

Package: epicrop (via r-universe)

October 5, 2026

Type Package

Title SEIR Simulation Modelling of Crop Diseases

Version 1.0.0

Description Generic simulation modelling of crop diseases using a Susceptible-Exposed-Infectious-Removed ('SEIR') model. This type of model was first described in botanical epidemiology by Zadoks (1971) <[doi:10.1094/Phyto-61-600](https://doi.org/10.1094/Phyto-61-600)> and implemented in the 'EPIRICE' model by Savary et al. (2012) <[doi:10.1016/j.cropro.2011.11.009](https://doi.org/10.1016/j.cropro.2011.11.009)> detailing the model's development, use and results from modelling global unmanaged epidemics of rice diseases for five major rice diseases, bacterial blight, brown spot, leaf blast, sheath blight and tungro. Specific functions are provided to simulate all of these diseases with a generic 'SEIR' model function that is suitable for parameterising for use with other pathosystems as demonstrated in Savary et al. 2012 and Kim et al. 2015 <[doi:10.1016/j.agrformet.2015.01.011](https://doi.org/10.1016/j.agrformet.2015.01.011)> along with two wheat diseases, brown (leaf) rust and Septoria tritici blotch as published in Savary et al. 2015 <[doi:10.1007/s10658-015-0650-7](https://doi.org/10.1007/s10658-015-0650-7)>. Based on a fork of the `cropsim` package that only provided the EPIRICE model; enhanced functions for fetching and formatting weather data from the NASA POWER and ERA5 databases are also provided to simplify the process of running simulations.

License GPL (>= 3)

URL <https://codeberg.org/adamhsparks/epicrop>,
<https://adamhsparks.codeberg.page/epicrop>

BugReports <https://codeberg.org/adamhsparks/epicrop/issues>

Depends R (>= 4.4.0)

Imports cli, data.table, digest, httr2, memoise, nasapower (>= 4.0),
rlang, yyjsonr

Suggests chirps, cropgrowdays, ggplot2, ggrepel, ggspatial, knitr, rmarkdown, roxyglobals, spelling, stats, testthat, vcr (>= 0.6.0), withr

VignetteBuilder knitr

ByteCompile TRUE

Config/Needs/build moodymudskipper/devtag

Config/roxygen2/version 8.1.0

Config/roxyglobals/filename globals.R

Config/roxyglobals/unique FALSE

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE, roclets = c("`collate", "`namespace", "`rd", "`roxyglobals::global_roclet", "`devtag::dev_roclet"))

X-schema.org-applicationCategory Tools

X-schema.org-keywords rice, wheat, botanical-epidemiology, disease, model, seir, agricultural-modelling, agricultural-modeling, crop-protection, agricultural-research, model, modelling, modeling, seir-model, plant-disease

Config/pak/sysreqs libssl-dev libx11-dev

Repository <https://adamhspark.r-universe.dev>

Date/Publication 2026-10-05 09:34:31 UTC

RemoteUrl <https://codeberg.org/adamhspark/epicrop>

RemoteRef HEAD

RemoteSha e519920314fd41cf34e3fbd1cc6fc13639f40894

Contents

audpc_dt	3
build_epicrop_emergence	5
disease_models	9
format_wth	21
get_audpc	26
get_epicrop_weather_list	28
get_wth	30
run_epicrop_model	36
seir	40

Index	48
--------------	-----------

audpc_dt	<i>Extract AUDPC Summary as data.table</i>
----------	--

Description

Extracts the Area Under Disease Progress Curve (AUDPC) from an **epicrop** simulation output along with spatial (lat/lon) and temporal (emergence) metadata. Returns a compact summary table suitable for analysis, mapping, or reporting.

Usage

```
audpc_dt(x)
```

Arguments

x An `epicrop.sim` object from `seir()`, disease model functions (`bacterial_blight()`, `leaf_blast()`, etc.), or `run_epicrop_model()` with `output = "full"`.

Value

A `data.table::data.table()` with one row containing:

- **lat**: Latitude (double; NA if not available)
- **lon**: Longitude (double; NA if not available)
- **AUDPC**: Area under disease progress curve (double)
- **emergence**: Emergence date (character, "YYYY-MM-DD")

Table is keyed on emergence for efficient operations.

Output Structure

The returned `data.table::data.table()` includes:

- **lat**: Latitude (if available in the model object)
- **lon**: Longitude (if available in the model object)
- **AUDPC**: Area under disease progress curve (numeric)
- **emergence**: Emergence date as character (YYYY-MM-DD)

Columns are present only if the corresponding data exist in the input object. The table is keyed on emergence for efficient sorting and joining.

Use Cases

- **Batch processing**: Quickly summarise results from `run_epicrop_model()`
- **Mapping**: Combine with spatial data for disease risk mapping
- **Reporting**: Generate summary tables for publications or extension
- **Analysis**: Join with cultivar, management, or climate data

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>

Examples

```
# Example 1: Extract AUDPC summary from single model run
wth <- get_wth(
  lonlat = c(121.25562, 14.6774),
  dates = "2000-06-30",
  duration = 120L
)

sim <- bacterial_blight(wth, emergence = "2000-06-30")
summary_table <- audpc_dt(sim)
summary_table
# Output includes lat, lon, AUDPC, emergence

# Example 2: Batch summarise multiple emergence dates
emergence_dates <- seq(as.Date("2000-06-15"), as.Date("2000-07-15"), by = 7)
summaries <- lapply(emergence_dates, function(ed) {
  sim <- bacterial_blight(wth, emergence = as.character(ed))
  audpc_dt(sim)
})
summary_all <- data.table::rbindlist(summaries)
summary_all

# Example 3: Compare multiple locations and emergence dates
locations <- data.frame(
  name = c("IRRI", "Manila"),
  lon = c(121.25562, 120.985),
  lat = c(14.6774, 14.6042)
)

emergence_dates <- c("2000-06-30", "2000-07-15")

results <- expand.grid(
  location = locations$name,
  emergence = emergence_dates,
  stringsAsFactors = FALSE
)

results$AUDPC <- mapply(function(loc_name, em_date) {
  loc <- locations[locations$name == loc_name, ]
  wth <- get_wth(
    lonlat = c(loc$lon, loc$lat),
    dates = em_date,
    duration = 120L
  )
  sim <- leaf_blast(wth, emergence = em_date)
  get_audpc(sim)
}, results$location, results$emergence)
```

```

results
# Identify highest risk scenario
worst <- results[which.max(results$AUDPC), ]
sprintf("Highest risk: %s, emergence %s, AUDPC = %.1f",
        worst$location, worst$emergence, worst$AUDPC)

# Example 4: Export for mapping or further analysis
wth <- get_wth(
  lonlat = c(121.25562, 14.6774),
  dates = "2000-06-30",
  duration = 120L
)
sim <- brown_spot(wth, emergence = "2000-06-30")
summary_dt <- audpc_dt(sim)

```

build_epicrop_emergence

Build Emergence Dates from Year and Month-Day Combinations

Description

Constructs a vector of emergence (planting) dates by combining years with month-day patterns. Returns a sorted `data.table::IDate` vector tagged with class `epicrop.emerge` for use with `run_epicrop_model()` and related batch modelling functions.

Usage

```
build_epicrop_emergence(years, month_day)
```

Arguments

- | | |
|-----------|--|
| years | Integer vector of years. Examples: <ul style="list-style-type: none"> • 2000 – single year • c(2000, 2002, 2005) – specific years • 2000:2005 – range (recommended for clarity) Dates are constructed as <code>paste0(year, month_day)</code> , so years can be any 4-digit integer. No validation against real calendar (<i>e.g.</i> , allows year 9999). |
| month_day | Character vector of month-day patterns. Each string specifies a fixed planting date recurring each year. Format (either): <ul style="list-style-type: none"> • "-MM-DD" with leading dash (<i>e.g.</i>, "-06-01", "-12-25") • "MM-DD" without dash (<i>e.g.</i>, "06-01", "12-25") Validation: <ul style="list-style-type: none"> • Month: 01-12 (leading zero required) • Day: 01-31 (leading zero required) • Separator: dash (-), not . or / |

Examples of valid input:

- `c("-06-01", "-07-15", "-08-31")`
- `c("01-15", "06-01", "12-25")` (no leading dash)

Invalid examples (will abort):

- `"6-1"` (no leading zeros)
- `"-06/01"` (wrong separator)
- `"-13-01"` (month > 12)
- `"-02-30"` (day > 29 for Feb; caught during date parsing)

Details

Internal Algorithm:

1. Normalize month_day strings:
 - If matches `"^-(0[1-9]|1[0-2])-(0[1-9]|[12][0-9]|3[01])$"`: use as-is
 - If matches `"^(0[1-9]|1[0-2])-(0[1-9]|[12][0-9]|3[01])$"`: prepend -
 - Otherwise: abort with error message
2. Create cross-product: every year $\times$ every normalized month-day
3. Construct ISO date strings: `sprintf("%s%s", year, month_day)`
4. Parse to `data.table::IDate` with format `"%Y-%m-%d"`
5. Sort by date and tag with `epicrop.emerge` class

Leap Year Handling:

R's Date class handles leap years automatically:

- `"2000-02-29"` is valid (2000 is leap year)
- `"2001-02-29"` fails to parse (2001 is not leap year)

Invalid dates (*e.g.*, Feb 30 for any year) are caught during parsing and trigger an abort.

Performance:

Fast even for large year/month-day combinations:

- Cross-product via `data.table::CJ()` (optimized)
- Vectorised date construction via `sprintf()`
- Sorting via base `sort()` on `IDate` (fast integer representation)

Example: 20 years $\times$ 10 month-days = 200 dates constructed in < 1 ms.

