θ_{fc} and θ_{wp} are in $m^3\ m^{-3}$, z_i is in metres, and the factor 1000 converts metres of water column to millimetres.

For matrix and data.frame inputs, theta, theta_fc, theta_wp, and z are read from the specified columns for rows indexed by iday, and the result is written to a column named RU_name, which is created if missing.

Value

For numeric input: a numeric vector of available water (mm). For `matrix` and `data.frame` input: the object `x` with a column `RU_name` created or updated.

See Also

- `calc_theta()`

Examples

```
# Numeric example: theta = 0.25, theta_fc = 0.30, theta_wp = 0.10, z = 0.6 m
# available fraction = clamp(0.25 - 0.10, 0, 0.20) = 0.15
# RU = 0.15 * 0.6 * 1000 = 90 mm
calc_RU_from_theta(0.25, theta_fc = 0.3, theta_wp = 0.1, z = 0.6)
```

---

calc_RUmax *Calculate maximum soil available water content (RUmax)*

Description

Generic function and methods to calculate the maximum soil available water content (`RUmax`) from field capacity (`theta_fc`), wilting point (`theta_wp`) and soil depth (`z`). `RUmax` represents the total amount of water that can be stored between `theta_wp` and `theta_fc`.

Usage

```
calc_RUmax(x, ...)

## S3 method for class 'numeric'
calc_RUmax(x, theta_fc, theta_wp, z = x, ...)

## S3 method for class 'matrix'
calc_RUmax(
  x,
  theta_fc = x[iday, theta_fc_name],
  theta_wp = x[iday, theta_wp_name],
  z = x[iday, z_name],
  iday = seq.int(nrow(x)),
  theta_fc_name = "theta_fc",
  theta_wp_name = "theta_wp",
  z_name = "z",
  RUmax_name = "RUmax",
  ...
)

## S3 method for class 'data.frame'
```

```
calc_RUmax(
  x,
  theta_fc = x[iday, theta_fc_name],
  theta_wp = x[iday, theta_wp_name],
  z = x[iday, z_name],
  iday = seq.int(nrow(x)),
  theta_fc_name = "theta_fc",
  theta_wp_name = "theta_wp",
  z_name = "z",
  RUmax_name = "RUmax",
  ...
)
```

Arguments

x	Input object. Can be a numeric vector, matrix or data.frame.
...	Additional arguments passed to methods.
theta_fc	numeric Volumetric water content at field capacity ($m^3\ m^{-3}$). For the numeric method, a single value is expected.
theta_wp	numeric Volumetric water content at wilting point ($m^3\ m^{-3}$). For the numeric method, a single value is expected.
z	numeric Soil depth (m). For the numeric method, defaults to x. For matrix/data.frame methods, values are taken from the column named z_name unless supplied explicitly.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
theta_fc_name	character Column name containing field capacity (theta_fc) values (default "theta_fc").
theta_wp_name	character Column name containing wilting point (theta_wp) values (default "theta_wp").
z_name	character Column name containing soil depth (z) values (default "z").
RUmax_name	character Column name used to store calculated maximum available water content (RUmax) (default "RUmax").

Details

For each time step i, the maximum available water (between field capacity and wilting point) is computed as:

$$RU_{max,i} = (\theta_{fc,i} - \theta_{wp,i})\ z_i \times 1000$$

where $\theta_{fc,i}$ and $\theta_{wp,i}$ are in $m^3\ m^{-3}$, z_i is in metres, and the factor 1000 converts metres of water column to millimetres.

For matrix and data.frame inputs, theta_fc, theta_wp and z are read from the specified columns for rows indexed by iday, and the result is written to a column named RUmax_name, which is created if missing.

Value

For numeric input: a numeric vector of maximum available water (mm). For `matrix` and `data.frame` input: the object `x` with a column `RUmax_name` created or updated.

Examples

```
# Numeric example:
# theta_fc = 0.30, theta_wp = 0.10, z = 0.6 m
# RUmax = (0.30 - 0.10) * 0.6 * 1000 = 120 mm
calc_RUmax(0.6, theta_fc = 0.30, theta_wp = 0.10)
```

---

calc_stress *Calculate LAI stress value*

Description

Generic and method-specific functions to calculate LAI stress values.

Usage

```
calc_stress(x, ...)

## S3 method for class 'numeric'
calc_stress(
  x,
  stress_T = NA,
  iday_crop_dev = rep(1, length(x)),
  stress_TP = x,
  ...
)

## S3 method for class 'matrix'
calc_stress(
  x,
  stress_TP = x[iday, stress_TP_name],
  stress_T = x[iday, stress_T_name],
  iday = seq.int(nrow(x)),
  stress_TP_name = "stress_TP",
  stress_T_name = "stress_T",
  stress_name = "stress",
  ...
)

## S3 method for class 'model_output'
calc_stress(x, ...)
```

Arguments

x	Input object. Can be numeric, matrix, etc.
...	Additional arguments passed to methods.
stress_T	numeric Temperature stress.
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
stress_TP	numeric Water stress from evapotranspiration ratio.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
stress_TP_name	character Column or layer name of evapotranspiration stress.
stress_T_name	character Column or layer name of temperature stress.
stress_name	character Column or layer name to store calculated final stress.

Value

An object of the same class as x, with the calculated LAI stress values.

---

calc_stress_noci_B *Calculate biomass stress nocivity (stress_noci_B)*

Description

Generic function and methods to calculate biomass stress nocivity (stress_noci_B) from potential biomass (Bp) and a maximum stress nocivity value (estress_b). The quantity stress_noci_B scales linearly with the fraction of potential biomass relative to a reference maximum (Bp_max).

Usage

```
calc_stress_noci_B(x, ...)

## S3 method for class 'numeric'
calc_stress_noci_B(x, Bp_max, estress_b, Bp = x, ...)

## S3 method for class 'matrix'
calc_stress_noci_B(
  x,
  Bp = x[iday, Bp_name],
  estress_b = x[iday, estress_b_name],
  Bp_max = max(x[, Bp_name], na.rm = TRUE),
  iday = seq.int(nrow(x)),
  Bp_name = "Bp",
  estress_b_name = "estress_b",
  stress_noci_B_name = "stress_noci_B",
```

```
  ...
)

## S3 method for class 'data.frame'
calc_stress_noci_B(
  x,
  Bp = x[iday, Bp_name],
  estress_b = x[iday, estress_b_name],
  Bp_max = max(x[, Bp_name], na.rm = TRUE),
  iday = seq.int(nrow(x)),
  Bp_name = "Bp",
  estress_b_name = "estress_b",
  stress_noci_B_name = "stress_noci_B",
  ...
)
```

Arguments

x	Input object. Can be numeric or matrix.
...	Additional arguments passed to methods.
Bp_max	numeric Maximum potential biomass value used as reference ($t\ ha^{-1}$). For matrix and data.frame methods, defaults to the maximum Bp over all rows.
estress_b	numeric Maximum stress nocivity value.
Bp	numeric Potential or reference biomass ($t\ ha^{-1}$).
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
Bp_name	character Column or layer name containing potential biomass values (default "Bp").
estress_b_name	character Name of the column containing maximum stress nocivity values (matrix/data.frame methods).
stress_noci_B_name	character Name of the column or layer used to store calculated stress nocivity values (default "stress_noci_B").

Details

For each time step or element i, the biomass stress nocivity is computed as:

$$\text{stress_noci_B}_i = \frac{Bp_i}{Bp_{max}}\, estress_b$$

where Bp_i is the current potential biomass, Bp_{max} is the reference maximum potential biomass (Bp_max), and $estress_b$ is the maximum stress nocivity value (estress_b).

For matrix and data.frame inputs, Bp is read from the column named Bp_name (default "Bp") and the result is written to a column named stress_noci_B_name, which is created if missing.

Value

An object of the same class as x, with biomass stress nocivity values added or updated:

- numeric: a numeric vector of stress nocivity values,
- matrix/data.frame: the object with a new or updated column stress_noci_B_name.

See Also

- calc_B()
- calc_Bp()

Examples

```
# Numeric example
calc_stress_noci_B(1:100, Bp_max = 200, estress_b = 0.5)

# Matrix example
calc_stress_noci_B(
  cbind(Bp = 1:10, estress_b = 0.5)
)
```

---

calc_stress_noci_LAI *Calculate leaf area index stress nocivity*

Description

Generic and method-specific functions to calculate leaf area index (LAI) stress nocivity.

Usage

```
calc_stress_noci_LAI(x, ...)

## S3 method for class 'numeric'
calc_stress_noci_LAI(
  x,
  estress_lai,
  iday_crop_dev = rep(1, length(x)),
  stress_LAI = x,
  ...
)

## S3 method for class 'matrix'
calc_stress_noci_LAI(
  x,
  stress_LAI = x[iday, stress_LAI_name],
  estress_lai = x[iday, estress_lai_name],
```

```
  iday,
  stress_LAI_name = "stress_LAI",
  estress_lai_name = "estress_lai",
  stress_noci_LAI_name = "stress_noci_LAI",
  ...
)
```

Arguments

x	Input object. Can be numeric, matrix, etc.
...	Additional arguments passed to methods.
estress_lai	numeric Maximum LAI stress nocivity value.
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
stress_LAI	numeric Leaf area index stress values.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
stress_LAI_name	character Column or layer name of LAI stress values.
estress_lai_name	character Column or layer name of maximum LAI stress nocivity values.
stress_noci_LAI_name	character Column or layer name to store LAI stress nocivity values.

Value

An object of the same class as x, with the calculated LAI stress nocivity values.

Examples

```
# Numeric example
calc_stress_noci_LAI(1:100, estress_lai = 0.5)
# Matrix example
calc_stress_noci_LAI(cbind(stress_LAI = 1:100, estress_lai = 0.5))
```

---

calc_stress_T *Calculate temperature stress value*

Description

Generic and method-specific functions for calculating temperature stress.

Usage

```
calc_stress_T(x, ...)

## S3 method for class 'numeric'
calc_stress_T(x, Tbase, Tmax, Tgel, T = x, ...)

## S3 method for class 'matrix'
calc_stress_T(
  x,
  T = x[iday, T_name],
  Tbase,
  Tmax,
  Tgel,
  iday = seq.int(nrow(x)),
  T_name = "T",
  stress_T_name = "stress_T",
  ...
)

## S3 method for class 'model_output'
calc_stress_T(x, crop, ...)
```

Arguments

x	Input object. Can be numeric, matrix, or model_output.
...	Additional arguments passed to methods.
Tbase	numeric Base temperature.
Tmax	numeric Maximum temperature.
Tgel	numeric Gel temperature.
T	numeric Temperature value.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
T_name	character Layer or column name of temperature.
stress_T_name	character Layer or column name to store calculated temperature stress.
crop	list Crop parameters.

Value

An object of the same class as x, with the calculated temperature stress values added.

`calc_stress_water` *Calculate the stress water ratio*

Description

This function calculates the stress water ratio for different data types.

Usage

```
calc_stress_water(x, ...)

## S3 method for class 'numeric'
calc_stress_water(
  x,
  potential,
  which,
  iday_crop_dev = rep(1, length(x)),
  real = x,
  ...
)

## S3 method for class 'matrix'
calc_stress_water(
  x,
  which,
  real = x[iday, which],
  potential = x[iday, switch(which, ES = "ES0", TP = "TP0", ETR = "ETM")],
  iday = seq.int(nrow(x)),
  stress_water_name = sprintf("ratio_%s", which),
  ...
)

## S3 method for class 'model_output'
calc_stress_water(x, which, iday, ...)
```

Arguments

`x`	Input object. Can be a numeric, matrix.
`...`	Additional arguments passed to methods.
`potential`	numeric The potential value of the variable.
`which`	character The variable to calculate the stress water ratio for. Accepted values are the strings contained in c("ES", "TP", "ETR").
`iday_crop_dev`	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as `x`.
`real`	numeric The actual value of the variable.

`iday` integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).

`stress_water_name` character Name of the column or layer to store the stress water ratio.

Value

A numeric, matrix with the stress water ratio.

`calc_sugar_accumulation` *Calculate sugar accumulation in roots*

Description

Generic S3 function and methods to compute sugar accumulation in root tissues based on thermal time and crop parameters.

Usage

```
calc_sugar_accumulation(x, ...)

## S3 method for class 'matrix'
calc_sugar_accumulation(x, crop, ...)
```

Arguments

`x`	A data object containing simulation time steps. Expected to be a numeric matrix with the columns named "Bp_root" (biomass partitioning to root) together with "TT_sowing" (thermal time since sowing).
`...`	Additional arguments passed to method-specific implementations.
`crop`	A list or named vector of crop parameters. Expected items: • TT_ms: thermal time start for sugar accumulation (optional) • TT_B_root5: fallback thermal time when TT_ms is NA • tau: time constant for accumulation (optional; estimated if NA) • max_sugar_rate: maximum sugar accumulation rate (percent)

Details

The matrix method identifies days with root biomass (Bp_root > 0) and where TT_sowing is at or after the start thermal time (TT_ms or TT_B_root5). Sugar accumulation is computed as a saturating exponential: sugar_rate = (max_sugar_rate / 100) * (1 - exp(-(TT_sowing - TT_ms) / tau)) If tau is not provided it is estimated from the first day with positive Bp_root.

