

Package ‘insectecol’

October 8, 2026

Type Package

Title Insect Ecology Data Analysis Toolkit

Version 1.1.2

Description A collection of analytical tools for insect ecology research, currently covering age-stage, two-sex life table analysis, dose-response bioassays, temperature-dependent development and insect phenology prediction. The life table module follows the age-stage, two-sex life table theory of Chi (1988) <[doi:10.1093/ee/17.1.26](https://doi.org/10.1093/ee/17.1.26)> and Chi et al. (2020) <[doi:10.1127/entomologia/2020/0936](https://doi.org/10.1127/entomologia/2020/0936)>. It supports fast batch processing of multi-group datasets, validates raw 'csv' data, computes cohort size, mean fecundity, age-stage survival rates, age-specific survival, age-specific fecundity, life expectancy, and derived population parameters (net reproductive rate, intrinsic and finite rates of increase, mean generation time), simultaneously generates age-stage survival curves for all groups, and exports all tabular results and plots to 'Excel' in a single run. The bioassay module estimates lethal concentrations by the traditional and the weighted (improved) linear regression methods and by probit analysis (Bliss, 1934) <[doi:10.1126/science.79.2037.38](https://doi.org/10.1126/science.79.2037.38)>, with the control-mortality correction of Abbott (1925) <[doi:10.1093/jee/18.2.265a](https://doi.org/10.1093/jee/18.2.265a)>, 95% confidence intervals and chi-square goodness-of-fit tests; the lethal proportion can be set freely (e.g., 25%, 50%, 70% or 90%), so any LC value such as the LC25, LC70 or LC90 can be computed, not only the LC50. The regression plots and tables are exported to 'Excel'. The degree-day module estimates the developmental threshold temperature and the effective accumulated temperature by the linear degree-day method (Campbell et al., 1974) <[doi:10.2307/2402197](https://doi.org/10.2307/2402197)>, and fits the common nonlinear temperature-dependent development models following Logan et al. (1976) <[doi:10.1093/ee/5.6.1133](https://doi.org/10.1093/ee/5.6.1133)> (Logan-6), Lactin et al. (1995) <[doi:10.1093/ee/24.1.68](https://doi.org/10.1093/ee/24.1.68)> and Briere et al. (1999) <[doi:10.1093/ee/28.1.22](https://doi.org/10.1093/ee/28.1.22)>, plus a 7-parameter Wang model; it selects the best model by AICc, predicts durations and accumulates field degree-days. The emergence module applies the stage-grading method to a single survey of the population stage structure, i.e. to

stage-frequency data (Kiritani and Nakasuji, 1967)
[doi:10.1007/bf02514921](https://doi.org/10.1007/bf02514921) and Manly (1974)
[doi:10.1007/bf00345751](https://doi.org/10.1007/bf00345751), and projects the beginning, peak and end of
the adult emergence period (the 16%, 50% and 84% quantiles),
optionally shifting the eclosion dates by the pre-oviposition period
and the egg duration to forecast larval hatch. Further extensions,
such as median lethal temperature/time (LT50), are planned.

License MIT + file LICENSE

Encoding UTF-8

Imports dplyr, ggplot2 (>= 3.5.0), grid, magrittr, openxlsx, ragg,
scales, sysfonts, systemfonts, readr, showtext, tidyr, utils

Config/roxygen2/version 8.1.0

Suggests readxl, testthat (>= 3.0.0), writexl

Config/testthat/edition 3

URL <https://github.com/SeaGhost-0/insectecol>

BugReports <https://github.com/SeaGhost-0/insectecol/issues>

Depends R (>= 3.5)

NeedsCompilation no

Author Wangyao Li [aut, cre, cph],
Yongsheng Zhang [aut],
Huan Yu [aut]