Value

A sorted vector of `data.table::IDate` emergence dates with class `epicrop.emerge`. Inherits from `base::Date` and `data.table::IDate`, so standard date operations (subsetting, arithmetic, formatting) work. `Length = length(years) * length(month_day)`.

Purpose

Simplifies creation of emergence date vectors for epidemiological modelling scenarios where you want to:

- Test multiple planting dates within a season (*e.g.*, June 1, 15, 30)
- Replicate across multiple years (*e.g.*, 2000-2005)
- Generate all combinations automatically

Eliminates manual date construction: instead of typing out 18 dates (6 month-days $\times$ 3 years), write:

```
build_epicrop_emergence(2000:2002, c("-06-01", "-06-15", "-07-01"))
```

Date Format

Month-day strings can be provided in two equivalent formats:

- **With leading dash** (recommended): "-06-01", "-12-31"
 - Matches ISO 8601 suffix for "YYYY-MM-DD"
 - Function uses these directly
- **Without leading dash**: "06-01", "12-31"
 - Function prepends - automatically
 - Less error-prone if mixed formats appear

Both `c("-06-01", "-06-15")` and `c("06-01", "06-15")` produce identical output.

Validation catches:

- Invalid months (not 01-12)
- Invalid days (not 01-31 for month; leap years handled by R's Date class)
- Malformed strings (typos, missing separators)

Output Class

The returned vector has class `epicrop.emerge` (subclass of `data.table::IDate` and `base::Date`). This tag enables:

- Input validation in `run_epicrop_model()` (checks class, not just type)
- Clear intent: a vector of planting dates, not arbitrary dates
- Potential future S3 methods (*e.g.*, `plot.epicrop.emerge()`)

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>

See Also

- `run_epicrop_model()` – primary consumer of `epicrop.emerge` vectors
- `get_epicrop_weather_list()` – companion for building weather lists
- `base::Date`, `data.table::IDate` – underlying date classes
- `base::seq.Date()` – alternative for building date sequences

Examples

```
# Example 1: Basic usage - two years, three planting dates
years <- 2000:2001
month_day <- c("-06-01", "-06-15", "-07-01")

em <- build_epicrop_emergence(years, month_day)
em

# Check class
class(em) # "epicrop.emerge" "IDate" "Date"
length(em) # 6 (2 years × 3 month-days)

# Example 2: Month-day without leading dash (auto-normalized)
em2 <- build_epicrop_emergence(2000, c("06-01", "06-15", "07-01"))
identical(em2, build_epicrop_emergence(2000, c("-06-01", "-06-15", "-07-01")))

# Example 3: Single year, multiple planting windows
# Early, mid, late June + early July
single_year <- build_epicrop_emergence(
  years = 2000,
  month_day = c("-06-01", "-06-10", "-06-20", "-07-01")
)
single_year
plot(single_year, rep(1, length(single_year)),
  main = "Planting Date Options for 2000",
  xlab = "Date", ylab = "")

# Example 4: Long-term study across 10 years
long_term <- build_epicrop_emergence(
  years = 1990:1999,
  month_day = c("-06-15", "-07-01") # Two standard planting dates
)
length(long_term) # 20 (10 years × 2 dates)

# Example 5: Use in batch modelling workflow
years <- 2000:2002
month_day <- c("-06-30", "-07-15")
emergence <- build_epicrop_emergence(years, month_day)

# Fetch weather data (one year per list element)
wth_list <- get_epicrop_weather_list(
  lonlat = c(121.255669, 14.167425),
  start_date = month_day,
  duration = 120L,
  years = years
)

# Run disease model across all emergence dates
results <- run_epicrop_model(
  model_fun = leaf_blast,
  emergence_dates = emergence,
  wth_list = wth_list,
```

```

    output = "audpc"
  )
  results # AUDPC for each emergence date

# Example 6: Subset and filter emergence dates
em <- build_epicrop_emergence(2000:2002, c("-06-01", "-07-01"))

# Keep only June plantings
june_only <- em[month(em) == 6]
june_only

# Keep only 2001 plantings
y2001 <- em[year(em) == 2001]
y2001

# Example 7: Error handling
## Not run:
# Invalid month (13)
build_epicrop_emergence(2000, "-13-01")
# Error: Invalid month_day value(s): -13-01

# Invalid day (32)
build_epicrop_emergence(2000, "-02-30")
# Error: Failed to construct emergence dates

# Typo in format (no dash, two-digit year in string)
build_epicrop_emergence(2000, "6-1")
# Error: Invalid month_day value(s): 6-1

## End(Not run)

```

Description

Convenience wrappers around `seir()` with pre-configured parameter tables for well-characterized crop diseases. Each function encapsulates parameters from published epidemiological studies, allowing one-line disease simulations.

`bacterial_blight()` models rice bacterial blight (*Xanthomonas oryzae* pv. *oryzae*). The disease requires high humidity and warm temperatures (optimum 25–30°C) and is most severe in the Philippines and Asia-Pacific region.

`brown_spot()` models rice brown spot (*Bipolaris oryzae*). A seed-borne fungal disease prevalent in tropical and subtropical regions, particularly damaging to older rice varieties.

`leaf_blast()` models rice leaf blast (*Magnaporthe oryzae*). The most economically important rice disease worldwide, favoured by cool (25°C), wet conditions. Rapid disease progression can cause total crop loss.

`sheath_blight()` models rice sheath blight (*Rhizoctonia solani*). A soil-borne disease that infects the leaf sheath; can cause catastrophic damage in high-N fertilized crops.

`tungro()` models rice tungro, a viral disease complex (*Tungro bacilliform* and *Tungro spherical viruses*) transmitted by the insect vector *Nephotettix* spp. Vector populations increase with temperature and humidity; disease pressure peaks mid-season.

`modified_kim_leaf_blast()` models rice leaf blast using parameters from Kim *et al.* (2015), calibrated for South Korean rice cultivars and growing conditions.

`modified_kim_sheath_blight()` models rice sheath blight using parameters from Kim *et al.* (2015), calibrated for South Korean rice cultivars. Uses slightly higher humidity threshold (95%) reflecting Korean monsoon climate.

`leaf_rust()` models wheat leaf rust (*Puccinia triticina*). A major wheat pathogen globally, favoured by moderate temperatures (15–25°C) and high humidity. #' DVS-based modelling accounts for developmental susceptibility.

`s_tritici_blotch()` models wheat Septoria tritici blotch (*Zymoseptoria tritici*), a major wheat pathogen in temperate regions. Features lesion expansion (direct H→I pathway) allowing rapid canopy senescence. DVS-based modelling accounts for developmental susceptibility.

`bb()` is an alias for `bacterial_blight()`.

`bs()` is an alias for `brown_spot()`.

`lb()` is an alias for `leaf_blast()`.

`sb()` is an alias for `sheath_blight()`.

`tg()` is an alias for `tungro()`.

`mk_lb()` is an alias for `modified_kim_leaf_blast()`.

`mk_sb()` is an alias for `modified_kim_sheath_blight()`.

`lr()` is an alias for `leaf_rust()`.

`brown_rust()` is an alias for `leaf_rust()`.

`stb()` is an alias for `s_tritici_blotch()`.

Usage

```
bacterial_blight(
  wth,
  emergence,
  RcA = NULL,
  RcT = NULL,
  RRS = NULL,
  RRG = NULL,
  rhlim = NULL,
  rainlim = NULL,
  ...
)
```

```
brown_spot(
  wth,
  emergence,
```

```
RcA = NULL,  
RcT = NULL,  
RRS = NULL,  
RRG = NULL,  
rhlim = NULL,  
rainlim = NULL,  
...  
)  
  
leaf_blast(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRS = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
sheath_blight(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRS = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
tungro(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRS = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
modified_kim_leaf_blast(  
  wth,  
  emergence,
```

```
RcA = NULL,  
RcT = NULL,  
RRS = NULL,  
RRG = NULL,  
rhlim = NULL,  
rainlim = NULL,  
...  
)  
  
modified_kim_sheath_blight(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRS = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
leaf_rust(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRPSN = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
s_tritici_blotch(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRPSN = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
bb(  
  wth,  
  emergence,
```

```
RcA = NULL,  
RcT = NULL,  
RRS = NULL,  
RRG = NULL,  
rhlim = NULL,  
rainlim = NULL,  
...  
)  
  
bs(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRS = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
lb(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRS = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
sb(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRS = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
tg(  
  wth,  
  emergence,
```

```
RcA = NULL,  
RcT = NULL,  
RRS = NULL,  
RRG = NULL,  
rhlim = NULL,  
rainlim = NULL,  
...  
)  
  
mk_lb(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRS = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
mk_sb(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRS = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
lr(  
  wth,  
  emergence,  
  RcA = NULL,  
  RcT = NULL,  
  RRPSN = NULL,  
  RRG = NULL,  
  rhlim = NULL,  
  rainlim = NULL,  
  ...  
)  
  
brown_rust(  
  wth,  
  emergence,
```

```

    RcA = NULL,
    RcT = NULL,
    RRPSN = NULL,
    RRG = NULL,
    rhlim = NULL,
    rainlim = NULL,
    ...
)

stb(
  wth,
  emergence,
  RcA = NULL,
  RcT = NULL,
  RRPSN = NULL,
  RRG = NULL,
  rhlim = NULL,
  rainlim = NULL,
  ...
)
```

Arguments

wth A data.frame of class epicrop.wth from [get_wth\(\)](#) or [format_wth\(\)](#) containing daily weather data with columns:

Required Columns:

Column	Description
YYYYMMDD	Date (ISO 8601: YYYYMMDD)
DOY	Day of year (1–366)
TEMP	Mean daily temperature (°.RC)
RHUM	Mean daily relative humidity (%)
RAIN	Daily rainfall (mm)

Optional Columns (for advanced features):

Column	Purpose
TMIN	Minimum daily temperature (°.RC); required if simple_wetness = FALSE
TMAX	Maximum daily temperature (°.RC); required if simple_wetness = FALSE
LAT	Latitude; required if simple_wetness = FALSE
LON	Longitude; included in output if present

emergence Plant emergence (transplanting for rice) date in YYYY-MM-DD format (character).