Value

For the matrix method, the input object 'x' with an updated 'sugar_accumulation' column (or field).

calc_tau *Calculate the tau parameter*

Description

Generic S3 function and methods to compute the tau parameter based on thermal time. Methods are provided for numeric and matrix objects. The numeric method handles a single value, while the matrix method processes a time series.

Usage

```
calc_tau(x, ...)

## S3 method for class 'numeric'
calc_tau(x, TT_B_root5, ...)

## S3 method for class 'matrix'
calc_tau(x, TT_B_root5, ...)
```

Arguments

x	Input object (numeric scalar or matrix).
...	Additional arguments passed to methods.
TT_B_root5	numeric Thermal-time threshold for 5% root fraction.

calc_taum *Calculate root growth taum parameter*

Description

Generic and method-specific functions for calculating root growth taum parameter.

Usage

```
calc_taum(x, ...)

## S3 method for class 'numeric'
calc_taum(
  x,
  racplus,
  vroot,
  vrac,
  iday_crop_dev = rep(1, length(x)),
  stress_root = x,
  ...
```

```
)

## S3 method for class 'matrix'
calc_taum(
  x,
  stress_root = x[iday, stress_root_name],
  racplus,
  vroot,
  vrac,
  iday = seq.int(nrow(x)),
  stress_root_name = "stress_ETR",
  taum_name = "taum",
  ...
)

## S3 method for class 'model_output'
calc_taum(x, crop, iday, ...)
```

Arguments

x	Input object. Can be numeric, matrix, or model_output.
...	Additional arguments passed to methods.
racplus	numeric Root growth parameter.
vroot	numeric Root growth parameter.
vrac	numeric Root growth parameter.
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
stress_root	numeric Root growth stress values.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
stress_root_name	character Layer or column name containing root growth stress values.
taum_name	character Layer or column name to store the calculated taum parameter.
crop	list Crop parameters.

Value

An object of the same class as x, with the calculated taum parameter added.

calc_Tcrit *Calculate Critical Temperature (Tcrit)*

Description

This is a generic function to calculate the critical temperature (Tcrit) for different types of input data. The method dispatches based on the class of the input `x`. The critical temperature is typically used in crop modeling to determine temperature thresholds for physiological processes.

Usage

```
calc_Tcrit(x, ...)

## S3 method for class 'numeric'
calc_Tcrit(x, ...)

## S3 method for class 'matrix'
calc_Tcrit(x, iday_sowing, Tcrit_name = "Tcrit", ...)
```

Arguments

`x`	Input object. Can be `numeric`, `matrix`, etc.
`...`	Additional arguments passed to specific methods.
`iday_sowing`	numeric The day index of sowing. Used in the matrix method to calculate Tcrit based on days since sowing.
`Tcrit_name`	character The name of the column to store Tcrit

Value

The critical temperature (Tcrit) calculated for the input object. The return type depends on the used method

See Also

`calc_Tcrit.numeric`, `calc_Tcrit.matrix`

calc_theta *Calculate soil water content*

Description

This function calculates the soil water content (theta) based on the input parameters. Called function depends on the 'from' argument.

- R: calculates theta from soil water reserve (R)

Usage

```
calc_theta(x, from, ...)

calc_theta_from_R(x, ...)

## S3 method for class 'numeric'
calc_theta_from_R(x, z, R = x, ...)

## S3 method for class 'matrix'
calc_theta_from_R(
  x,
  R = x[iday, R_name],
  z = x[iday, z_name],
  iday = iday,
  R_name = "R",
  z_name = "z",
  theta_name = "theta",
  ...
)
```

Arguments

x	Input object. Can be a numeric vector or a data frame.
from	character Input type. Can be one of "R".
...	Additional arguments passed to methods.
z	numeric Soil depth.
R	numeric Soil water reserve (only for from = "R").
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
R_name	character Name of the column or array for R (default: "R").
z_name	character Name of the column or array for
theta_name	character Name of the column or array for output theta (default: "theta").

---

calc_theta_fc *Calculate available water capacity at field capacity*

Description

Generic and method-specific functions to calculate the available water capacity at field capacity (theta_fc) from different types of input data:

- soil texture (percentage of sand and clay): the default method

Usage

```
calc_theta_fc(x, from = "texture", ...)

calc_theta_fc_from_texture(x, ...)

## S3 method for class 'numeric'
calc_theta_fc_from_texture(x, ratio_sand, ratio_clay = x, ...)
```

Arguments

x	Inputs object.
from	character Indicating the type of input data. Possible values are: "texture" (default).
...	Further arguments passed to method-specific functions.
ratio_sand	numeric Percentage of sand in the soil.
ratio_clay	numeric Percentage of clay in the soil.

Details

The formula used in the texture method is from Roman Dobarco, Mercedes; Bourennane, Hocine; Arrouays, Dominique; Saby, Nicolas; Cousin, Isabelle; Martin, Manuel P., 2021, "Réservoir utile des sols de la France métropolitaine", https://doi.org/10.15454/9IRARJ, Recherche Data Gouv, V1: # nolint: line_length_linter. It is given by:

$$theta_wp = 0.278 + 0.245 * ratio_clay - 0.135 * ratio_sand$$

where ratio_clay and ratio_sand are the fractions of clay and sand in the soil, respectively.

Value

An object of the same class as x containing the calculated available water capacity at field capacity (theta_fc).

Examples

```
# Numeric method
calc_theta_fc(0.2, ratio_sand = 0.4, from = "texture")
```

---

calc_theta_r *Calculate Residual Soil Water Content (Theta_r)*

Description

This set of functions calculates the residual soil water content (θ_r) based on various input parameters.

Usage

```
calc_theta_r(x, from = "texture", ...)

calc_theta_r_from_texture(x, ...)

## S3 method for class 'numeric'
calc_theta_r_from_texture(x, ratio_clay = x, ...)
```

Arguments

x	Input value(s). Interpretation depends on the method used (see from argument).
from	Character string specifying the method to use for calculation. Default is "texture".
...	Additional arguments passed to method-specific functions.
ratio_clay	Numeric. Percentage of clay in the soil. Defaults to x.

Value

Numeric value(s) of residual soil water content (θ_r).

References

Carsel, R.F. and Parrish, R.S. (1988). Developing joint probability distributions of soil water retention characteristics. Water Resources Research, 24(5), 755-769. SoilGrids: doi:10.5194/essd15-44172023

Examples

```
calc_theta_r(25, from = "texture", ratio_sand = 60, ratio_clay = 25)
```

---

calc_theta_rel *Calculate Relative Soil Water Content (Theta_rel)*

Description

Computes the relative soil water content (θ_{rel}) by subtracting a reference value (θ_{ref}) from the observed soil water content (θ), with a lower bound of zero.

Usage

```
calc_theta_rel(x, ...)

## S3 method for class 'numeric'
calc_theta_rel(x, theta_ref, theta = x, ...)

## S3 method for class 'matrix'
```

```
calc_theta_rel(
  x,
  theta = x[iday, theta_name],
  theta_ref,
  iday = seq.int(nrow(x)),
  theta_name = "theta",
  theta_rel_name = "theta_rel",
  ...
)
```

Arguments

x	A numeric vector or a matrix containing soil water content data.
...	Further arguments passed to or from other methods.
theta_ref	numeric A reference soil water content value used for the calculation.
theta	numeric Soil water content. For numeric input, this is the observed soil water content value. For matrix input, this is the column name containing the observed soil water content values.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
theta_name	character Name of the column or array for theta (default: "theta").
theta_rel_name	character Name of the column or array for output theta_rel (default: "theta_rel").

Details

The relative soil water content is calculated as $\max(\theta - \theta_{ref}, 0)$.

Value

For numeric input, returns a numeric vector of relative soil water content. For matrix input, returns a matrix with an additional column containing the relative soil water content.

---

calc_theta_sat *Calculate Saturated Water Content (Theta_sat)*

Description

This function estimates the saturated water content (theta_sat) of soil based on input parameters. The calculation method can be specified via the `from` argument, with the default being "texture".

Usage

```
calc_theta_sat(x, from = "texture", ...)

calc_theta_sat_from_texture(x, ...)

## S3 method for class 'numeric'
calc_theta_sat_from_texture(x, ratio_sand, ratio_clay = x, ratio_om = 1.5, ...)
```

Arguments

x	Numeric value or object containing soil data.
from	Character string specifying the method to use for calculation. Default is "texture".
...	Additional arguments passed to the specific calculation method.
ratio_sand	Numeric. Percent sand in the soil sample.
ratio_clay	Numeric. Percent clay in the soil sample. If not provided, defaults to x.
ratio_om	Numeric. Percent organic matter in the soil sample. Default is 1.5%.

Details

The default method ("texture") uses the Saxton and Rawls (2006) regression equations to estimate theta_sat from soil texture (sand, clay) and organic matter content.

Value

Estimated saturated water content (theta_sat).

References

Saxton, K. E., and Rawls, W. J. (2006). Soil Water Characteristic Estimates by Texture and Organic Matter for Hydrologic Solutions. Soil Science Society of America Journal, 70(5), 1569-1578. https://doi.org/10.2136/sssaj2005.0117

Examples

```
calc_theta_sat(40, ratio_sand = 40, ratio_clay = 20, from = "texture")
```

---

calc_theta_wp *Calculate available water capacity at wilting point*

Description

Generic and method-specific functions to calculate the available water capacity at wilting point (theta_wp) from different types of input data:

- soil texture (percentage of sand and clay): the default method

Usage

```
calc_theta_wp(x, from = "texture", ...)

calc_theta_wp_from_texture(x, ...)

## S3 method for class 'numeric'
calc_theta_wp_from_texture(x, ratio_sand, ratio_clay = x, ...)
```

Arguments

x	Inputs object.
from	character Indicating the type of input data. Possible values are: "texture" (default).
...	Further arguments passed to method-specific functions.
ratio_sand	numeric Percentage of sand in the soil.
ratio_clay	numeric Percentage of clay in the soil.

Details

The formula used in the texture method is from Roman Dobarco, Mercedes; Bourennane, Hocine; Arrouays, Dominique; Saby, Nicolas; Cousin, Isabelle; Martin, Manuel P., 2021, "Réservoir utile des sols de la France métropolitaine", https://doi.org/10.15454/9IRARJ, Recherche Data Gouv, V1: # nolint: line_length_linter. It is given by:

$$theta_wp = 0.08 + 0.401 * ratio_clay - 0.293 * ratio_sand$$

where ratio_clay and ratio_sand are the fractions of clay and sand in the soil, respectively.

Value

An object of the same class as x containing the calculated available water capacity at wilting point (theta_wp).

Examples

```
# Numeric method
calc_theta_wp(0.2, ratio_sand = 0.4, from = "texture")
```

---

calc_threshold *Calculate Irrigation Trigger Conditions*

Description

Evaluates multiple threshold-based conditions to determine whether irrigation should be triggered on a given day. Conditions include readily available water, soil moisture, precipitation, temperature, water stress, and evapotranspiration.

Usage

```
calc_threshold(
  x,
  iday,
  RU_thrld,
  RU_units,
```

```
  THETA_thrld,
  PI_thrld,
  PI_ndays,
  SW_thrld,
  SW_ndays,
  T_thrld,
  T_ndays,
  ETM_thrld,
  ETM_ndays,
  theta_fc,
  theta_wp
)
```

Arguments

x	A data frame containing daily time series data with columns for soil water variables (RU1, RU2, R1, R2, z1, z2), weather (P, T), irrigation (I1), evapotranspiration (ETR), and stress indicators (stressTP).
iday	Integer. The current day index in the time series for which to evaluate trigger conditions.
RU_thrld	Numeric or NA. Threshold for readily available water (RU). Units depend on RU_units. If NA, this condition is not evaluated.
RU_units	Character or numeric. Unit for RU threshold: "1" for mm, "2" for percent of maximum available water.
THETA_thrld	Numeric or NA. Threshold for volumetric soil moisture content (m^3/m^3). If NA, this condition is not evaluated.
PI_thrld	Numeric or NA. Threshold for cumulative precipitation plus irrigation (mm) over the past PI_ndays. If NA, this condition is not evaluated.
PI_ndays	Integer. Number of days to sum precipitation and irrigation for the PI condition.
SW_thrld	Numeric or NA. Threshold for average water stress over the past SW_ndays. If NA, this condition is not evaluated.
SW_ndays	Integer. Number of days to average water stress for the SW condition.
T_thrld	Numeric or NA. Temperature threshold (°C). If NA, this condition is not evaluated.
T_ndays	Integer. Number of days to average temperature for the T condition.
ETM_thrld	Numeric or NA. Threshold for the ratio of precipitation+irrigation to evapotranspiration (as percentage) over the past ETM_ndays. If NA, this condition is not evaluated.
ETM_ndays	Integer. Number of days to evaluate the ETM ratio.
theta_fc	Numeric. Volumetric soil moisture content at field capacity (m^3/m^3).
theta_wp	Numeric. Volumetric soil moisture content at wilting point (m^3/m^3).