Maintainer Wangyao Li <1561941342@qq.com>

Repository CRAN

Date/Publication 2026-10-08 14:31:07 UTC

Contents

calc_exj	3
check_path_type	4
default_stage_names	5
emergence_analyze	6
emergence_calc	9
emergence_export	11
emergence_export_plot	12
emergence_read	13
gdd_analyze	14
gdd_calc	18
gdd_check	20
gdd_compare	21
gdd_daily	22
gdd_export	22
gdd_export_plot	23
gdd_plot	24

gdd_predict	26
gdd_read	27
get_stage_names	28
lc50_analyze	29
lc50_calculate	32
lc50_export	33
lc50_export_auto	34
lc50_export_plot	35
lc50_export_plot_auto	36
lc50_plot	37
lc50_read	40
lifeTable_analyze	41
lifeTable_bootstrap	45
lifeTable_boot_test	47
lifeTable_build	48
lifeTable_calculate	50
lifeTable_calculate_all	51
lifeTable_check	52
lifeTable_export	53
lifeTable_plot	55
lifeTable_read	56
plot.emergence	57
predict.emergence	59
print.emergence	60
summary.emergence	60
Index	61

calc_exj	<i>Age-Stage Life Expectancy e_{xj}</i>
----------	--

Description

The life expectancy of an individual that has reached age x and is in stage j : the expected total number of days still to be lived by such an individual, counting the current day, computed exactly as the mean remaining lifespan of the cohort members that occupy age-stage cell (x, j) . This is the exact value of the age-stage life expectancy defined by Chi and Su (2006) as $e_{xj} = \sum_{i \geq x} \sum_{y \geq j} s'_{iy}$, where s'_{iy} is the probability that an individual of age x and stage j survives to age i and stage y . Cells that no individual can occupy return 0.

Usage

```
calc_exj(lt)
```

Arguments

lt A life_table object returned by [lifeTable_read](#).

Details

The raw individual records are used, so the result is exact. Note that when the `s_xj` matrix alone is used to evaluate the double sum (as done by matrix-based programs), small deviations can appear at ages where several stages are occupied simultaneously, because the within-cell flows cannot be recovered from `s_xj`; the values returned here do not have that ambiguity. In all other cells the two approaches coincide.

Value

A data frame with the column `Age` followed by one `e_xj` column per stage; one row per age class.

References

Chi, H. and Su, J. Y. (2006) Age-stage, two-sex life tables of *Aphidius gifuensis* (Ashmead) (Hymenoptera: Braconidae) and its host *Myzus persicae* (Sulzer) (Homoptera: Aphididae). *Environmental Entomology* 35(1), 10-21.

See Also

[calc_ex](#), [calc_sxj](#)

Examples

```
f <- system.file("extdata", "lifetable_example.csv", package = "insectecol")
lt <- lifeTable_read(f)
head(calc_exj(lt))
```

check_path_type	<i>Check the Type of an Input Path</i>
-----------------	--

Description

Checks whether the input path points to a folder or to a file. Before the check, the path is cleaned automatically: backslashes are converted to forward slashes, invisible characters that are often copied along with paths from 'Windows' dialogs (U+202A) are removed, and a redundant trailing "/" is dropped.

Usage

```
check_path_type(path)
```

Arguments

path	Character string; the path to check. Anything that is not a single, non-missing, non-empty character string is treated as invalid.
------	--

Details

The function never stops: it always returns one of the four type labels below, so callers can branch directly on the result. The cleaned path is used for the existence checks, which makes the function robust against paths copied out of the 'Windows' Explorer address bar. It is used internally by the reading functions of the package to support both folder input (batch mode) and single-file input.

Value

A character string, one of

"folder"	the path exists and is a folder
"csv file"	the path exists, is a file and has the extension csv
"other file"	the path exists and is a file with another extension
"invalid path"	the path does not exist or is not a valid path string

See Also

[lc50_read](#)

Examples

```
check_path_type(tempdir())
check_path_type(system.file("extdata", "lc50_example.csv", package = "insectecol"))
```

default_stage_names	<i>Default Developmental Stage Names</i>
---------------------	--

Description

Generates the default stage names used throughout the package: "Egg" followed by "1st instar", "2nd instar", ... one entry per immature stage. The adult labels "Female"/"Male" are appended automatically by [get_stage_names](#) and need not be included here.