RcA Optional custom age modifier table (2-column numeric matrix). Overrides the published default for this disease.

RcT	Optional custom temperature modifier table (2-column numeric matrix). Overrides the published default for this disease.
RRS	Optional custom relative senescence rate (scalar numeric, vector, function, or matrix). Overrides the published default. Applies to rice disease models only.
RRG	Optional custom relative growth rate (scalar numeric, vector, or function). Overrides the published default.
rhlim	Optional override for relative humidity threshold (%) for leaf wetness. Defaults to disease-specific value (typically 90).
rainlim	Optional override for rainfall threshold (mm) for leaf wetness. Defaults to disease-specific value (typically 5).
...	Additional parameters passed to seir() , allowing full customization of any parameter (onset, duration, H0, I0, p, i, Sx, a, simple_wetness, etc.).
RRPSN	Optional custom relative photoperiod-adjusted senescence rate (scalar numeric, vector, function, or matrix). Overrides the published default. Applies to wheat disease models only.

Details

All functions support parameter override via ... arguments, enabling sensitivity analysis and custom cultivar/environment modelling.

Value

A `data.table::data.table()` of class `epicrop.sim` with disease progression over time. See [seir\(\)](#) for output details.

Available Models

Rice Diseases (Savary *et al.* 2012):

- [bacterial_blight\(\)](#) – Bacterial blight (*Xanthomonas oryzae* pv. *oryzae*)
- [brown_spot\(\)](#) – Brown spot (*Bipolaris oryzae*)
- [leaf_blast\(\)](#) – Leaf blast (*Magnaporthe oryzae*)
- [sheath_blight\(\)](#) – Sheath blight (*Rhizoctonia solani*)
- [tungro\(\)](#) – Tungro (viral, vector-borne)

Rice Diseases (Kim *et al.* 2015 modifications):

- [modified_kim_leaf_blast\(\)](#) – Cultivar-adjusted leaf blast
- [modified_kim_sheath_blight\(\)](#) – Cultivar-adjusted sheath blight

Wheat Diseases (Savary *et al.* 2015):

- [leaf_rust\(\)](#) – Leaf rust (*Puccinia tritica*)
- [s_tritici_blotch\(\)](#) – Septoria tritici blotch (*Zymoseptoria tritici*)

Parameter Sources

Rice models use baseline parameters from:

- Savary *et al.* (2012) – Original parameterisation from literature
- Kim *et al.* (2015) – Versions calibrated for Korean rice cultivars

Wheat models use:

- Savary *et al.* (2015) – European calibration

TMIN/TMAX Details

When `simple_wetness = FALSE`, the function calculates hourly leaf wetness for 24 hours using TMIN, TMAX, LAT, and DOY in the `wth` object. The `rhl` and `rainlim` thresholds determine wet hours. A leaf-wetness modifier (`RcW`) converts wet hours to a 0–1 *Rc* scale.

When `simple_wetness = TRUE`, this calculation is skipped; leaf wetness is binary (0 or 1) based solely on daily RHUM and RAIN.

LAT/LON Details

If the `wth` object contains LAT and LON columns, they are appended to each row of the output for mapping or spatial analysis. Both must be present to be included. By default, `get_wth()` and `format_wth()` provide these.

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>

References

- Savary, S., Nelson, A., Willocquet, L., Pangga, I., and Aunario, J. 2012. Modeling and mapping potential epidemics of rice diseases globally. *Crop Protection*, 34: 6–17. doi:10.1016/j.cropro.2011.11.009
- Kim, K. H., Cho, J., Lee, Y. H., and Lee, W. S. 2015. Predicting potential epidemics of rice leaf blast and sheath blight in South Korea. *Agricultural and Forest Meteorology*, 203: 191–207. doi:10.1016/j.agrformet.2015.01.011
- Savary, S., Stetkiewicz, S., Brun, F., and Willocquet, L. 2015. Modelling and mapping potential epidemics of wheat diseases. *European Journal of Plant Pathology*, 142: 771–790. doi:10.1007/s1065801506507

Examples

```
# Fetch weather data for a location in the Philippines
wth <- get_wth(
  lonlat = c(121.255669, 14.16742),
  dates = "2000-06-30",
  duration = 120L
)

# Run bacterial blight model with default parameters (Savary et al.* 2012)
bb_sim <- bacterial_blight(
```

```

    wth = wth,
    emergence = "2000-06-30"
  )

# Inspect results
head(bb_sim)
plot(bb_sim$simday, bb_sim$intensity, type = "l",
     main = "Rice Bacterial Blight Progression",
     xlab = "Days", ylab = "Disease Intensity")

# Simulate brown spot for a standard rice season
wth <- get_wth(
  lonlat = c(121.255669, 14.16742),
  dates = "2000-06-30",
  duration = 120L
)

bs_sim <- brown_spot(
  wth = wth,
  emergence = "2000-06-30"
)

# Brown spot develops slowly compared to leaf blast
plot(bs_sim$simday, bs_sim$intensity, type = "l",
     main = "Rice Brown Spot Progression",
     xlab = "Days After Emergence", ylab = "Disease Intensity")

# Simulate leaf blast under favourable conditions
wth <- get_wth(
  lonlat = c(121.255669, 14.16742),
  dates = "2000-06-30",
  duration = 120L
)

lb_sim <- leaf_blast(
  wth = wth,
  emergence = "2000-06-30"
)

# Leaf blast shows rapid increase in intensity after onset
plot(lb_sim$simday, lb_sim$intensity, type = "l", ylim = c(0, 1),
     main = "Rice Leaf Blast Progression",
     xlab = "Days After Emergence", ylab = "Disease Intensity")

wth <- get_wth(
  lonlat = c(121.255669, 14.16742),
  dates = "2000-06-30",
  duration = 120L
)

```

```

sb_sim <- sheath_blight(
  wth = wth,
  emergence = "2000-06-30"
)

# Sheath blight has long infectious period and delayed onset
plot(sb_sim$simday, sb_sim$intensity, type = "l", ylim = c(0, 1),
     main = "Rice Sheath Blight Progression",
     xlab = "Days After Emergence", ylab = "Disease Intensity")

# Simulate tungro (vector-borne viral disease)
wth <- get_wth(
  lonlat = c(121.255669, 14.16742),
  dates = "2000-06-30",
  duration = 120L
)

tg_sim <- tungro(
  wth = wth,
  emergence = "2000-06-30"
)

# Tungro shows plateau phase (low Sx due to limited vector population)
plot(tg_sim$simday, tg_sim$intensity, type = "l",
     main = "Rice Tungro Progression",
     xlab = "Days After Emergence", ylab = "Disease Intensity")

# Simulate using Kim *et al.* (2015) cultivar-adjusted parameters
wth <- get_wth(
  lonlat = c(126.9780, 37.5665), # Seoul, South Korea
  dates = "2000-06-01",
  duration = 77L
)

kim_lb <- modified_kim_leaf_blast(
  wth = wth,
  emergence = "2000-06-01"
)

# Kim version has shorter duration and higher Sx (larger canopy model)
plot(kim_lb$simday, kim_lb$intensity, type = "l",
     main = "Rice Leaf Blast (Kim *et al.* 2015)",
     xlab = "Days", ylab = "Intensity")

# Simulate sheath blight with Kim *et al.* (2015) parameters
wth <- get_wth(
  lonlat = c(126.9780, 37.5665),
  dates = "2000-06-01",
  duration = 88L
)

```

```

)

kim_sb <- modified_kim_sheath_blight(
  wth = wth,
  emergence = "2000-06-01"
)

plot(kim_sb$simday, kim_sb$intensity, type = "l",
     main = "Rice Sheath Blight (Kim et al.* 2015)",
     xlab = "Days", ylab = "Intensity")

# Simulate wheat leaf rust with DVS (development stage) modelling
wth <- get_wth(
  lonlat = c(8.5, 50.1), # Example: France
  dates = "2000-10-15", # Winter wheat sowing
  duration = 240L       # Full growing season
)

lr_sim <- leaf_rust(
  wth = wth,
  emergence = "2000-10-15"
)

# Leaf rust shows gradual progression across growing season
plot(lr_sim$simday, lr_sim$intensity, type = "l",
     main = "Wheat Leaf Rust Progression (DVS-based)",
     xlab = "Days After Sowing", ylab = "Disease Intensity")

# Example: Septoria tritici blotch - compare lesion expansion effect

# Fetch weather for a wheat-growing region (France, Oct-May season)
wth <- get_wth(
  lonlat = c(8.5, 50.1),
  dates = "2000-10-15",
  duration = 240L
)

stb <- s_tritici_blotch(
  wth = wth,
  emergence = "2000-10-15"
)

# Create single plot with both lines for comparison
plot(stb$simday, stb$intensity,
     type = "l", col = "black", lty = 1,
     main = "Wheat Septoria tritici Blotch",
     xlab = "Days After Sowing",
     ylab = "Disease Intensity")

```

format_wth

Format Weather Data for Use in seir() Simulations

Description

Reformats an existing `data.frame()` of weather data into the standard **epicrop** format for use in `seir()` and disease model functions. Selects and renames columns to match model expectations, validates data quality (no gaps, no duplicates), and adds derived fields (DOY).