Details

The function evaluates up to six different irrigation trigger conditions:

- **RU**: Readily available water in root zone layers
- **THETA**: Combined soil moisture content
- **PI**: Recent precipitation and irrigation inputs
- **SW**: Recent water stress levels
- **T**: Recent temperature conditions
- **ETM**: Water supply vs. demand ratio

All evaluations use data from day `iday - 1` or earlier to avoid lookahead bias. For conditions requiring averaging or summation over multiple days, the period extends from `iday - ndays` to `iday - 1`.

Value

A named list of logical values indicating whether each threshold condition is met. Possible elements include:

RUcond TRUE if readily available water is below threshold

THETAcond TRUE if soil moisture is below threshold

PIcond TRUE if cumulative precipitation+irrigation is below threshold

SWcond TRUE if water stress is below threshold

Tcond TRUE if temperature is above threshold

ETMcond TRUE if precipitation/evapotranspiration ratio is below threshold

Only conditions with non-NA thresholds are included in the returned list.

`calc_TP` *Calculate crop transpiration*

Description

Generic S3 method to compute crop transpiration (TP).

Usage

```
calc_TP(x, ...)

## S3 method for class 'numeric'
calc_TP(x, TP0, KTP, Rmin = NULL, R = x, ...)

## S3 method for class 'model_output'
calc_TP(x, ...)
```

Arguments

x	Input object. Can be a numeric vector, data.frame, matrix, etc.
...	Additional arguments.
TP0	numeric Potential crop transpiration
KTP	numeric Crop transpiration coefficient
Rmin	numeric Minimum soil water reserve
R	numeric Soil water reserve

Value

Depending on the input type, the function returns a numeric vector, data.frame or matrix with the calculated crop transpiration (TP).

---

calc_TP0 *Calculate potential transpiration (TP0)*

Description

Generic and method-specific functions to calculate potential transpiration (TP0).

Usage

```
calc_TP0(x, ...)

## S3 method for class 'numeric'
calc_TP0(x, Kc, Cp, Ksol, ET0 = x, ...)

## S3 method for class 'matrix'
calc_TP0(
  x,
  ET0 = x[iday, ET0_name],
  Kc = x[iday, Kc_name],
  Cp = x[iday, Cp_name],
  Ksol,
  iday = seq.int(nrow(x)),
  ET0_name = "ET0",
  Kc_name = "Kc",
  Cp_name = "Cp",
  TP0_name = "TP0",
  ...
)
```

Arguments

x	Input object. Can be a numeric vector or a matrix.
...	Additional arguments passed to methods.
Kc	numeric Crop coefficient
Cp	numeric Canopy partition coefficient
Ksol	numeric Maximum solar radiation
ET0	numeric vector or matrix column. The reference evapotranspiration values. Defaults to x for numeric and matrix methods.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
ET0_name	character Layer or column name containing ET0 values.
Kc_name	character Layer or column name containing Kc values.
Cp_name	character Layer or column name containing Cp values.
TP0_name	character Layer or column name to store calculated TP0.

Value

The updated input object with potential transpiration (TP0) values.

---

calc_TT *Calculate Cumulative Thermal Time (TT)*

Description

The calcThermTime function calculates the cumulative thermal time (TT) for the current day based on the daily temperature data and a specified base temperature. It updates the array initModel by computing the thermal time using the previous day's value and today's climatic data.

Usage

```
calc_TT(x, ...)

## S3 method for class 'numeric'
calc_TT(x, Tbase, T = x, ...)

## S3 method for class 'matrix'
calc_TT(
  x,
  T = x[iday, T_name],
  Tbase,
  iday = seq.int(nrow(x)),
  T_name = "T",
  TT_name = "TT",
```

```
  ...
)

## S3 method for class 'data.frame'
calc_TT(
  x,
  T = x[iday, T_name, drop = TRUE],
  Tbase,
  iday = seq.int(nrow(x)),
  T_name = "T",
  TT_name = "TT",
  ...
)

## S3 method for class 'model_init'
calc_TT(x, crop, iday_sowing, ...)
```

Arguments

x	An object representing the data for which thermal time is to be calculated.
...	Additional arguments passed to or from other methods.
Tbase	numeric Base temperature below which no thermal time accumulates.
T	numeric Daily temperature.
iday	integer Indices of days to process. Defaults to all rows.
T_name	character Name of the column containing daily temperature.
TT_name	character Name of the column to store the calculated thermal time.
crop	list Crop parameters, used in the `model_init` method.
iday_sowing	integer Sowing day index, used in the `model_init` method.

Value

A numeric vector or data frame with the cumulative thermal time

---

`calc_TT_norm` *Calculate normalized thermal time*

Description

This function calculates the normalized thermal time (TT) by dividing the observed TT by the maximum TT (TTmax).

Usage

```
calc_TT_norm(x, ...)

## Default S3 method:
calc_TT_norm(x, TTmax, ...)

## S3 method for class 'data.frame'
calc_TT_norm(
  x,
  TT = x[iday, TT_name, drop = TRUE],
  TTmax,
  iday = seq.int(nrow(x)),
  TT_name = "TT",
  TT_norm_name = "TT_norm",
  ...
)

## S3 method for class 'matrix'
calc_TT_norm(
  x,
  TT = x[iday, TT_name],
  TTmax,
  iday = seq.int(nrow(x)),
  TT_name = "TT",
  TT_norm_name = "TT_norm",
  ...
)
```

Arguments

x	Input object. Can be a numeric vector, data frame, matrix object.
...	Additional arguments.
TTmax	numeric Maximum thermal time.
TT	numeric Thermal time.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
TT_name	character Column or attribute name to store the normalized TT.
TT_norm_name	character Column or attribute name to store the normalized

Value

numeric, data.frame, matrix, depending on the input.

calc_TT_rel *Calculate Relative Thermal Time (TT_rel)*

Description

Generic function to calculate relative thermal time (TT_rel) based on input.

Usage

```
calc_TT_rel(x, ...)

## S3 method for class 'numeric'
calc_TT_rel(x, TT_ref, TT = x, ...)

## S3 method for class 'data.frame'
calc_TT_rel(
  x,
  TT = x[iday, TT_name, drop = TRUE],
  TT_ref,
  iday = seq.int(nrow(x)),
  TT_name = "TT",
  TT_rel_name = "TT_rel",
  ...
)

## S3 method for class 'matrix'
calc_TT_rel(
  x,
  TT = x[iday, TT_name],
  TT_ref,
  iday = seq.int(nrow(x)),
  TT_name = "TT",
  TT_rel_name = "TT_rel",
  ...
)
```

Arguments

x	Object containing thermal time data, either numeric or data frame.
...	Additional arguments passed to methods.
TT_ref	numeric Reference thermal time value for calculation.
TT	numeric Thermal time values.
iday	integer Indices of days to process. Defaults to all
TT_name	character Column name for thermal time in data frame or matrix.
TT_rel_name	character Column name for relative thermal time.

Value

Relative thermal time values as numeric or data.frame with new column.

Examples

```
TT_ref <- 15

# Numeric method
calc_TT_rel(c(10, 20, 30), TT_ref = 15)

# Data frame method
df <- data.frame(TT = c(10, 20, 30))
calc_TT_rel(df, TT_ref = 15)
```

---

calc_TT_sowing *Calculate relative thermal time since sowing*

Description

This function calculates the relative thermal time since sowing by comparing the current thermal time to the thermal time at sowing.

Usage

```
calc_TT_sowing(x, ...)

## S3 method for class 'numeric'
calc_TT_sowing(x, iday_sowing, TT = x, ...)

## S3 method for class 'matrix'
calc_TT_sowing(
  x,
  TT = x[iday, TT_name],
  iday_sowing,
  iday = seq.int(nrow(x)),
  TT_name = "TT",
  TT_sowing_name = "TT_sowing",
  ...
)

## S3 method for class 'data.frame'
calc_TT_sowing(
  x,
  TT = x[iday, TT_name, drop = TRUE],
  iday_sowing,
  iday = seq.int(nrow(x)),
```

```
  TT_name = "TT",
  TT_sowing_name = "TT_sowing",
  ...
)
```

Arguments

x	Input object. Can be a numeric vector, data frame, matrix object.
...	Additional arguments.
iday_sowing	integer Sowing day index, used in the model_init method.
TT	numeric Thermal time.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
TT_name	character Column or attribute name containing the thermal time.
TT_sowing_name	character Column or attribute name to store the relative thermal time since sowing.

Value

numeric, data.frame, matrix, depending on the input.

---

calc_Y_grain *Calculate maize grain yield*

Description

Calculates maize grain yield from total dry biomass. Before flowering (R1), all dry biomass is assigned to leaves and stems. After R1, every new increment of total dry biomass is assigned to grain.

Usage

```
calc_Y_grain(x, ...)

## S3 method for class 'numeric'
calc_Y_grain(
  x,
  TT,
  TT_R1,
  TT_R6,
  HUg_R1,
  HUg_R4,
  HUg_R6,
  HUg_min,
  estress_b,
```

```
  B = x,
  HUg,
  B_previous = 0,
  Bgs_previous = 0,
  TT_previous = NA_real_,
  ...
)

## S3 method for class 'matrix'
calc_Y_grain(
  x,
  TT_R1,
  TT_R6,
  HUg_R1,
  HUg_R4,
  HUg_R6,
  HUg_min,
  estress_b,
  B = x[iday, B_name],
  TT = x[iday, TT_name],
  HUg = x[iday, HUg_name],
  iday = seq_len(nrow(x)),
  B_name = "B",
  TT_name = "TT_sowing",
  BFTS_name = "BFTS",
  Bgs_name = "Bgs",
  Y_name = "Y",
  Y_HU_0_name = "Y_HU_0",
  Y_HU_15_name = "Y_HU_15",
  Y_HU_tot_name = "Y_HU_tot",
  HUg_name = "HUg",
  stress_noci_B_name = "stress_noci_B",
  ...
)
```

Arguments

x	A numeric vector of total dry biomass, or a matrix containing it.
...	Additional arguments passed to methods.
TT	numeric Ordered thermal time since sowing (degree-days).
TT_R1, TT_R6	numeric Thermal times at flowering and physiological maturity (degree-days).
HUg_R1, HUg_R6	numeric Grain moisture at R1 and at R6 (percent, wet basis).
HUg_R4	numeric Grain moisture at R4 (percent, wet basis), used to locate the end of maximum biomass-stress nocivity.
HUg_min	numeric Minimum grain moisture that can be reached while the crop remains standing (percent, wet basis). It is the lower asymptote of field drying, not a harvest target or a post-harvest drying target.

`estress_b` numeric Maximum biomass-stress nocivity.

`B` numeric Total dry biomass (`t ha^-1`).

`HUg` numeric Grain moisture precomputed by `calc_HUg()` (percent, wet basis).

`B_previous`, `Bgs_previous`, `TT_previous`
numeric State immediately before the first value supplied to the numeric method.

`iday` integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).

`B_name`, `TT_name`, `BFTS_name`, `Bgs_name`, `Y_name`, `Y_HU_0_name`, `Y_HU_15_name`, `Y_HU_tot_name`, `HUg_name`
character Input and output column names for matrix inputs.

`stress_noci_B_name`
character Output column containing dynamic biomass-stress nocivity.

Details

Grain moisture must be computed beforehand with `calc_HUg()`. The moisture supplied for R4 determines its thermal time from the inverse of that curve. Biomass-stress nocivity rises linearly from zero at sowing to `estress_b` at R1, remains constant until R4, then decreases linearly to zero at R6.

`Bgs` is dry grain biomass. `Y_HU_0` is expressed on a dry basis, `Y_HU_15` at 15 percent moisture, and `Y_HU_tot` at the simulated grain moisture. `Y` remains equal to `Y_HU_0` unless another basis is selected by `run_model()`:

$$Y = Bgs$$

$$Y_{HU,15} = Y_{HU,0} \times 100/85$$

$$Y_{HU,tot} = Y_{HU,0} \times 100/(100 - HUg).$$

Value

The numeric method returns a numeric matrix with columns `stress_noci_B`, `BFTS`, `Bgs`, `Y`, `Y_HU_0`, `Y_HU_15`, and `Y_HU_tot`. The matrix method returns `x` with the corresponding output columns updated.

Examples

```
TT <- c(0, 1000, 1400, 2000)
HUg <- calc_HUg(
  TT, TT_R1 = 1200, TT_R6 = 2000,
  HUg_R1 = 85, HUg_R6 = 32, HUg_min = 15
)
calc_Y_grain(
  c(0, 4, 8, 12),
  TT = TT, HUg = HUg,
  TT_R1 = 1200, TT_R6 = 2000,
  HUg_R1 = 85, HUg_R4 = 70, HUg_R6 = 32, HUg_min = 15,
  estress_b = 1
)
```

`calc_Y_hi` *Calculate harvest-index yield*

Description

Generic and method-specific functions to calculate crop yield from total biomass and a harvest index (`Y = B * HI`).

Usage

```
calc_Y_hi(x = NULL, ...)

## S3 method for class '`NULL`'
calc_Y_hi(x = NULL, HI, ...)

## S3 method for class 'numeric'
calc_Y_hi(x, HI, iday_crop_dev = rep(1, length(x)), B = x, ...)

## S3 method for class 'matrix'
calc_Y_hi(
  x,
  B = x[iday, B_name],
  HI = x[iday, HI_name],
  iday = seq.int(nrow(x)),
  B_name = "B",
  HI_name = "HI",
  Y_name = "Y",
  ...
)
```

Arguments

`x`	Input object. Can be `numeric` or `matrix`.
`...`	Additional arguments passed to methods.
`HI`	numeric Harvest Index values.
`iday_crop_dev`	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as `x`.
`B`	numeric Total above-ground biomass ($t\ ha^{-1}$). For the numeric method, defaults to `x`.
`iday`	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
`B_name`	character Column or layer name containing biomass values (default `"B"`).
`HI_name`	character Column or layer name used to store calculated harvest index values (default `"HI"`).
`Y_name`	character Column or layer name to store calculated yield values.

Value

An object of the same class as x with calculated yield (Y) values.

Examples

```
# Numeric example
calc_Y_hi(1:100, HI = 0.5)
# Matrix example
calc_Y_hi(cbind(B = 1:100, HI = rep(0.5, 100)))
```

`calc_Y_sugar` *Calculate sugar yield (Y_sugar)*

Description

Generic and methods to compute Y_sugar = B_root * sugar_accumulation. Methods are provided for numeric, matrix/data.frame.

Usage

```
calc_Y_sugar(x, ...)

## S3 method for class 'numeric'
calc_Y_sugar(x, B_root = x, sugar_accumulation, ...)

## S3 method for class 'matrix'
calc_Y_sugar(
  x,
  B_root = x[iday, B_root_name],
  sugar_accumulation = x[iday, sugar_accumulation_name],
  iday = seq.int(nrow(x)),
  B_root_name = "B_root",
  sugar_accumulation_name = "sugar_accumulation",
  Y_sugar_name = "Y_sugar",
  ...
)
```

Arguments

x	Object to dispatch on.
...	Additional arguments passed to methods.
B_root	numeric Root biomass (numeric or extracted from x).
sugar_accumulation	numeric Sugar accumulation factor (numeric or extracted from x).

`iday` integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).

`B_root_name, sugar_accumulation_name, Y_sugar_name` character Names of layers/columns in x.

Value

Numeric result (for numeric method) or modified x with Y_sugar updated.

`calc_zroot` *Calculate root depth*

Description

This function calculates the root depth of a plant.

Usage

```
calc_zroot(x, ...)

## S3 method for class 'numeric'
calc_zroot(
  x,
  zroot_init,
  zmax,
  iday_crop_dev = rep(1, length(x)),
  dzroot = x,
  ...
)

## S3 method for class 'matrix'
calc_zroot(
  x,
  dzroot = x[iday, dzroot_name],
  zroot_init,
  zmax,
  iday = seq.int(nrow(x)),
  dzroot_name = "dzroot",
  zroot_name = "zroot",
  ...
)
```

Arguments

`x` Input object. Can be a numeric, data.frame, or matrix.

`...` Further arguments passed to or from other methods.

`zroot_init` numeric Initial root depth.

zmax	numeric Maximum root or soil depth.
iday_crop_dev	integer Used like a boolean to indicate a vector of days within the crop development period (1) or outside (0). Default is a vector of 1s with the same length as x.
dzroot	numeric Increment of root depth.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
dzroot_name	character Name of the column in x representing daily root depth increments.
zroot_name	character Layer or column name to store calculated root depth.

Value

A numeric, matrix, data.frame or depending on the input type.

`calc_zrootp` *Calculate potential root depth*

Description

This function calculates the potential root depth of a plant.

Usage

```
calc_zrootp(x, ...)

## S3 method for class 'numeric'
calc_zrootp(
  x,
  degred,
  TT_root_install,
  sowing_depth,
  seuil,
  zroot_max,
  TT = x,
  ...
)

## S3 method for class 'matrix'
calc_zrootp(
  x,
  TT = x[iday, TT_name],
  degred = x[iday, degred_name],
  TT_root_install,
  sowing_depth = 0,
  seuil,
  zroot_max,
```

```
  iday = seq.int(nrow(x)),
  TT_name = "TT",
  degred_name = "degred",
  zrootp_name = "zrootp",
  ...
)

## S3 method for class 'model_init'
calc_zrootp(x, crop, soil, dates, iday_sowing, iday_harvest, ...)
```

Arguments

x	Input object. Can be a numeric, data.frame, or matrix.
...	Further arguments passed to or from other methods.
degred	numeric Daily root depth growth rate per degree day.
TT_root_install	numeric Thermal time for root installation.
sowing_depth	numeric Depth of sowing.
seuil	numeric Initial root depth at sowing.
zroot_max	numeric Maximum root or soil depth.
TT	numeric Thermal time.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).
TT_name	character Name of the column in x representing thermal time.
degred_name	character Name of the column in x representing daily root depth growth rate per degree day.
zrootp_name	character Layer or column name to store calculated potential root depth.
crop	list Crop parameters (for model_init method).
soil	list Soil parameters (for model_init method).
dates	data.frame Data frame with simulation dates (for model_init method).
iday_sowing	integer Day of year of sowing (for model_init method).
iday_harvest	integer Day of year of harvest (for model_init method).

Value

A numeric, matrix, data.frame or depending on the input type.

`create_model_inputs` *Create Model Inputs (docs from createInputsModel)*

Description

Generic and S3 methods to create model input objects for the Optirrig model. These methods handle different input types, such as file paths, data frames, and lists, and transform them into a standardized model input structure.

Usage

```
create_model_inputs(x, ...)

## S3 method for class 'character'
create_model_inputs(x, options = list(log = TRUE), ..., cfg = load_config())

## S3 method for class 'data.frame'
create_model_inputs(x, options = list(log = TRUE), ..., cfg = load_config())

## S3 method for class 'list'
create_model_inputs(x, options = list(log = TRUE), ..., cfg = load_config())
```

Arguments

`x`	An object to create model inputs from. Can be a file path (character), a data.frame, or a list. Flat lists are supported for single runs, including in-memory `data.frame` values for inputs such as `climate` and `irrigation`.
`...`	Additional arguments passed to methods.
`options`	list Options for processing model inputs. Currently supports: `log` (logical) to enable or disable logging (default: `TRUE`). `inputs_path` (character) to specify a base directory where observed/input files referenced by name (e.g. climate) are located.
`cfg`	A configuration list, typically loaded using `load_config()`, which contains user-defined settings and paths.

Details

create_model_inputs.character Reads a file from the given path and processes it as model input. The file must exist.

create_model_inputs.data.frame Transforms a data frame into model input objects, mapping column names and categories using data correspondence tables. Duplicate `run_id` values are suffixed in row order (`Run_1`, `Run_2`, ...) so that every row remains a distinct simulation. Handles loading of external data files (e.g., climate) as needed.

create_model_inputs.list Validates and processes a list of model input objects. Single flat run lists are normalized to the categorized structure expected downstream, including direct `data.frame` climate and irrigation inputs. Returns a list of standardized model input objects with class `model_inputs`.

Value

A list of model input objects with class `model_inputs`, each containing the components: `climate`, `crop`, `soil`, and `run`.

See Also

`get_vars_data`, `get_data_path`, `load_config`

Examples

```
## Not run:
# From a data frame
df <- data.frame(run_id = 1, ...)
model_inputs <- create_model_inputs(df)

# From a file path
model_inputs <- create_model_inputs("inputs/model_data.csv")

## End(Not run)
```

---

create_model_inputs_climate

Create climate model inputs

Description

Generic and method-specific functions for processing climate model inputs.

Usage

```
create_model_inputs_climate(x, ...)

## S3 method for class 'list'
create_model_inputs_climate(x, ..., cfg = load_config())

## S3 method for class 'character'
create_model_inputs_climate(x, ..., options = list(), cfg = load_config())

## S3 method for class 'data.frame'
create_model_inputs_climate(x, ...)
```

Arguments

`x`	Input object. Supported classes are `list`, `character`, and `data.frame`.
`...`	Additional arguments passed to methods.

cfg	A configuration list, typically loaded using load_config(), which contains user-defined settings and paths.
options	list Optional options. When x is a file name, you can set options$inputs_path to a base directory where input files are located.

Value

A data.frame or climate object.

`create_model_inputs_crop` *Create crop model inputs*

Description

Generic and method-specific functions for processing crop model inputs.

Usage

```
create_model_inputs_crop(x, ...)

## S3 method for class 'list'
create_model_inputs_crop(x, ..., cfg = load_config())
```

Arguments

x	Input object. Can be only a list for now.
...	Additional arguments passed to methods.
cfg	A configuration list, typically loaded using load_config(), which contains user-defined settings and paths.

Details

Responsible for adding any missing parameters using the default configuration (previously handled in createInputsParams).

Value

A list of crop parameter

`create_model_inputs_irrigation` *Create Irrigation Model Inputs*

Description

This function serves as a generic interface for creating and processing input parameters required by irrigation models. It supports multiple input types, including `list`, `character` (e.g., file paths), and `data.frame` objects, allowing for flexible integration with various data sources and formats. Method-specific implementations handle the conversion and validation of these inputs to ensure they conform to the requirements of the irrigation modeling workflow.

Usage

```
create_model_inputs_irrigation(x, ...)

## S3 method for class 'list'
create_model_inputs_irrigation(x, ..., cfg = load_config())

## S3 method for class 'character'
create_model_inputs_irrigation(x, ..., options = list(), cfg = load_config())

## S3 method for class 'data.frame'
create_model_inputs_irrigation(x, ...)
```

Arguments

`x`	An input object containing data for the irrigation model. Supported types include: • `list`: A named list of parameter values. • `character`: A file path or identifier for loading input data. • `data.frame`: A tabular dataset with columns representing model parameters used by the irrigation model.
`...`	Additional arguments passed to specific methods for further customization or processing.
`cfg`	A configuration list, typically loaded using `load_config()`, which contains user-defined settings and paths.
`options`	list Optional options. When x is a file name, you can set `options$inputs_path` to a base directory where input files are located.

Details

The function is designed to be extended via S3 methods, enabling custom handling for different input types. Each method is responsible for extracting, validating, and formatting the input data into a standardized list of irrigation parameters. This ensures consistency and reliability in downstream modeling tasks.

Value

A `list` containing the processed irrigation model parameters, ready for use in simulation or analysis.

Examples

```
irrigation <- data.frame(
  Date = c("2020-06-01", "2020-06-08"),
  Dose = c(20, 25)
)

create_model_inputs_irrigation(irrigation)
```

`create_model_inputs_run` *Create run model inputs*

Description

Generic and method-specific functions for processing run model inputs.

Usage

```
create_model_inputs_run(x, ...)

## S3 method for class 'list'
create_model_inputs_run(x, ..., cfg = load_config())
```

Arguments

`x`	Input object. Must be a `list` describing the run configuration.
`...`	Additional arguments passed to methods.
`cfg`	A configuration list, typically loaded using `load_config()`, which contains user-defined settings and paths.

Value

A list of run parameters.

`create_model_inputs_soil` *Create soil model inputs*

Description

Generic and method-specific functions for processing soil model inputs.

Usage

```
create_model_inputs_soil(x, ...)

## S3 method for class 'list'
create_model_inputs_soil(x, ..., options = list(), cfg = load_config())
```

Arguments

`x`	Input object. Must be a `list` describing soil parameters or file references.
`...`	Additional arguments passed to methods.
`options`	list Optional options. Some soil parameters can be provided as filenames; set `options$inputs_path` to a base directory where those files are located.
`cfg`	A configuration list, typically loaded using `load_config()`, which contains user-defined settings and paths.

Value

A list of soil parameters.

`get_data_path` *Give the complete path of the data and cache data on local disk*

Description

It supports Nextcloud public links in case of using cloud. In this case, this function can download the file in a temporary folder and returns the path of the downloaded file. This functionality is useful for `read_excel` from the package *readxl* (See: https://stackoverflow.com/a/41368947/5300212).