Usage

```
format_wth(wth, date, temp, rhum, rain, tmin, tmax, lat, lon)
```

Arguments

wth	A <code>data.frame()</code> or <code>data.table::data.table()</code> containing raw weather observations. Must have one row per day with no gaps.
date	Character. Name of column in wth containing dates. Dates should be in ISO format (YYYY-MM-DD), integer (YYYYMMDD), or already Date class. Required.
temp	Character. Name of column containing mean daily temperature (degrees C). Required.
rhum	Character. Name of column containing mean daily relative humidity (%). Required.
rain	Character. Name of column containing total daily rainfall (mm). Required.
tmin	Character. Name of column containing minimum daily temperature (degrees C). Optional; required only if using <code>simple_wetness = FALSE</code> in <code>seir()</code> .
tmax	Character. Name of column containing maximum daily temperature (degrees C). Optional; required only if using <code>simple_wetness = FALSE</code> in <code>seir()</code> .
lat	Character. Name of column containing latitude (degrees). Optional; improves hourly wetness calculation and enables mapping.
lon	Character. Name of column containing longitude (degrees). Optional; enables spatial mapping of results.

Details

Data Source Examples:

Common data sources and required column names:

- **NASA POWER** (via `nasapower::get_power()`): Use `date = "YYYYMMDD"`, `temp = "T2M"`, `rhum = "RH2M"`, `rain = "PRECTOTCORR"`, etc.
- **CHIRPS rainfall** (via `chirps::get_chirps()`): Combine with temperature data from another source
- **Local weather station data**: Map station file columns to parameter names

Caveats:

- Sub-daily weather data must be aggregated to daily first. See `cropgrowdays::daily_mean()` for hourly data aggregation.
- Dates must be complete and continuous (no missing days). Interpolation is not performed.
- Temperature, RH, and rainfall must be numeric (integers or floats). Character/factor columns cause errors.
- For `simple_wetness = FALSE` in `seir()`, both TMIN and TMAX are required.

Value

An `epicrop.wth` object (`data.table::data.table()` subclass) containing reformatted weather data. Compatible with `seir()` and all disease model functions (`bacterial_blight()`, `leaf_blast()`, etc.). Automatically includes DOY (day of year) derived from date.

Input Requirements

Weather data must contain **daily observations** (one row per day) with these required columns (names arbitrary):

- **Date** column with full date (year, month, day)
- **Temperature** column with mean daily temperature (degrees C)
- **Relative humidity** column with mean daily RH (%)
- **Rainfall** column with total daily precipitation (mm)

Optional columns that enhance model functionality:

- **Minimum temperature** (degrees C; required if `simple_wetness = FALSE`)
- **Maximum temperature** (degrees C; required if `simple_wetness = FALSE`)
- **Latitude** (degrees; used for hourly wetness and mapping)
- **Longitude** (degrees; used for mapping and spatial analysis)

Data Validation

This function validates that:

1. All required columns exist in the input `data.frame`
2. No duplicate dates are present
3. Dates form a contiguous sequence with no gaps (true daily data)
4. Date column is properly formatted as Date class

If validation fails, detailed error messages pinpoint the issue.

Output Format

Returns a `data.table::data.table()` with class `epicrop.wth` containing the following columns (in order):

Column	Description
YYYYMMDD	Date (ISO 8601 format, Date class)
DOY	Day of year (1-365/366, integer)
TEMP	Mean daily temperature (degrees C)
TMIN	Minimum daily temperature (degrees C) [optional]
TMAX	Maximum daily temperature (degrees C) [optional]
RHUM	Mean daily relative humidity (%)
RAIN	Total daily rainfall (mm)
LAT	Latitude [optional]
LON	Longitude [optional]

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>

Examples

```
# Example 1: Format weather from NASA POWER API
# (Requires nasapower package; this example shows typical usage)

# Simulated POWER data (in practice from get_power())
wth_raw <- data.frame(
  DATE = seq(as.Date("2000-06-30"), as.Date("2000-12-31"), by = "day"),
  TAVG = rnorm(185, mean = 25, sd = 3),
  TMIN = rnorm(185, mean = 20, sd = 3),
  TMAX = rnorm(185, mean = 30, sd = 3),
  RH = rnorm(185, mean = 80, sd = 10),
  PRECIP = rgamma(185, shape = 2, rate = 0.5),
  LAT = 14.6774,
  LON = 121.25562
)

# Format for epicrop
wth <- format_wth(
  wth = wth_raw,
  date = "DATE",
  temp = "TAVG",
  tmin = "TMIN",
  tmax = "TMAX",
  rhum = "RH",
  rain = "PRECIP",
  lat = "LAT",
  lon = "LON"
)

# Check class and structure
class(wth)
head(wth)
# Ready for use in disease models
sim <- bacterial_blight(wth, emergence = "2000-07-01")
```

```

# Example 2: Minimal data (no lat/lon, no TMIN/TMAX)
# Only requires temp, rhum, rain (simple_wetness = TRUE)
wth_minimal <- data.frame(
  obs_date = seq(as.Date("2000-06-30"), as.Date("2000-08-28"), by = "day"),
  mean_temp = rnorm(60, mean = 25, sd = 2),
  mean_rh = rnorm(60, mean = 85, sd = 5),
  daily_rain = rpois(60, lambda = 3)
)

wth_fmt <- format_wth(
  wth = wth_minimal,
  date = "obs_date",
  temp = "mean_temp",
  rhum = "mean_rh",
  rain = "daily_rain"
)

# Simulate with binary wetness (no hourly calculation)
sim <- leaf_blast(wth_fmt, emergence = "2000-07-01", simple_wetness = TRUE)

# Example 3: Integer date format (YYYYMMDD)
wth_int_date <- data.frame(
  date_int = c(20000630L, 20000701L, 20000702L),
  temp = c(25, 26, 24),
  rhum = c(85, 80, 90),
  rain = c(5, 0, 8)
)

# Convert to character first (data.table will handle conversion)
wth_int_date$date_int <- as.character(wth_int_date$date_int)

wth_formatted <- format_wth(
  wth = wth_int_date,
  date = "date_int",
  temp = "temp",
  rhum = "rhumb",
  rain = "rain"
)

# Example 4: Subset and format existing epicrop.wth
# (*e.g.*, use only part of a longer dataset)
wth_full <- get_wth(
  lonlat = c(121.25562, 14.6774),
  dates = c("2000-06-30", "2000-12-31")
)

# Subset to June-July only
wth_subset <- wth_full[month(YYYYMMDD) %in% 6:7]

# Re-validate formatting
wth_check <- format_wth(
  wth = wth_subset,
  date = "YYYYMMDD",

```

```

    temp = "TEMP",
    rhum = "RHUM",
    rain = "RAIN",
    tmin = "TMIN",
    tmax = "TMAX",
    lat = "LAT",
    lon = "LON"
  )

# Example 5: Error handling - demonstrating validation
# This will fail (missing column)
## Not run:
bad_wth <- data.frame(
  date = seq(as.Date("2000-06-30"), as.Date("2000-07-05"), by = "day"),
  temp = c(25, 26, 24, 25, 27, 26),
  rhum = c(85, 80, 90, 85, 75, 80)
  # Missing 'rain' column!
)

format_wth(
  wth = bad_wth,
  date = "date",
  temp = "temp",
  rhum = "rhum",
  rain = "rain" # This column doesn't exist
)
# Error: The following columns are not in wth: rain

## End(Not run)

# This will fail (duplicate dates)
## Not run:
dup_wth <- data.frame(
  date = c(as.Date("2000-06-30"),
           as.Date("2000-06-30"),
           as.Date("2000-07-01")),
  temp = c(25, 26, 24),
  rhum = c(85, 80, 90),
  rain = c(5, 0, 8)
)

format_wth(
  wth = dup_wth,
  date = "date",
  temp = "temp",
  rhum = "rhum",
  rain = "rain"
)
# Error: Weather data contains duplicate dates

## End(Not run)

# This will fail (gaps in dates)

```

```
## Not run:
gap_wth <- data.frame(
  date = c(
    as.Date("2000-06-30"),
    as.Date("2000-07-01"), # Skip July 1
    as.Date("2000-07-03") # Gap of 2 days!
  ),
  temp = c(25, 24, 26),
  rhum = c(85, 90, 80),
  rain = c(5, 8, 0)
)

format_wth(
  wth = gap_wth,
  date = "date",
  temp = "temp",
  rhum = "rhum",
  rain = "rain"
)
# Error: Weather data is not on a daily time-step or has gaps

## End(Not run)
```

get_audpc

Extract AUDPC Value from Disease Model Output

Description

Retrieves the pre-calculated Area Under the Disease Progress Curve (AUDPC) from an `epicrop.sim` object. AUDPC is automatically computed by `seir()` and stored as an object attribute.

Usage

```
get_audpc(x)
```

Arguments

`x` An `epicrop.sim` object returned by `seir()`, `bacterial_blight()`, `leaf_blast()`, or other disease model functions.

Value

Numeric AUDPC value (double).

Purpose

This function provides a safe, documented way to access AUDPC without directly querying attributes. It includes validation to ensure the object is a valid model result with a calculated AUDPC.

AUDPC Interpretation

AUDPC values:

- Near 0: Minimal disease throughout season (resistant or unfavorable conditions)
- Mid-range: Moderate disease development
- Near duration: Severe, sustained disease pressure

Typical ranges depend on disease and duration:

- 120-day rice season: 0-120 (perfect health to constant epidemic)
- 240-day wheat season: 0-240

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>

Examples

```
# Example 1: Basic usage - extract AUDPC from model output
wth <- get_wth(
  lonlat = c(121.25562, 14.6774),
  dates = "2000-06-30",
  duration = 120L
)

sim <- bacterial_blight(wth, emergence = "2000-06-30")
audpc <- get_audpc(sim)
paste("Bacterial blight AUDPC:", round(audpc, 2))

# Example 2: Compare AUDPC across multiple diseases
diseases <- list(
  bb = bacterial_blight(wth, emergence = "2000-06-30"),
  bs = brown_spot(wth, emergence = "2000-06-30"),
  lb = leaf_blast(wth, emergence = "2000-06-30"),
  sb = sheath_blight(wth, emergence = "2000-06-30")
)

audpc_values <- data.frame(
  disease = names(diseases),
  AUDPC = sapply(diseases, get_audpc)
)
audpc_values
# Identify most damaging disease
worst <- audpc_values[which.max(audpc_values$AUDPC), ]
paste("Most severe:", worst$disease, "AUDPC =", round(worst$AUDPC, 1))

# Example 3: Use in analysis pipeline
locations <- list(
  IRRI = c(121.25562, 14.6774),
  Manila = c(120.985, 14.6042)
)
```

```

audpc_by_location <- sapply(locations, function(lonlat) {
  wth <- get_wth(lonlat = lonlat, dates = "2000-06-30", duration = 120L)
  sim <- leaf_blast(wth, emergence = "2000-06-30")
  get_audpc(sim)
})

audpc_by_location

```

get_epicrop_weather_list

Get Weather for One or More Locations and Split by Calendar Year

Description

Retrieves one or more continuous blocks of weather data for one or more locations from given start dates and durations, then splits each result into a named list of per-year tables. Each (location, start, duration) result is an `epicrop.wth.list`. The top-level return is an `epicrop.wth.bundle` when multiple requests are processed, otherwise a single `epicrop.wth.list`.

Usage

```

get_epicrop_weather_list(
  lonlat,
  start_date,
  duration,
  years = NULL,
  mode = "pairwise",
  location_names = NULL,
  source = c("power", "era5")
)

```

Arguments

lonlat	Numeric length-2 vector <code>c(lon, lat)</code> , OR multiple locations as: numeric vector of length <code>2n</code> , a 2-column matrix/data.frame, or a list of numeric length-2 vectors.
start_date	A starting date (character <code>"YYYY-MM-DD"</code> , Date , or data.table::IDate) or a month-day string like <code>"06-30"</code> (leading <code>"-"</code> optional) used together with years.
duration	Integer(s). Number of days to retrieve from <code>start_date</code> (inclusive). Single value or vector (recycled).
years	Optional integer vector of years. Required if <code>start_date</code> contains any month-day values without a year.
mode	Character scalar: <code>"pairwise"</code> (default) pairs locations and dates by position (recycling length-1), or <code>"cross"</code> forms all combinations (locations $\times$ dates).

location_names Optional character names for locations (length = number of unique locations).

source Character scalar: weather data source, "power" (default, NASA POWER) or "era5" (ERA5 reanalysis via Open-Meteo). See [get_wth\(\)](#) for details on the differences between sources.

Value

- If only one (location, start, duration) is requested: an epicrop.wth.list.
- Otherwise: an epicrop.wth.bundle (a named list of epicrop.wth.list).

Flexible Input Formats

Locations (lonlat):

- Single numeric vector: c(lon, lat)
- Multiple locations as: numeric vector of length 2n, a 2-column matrix/data.frame, or a list of list(c(lon, lat))

Start dates (start_date):

- Full ISO date: "2000-06-30" or Date/IDate object
- Month-day string: "06-30", "6-30", "-06-30" (with years parameter)
- Vector of dates (recycled in pairwise mode or crossed in cross mode)

Duration:

- Single integer or vector (recycled)
- Interpreted as inclusive days: end = start + duration - 1

Years (optional):

- Only used when start_date contains month-day values without a year
- Computes a **cover window** per distinct month-day:
 - start = min(paste(years, md))
 - end = max(paste(years, md)) + (duration - 1).

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>

Examples

```
# Single location, month-day start date
w1 <- get_epicrop_weather_list(
  lonlat = c(121.255669, 14.16742),
  start_date = "6/30",
  duration = 120L,
  years = 2000:2001
)
```

```

# ISO start date
w2 <- get_epicrop_weather_list(
  lonlat = c(121.255669, 14.16742),
  start_date = "2000-06-30",
  duration = 120L
)

# Multiple locations × multiple dates (cross mode)
locations <- rbind(
  "IRRI_ZES" = c(121.255669, 14.16742),
  "Metro_Manila" = c(120.985, 14.6042)
)
w3 <- get_epicrop_weather_list(
  lonlat = locations,
  start_date = c("2000-06-30", "6-30"),
  duration = 120L,
  years = 2000:2001,
  mode = "cross"
)

# Multiple start dates for same location
w4 <- get_epicrop_weather_list(
  lonlat = c(121.255669, 14.16742),
  start_date = c("-06-01", "-06-14", "-06-30"),
  duration = 90L,
  years = 2001:2003
)

# ERA5 instead of NASA POWER
w5 <- get_epicrop_weather_list(
  lonlat = c(121.255669, 14.16742),
  start_date = "2000-06-30",
  duration = 120L,
  source = "era5"
)

```

get_wth

Get Weather Data for Disease Modelling

Description

Retrieves daily weather data from either the NASA POWER database or ERA5 reanalysis (via the Open-Meteo Archive API) and formats it for use in [seir\(\)](#) and **epicrop** disease models.

Usage

```
get_wth(lonlat, dates, duration = NULL, source = c("power", "era5"))
```

Arguments

lonlat	Numeric vector of coordinates as <code>c(longitude, latitude)</code> . Longitude: -180 to 180 degrees (West/East) Latitude: -90 to 90 degrees (South/North) Example: <code>c(121.255669, 14.167425)</code> for IRRI, Philippines
dates	Character vector of start date and optional end date, or character scalar start date. Format: "YYYY-MM-DD" (ISO 8601). - Length 1: Use with duration parameter - Length 2+: First is start, second is end (used only if duration is not supplied)
duration	Optional integer. Number of days to retrieve (inclusive). If supplied, it takes precedence over any end date in dates. Must be positive (> 0). If not provided and dates has only one element, an error is raised.
source	Character. Weather data source, one of "power" (default) or "era5".

Value

An `epicrop.wth` object (`data.table::data.table()` subclass) containing daily weather data with columns:

Column	Description	Units
YYYYMMDD	Date	ISO 8601 (YYYY-MM-DD)
DOY	Day of year	Integer (1-365/366)
TEMP	Mean daily temperature	Degrees C
TMIN	Minimum daily temperature	Degrees C
TMAX	Maximum daily temperature	Degrees C
RHUM	Mean daily relative humidity	Percent (0-100)
RAIN	Total daily rainfall	Millimeters
LAT	Latitude of query location	Degrees (-90 to 90)
LON	Longitude of query location	Degrees (-180 to 180)

Ready for direct use in `seir()`, `bacterial_blight()`, `leaf_blast()`, and other disease models.

Data Source

- "power" (default): NASA POWER via `nasapower::get_power()`, global daily data from 1984 onwards at 0.5-degree resolution.
- "era5": ERA5 from 1940 onwards at about 0.25-degree resolution, roughly 25 km, via the Open-Meteo Archive API (<https://open-meteo.com/en/docs/historical-weather-api>). Hourly values are aggregated to daily values (mean/min/max temperature, mean relative humidity, total precipitation) by local calendar day so that the output matches the POWER-derived columns. ERA5 data are typically available with a lag of about five days from 1940 onwards.

Both sources are modelled/reanalysis products and may differ from local weather station observations. Note that ERA5 precipitation is not the same quantity as POWER's bias-corrected PRECTOTCORR, so rainfall totals will differ between sources.

Citing Weather Data

Please be kind and cite the data and their sources. To do so, please see the [NASA POWER](#) and [Openmeteo](#) websites for how to cite the data when you use them.

Retrieved Variables

This function automatically provides:

- Mean daily temperature (2-meter height)
- Daily minimum temperature
- Daily maximum temperature
- Mean daily relative humidity (2-meter height)
- Total daily precipitation (bias-corrected for POWER)

All variables are provided in standard units: temperature in degrees C, humidity in %, rainfall in mm.

Date Handling

Two methods for specifying the weather window:

1. **Start and end dates** (character vector): `dates = c("2000-06-30", "2000-12-31")`
2. **Start date + duration** (integer): `dates = "2000-06-30", duration = 120`

Duration is interpreted as inclusive days: `end = start + duration - 1`. For example, `duration = 120` starting on June 30 ends on October 27 (120 days total).

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com> with ERA5 support adapted from code by Emerson M. Del Ponte <delponte@ufv.br> from the **r4pde**.

References

- NASA POWER project (2019). Agroclimatology, Prediction Of Worldwide Energy Resource. Funded by NASA's Applied Science Program. <https://power.larc.nasa.gov/>.
- Hersbach, H. et al. (2020). The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146, 1999-2049. doi:10.1002/qj.3803.
- Sparks, A. H., P. D. Esker, M. Bates, W. Dall'Acqua, Z. Guo, V. Segovia, S. D. Silwal, S. Tolos, and K. A. Garrett. (2008). Ecology and Epidemiology in R: Disease Progress over Time. *The Plant Health Instructor*. doi:10.1094/PHIA2008012902.
- Zippenfenig, P. (2023) Open-Meteo.com Weather API. Zenodo. doi: doi:10.5281/ZENODO.7970649.