Usage

```
get_data_path(
  path = NULL,
  ...,
  base_dir = NULL,
  resolve = FALSE,
  use_cache = FALSE,
  cache = Sys.getenv("PKG_DATA_CACHE", file.path(dirname(tempdir()),
    utils::packageName())),
  cfg = load_config()
)
```

Arguments

path	character representing the path of the data in the database
...	character path elements to add after path
base_dir	character Optional base directory used when resolve = TRUE. Typically set to options$inputs_path to locate input files not stored under the configured project inputs folder.
resolve	logical If TRUE, try to resolve the file by checking whether the provided path (via ...) exists as-is, then whether it exists under base_dir, before falling back to the configured paths.
use_cache	logical download the data and use cache location
cache	character path where downloaded files from cloud are cached (See details)
cfg	a config object. Configuration to use. See load_config for details

Details

The cache directory is by default located in the system temporary folder in a subdirectory named as the package name or the value of the environment variable "PKG_DATA_CACHE" if it is defined.

Value

character with the path of the file to read either on:

- the local storage if cfg$data$mode == "local"
- the cache directory if cfg$data$mode != "local" and use_cache == TRUE
- the cloud URL if cfg$data$mode != "local" and use_cache == FALSE

`get_iday` *Get model time indices*

Description

Generic and method-specific functions to get model time indices. These indices include:

- `iday_emerg`: Day of emergence (first day with `TT_LAI` > 0).
- `iday_root_install`: Day of root installation (first day with `TT_root` > 0).
- `iday_crop_dev`: Days of crop development (from emergence or root installation to harvest).
- `iday_strategy`: Irrigation strategy period identifier (0 = none, k = period k).
- `iday_constraints`: Irrigation constraints period identifier.
- `iday_no_irrig`: Binary indicator for irrigation allowed (1) or not (0).
- `iday_waterturn`: Binary indicator for water turn days.

Usage

```
get_iday(x, ...)

## S3 method for class 'model_init'
get_iday(x, crop, irrigation, dates, ...)

## S3 method for class 'matrix'
get_iday(
  x,
  iday_harvest,
  iday_strategy_start,
  iday_strategy_end,
  iday_constraints_start,
  iday_constraints_end,
  iday_no_irrig,
  WTurn_ndays,
  LTurn_ndays,
  tbis_mod,
  TT_d,
  ...
)
```

Arguments

`x`	Input object. Must be of class `model_init` or `matrix`.
`...`	Additional arguments passed to methods.
`crop`	list Crop parameters.
`irrigation`	list Irrigation parameters.

`dates`	base::Date Vector of dates corresponding to the simulation period.
`iday_harvest`	integer Harvest day index.
`iday_strategy_start`	integer Start of irrigation strategy period(s).
`iday_strategy_end`	integer End of irrigation strategy period(s).
`iday_constraints_start`	integer Start of irrigation constraints period(s).
`iday_constraints_end`	integer End of irrigation constraints period(s).
`iday_no_irrig`	integer Day indices where irrigation is not allowed.
`WTurn_ndays`	integer Water turn period in days (can be vector for multiple periods).
`LTurn_ndays`	integer Logical turn period in days (can be vector for multiple periods).
`tbis_mod`	integer TBIS (Timing-Based Irrigation Strategy) mode (0 = off, >0 = on).
`TT_d`	numeric Thermal time threshold for TBIS mode (degrees-days).

Value

An object of the same class as `x` with added time index columns/layers.

`get_model_output_path` *Generate Output File Path for Model Results*

Description

Constructs a file path for saving model output based on the output's type and a given name. The file extension is determined by the class of `model_output`:

- `"csv"` for matrices
- `"RDS"` for all other types

Usage

```
get_model_output_path(model_output, model_output_name, path_outputs)
```

Arguments

`model_output`	The model output object whose type determines the file extension.
`model_output_name`	character string specifying the base name for the output file.
`path_outputs`	character string specifying the directory path where the output file will be saved.

Value

character string representing the full file path for the model output.

get_vars_data *Retrieve Metadata for Parameters and Variables*

Description

Retrieves metadata for parameters and variables from a CSV file included in the package. The function reads the specified CSV file and extracts information from specified columns based on matching values.

Usage

```
get_vars_data(
  filename = "params.desc.csv",
  df = read(system.file("extdata/", filename, package = "OptirrigCORE"), show_col_types =
    FALSE),
  match_from,
  match_with,
  values = na.omit(unique(df[, match_from, drop = TRUE])),
  ...,
  cfg = load_config()
)
```

Arguments

filename	Character. Name of the CSV file containing metadata. Defaults to "params.desc.csv". Can also be "vars.desc.csv".
df	Data frame. Metadata table. By default, it is read from the specified CSV file.
match_from	Character. Name of the column in df to match values against.
match_with	Character. Name of the column in df from which to retrieve values.
values	Character vector. Values to match in the match_from column. By default, all unique non-NA values from that column are used.
...	Additional arguments (currently unused).
cfg	List. Configuration object, typically loaded via load_config().

Value

Named vector of values from the match_with column, matched by values in the match_from column. Names correspond to the matched values.

`init_model` *Initialize the model simulation*

Description

Generic and method-specific functions to initialize the model simulation.

Usage

```
init_model(x, ...)

## S3 method for class 'model_inputs'
init_model(x, ...)

## S3 method for class 'model_input'
init_model(x, options = list(log = TRUE, yield_method = "hi"), ...)

## S3 method for class 'matrix'
init_model(x, crop, soil, run, dates, irrigation, ...)
```

Arguments

`x`	Input object. Can be `model_inputs` or `model_input`. See `create_model_inputs` output.
`...`	Additional arguments passed to methods.
`options`	list Initialization options: `log` enables logging and `yield_method` selects `"hi"`, `"grain"`, or `"sugar"`.
`crop`	list Crop-specific parameters.
`soil`	list Soil-specific parameters.
`run`	list Run-specific parameters.
`dates`	base::Date Simulation dates.
`irrigation`	list Irrigation-specific parameters.

Value

An object of class `model_inits` or `model_init` named with the run identifier. The `model_inits` object is a list of `model_init` objects. The `model_init` object is a list of:

- `init`: The initialized model parameters. This is the main object which is processed by the model, starting with the climate data.
- `crop`: The crop-specific parameters.
- `soil`: The soil-specific parameters.
- `run`: The run-specific parameters.
- `dates`: The simulation dates.
- `irrigation`: The irrigation-specific parameters.
- `iday`: The day-specific parameters.

`init_model_climate` *Initialize the model simulation - climate*

Description

Generic and method-specific functions to initialize the model simulation for climate data.

Usage

```
init_model_climate(x, ...)

## S3 method for class 'model_input'
init_model_climate(x, soil, irrigation, ...)

## S3 method for class 'data.frame'
init_model_climate(
  x,
  date_start,
  date_end,
  mulch,
  gge,
  date_name = "Date",
  ...
)

## S3 method for class 'matrix'
init_model_climate(x, ...)

## S3 method for class 'list'
init_model_climate(x, ...)
```

Arguments

`x`	Input object. Can be `model_inputs` or `model_input`. See `create_model_inputs` output.
`...`	Additional arguments passed to methods.
`soil`	list Soil parameters.
`irrigation`	list Irrigation parameters.
`date_start`	base::Date Start date of the simulation.
`date_end`	base::Date End date of the simulation.
`mulch`	numeric Soil mulch parameter.
`gge`	numeric Irrigation gge parameter.
`date_name`	character Name of the date column or dimension.

Value

An object of class `model_init` backed by a matrix.

Examples

```
climate <- data.frame(
  Date = as.Date("2020-01-01") + 0:4,
  ETP = c(1.2, 1.3, 1.1, 1.4, 1.2),
  Rain = c(0, 2, 0, 1, 0)
)

init_model_climate(
  climate,
  date_start = as.Date("2020-01-02"),
  date_end = as.Date("2020-01-04"),
  mulch = 0,
  gge = 1
)
```

`init_model_crop_dev` *Initialize the model simulation - crop development*

Description

Generic and method-specific functions to initialize the model simulation for crop development.

Usage

```
init_model_crop_dev(x, ...)

## S3 method for class 'model_init'
init_model_crop_dev(x, crop, soil, irrigation, dates, yield_method = "hi", ...)
```

Arguments

`x`	Input object. Can be `model_inputs` or `model_input`. See `create_model_inputs` output.
`...`	Additional arguments passed to methods.
`crop`	list Crop parameters.
`soil`	list Soil parameters.
`irrigation`	list Irrigation parameters.
`dates`	base::Date Dates for the simulation.
`yield_method`	character Yield calculation method: `"hi"`, `"grain"`, or `"sugar"`.

Details

This function initializes the model simulation for crop development by calling:

- `calc_ET0()`: calculate reference evapotranspiration
- `calc_TT()`: calculate thermal time from sowing, emergence and root installation
- `calc_LAIp()`: calculate potential leaf area index (including TT_norm, LT, alpha, etc.)
- `calc_zrootp()`: calculate potential root depth (including degred, etc.)
- `calc_Bp()`: calculate potential biomass (including IR, PAR, etc.)
- `get_iday()`: get crop development periods

Value

An object of class `model_init` backed by a matrix.

`init_model_irrigation` *Initialize the model simulation - climate*

Description

Generic and method-specific functions to initialize the model simulation for climate data.

Usage

```
init_model_irrigation(x, ...)

## S3 method for class 'model_input'
init_model_irrigation(x, dates, irrigation, ...)

## S3 method for class 'matrix'
init_model_irrigation(x, dates, irrigation, ...)
```

Arguments

`x`	Input object. Can be `model_inputs` or `model_input`. See `create_model_inputs` output.
`...`	Additional arguments passed to methods.
`dates`	base::Date Simulation dates.
`irrigation`	list Irrigation parameters.

Value

An object of class `model_init` backed by a matrix.

init_model_vars *Initialize variables for the model simulation*

Description

Generic and method-specific functions to initialize variables for the model simulation.

Usage

```
init_model_vars(x, ...)

## S3 method for class 'model_init'
init_model_vars(x, ...)

## S3 method for class 'matrix'
init_model_vars(x, vars, ...)
```

Arguments

x	Input object of class model_init (wrapping a matrix) or a plain matrix.
...	Additional arguments passed to specific methods.
vars	character Vector of variable names to ensure are present in the output.

Value

An object of class model_init with the specified variables initialized.

inputs_meta *Retrieve inputs metadata (categories, types, subcategories)*

Description

Retrieve inputs metadata (categories, types, subcategories)

Usage

```
inputs_meta()
```

Value

A list with elements:

- categories: named character vector of parameter categories
- types: named character vector of parameter types
- subcategories: named character vector of parameter subcategories

load_config *Load Package Configuration File*

Description

Loads and merges the default and user configuration files for the package. The function determines the appropriate configuration file to use, merges it with the package default, and sets up internal paths and metadata.

Usage

```
load_config(
  path_config_ini = "config.user.ini",
  path_directory = system.file("extdata", package = pkg),
  pkg = utils::packageName(),
  ini_sources = NULL,
  ...,
  path_default_cfg = system.file("config.user.ini", package = pkg)
)
```

Arguments

path_config_ini	Path to the user-specific .ini configuration file. Default is "config.user.ini".
path_directory	Optional. Directory to look for the configuration file. Defaults to the package's extdata directory.
pkg	Name of the package. Defaults to the current package name.
ini_sources	Optional named list defining where reference .ini files are searched for each input type. Supported names are run, crop, soil, and irrigation. Each entry can be a package name, a directory path, or a character vector mixing both. Sources are searched in order, so earlier entries take precedence and later entries act as fallbacks.
...	Additional arguments passed to ini::read.ini.
path_default_cfg	Path to the default configuration file within the package. Defaults to "config.user.ini" in the package.

Details

The function first attempts to load a user configuration file from the specified path. If not found, it falls back to the default configuration file provided by the package. The two configurations are merged, with user settings taking precedence. Internal paths and package metadata (such as version and name) are added to the configuration list. If a valid configuration cannot be found, the function stops with an error.

Value

A list containing the merged configuration settings, including internal paths and package metadata.

Examples

```
# Load configuration with defaults
cfg <- load_config()
str(cfg)
```

MAIZE_21LAVALETTE_B *Observed biomass for maize trial (2021, Lavalette)*

Description

Daily observed biomass values for the maize 2021 Lavalette project.

Usage

```
data(MAIZE_21LAVALETTE_B)
```

Format

A data.frame

Source

Generated as part of the Maize 2021 project.

Examples

```
data(MAIZE_21LAVALETTE_B)
head(MAIZE_21LAVALETTE_B)
```

MAIZE_21LAVALETTE_clim *Climate data for maize trial (2021, Lavalette)*

Description

Daily climate inputs (e.g., temperature, radiation, ETP, rainfall) used for the maize 2021 Lavalette project.

Usage

```
data(MAIZE_21LAVALETTE_clim)
```

Format

A data.frame

Source

Generated as part of the Maize 2021 project.

Examples

```
data(MAIZE_21LAVALETTE_clim)
head(MAIZE_21LAVALETTE_clim)
```

MAIZE_21LAVALETTE_irrig

Irrigation schedule for maize trial (2021, Lavalette)

Description

Irrigation schedule used for the maize 2021 Lavalette project.

Usage

```
data(MAIZE_21LAVALETTE_irrig)
```

Format

A data.frame

Source

Generated as part of the Maize 2021 project.

Examples

```
data(MAIZE_21LAVALETTE_irrig)
head(MAIZE_21LAVALETTE_irrig)
```

MAIZE_21LAVALETTE_LAI *Observed leaf area index for maize trial (2021, Lavalette)*

Description

Daily observed leaf area index (LAI) for the maize 2021 Lavalette project.

Usage

```
data(MAIZE_21LAVALETTE_LAI)
```

Format

A data.frame

Source

Generated as part of the Maize 2021 project.

Examples

```
data(MAIZE_21LAVALETTE_LAI)
head(MAIZE_21LAVALETTE_LAI)
```

MAIZE_21LAVALETTE_RES *Observed water reservoir for maize trial (2021, Lavalette)*

Description

Daily observed water reservoir values for the maize 2021 Lavalette project.

Usage

```
data(MAIZE_21LAVALETTE_RES)
```

Format

A data.frame

Source

Generated as part of the Maize 2021 project.

Examples

```
data(MAIZE_21LAVALETTE_RES)
head(MAIZE_21LAVALETTE_RES)
```

`parse_date_time` *Parse date strings into Date objects*

Description

This function attempts to parse character date strings using lubridate::parse_date_time with a set of common orders. If a string cannot be parsed it yields NA in the result.

Usage

```
parse_date_time(date_str)
```

Arguments

`date_str`	character vector of date/time strings to parse.

Details

The parsing orders tried (in no particular priority) are stored in c("dmy", "mdy", "ymd", "Ymd", "dmY", "mdY", "Ydm"). Parsing is performed with exact = FALSE to allow flexible separators and variants supported by lubridate.

Value

A base::Date vector corresponding to parsed input. Unparseable entries are returned as NA.

See Also

lubridate::parse_date_time

Examples

```
# Single string
parse_date_time("01-02-2003")
# Vector of mixed formats
parse_date_time(c("2003/02/01", "01 Mar 2004", "31-12-1999"))
```

`read` *Read a file by inferring the reader from the file extension*

Description

A convenience wrapper that dispatches to an appropriate file-reading function based on the extension of the supplied path. Supported extensions and their readers:

- csv, dat, tsv: readr::read_delim
- ini: ini::read.ini
- xls, xlsx: readxl::read_excel
- ods: readODS::read_ods
- RDS: base::readRDS

The function simply determines the file extension, selects the reader, and forwards the path and any additional arguments to that reader.

Usage

```
read(path, ..., cache_enabled = TRUE)
```

Arguments

`path`	character Path to the file to be read. Must include a file extension (e.g. "data.csv"). If the path has no extension the function throws an error.
`...`	Additional arguments passed through to the underlying reader function (for example, delim, col_types for readr or sheet for readxl).
`cache_enabled`	logical Whether to reuse a cached object when the file path, size and modification time are unchanged.

Details

If the extension is not recognized (i.e. there is no matching reader in the internal switch), the call will fail when attempting to invoke the reader. Ensure the necessary packages (readr, readxl, readODS, ini) are installed when reading those formats.

Value

The object returned by the chosen reader. Typically a data frame or tibble for delimited and spreadsheet formats, a list for INI files, or any R object stored in an RDS file.

See Also

readr::read_delim, readxl::read_excel, readODS::read_ods, ini::read.ini, base::readRDS

Examples

```
## Not run:
# Read a comma-separated file
read("path/to/data.csv")

# Read the first sheet of an Excel workbook
read("path/to/data.xlsx", sheet = 1)

# Read an RDS file
read("path/to/object.RDS")

## End(Not run)
```

`run_model` *Run Optirrig model*

Description

Generic and method-specific functions to run Optirrig model simulations. These functions handle different input types, such as file paths, data frames, and lists.

Usage

```
run_model(x = NULL, ...)

## S3 method for class '`NULL`'
run_model(x, options = list(), ...)

## S3 method for class 'character'
run_model(
  x,
 options = list(log = TRUE, write_outputs = TRUE, force = FALSE, return_outputs = TRUE,
   inputs_path = NULL, outputs_path = NULL, plan = NULL, workers = NULL, yield_method =
    "hi", yield_humidity = "0"),
  ...
)

## S3 method for class 'data.frame'
run_model(
  x,
 options = list(log = TRUE, write_outputs = TRUE, force = FALSE, return_outputs = TRUE,
   inputs_path = NULL, outputs_path = NULL, plan = NULL, workers = NULL, yield_method =
    "hi", yield_humidity = "0"),
  ...
)
```

```
## S3 method for class 'list'
run_model(
  x,
 options = list(log = TRUE, write_outputs = TRUE, force = FALSE, return_outputs = TRUE,
   inputs_path = NULL, outputs_path = NULL, plan = NULL, workers = NULL, yield_method =
    "hi", yield_humidity = "0"),
  ...
)

## S3 method for class 'model_inputs'
run_model(
  x,
 options = list(log = TRUE, write_outputs = TRUE, force = FALSE, return_outputs = TRUE,
    outputs_path = NULL, yield_method = "hi", yield_humidity = "0"),
  ...
)

## S3 method for class 'model_input'
run_model(
  x,
 options = list(log = TRUE, write_outputs = TRUE, force = FALSE, return_outputs = TRUE,
    outputs_path = NULL, yield_method = "hi", yield_humidity = "0"),
  ...
)

## S3 method for class 'model_inits'
run_model(
  x,
 options = list(log = TRUE, write_outputs = TRUE, force = FALSE, return_outputs = TRUE,
    outputs_path = NULL, yield_method = "hi", yield_humidity = "0"),
  ...
)

## S3 method for class 'model_init'
run_model(x, options = list(), ...)
```

Arguments

x	Input object. Can be character, data.frame, or list. Flat lists can describe a single run directly, including in-memory data.frame values for climate and irrigation.
...	Additional arguments passed to methods as well as to create_model_inputs() and init_model().
options	list A list of options for running the model: • log: logical Whether to log progress messages (default: TRUE). • write_outputs: logical Whether to write the model outputs to files (default: TRUE). One CSV file is written per simulation.

- force: logical Whether to run the simulation even when its output file already exists (default: FALSE).
- return_outputs: logical Whether to return the model outputs in R (default: TRUE). When FALSE, run_model() returns NULL.
- inputs_path: character Optional base directory to resolve observed/input files referenced by name in parameters (e.g. climate = "my.csv"). It does not alter the path supplied as x.
- outputs_path: character Optional directory where simulation output files are looked up and written. When NULL, the directory from the project configuration is used (default: NULL).
- plan: character A future execution plan. NULL uses sequential execution (default: NULL).
- workers: integer Number of workers for a non-sequential plan. NULL uses parallel::detectCores() - 1 (default: NULL).
- output_vars: character Optional names of output variables to keep. When provided, include "Date" explicitly to retain it (default: NULL, meaning all variables).
- overwrite: logical Whether a completed simulation may replace an existing output file. FALSE preserves the existing file; NULL allows replacement (default: NULL).
- yield_method: character Yield calculation method. One of "hi", "grain", or "sugar" (default: "hi").
- yield_humidity: character Grain-moisture basis assigned to Y. One of "0", "15", or "tot" (default: "0").

Details

Input and output paths:

inputs_path is a base directory for files named *inside* a scenario, such as climate, irrigation, or soil files. If x itself is a parameter-file path, that path must already exist; inputs_path is not prepended to x.

outputs_path overrides the output directory stored in the initialized run. The same directory is used both to find a previous result and to write a new result.

Returning, writing, and reusing outputs:

return_outputs and write_outputs are independent:

- both TRUE: return the results in R and write them to disk;
- return_outputs = TRUE, write_outputs = FALSE: return results only;
- return_outputs = FALSE, write_outputs = TRUE: write results and return NULL;
- both FALSE: run the simulation, discard the result, and return NULL.

Before running, run_model() checks for an existing output file in outputs_path. With force = FALSE, that file is reused and the simulation is skipped, independently of write_outputs. Set force = TRUE to recompute. If recomputing encounters an existing file, overwrite = FALSE keeps that file unchanged; overwrite = TRUE or NULL allows writing the new result.

Workflow:

The simulation workflow consists of `create_model_inputs()`, `init_model()`, and the internal `model_init` method of `run_model()`. The `model_inputs`, `model_input`, `model_inits`, and `model_init` methods connect these stages.

Value

If `options$return_outputs = TRUE`, an object of class `"model_outputs"`: a named list containing one output data.frame per simulation. If `options$return_outputs = FALSE`, `NULL`.

Examples

```
## Not run:
outputs <- run_model(
  input,
  options = list(
    inputs_path = "data/inputs",
    outputs_path = "data/outputs",
    write_outputs = TRUE,
    return_outputs = TRUE,
    force = FALSE
  )
)

# In-memory results without writing files
outputs <- run_model(
  input,
  options = list(write_outputs = FALSE, return_outputs = TRUE)
)

## End(Not run)
```

---

run_model_crop_dev *Crop development module*

Description

Generic and method-specific functions to run the crop development module of the Optirrig model. This module includes calculations for stress factors, leaf area index (LAI), biomass (B), yield (Y), and root depth (zroot).

Usage

```
run_model_crop_dev(x, ...)

## S3 method for class 'model_output'
run_model_crop_dev(
  x,
```

```
  crop,
  soil,
  iday,
  yield_method = "hi",
  yield_humidity = "0",
  ...
)
```

Arguments

x	Input object. Must be of class model_output (matrix handling occurs downstream).
...	Additional arguments passed to methods.
crop	list Crop parameters.
soil	list Soil parameters.
iday	integer Current day index of the simulation.
yield_method	character Yield calculation method: "hi", "grain", or "sugar".
yield_humidity	character Grain-moisture basis assigned to Y: "0", "15", or "tot" for simulated moisture. This option does not alter sugar yield.

Value

An object of class model_output with updated daily variables.

---

run_model_crop_dev_B *Biomass part of model crop development module*

Description

Generic and method-specific functions to compute biomass (B) during crop development.

Usage

```
run_model_crop_dev_B(x, ...)

## S3 method for class 'model_output'
run_model_crop_dev_B(x, crop, iday, ...)

## S3 method for class 'matrix'
run_model_crop_dev_B(x, iday, RUE, ...)
```

Arguments

x	Input object. Must be of class model_output (matrix handling occurs downstream).
...	Additional arguments passed to methods.
crop	list Crop parameters.
iday	integer Current day index of the simulation.
RUE	numeric Radiation use efficiency ($g\ MJ^{-1}$ of PAR).

Value

An object of class model_output with updated daily variables.

---

run_model_crop_dev_B_root

Run/update crop root biomass (B_root) for a single day

Description

Generic S3 dispatcher and method implementations to compute or update daily root biomass (B_root) within a crop simulation state.

Usage

```
run_model_crop_dev_B_root(x, ...)

## S3 method for class 'model_output'
run_model_crop_dev_B_root(x, crop, iday, ...)

## S3 method for class 'matrix'
run_model_crop_dev_B_root(x, crop, iday, ...)
```

Arguments

x	Object to dispatch on. For methods, this is typically: • a numeric matrix of model state variables (rows = days, cols = variables), • a 'model_output' object that forwards to the matrix method,
...	Additional arguments passed to internal helpers (e.g. calc_B_root) or to other methods.
crop	List or similar containing crop-specific parameters. The matrix method expects at least TT_B_root5 and TT_B_root70 to compute development.
iday	Integer. Index of the current day (row) to update in the state.

Details

The matrix method computes B_root for the given day using the helper calc_B_root. It reads the current total biomass (B) at row `iday` and uses previous-day values (when available) to set B0 and B_root0; for day 1 these initial values default to 0. The 'model_output' method forwards to the matrix implementation. The method is currently a placeholder that invokes an interactive debugger.

Value

The updated object of the same class with B_root at row/index

is for debugging and does not currently return a transformed object.

`run_model_crop_dev_LAI`

LAI part of the crop development module

Description

Generic and method-specific functions to compute leaf area index (LAI) during crop development.

Usage

```
run_model_crop_dev_LAI(x, ...)

## S3 method for class 'model_output'
run_model_crop_dev_LAI(x, crop, soil, iday, ...)

## S3 method for class 'matrix'
run_model_crop_dev_LAI(
  x,
  iday,
  Kc_max,
  c1,
  c2,
  Ksol,
  estress_lai,
  LAImin = NA_real_,
  k = NA_real_,
  ...
)
```

Arguments

`x`	Input object. Must be of class `model_output` (matrix handling occurs downstream).
`...`	Additional arguments passed to methods.
`crop`	list Crop parameters.

soil	list Soil parameters.
iday	integer Current day index of the simulation.
Kc_max	numeric Maximum crop coefficient.
c1	numeric Coefficient related to the crop.
c2	numeric Coefficient related to the crop.
Ksol	numeric Maximum soil evaporation coefficient.
estress_lai	numeric Maximum LAI stress nocivity value.
LAImin	numeric Lower asymptote used to damp LAI decline during senescence.
k	numeric Damping coefficient controlling the maximum daily LAI loss when LAImin is active.