Examples

```

# Example 1: Get weather for a date range
# IRRI Zeigler Experiment Station, wet season 2000
wth_range <- get_wth(
  lonlat = c(121.255669, 14.167425),
  dates = c("2000-06-30", "2000-12-31")
)
head(wth_range)
nrow(wth_range) # Number of days

# Example 2: Get weather by start date and duration
# 120-day window from June 30, 2000
wth_duration <- get_wth(
  lonlat = c(121.255669, 14.167425),
  dates = "2000-06-30",
  duration = 120L
)
nrow(wth_duration) # Should be 120 rows

# Example 3: Verify both methods return equivalent data
# (within floating-point precision)
all.equal(wth_range[1:120], wth_duration)

# Example 4: Use retrieved weather in disease model
wth <- get_wth(
  lonlat = c(121.255669, 14.167425),
  dates = "2000-06-30",
  duration = 120L
)

# Run bacterial blight model
bb_sim <- bacterial_blight(wth, emergence = "2000-07-01")
head(bb_sim)
plot(
  bb_sim$simday,
  bb_sim$intensity,
  type = "l",
  xlab = "Days After Emergence",
  ylab = "Disease Intensity"
)

# Example 5: Multiple locations - compare disease risk
locations <- data.frame(
  name = c("IRRI_Laguna", "BRRI_Gazipur", "CIMMYT_Mexico"),
  lon = c(121.255669, 90.2167, -99.1828),
  lat = c(14.167425, 24.0028, 19.1296)
)

# Get weather for each location
wth_list <- lapply(
  1:nrow(locations),
  function(i) {

```

```

    get_wth(
      lonlat = c(locations$lon[i], locations$lat[i]),
      dates = "2000-06-30",
      duration = 120L
    )
  }
)
names(wth_list) <- locations$name

# Run leaf blast model at each location
lb_results <- lapply(
  wth_list,
  function(w) leaf_blast(w, emergence = "2000-07-01")
)

# Compare AUDPC (disease intensity summary)
audpc_comparison <- data.frame(
  location = names(lb_results),
  AUDPC = sapply(lb_results, function(x) attr(x, "AUDPC"))
)
audpc_comparison

# Example 6: Cross-year comparison (same location, different years)
years <- 2000:2002
wth_by_year <- lapply(
  years,
  function(y) {
    get_wth(
      lonlat = c(121.255669, 14.167425),
      dates = paste0(y, "-06-30"),
      duration = 120L
    )
  }
)
names(wth_by_year) <- as.character(years)

# Run model for each year
bb_by_year <- lapply(
  wth_by_year,
  function(w) bacterial_blight(w, emergence = unique(w$YYYYMMDD)[1])
)

# Compare disease progression across years
plot_data <- data.table::rbindlist(lapply(
  seq_along(bb_by_year),
  function(i) {
    data.frame(
      year = names(bb_by_year)[i],
      simday = bb_by_year[[i]]$simday,
      intensity = bb_by_year[[i]]$intensity
    )
  }
))

```

```

plot(
  plot_data$simday,
  plot_data$intensity,
  col = as.factor(plot_data$year),
  pch = 16
)
legend(
  "topleft",
  legend = names(bb_by_year),
  col = seq_along(bb_by_year),
  pch = 16
)

# Example 7: Error handling
## Not run:
# This will fail (invalid coordinates)
get_wth(
  lonlat = c(200, 95), # Out of range
  dates = "2000-06-30",
  duration = 120
)
# Error: Longitude must be in [-180, 180], got 200

# This will fail (invalid date format)
get_wth(
  lonlat = c(121.255669, 14.167425),
  dates = "30-06-2000", # Wrong format
  duration = 120
)
# Error: dates[1] must be an ISO date ('YYYY-MM-DD')

# This will fail (missing date specification)
get_wth(
  lonlat = c(121.255669, 14.167425),
  dates = "2000-06-30" # No end date or duration
)
# Error: Provide dates as c(start, end) or supply duration

## End(Not run)

# Example 8: ERA5 data
# Get ERA5 data in the same format as POWER
wth_era5 <- get_wth(
  lonlat = c(121.255669, 14.167425),
  dates = "2000-06-30",
  duration = 120L,
  source = "era5"
)
head(wth_era5)

```

run_epicrop_model	<i>Run an epicrop Disease Model Across Emergence Dates and Locations</i>
-------------------	--

Description

A unified interface to execute any **epicrop** disease model ([bacterial_blight\(\)](#), [brown_spot\(\)](#), [leaf_blast\(\)](#), etc.) over multiple emergence dates and/or locations. Handles cross-year and within-year weather windows automatically without global weather binding.

Usage

```
run_epicrop_model(
  model_fun,
  emergence_dates,
  wth_list,
  window_days = 120L,
  output = "audpc",
  ...
)
```

Arguments

model_fun	Function. An epicrop model function such as: bacterial_blight() , brown_spot() , leaf_blast() , sheath_blight() , tungro() , modified_kim_leaf_blast() , modified_kim_sheath_blight() , leaf_rust() , or s_tritici_blotch() .
emergence_dates	Vector of emergence dates (character "YYYY-MM-DD", Date , or data.table::IDate ; must be of class epicrop.emerge from build_epicrop_emergence()).
wth_list	Named list of weather data.table::data.tables , one per year. • Names must be year strings (<i>e.g.</i>, "2001", "2002") • Each table must have a YYYYMMDD column (Date, IDate, or ISO string) • Typically created by get_epicrop_weather_list() • Can also be an epicrop.wth.bundle (multiple locations)
window_days	Integer. Days after emergence to simulate (default 120L). The model will run for exactly window_days rows when sufficient weather is available. If emergence_dates is at year-end, spans into next year.
output	Character. Output format: • "audpc" (default): Compact summary with emergence date and AUDPC • "full": All daily model columns plus emergence and AUDPC
...	Additional arguments forwarded to model_fun (<i>e.g.</i> , RcOpt, simple_wetness, custom modifier tables). Overrides model defaults.

Value

A `data.table::data.table`:

- If output = "audpc":
 - Columns: emergence (IDate), AUDPC (numeric)
 - One row per emergence date
 - AUDPC = NA_real_ if weather data are insufficient
- If output = "full":
 - All columns from the disease model (sites, latent, infectious, etc.)
 - Plus emergence (IDate) and AUDPC (numeric)
 - Multiple rows per emergence date (one per simulated day)
 - Class set to `epicrop.sim` for S3 methods
- If `wth_list` is a bundle (multiple locations):
 - Additional location column identifying source bundle member

Workflow

This function bridges `get_epicrop_weather_list()` (weather prep) and `build_epicrop_emergence()` (emergence date generation) with disease models. For each emergence date, it:

1. Slices weather data spanning emergence to emergence + window_days - 1
2. Handles year boundaries (*e.g.*, Dec 15 – Jan 15)
3. Executes the model with the sliced weather
4. Returns either a compact AUDPC summary or full daily progression

Supports weather bundles (multiple locations) by processing each location independently and combining results with a `location` column.

Cross-Year Windows

When an emergence date + window_days spans two calendar years (*e.g.*, Dec 15 – Feb 15), the function automatically:

- Fetches weather from both years
- Stitches them in chronological order
- Returns a contiguous window

Missing data (no weather for a required year) produces a warning and AUDPC = NA_real_.

Bundle Processing

If `wth_list` is an `epicrop.wth.bundle` (from `get_epicrop_weather_list()` with multiple locations or location-date pairs):

- Each bundled weather object is processed independently
- Results are combined with a `location` column
- Location labels extracted from bundle names (text before "__")

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>

See Also

- [build_epicrop_emergence\(\)](#) – Construct emergence date vectors
- [get_epicrop_weather_list\(\)](#) – Prepare weather data
- [get_audpc\(\)](#) – Extract/compute AUDPC from results
- [seir\(\)](#) – Core SEIR model
- [leaf_blast\(\)](#), [sheath_blight\(\)](#), etc. – Individual disease models

Examples

```
# --- Example 1: Single location, multiple emergence dates ---

# Build emergence dates (every 15 days, June-July, years 2001-2003)
years <- 2001:2003
month_days <- c("-06-01", "-06-15", "-07-01")
emergence <- build_epicrop_emergence(years, month_days)

# Fetch weather (one year per element)
lonlat <- c(121.255669, 14.16742)
wth_list <- get_epicrop_weather_list(
  lonlat = lonlat,
  start_date = month_days,
  duration = 120L,
  years = years
)

# Run leaf blast model across all emergence dates
results <- run_epicrop_model(
  model_fun = leaf_blast,
  emergence_dates = emergence,
  wth_list = wth_list,
  window_days = 120L,
  output = "audpc"
)
head(results)

# --- Example 2: Full progression output for detailed analysis ---

full_results <- run_epicrop_model(
  model_fun = leaf_blast,
  emergence_dates = emergence[1:5], # First 5 dates
  wth_list = wth_list,
  window_days = 120L,
  output = "full"
)

# Plot disease progression for first emergence date
```

```

first_date <- emergence[1]
sub <- full_results[emergence == first_date]
plot(sub$simday, sub$intensity, type = "l",
      main = paste("Leaf Blast:", first_date),
      xlab = "Days After Emergence", ylab = "Disease Intensity")

# --- Example 3: Sensitivity analysis (modify RcOpt) ---

# Susceptible cultivar
susceptible <- run_epicrop_model(
  model_fun = leaf_blast,
  emergence_dates = emergence,
  wth_list = wth_list,
  window_days = 120L,
  output = "audpc",
  RcOpt = 1.14 # Default
)

# Resistant cultivar (lower infection rate)
resistant <- run_epicrop_model(
  model_fun = leaf_blast,
  emergence_dates = emergence,
  wth_list = wth_list,
  window_days = 120L,
  output = "audpc",
  RcOpt = 0.8 # Lower
)

comparison <- data.table(
  emergence = susceptible$emergence,
  susceptible_AUDPC = susceptible$AUDPC,
  resistant_AUDPC = resistant$AUDPC
)
comparison[, reduction := 1 - (resistant_AUDPC / susceptible_AUDPC)]
comparison

# --- Example 4: Bundle processing (multiple locations) ---

locations <- rbind(
  "IRRI" = c(121.255669, 14.16742),
  "Laguna" = c(121.2, 14.1)
)

wth_bundle <- get_epicrop_weather_list(
  lonlat = locations,
  start_date = month_days,
  duration = 120L,
  years = years,
  mode = "cross" # All location x date combinations
)

bundle_results <- run_epicrop_model(
  model_fun = leaf_blast,