Value

An object of class model_output with updated daily variables.

---

run_model_crop_dev_stress_water

Water stress part of the crop development module

Description

Generic and method-specific functions to compute water stress factors during crop development.

Usage

```
run_model_crop_dev_stress_water(x, ...)

## S3 method for class 'matrix'
run_model_crop_dev_stress_water(x, iday, ...)
```

Arguments

x	Input object. Must be of class model_output (matrix handling occurs downstream).
...	Additional arguments passed to methods.
iday	integer Current day index of the simulation.

Value

An object of class model_output with updated daily variables.

run_model_crop_dev_Y *Yield part of the crop development module*

Description

Generic and method-specific functions to compute yield (Y) during crop development.

Usage

```
run_model_crop_dev_Y(x, ...)

## S3 method for class 'model_output'
run_model_crop_dev_Y(x, crop, iday, ...)

## S3 method for class 'matrix'
run_model_crop_dev_Y(x, iday, HIp, ...)
```

Arguments

x	Input object. Must be of class model_output (matrix handling occurs downstream).
...	Additional arguments passed to methods.
crop	list Crop parameters.
iday	integer Current day index of the simulation.
HIp	numeric Potential harvest index. Can be scalar or time-varying.

Value

An object of class model_output with updated daily variables.

run_model_crop_dev_Y_grain *Grain-yield part of the crop development module*

Description

Updates maize grain yield and its associated dry-biomass partition, grain-moisture and stress-nocivity variables.

Usage

```
run_model_crop_dev_Y_grain(x, ...)

## S3 method for class 'model_output'
run_model_crop_dev_Y_grain(x, crop, iday, ...)

## S3 method for class 'matrix'
run_model_crop_dev_Y_grain(x, iday, ...)
```

Arguments

x	Input object. Must be of class model_output (matrix handling occurs downstream).
...	Additional arguments passed to methods.
crop	list Crop parameters.
iday	integer Current day index of the simulation.

Value

An object of class model_output with updated maize grain-yield variables.

---

run_model_crop_dev_Y_sugar

Run model for crop development stage Y (sugar)

Description

Generic S3 dispatcher and methods to compute sugar accumulation (Y_sugar) for sugar beet. The calculation uses calc_Y_sugar and updates the input object (matrix or model_output) in-place (returns the updated object).

Usage

```
run_model_crop_dev_Y_sugar(x, ...)

## S3 method for class 'model_output'
run_model_crop_dev_Y_sugar(x, crop, iday, Y_name = "Y", ...)

## S3 method for class 'matrix'
run_model_crop_dev_Y_sugar(x, iday, Y_name = "Y", ...)
```

Arguments

x	Object to update: a matrix or a model_output.
...	Additional arguments passed to methods.
crop	character Crop identifier (used by the model_output method).
iday	integer Day index to update.
Y_name	character Output column name.

Details

- The matrix method expects the columns "B_root" together with "sugar_accumulation" and writes the result to Y_name for the given row iday.
- The model_output method delegates to NextMethod, forwarding iday.

Value

The input object x with its selected yield component updated.

See Also

calc_Y_sugar

---

run_model_crop_dev_zroot

Root development part of the crop development module

Description

Generic and method-specific functions to compute root depth (zroot) during crop development.

Usage

```
run_model_crop_dev_zroot(x, ...)

## S3 method for class 'model_output'
run_model_crop_dev_zroot(x, crop, soil, iday, ...)

## S3 method for class 'matrix'
run_model_crop_dev_zroot(
  x,
  iday,
  racplus,
  vroot,
  vrac,
  seuil,
  zmax,
  theta_raw,
  ...
)
```

Arguments

x	Input object. Must be of class model_output (matrix handling occurs downstream).
...	Additional arguments passed to methods.
crop	list Crop parameters.
soil	list Soil parameters.
iday	integer Current day index of the simulation.
racplus	numeric Root growth parameter.
vroot	numeric Root growth parameter.
vrac	numeric Root growth parameter.
seuil	numeric Root growth parameter.
zmax	numeric Maximum root or soil depth.
theta_raw	numeric Raw soil moisture content.

Value

An object of class model_output with updated daily variables.

---

run_model_finance *Run Financial Model*

Description

Generic function to run a financial model. This function dispatches methods based on the class of the input object.

Usage

```
run_model_finance(x, ...)

## S3 method for class 'model'
run_model_finance(x, ..., cfg = load_config())
```

Arguments

x	An object representing the model or data to be used.
...	Additional arguments passed to specific methods.
cfg	A configuration list, typically loaded using load_config(), which contains user-defined settings and paths.

Value

The result of the financial model run, depending on the method.

run_model_irrig_strat *Run Model for Irrigation Strategy*

Description

This function serves as a generic method dispatcher for running a model with an irrigation strategy.

Usage

```
run_model_irrig_strat(x, ...)

## S3 method for class 'model_output'
run_model_irrig_strat(x, soil, irrigation, iday, ...)
```

Arguments

x	An object representing the irrigation strategy or model input.
...	Additional arguments passed to specific methods.
soil	list Soil parameters.
irrigation	list Irrigation parameters.
iday	integer Integer vector. Indices of rows to process (for data.frames) or time steps (f).

Details

The run_model_irrig_strat function is a generic S3 method dispatcher. It allows for the implementation of specific methods for different classes of irrigation strategy objects. By defining methods for various classes, users can customize how the model is run depending on the type of irrigation strategy provided.

Value

The output depends on the specific method implemented for the class of x.

run_model_irrig_strat_dose *Run Irrigation Strategy Dose Model*

Description

Generic and S3 methods to run an irrigation strategy dose model on simulation outputs.

Usage

```
run_model_irrig_strat_dose(x, ...)

## S3 method for class 'model_output'
run_model_irrig_strat_dose(x, soil, strategy, constraints, iday, ...)

## S3 method for class 'matrix'
run_model_irrig_strat_dose(
  x,
  theta_fc,
  theta_wp,
  RU_target,
  RU_units,
  THETA_target,
  RU_thrld,
  THETA_thrld,
  ETM_thrld,
  ETM_ndays,
  SW_thrld,
  SW_ndays,
  PI_thrld,
  PI_ndays,
  T_thrld,
  T_ndays,
  dose_fix,
  dose_min,
  dose_max,
  WTurn_mod,
  LTurn_ndays,
  quota_max,
  quota_wind,
  iday,
  ...
)
```

Arguments

x	An object containing model output data. For the default method, a matrix of simulation results.
...	Additional arguments passed to methods.
soil	A list or object containing soil parameters, including field capacity (theta_fc) and wilting point (theta_wp).
strategy	A list that contains the strategy-specific thresholds, targets, turn configuration, and other rule parameters passed to the matrix method.
constraints	A list that defines dose and quota constraints (e.g., fixed/min/max dose, quota mode, quota partitions) applied when iday_constraints indicates they are active.

iday	Integer. The current simulation day index.
theta_fc	Numeric. Soil field capacity (volumetric water content).
theta_wp	Numeric. Soil wilting point (volumetric water content).
RU_target	Numeric vector. Target readily available water (RU) for each strategy.
RU_units	Integer vector. Units for RU thresholds (1: mm, 2: %).
THETA_target	Numeric vector. Target soil moisture (RFU) for each strategy.
RU_thrld	Numeric vector. Thresholds for RU triggering irrigation.
THETA_thrld	Numeric vector. Thresholds for soil moisture triggering irrigation.
ETM_thrld	Numeric vector. Thresholds for ETM (evapotranspiration) ratio triggering irrigation.
ETM_ndays	Integer vector. Number of days for ETM calculation.
SW_thrld	Numeric vector. Thresholds for water stress triggering irrigation.
SW_ndays	Integer vector. Number of days for water stress calculation.
PI_thrld	Numeric vector. Thresholds for cumulative precipitation + irrigation triggering irrigation.
PI_ndays	Integer vector. Number of days for PI calculation.
T_thrld	Numeric vector. Thresholds for temperature triggering irrigation.
T_ndays	Integer vector. Number of days for temperature calculation.
dose_fix	Numeric vector. Fixed irrigation dose for each strategy.
dose_min	Numeric vector. Minimum irrigation dose for each strategy.
dose_max	Numeric vector. Maximum irrigation dose for each strategy.
WTurn_mod	Integer or logical vector. Irrigation turn mode (0/FALSE = strict mode: only irrigate on marked water turn days; 1/TRUE = flexible mode: irrigate if there has been a water turn since last irrigation).
LTurn_ndays	Integer vector. Minimum number of days between irrigations for each strategy.
quota_max	Numeric. Total available irrigation water quota.
quota_wind	Numeric vector. Limited quota for applying on a constraint window (fraction of quota_max).

Details

The function applies a set of irrigation decision rules based on soil, weather, and management parameters. It supports multiple strategies and thresholds, including soil moisture, water stress, precipitation, temperature, and ETM. The function manages irrigation turns, quotas, and dose calculations, and updates the simulation matrix accordingly.

Value

An updated simulation matrix or model output object with irrigation applied according to the strategy.

See Also

calc_dose, calc_theta, calc_threshold, calc_quota

`run_model_water_balance` *Water balance module*

Description

Generic and method-specific functions to compute daily water balance. This module includes calculations for potential evapotranspiration (ETM), soil hydric states (z, theta, R) update, water routing (precipitation and irrigation), drainage, soil evaporation (ES), and plant transpiration (TP).

Usage

```
run_model_water_balance(x, ...)

## S3 method for class 'model_output'
run_model_water_balance(x, soil, crop, iday, R0 = NULL, ...)

## S3 method for class 'model_output'
run_model_water_balance_others(x, soil, crop, iday, ...)

## S3 method for class 'matrix'
run_model_water_balance_others(x, theta_fc, theta_wp, iday, ...)
```

Arguments

`x`	Input object. Must be of class `model_output` (matrix handling occurs downstream).
`...`	Additional arguments passed to methods.
`soil`	list Soil parameters.
`crop`	list Crop parameters.
`iday`	integer Current day index of the simulation.
`R0`	numeric Optional initial soil water reservoir used when routing water for the current simulation day.
`theta_fc`	numeric Soil field capacity (volumetric water content).
`theta_wp`	numeric Soil wilting point (volumetric water content).

Value

An object of class `model_output` with updated daily variables.

`run_model_water_balance_drainage` *Drainage part of the water balance module*

Description

Generic and method-specific functions to compute drainage during daily water balance.

Usage

```
run_model_water_balance_drainage(x, ...)

## S3 method for class 'model_output'
run_model_water_balance_drainage(x, soil, iday, ...)

## S3 method for class 'matrix'
run_model_water_balance_drainage(
  x,
  theta_fc,
  theta_wp,
  theta_sat,
  Ks,
  iday,
  ...
)
```

Arguments

`x`	Input object. Must be of class `model_output` (matrix handling occurs downstream).
`...`	Additional arguments passed to methods.
`soil`	list Soil parameters.
`iday`	integer Current day index of the simulation.
`theta_fc`	numeric Water content at field capacity (cm^3/cm^3).
`theta_wp`	numeric Water content at permanent wilting point (cm^3/cm^3).
`theta_sat`	numeric Saturated water content (cm^3/cm^3).
`Ks`	numeric Non-Saturated hydraulic conductivity (mm/h).

Value

An object of class `model_output` with updated daily variables.

`run_model_water_balance_ES` *Evaporation part of the water balance module*

Description

Generic and method-specific functions to compute soil evaporation (ES) during daily water balance.

Usage

```
run_model_water_balance_ES(x, ...)

## S3 method for class 'model_output'
run_model_water_balance_ES(x, soil, iday, ...)

## S3 method for class 'matrix'
run_model_water_balance_ES(x, theta_r, iday, ...)
```

Arguments

`x`	Input object. Must be of class `model_output` (matrix handling occurs downstream).
`...`	Additional arguments passed to methods.
`soil`	list Soil parameters.
`iday`	integer Current day index of the simulation.
`theta_r`	numeric Residual soil water content (cm^3/cm^3).

Value

An object of class `model_output` with updated daily variables.

`run_model_water_balance_ETM` *Potential evapotranspiration (ETM) part of the water balance module*

Description

Generic and method-specific functions to compute potential evapotranspiration (ETM) during daily water balance.

Usage

```
run_model_water_balance_ETM(x, ...)

## S3 method for class 'model_output'
run_model_water_balance_ETM(x, soil, iday, ...)

## S3 method for class 'matrix'
run_model_water_balance_ETM(x, Ksol, iday, ...)
```

Arguments

x	Input object. Must be of class model_output (matrix handling occurs downstream).
...	Additional arguments passed to methods.
soil	list Soil parameters.
iday	integer Current day index of the simulation.
Ksol	numeric Maximum soil evaporation coefficient.

Value

An object of class model_output with updated daily variables.

---

run_model_water_balance_init_states

Initialize soil hydric states for the water balance module

Description

Generic and method-specific functions to initialize soil hydric states at the beginning of the water balance module.

Usage

```
run_model_water_balance_init_states(x, ...)

## S3 method for class 'model_output'
run_model_water_balance_init_states(x, soil, iday, ...)

## S3 method for class 'matrix'
run_model_water_balance_init_states(
  x,
  theta_fc,
  R_init = NA,
  ratio_R_init = NA,
  zES,
```

```
  zmax,
  iday,
  ...
)
```

Arguments

x	Input object. Must be of class model_output (matrix handling occurs downstream).
...	Additional arguments passed to methods.
soil	list Soil parameters.
iday	integer Current day index of the simulation.
theta_fc	numeric Soil field capacity (volumetric water content).
R_init	numeric Initial soil water reserve (mm).
ratio_R_init	numeric Percentage of soil water reserve (R) at the beginning of the simulation.
zES	numeric Depth of soil evaporation front (m).
zmax	numeric Maximum root or soil depth.

Value

An object of class model_output with updated daily variables.

---

run_model_water_balance_others

Others part of the water balance module

Description

Generic and method-specific functions to compute other daily variables related to water balance, such as actual evapotranspiration (ETR), water reserve (R) over all soil layers, available water capacity (RU) over all soil layers, total available water capacity (RUmax) over all soil layers.

Usage

```
run_model_water_balance_others(x, ...)
```

Arguments

x	Input object. Must be of class model_output (matrix handling occurs downstream).
...	Additional arguments passed to methods.

Value

An object of class model_output with updated daily variables.

`run_model_water_balance_routing_water` *Route water through the soil layers*

Description

Generic and method-specific functions to route water (precipitation and irrigation) through the soil layers during daily water balance.

Usage

```
run_model_water_balance_routing_water(x, ...)

## S3 method for class 'model_output'
run_model_water_balance_routing_water(x, soil, iday, ...)

## S3 method for class 'matrix'
run_model_water_balance_routing_water(
  x,
  ratio_R_init,
  R_init,
  zmax,
  theta_fc,
  iday,
  R0 = NULL,
  ...
)
```

Arguments

`x`	Input object. Must be of class `model_output` (matrix handling occurs downstream).
`...`	Additional arguments passed to methods.
`soil`	list Soil parameters.
`iday`	integer Current day index of the simulation.
`ratio_R_init`	numeric initial reservoir ratio (in %)
`R_init`	numeric initial reservoir volume (in mm)
`zmax`	numeric maximum soil root depth (in m)
`theta_fc`	numeric Soil field capacity (volumetric water content).
`R0`	numeric Optional initial soil water reservoir used when routing water for the current simulation day.

Value

An object of class `model_output` with updated daily variables.

`run_model_water_balance_TP` *Transpiration part of the water balance module*

Description

Generic and method-specific functions to compute transpiration (TP) during daily water balance.

Usage

```
run_model_water_balance_TP(x, ...)

## S3 method for class 'model_output'
run_model_water_balance_TP(x, soil, iday, ...)

## S3 method for class 'matrix'
run_model_water_balance_TP(x, theta_r, theta_wp, theta_raw, iday, ...)
```

Arguments

`x`	Input object. Must be of class `model_output` (matrix handling occurs downstream).
`...`	Additional arguments passed to methods.
`soil`	list Soil parameters.
`iday`	integer Current day index of the simulation.
`theta_r`	numeric Residual soil water content (cm^3/cm^3).
`theta_wp`	numeric Volumetric soil water content at wilting point ($m^3\ m^{-3}$).
`theta_raw`	numeric Differencial of soil water content between readily available water and actual water content.

Value

An object of class `model_output` with updated daily variables.

`run_model_water_balance_update_R` *Update soil hydric states for the water balance module*

Description

Generic and method-specific functions to update soil hydric states (z, theta, R) with updated root depth (zroot).

Usage

```
run_model_water_balance_update_R(x, ...)

## S3 method for class 'model_output'
run_model_water_balance_update_R(x, soil, iday, ...)

## S3 method for class 'matrix'
run_model_water_balance_update_R(x, zES, zmax, iday, ...)
```

Arguments

x	Input object. Must be of class model_output (matrix handling occurs downstream).
...	Additional arguments passed to methods.
soil	list Soil parameters.
iday	integer Current day index of the simulation.
zES	numeric Depth of soil evaporation front (m).
zmax	numeric Maximum root or soil depth.

---

save_model_outputs *Save Model Outputs*

Description

Generic function to save model outputs. This function dispatches methods based on the class of the input object x.

Usage

```
save_model_outputs(x, ...)

## S3 method for class 'model_outputs'
save_model_outputs(x, ...)

## S3 method for class 'model_output'
save_model_outputs(x, ...)

## S3 method for class 'matrix'
save_model_outputs(x, run_id, path_outputs, overwrite, ...)

## S3 method for class 'data.frame'
save_model_outputs(x, run_id, path_outputs, overwrite, ...)
```

Arguments

x	An object containing model outputs to be saved. Supported classes include model_outputs, model_output, and matrix.
...	Additional arguments passed to specific methods.
run_id	character string specifying the run identifier (required for the matrix method).
path_outputs	character directory where outputs will be saved (required for the matrix method).
overwrite	logical whether to overwrite existing files (required for the matrix method).

Value

The result of the method dispatched for the class of x.

---

select_output_vars *Select variables to return from model outputs*

Description

Generic and method-specific functions to select specific variables from model outputs.

Usage

```
select_output_vars(x, ...)

## S3 method for class 'matrix'
select_output_vars(x, dates, vars = NULL, ...)
```

Arguments

x	Input object. Must be a matrix of simulation outputs.
...	Additional arguments passed to methods.
dates	base::Date Vector of dates corresponding to the rows of x
vars	character Names of variables to select. If NULL, all variables are returned.

Value

An object of the same class as x with selected variables.

Index

add_missing_params, 4

base::Date, *18*, *107*, *109–112*, *118*, *143*

calc_alpha, 5
calc_B, 7
calc_B_root, 9
calc_Bp, 11
calc_Bp_root, 11
calc_case, 12
calc_Cp, 13
calc_crac, 14
calc_d, 16
calc_d_from_Rgravity *(*calc_d*)*, 16
calc_degred, 17
calc_dose, 18, *134*
calc_dtheta_RU, 20
calc_dzroot, 21
calc_dzrootp, 23
calc_ES, 24
calc_ES0, 26
calc_ET0, 28
calc_ET0_mulch_effect, *28*, 29
calc_ETM, 30
calc_ETR, 32
calc_HI, 33
calc_HUg, 35
calc_HUg(), *93*
calc_IR, 36
calc_Kc, 38
calc_Kd, 39
calc_Kd_from_saxton *(*calc_Kd*)*, 39
calc_KES, 41
calc_Ks, 43
calc_Ks_from_cosby *(*calc_Ks*)*, 43
calc_Ks_from_ottoni *(*calc_Ks*)*, 43
calc_Ks_from_saxton *(*calc_Ks*)*, 43
calc_ks_root, 45
calc_KTP, 46
calc_LAI, 47
calc_LAIp, 51
calc_LT, 53
calc_PAR, 55
calc_quota, 57, *134*
calc_R, 58
calc_R.default *(*calc_R*)*, 58
calc_R.model_output *(*calc_R*)*, 58
calc_R_from_theta *(*calc_R*)*, 58
calc_Rgravity, 60
calc_Rgravity_from_R *(*calc_Rgravity*)*, 60
calc_Rgravity_from_theta
        *(*calc_Rgravity*)*, 60
calc_RU, 62
calc_RU_from_theta *(*calc_RU*)*, 62
calc_RUmax, 64
calc_stress, 66
calc_stress_noci_B, 67
calc_stress_noci_LAI, 69
calc_stress_T, 70
calc_stress_water, 72
calc_sugar_accumulation, 73
calc_tau, 74
calc_taum, 74
calc_Tcrit, 76
calc_Tcrit.matrix, *76*
calc_Tcrit.numeric, *76*
calc_theta, 76, *134*
calc_theta_fc, 77
calc_theta_fc_from_texture
        *(*calc_theta_fc*)*, 77
calc_theta_from_R *(*calc_theta*)*, 76
calc_theta_r, 78
calc_theta_r_from_texture
        *(*calc_theta_r*)*, 78
calc_theta_rel, 79
calc_theta_sat, 80
calc_theta_sat_from_texture
        *(*calc_theta_sat*)*, 80
calc_theta_wp, 81

calc_theta_wp_from_texture
(calc_theta_wp), 81
calc_threshold, 82, *134*
calc_TP, 84
calc_TP0, 85
calc_TT, 86
calc_TT_norm, 87
calc_TT_rel, 89
calc_TT_sowing, 90
calc_Y_grain, 91
calc_Y_hi, 94
calc_Y_sugar, 95
calc_zroot, 96
calc_zrootp, 97
character, *6*, *8*, *10*, *13*, *15, 16*, *18*, *21, 22*, *24, 25*, *27, 28*, *31–34*, *36, 37*, *40*, *42, 43*, *45, 46*, *49*, *52*, *54*, *56*, *59*, *61*, *63*, *65*, *67, 68*, *70–73*, *75–78*, *80*, *82*, *86–89*, *91*, *93, 94*, *96–98*, *105*, *107*, *110, 111*, *113*, *118, 119*, *122*, *124*, *130*, *143*
create_model_inputs, 99
create_model_inputs_climate, 100
create_model_inputs_crop, 101
create_model_inputs_irrigation, 102
create_model_inputs_run, 103
create_model_inputs_soil, 104

data.frame, *44*, *88*, *91*, *96–99*, *101*, *123*

get_data_path, *100*, 104
get_iday, 106
get_model_output_path, 107
get_vars_data, *100*, 108

init_model, 109
init_model_climate, 110
init_model_crop_dev, 111
init_model_irrigation, 112
init_model_vars, 113
inputs_meta, 113
integer, *6*, *8*, *10*, *12–16*, *18–20*, *22*, *24, 25*, *27, 28*, *31, 32*, *34*, *36–38*, *42*, *45, 46*, *49*, *52*, *54*, *56*, *59*, *61*, *63*, *65*, *67, 68*, *70–73*, *75*, *77*, *80*, *86–89*, *91*, *93, 94*, *96–98*, *107*, *122*, *124, 125*, *127–132*, *135–142*

list, *15*, *18*, *22*, *52*, *71*, *75*, *87*, *98, 99*, *101–104*, *106*, *109–112*, *121*, *123–129*, *131, 132*, *135–142*
load_config, *100*, *105*, 114
logical, *105*, *119*, *121, 122*, *143*

MAIZE_21LAVALETTE_B, 115
MAIZE_21LAVALETTE_clim, 115
MAIZE_21LAVALETTE_irrig, 116
MAIZE_21LAVALETTE_LAI, 117
MAIZE_21LAVALETTE_RES, 117
matrix, *72, 73*, *88*, *91*, *96–98*, *111, 112*

numeric, *6*, *8*, *10*, *13–16*, *18–20*, *22, 23*, *25*, *27–29*, *31, 32*, *34–38*, *40*, *42–46*, *49*, *52*, *54*, *56*, *59*, *61*, *63*, *65*, *67, 68*, *70–78*, *80*, *82*, *85–89*, *91–98*, *107*, *110*, *125*, *127, 128*, *131*, *135–142*

parse_date_time, 118

read, 119
run_model, 120
run_model(), *93*
run_model_crop_dev, 123
run_model_crop_dev_B, 124
run_model_crop_dev_B_root, 125
run_model_crop_dev_LAI, 126
run_model_crop_dev_stress_water, 127
run_model_crop_dev_Y, 128
run_model_crop_dev_Y_grain, 128
run_model_crop_dev_Y_sugar, 129
run_model_crop_dev_zroot, 130
run_model_finance, 131
run_model_irrig_strat, 132
run_model_irrig_strat_dose, 132
run_model_water_balance, 135
run_model_water_balance.model_output
(run_model_water_balance), 135
run_model_water_balance_drainage, 136
run_model_water_balance_ES, 137
run_model_water_balance_ETM, 137
run_model_water_balance_init_states, 138
run_model_water_balance_others, 139
run_model_water_balance_others.matrix
(run_model_water_balance), 135
run_model_water_balance_others.model_output
(run_model_water_balance), 135
run_model_water_balance_routing_water, 140

run_model_water_balance_TP, 141
run_model_water_balance_update_R, 141

save_model_outputs, 142
select_output_vars, 143