```

```

    emergence_dates = emergence,
    wth_list = wth_bundle,
    window_days = 120L,
    output = "audpc"
)

# Results now have a 'location' column
bundle_results[, .N, by = location]

```

seir	<i>Susceptible-Exposed-Infectious-Removed (SEIR) Model Framework with Lesion Expansion</i>
------	--

Description

A dynamical systems model for plant disease epidemiology that tracks disease progression through latent, infectious, and removed stages. Supports flexible aging (day or development stage), lesion expansion, and hourly leaf wetness calculations.

Usage

```

seir(
  wth,
  emergence,
  onset,
  duration,
  H0,
  I0,
  RcA,
  RcT,
  RcOpt,
  p,
  i,
  Sx,
  a = 1,
  rhlim = 90,
  rainlim = 5,
  RRS = NULL,
  RRG = NULL,
  RcW = NULL,
  RRLEX = NULL,
  simple_wetness = TRUE,
  weighted_wetness = FALSE,
  age_driver = c("day", "dvs"),
  thermal_time = NULL,
  infection_window = NULL
)

```

Arguments

wth A data.frame of class `epicrop.wth` from `get_wth()` or `format_wth()` containing daily weather data with columns:

Required Columns:

Column	Description
YYYYMMDD	Date (ISO 8601: YYYYMMDD)
DOY	Day of year (1–366)
TEMP	Mean daily temperature (°C)
RHUM	Mean daily relative humidity (%)
RAIN	Daily rainfall (mm)

Optional Columns (for advanced features):

Column	Purpose
TMIN	Minimum daily temperature (°C); required if <code>simple_wetness = FALSE</code>
TMAX	Maximum daily temperature (°C); required if <code>simple_wetness = FALSE</code>
LAT	Latitude; required if <code>simple_wetness = FALSE</code>
LON	Longitude; included in output if present

emergence	Plant emergence (transplanting for rice) date in YYYY-MM-DD format (character).
onset	Days until disease onset after emergence date (1-indexed integer). Value of 1 means disease begins on emergence day. When comparing with legacy SEIR(), use <code>legacy_onset = onset - 1</code> .
duration	Growing season length / simulation duration (integer days). Table 1 in Savary <i>et al.</i> (2012) discusses typical ranges per crop.
H0	Initial healthy sites (integer). Starting condition for compartment H at emergence. Typically 1000 or proportional to plant canopy size.
I0	Initial infective sites (integer). Inoculum at onset day. Becomes infectious after <code>p</code> days.
RcA	Age-based modifier for basic infection rate R_c (numeric matrix). Two-column format: • Column 1: crop age (days 0 to duration-1 if <code>age_driver="day"</code>; or DVS 0–2 if <code>age_driver="dvs"</code>) • Column 2: modifier (0–1, bounded) Tables 1 and 2 in Savary <i>et al.</i> (2012).
RcT	Temperature modifier for R_c (numeric matrix). Two-column format: • Column 1: temperature (°C) • Column 2: modifier (0–1, bounded) Tables 1 and 2 in Savary <i>et al.</i> (2012).
RcOpt	Potential infection rate (numeric scalar, unbounded). R_c corrected for removals and optimal conditions. Can be modified to reflect cultivar resistance (see Kim <i>et al.</i> 2015 for details).
p	Latent period duration (days). Can be:

	• Integer: constant duration across growing season • Numeric vector: interpolated against age values • Function: $f(\text{TEMP})$ to calculate latent days based on daily temperature Table 1 in Savary <i>et al.</i> (2012).
i	Infectious period duration (days). Same flexibility as p. Table 1 in Savary <i>et al.</i> (2012).
Sx	Maximum number of sites per plant/unit canopy (integer). Carrying capacity constraining growth. Table 1 in Savary <i>et al.</i> (2012).
a	Aggregation coefficient (numeric, vector, or function). Controls spatial aggregation of infection: • $a = 1$: random infection (binomial); default • $a > 1$: aggregated infection (negative binomial) See <i>a</i> - Aggregation Details for full explanation. Can be interpolated against age. Defaults to 1.
rhlim	Relative humidity threshold (%) for leaf wetness (numeric). If RHUM $\geq$ rhlim or RAIN $\geq$ rainlim, leaves are wet. Savary <i>et al.</i> (2012) used 90. Defaults to 90.
rainlim	Rainfall threshold (mm) for leaf wetness (numeric). If RAIN $\geq$ rainlim, leaves are wet regardless of RH. Savary <i>et al.</i> (2012) used 5. Defaults to 5.
RRS	Relative rate of physiological senescence (numeric, vector, or function). Causes healthy sites to senesce independent of disease. Can be: • Scalar: constant rate (per day) • Vector or matrix: interpolated against age or DVS • Function: $f(\text{DVS_or_days})$ for dynamic senescence Table 1 in Savary <i>et al.</i> (2012).
RRG	Relative rate of growth of healthy sites (numeric, vector, or function). Can be: • Scalar: constant growth rate • Vector or matrix: interpolated against age or DVS • Function: $f(\text{time})$ for flexible growth curves Table 1 in Savary <i>et al.</i> (2012).
RcW	Optional modifier curve for leaf wetness effect (numeric matrix). Only used when <code>simple_wetness = FALSE</code> (hourly wetness calculation). Two-column format: • Column 1: hours of wetness (0–24) • Column 2: R_c modifier (0–1, bounded) If NULL (default), linear relationship: 0 hrs $\rightarrow$ modifier 0; 24 hrs $\rightarrow$ 1. Ignored if <code>simple_wetness = TRUE</code>.
RRLEX	Rate of lesion expansion (numeric, vector, or function). Allows direct $H \rightarrow I$ conversion bypassing the latent period. Can be: • Scalar or vector: constant or age-dependent rate • Function: $f(\text{TEMP})$ or $f(\text{DVS})$ for temperature/age-dependent expansion Set to 0 (default) for standard SEIR (no lesion expansion).
simple_wetness	Logical. Leaf wetness calculation method: • TRUE (default): Binary (0 or 1) based on rhlim and rainlim thresholds

	• FALSE: Hourly calculation (requires TMIN, TMAX, LAT, DOY in wth); applies RcW curve to 0–24 hour range
weighted_wetness	Logical. Only used when simple_wetness = FALSE. • FALSE (default): Count hours where hourly RH $\geq$ rhlim (0–24) • TRUE: Compute weighted average RH over 24-hour day Defaults to FALSE.
age_driver	Character. Age variable for modifiers: • "day" (default): 0-indexed days from emergence; RcA and RRS are functions of days • "dvs": Development stage (0–2); RcA and RRS are functions of DVS; requires thermal_time parameter
thermal_time	List with thermal accumulation parameters (required if age_driver = "dvs"). Must contain: • base_temp: Base temperature for GDD accumulation ($^{\circ}$.RC) • gdd_to_maturity: Growing degree-days to reach maturity ($^{\circ}$.RC·days) Internally computed as: $DVS = \min \left(2, 2 \times \frac{\text{GDD}}{\text{gdd to maturity}} \right)$
infection_window	Optional list restricting disease onset to a DVS window. Must contain: • dvs_start: DVS when susceptibility begins (0–2) • dvs_end: DVS when susceptibility ends (0–2) • base_temp (optional): Base temperature for GDD (overrides thermal_time if present) • gdd_to_maturity (optional): GDD to maturity (overrides thermal_time if present) When set, Rc is multiplied by 0 outside the window, preventing infection.

Details

This function originated from specific disease models in ‘EPIRICE’ and ‘EPIWHEAT’ to model disease intensity of several rice diseases and two wheat diseases respectively. Given proper values it can be used with other pathosystems as well.

Value

A `data.table::data.table()` object of class `epicrop.sim` with one row per day and columns:

State Variables (compartment sizes at end of day):

- simday: Simulation day (1-indexed; 1 = emergence day)
- dates: Calendar date (Date class)
- sites: Healthy sites remaining
- latent: Latent (E) sites

- **infectious:** Infectious (I) sites
- **removed:** Removed (R) sites
- **senesced:** Senesced sites

Flow Variables (daily rate/change):

- **rateinf:** New infections (H→E) on this day
- **rlex:** Lesion expansion (H→I directly) on this day
- **rtransfer:** Transfer from latent to infectious (E→I) on this day
- **rremoved:** Transfer from infectious to removed (I→R) on this day
- **rgrowth:** Growth of healthy sites on this day
- **rsenesced:** Senescence of healthy sites on this day

Derived Variables:

- **diseased:** Total diseased sites (latent + infectious + removed)
- **intensity:** Proportion diseased = (latent + infectious) / (sites + latent + infectious); excludes removed and senesced
- **lat:** Latitude (if provided by wth)
- **lon:** Longitude (if provided by wth)

Area Under Disease Progress Curve (AUDPC): Accessed via `attr(result, "AUDPC")`. Computed from intensity using the trapezoidal rule over all days.

Model Structure

The model tracks five site compartments over time:

- **Healthy sites (H):** Susceptible to infection
- **Latent sites (E):** Infected but not yet infectious (duration = p days)
- **Infectious sites (I):** Can transmit infection (duration = i days)
- **Removed sites (R):** Diseased but no longer infectious
- **Senesced sites (S):** Physiologically dead

Daily transitions are driven by:

- Infection rate: $R_c \times I \times \left(1 - \frac{D}{S_x}\right)^a$ (corrected for crowding & aggregation)
- Latency progression: E→I after p days
- Recovery: I→R after i days
- Growth: H increases based on RRG (if room available)
- Senescence: H and R decrease based on RRS
- Lesion expansion (optional): H→I directly via RRLEX

Mass Balance

At any day, the constraint is:

$$\text{sites}_t + \text{latent}_t + \text{infectious}_t + \text{removed}_t + \text{senesced}_t = H_0 + \text{growth}_t - \text{senescence}_t$$

Accumulated growth is `sum(rgrowth)` and accumulated senescence is `sum(rsenesced)`. Small numerical deviations may occur due to floating-point arithmetic.

α – Aggregation Details

The infection rate is computed as:

$$\text{new infections} = I \times R_c \times \left(1 - \frac{D}{S_x}\right)^a$$

where D = diseased sites, S_x = carrying capacity, and a is the aggregation coefficient:

- **$a = 1$** (random, binomial): Assumes pathogen can access all healthy sites equally. Maximum infection rate per unit of disease.
- **$a > 1$** (aggregated, negative binomial): Pathogen cannot access all healthy sites; infections cluster spatially. Reduces effective infection rate. Refer to Savary *et al.* (2012) for biological interpretation and fitting guidance.

TMIN/TMAX Details

When `simple_wetness = FALSE`, the function calculates hourly leaf wetness for 24 hours using TMIN, TMAX, LAT, and DOY in the `wth` object. The `rhlim` and `rainlim` thresholds determine wet hours. A leaf-wetness modifier (`RcW`) converts wet hours to a 0–1 R_c scale.

When `simple_wetness = TRUE`, this calculation is skipped; leaf wetness is binary (0 or 1) based solely on daily RHUM and RAIN.

LAT/LON Details

If the `wth` object contains LAT and LON columns, they are appended to each row of the output for mapping or spatial analysis. Both must be present to be included. By default, `get_wth()` and `format_wth()` provide these.

Lesion Expansion Dynamics

When $RRLEX > 0$, the model adds a direct pathway $H \rightarrow I$, bypassing latency:

$$\text{rlex}_t = RRLEX \times I_t \times \left(1 - \frac{D}{S_x}\right)$$

Lesion expansion is computed in a two-stage update to avoid circular dependency: (1) compute pre-expansion infectious sites, (2) compute `rlex`, (3) finalize infectious count. This ensures that R_c -driven infections and lesion expansion are properly sequenced.

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>, derived from original code by Robert J. Hijmans, Rene Pangga, Jorrel Aunario, and Kwang-Hyung Kim.

References

- Sparks, A. H., P. D. Esker, M. Bates, W. Dall'Acqua, Z. Guo, V. Segovia, S.D. Silwal, S. Tolos, and K. A. Garrett, 2008. Ecology and Epidemiology in R: Disease Progress over Time. *The Plant Health Instructor*. doi:10.1094/PHIA2008012902.
- Madden, L. V., G. Hughes, and F. van den Bosch. 2007. *The Study of Plant Disease Epidemics*. American Phytopathological Society, St. Paul, MN. doi:10.1094/9780890545058.
- Savary, S., Nelson, A., Willocquet, L., Pangga, I., and Aunario, J. 2012. Modeling and mapping potential epidemics of rice diseases globally. *Crop Protection*, 34: 6–17. doi:10.1016/j.cropro.2011.11.009
- Kim, K. H., Cho, J., Lee, Y. H., and Lee, W. S. 2015. Predicting potential epidemics of rice leaf blast and sheath blight in South Korea. *Agricultural and Forest Meteorology*, 203: 191–207. doi:10.1016/j.agrformet.2015.01.011
- Savary, S., Stetkiewicz, S., Brun, F., and Willocquet, L. 2015. Modelling and mapping potential epidemics of wheat diseases. *European Journal of Plant Pathology*, 142: 771–790. doi:10.1007/s1065801506507

See Also**Disease-specific helper functions:**

- `bacterial_blight()` – Rice bacterial blight
- `brown_spot()` – Rice brown spot
- `leaf_blast()` – Rice leaf blast
- `sheath_blight()` – Rice sheath blight
- `tungro()` – Rice tungro
- `modified_kim_leaf_blast()` – Cultivar-adjusted leaf blast
- `modified_kim_sheath_blight()` – Cultivar-adjusted sheath blight
- `leaf_rust()` – Wheat leaf rust
- `s_tritici_blotch()` – Wheat septoria tritici blotch

Utilities:

- `get_wth()` – Fetch weather data
- `format_wth()` – Format weather data

Examples

```
# Create sample weather data
wth <- get_wth(
  lonlat = c(121.255669, 14.16742),
  dates = "2000-06-30",
  duration = 120L
```

```

)

# Run SEIR model with standard parameters (rice leaf blast example)
res <- seir(
  wth = wth,
  emergence = "2000-06-30",
  onset = 20,      # disease begins on day 20
  duration = 120,  # 120-day growing season
  H0 = 1000,       # 1000 initial healthy sites
  I0 = 1,          # 1 initial infected site
  RcA = rbind(c(0, 0.1), c(120, 0.1)), # age modifier (constant 0.1)
  RcT = rbind(c(25, 1), c(30, 1)),      # temp modifier (optimum at 25\ifelse{latex}{\out{$^\circ$}}{°}.RC)
  RcOpt = 0.5,    # infection rate optimum
  p = 5,          # 5-day latent period
  i = 10,         # 10-day infectious period
  Sx = 10000,     # max 10,000 sites per plant
  a = 1,          # random infection (no aggregation)
  rhlim = 90,     # wetness threshold at 90% RH
  rainlim = 5     # or 5mm rain
)

# Inspect results
head(res)
attr(res, "AUDPC") # area under disease progress curve

# Example with lesion expansion (H→I bypass)
res_lex <- seir(
  wth = wth,
  emergence = "2000-06-30",
  onset = 20,
  duration = 120,
  H0 = 1000,
  I0 = 1,
  RcA = rbind(c(0, 0.1), c(120, 0.1)),
  RcT = rbind(c(25, 1), c(30, 1)),
  RcOpt = 0.5,
  p = 5,
  i = 10,
  Sx = 10000,
  RRLEX = 0.01    # lesion expansion rate (1% per infectious site per day)
)

# Compare disease intensity with and without lesion expansion
plot(res$simday, res$intensity, type = "l", main = "SEIR Model Comparison")
lines(res_lex$simday, res_lex$intensity, col = "red", lty = 2)
legend("topright", c("Standard SEIR", "With Lesion Expansion"),
      lty = c(1, 2), col = c("black", "red"))

```

Index

audpc_dt, 3

bacterial_blight (disease_models), 9
bacterial_blight(), 3, 10, 16, 22, 26, 31, 36, 46

base::Date, 6, 7
base::seq.Date(), 7

bb (disease_models), 9

brown_rust (disease_models), 9
brown_spot (disease_models), 9
brown_spot(), 10, 16, 36, 46

bs (disease_models), 9

build_epicrop_emergence, 5
build_epicrop_emergence(), 36–38

chirps::get_chirps(), 21
cropgrowdays::daily_mean(), 22

data.frame(), 21
data.table::CJ(), 6
data.table::data.table, 36, 37
data.table::data.table(), 3, 16, 21, 22, 31, 43
data.table::IDate, 5–7, 28, 36
Date, 28, 36
disease_models, 9

format_wth, 21
format_wth(), 15, 17, 41, 45, 46

get_audpc, 26
get_audpc(), 38
get_epicrop_weather_list, 28
get_epicrop_weather_list(), 7, 36–38
get_wth, 30
get_wth(), 15, 17, 29, 41, 45, 46

lb (disease_models), 9
leaf_blast (disease_models), 9
leaf_blast(), 3, 10, 16, 22, 26, 31, 36, 38, 46
leaf_rust (disease_models), 9
leaf_rust(), 10, 16, 36, 46
lr (disease_models), 9

mk_lb (disease_models), 9
mk_sb (disease_models), 9
modified_kim_leaf_blast (disease_models), 9
modified_kim_leaf_blast(), 10, 16, 36, 46
modified_kim_sheath_blight (disease_models), 9
modified_kim_sheath_blight(), 10, 16, 36, 46

nasapower::get_power(), 21, 31

run_epicrop_model, 36
run_epicrop_model(), 3, 5, 7

s_tritici_blotch (disease_models), 9
s_tritici_blotch(), 10, 16, 36, 46
sb (disease_models), 9
seir, 40
seir(), 3, 9, 16, 21, 22, 26, 30, 31, 38
sheath_blight (disease_models), 9
sheath_blight(), 10, 16, 36, 38, 46
sort(), 6
sprintf(), 6
stb (disease_models), 9

tg (disease_models), 9
tungro (disease_models), 9
tungro(), 10, 16, 36, 46