Usage

```
default_stage_names(k)
```

Arguments

k Integer; number of immature stages (≥ 1).

Value

A character vector of length k.

Examples

```
default_stage_names(4)    # "Egg", "1st instar", "2nd instar", "3rd instar"
```

emergence_analyze

Predict the Emergence Period from a Stage-Structure Survey

Description

Non-interactive, fully parameter-driven entry point for the emergence-period module (the stage-grading method: one field survey of the population stage structure — e.g. a dissected-sample count of pupal grades — is turned into the projected eclosion dates of the 16 the beginning, peak and end of the adult emergence period). In the same style as [gdd_analyze](#) and [lc50_analyze](#), it (1) obtains the data — user column vectors (stage = d\$stage, count = d\$n, days = d\$days), a whole data frame, or a csv/xlsx file / folder read via [emergence_read](#) —, (2) computes the cumulative development and the interpolated quantile dates via [emergence_calc](#) and (3) optionally writes the projection figure to disk (plot_file, or the working directory under a default name). Larval hatch dates are projected as well when pre_ovip / egg_days are supplied. Tabular export is handled separately by [emergence_export](#).

Usage

```
emergence_analyze(
  stage = NULL,
  count = NULL,
  percent = NULL,
  days = NULL,
  data = NULL,
  path = NULL,
  stage_col = NULL,
  count_col = NULL,
  percent_col = NULL,
  days_col = NULL,
  survey_date,
  p = c(0.16, 0.5, 0.84),
  labels = NULL,
  pre_ovip = 0,
  egg_days = 0,
  encoding = "UTF-8",
  header = TRUE,
  pattern = "\\.(csv|xlsx|xls)$",
  plot = FALSE,
  plot_file = NULL,
  show_hatch = TRUE,
  plot_title = NULL,
  plot_sub = NULL,
  plot_xlab = NULL,
  plot_ylab = NULL,
  plot_family = NULL,
  plot_width = 10.67,
  plot_height = 6,
```

```

    plot_units = c("in", "cm", "px"),
    plot_res = 150,
    export = FALSE,
    export_path = NULL,
    export_file = NULL,
    ...
)

```

Arguments

stage, count, days	User-supplied column vectors, e.g. stage = d\$stage, count = d\$count, days = d\$days after <code>d <- read.csv("XXX.csv")</code> : the stage names, the individuals per stage and the average days from that stage to adult eclosion. percent may be used instead of count when the survey recorded percentages. When supplied, these vectors take precedence over data and path.
percent	Alternative to count: the stage shares in percent (any scaling works; the shares are normalised).
data	A data.frame with a stage column, a count (or percent) column and a days column. Used when the vectors are not supplied; takes precedence over path.
path	Optional; path to a csv/xlsx file or a folder (batch mode), read with emergence_read . Used only when neither the column vectors nor data are supplied.
stage_col, count_col, percent_col, days_col	Column names; auto-detected by default (ignored when the column vectors are supplied).
survey_date	The survey date: a Date or a character string ("2026-03-20", "2026/3/20"). Required.
p	Numeric vector of emergence quantiles, default <code>c(0.16, 0.5, 0.84)</code> (beginning / peak / end).
labels	Optional labels of the quantiles, see emergence_calc .
pre_ovip	Pre-oviposition period in days (default 0 = not used). Counted from female eclosion to egg deposition.
egg_days	Egg duration in days (default 0 = not used).
encoding, header, pattern	Reading options for emergence_read (only used when path is supplied).
plot	Logical; whether to write the projection figure to disk (default FALSE). With <code>plot = TRUE</code> the figure is always written, to <code>plot_file</code> when supplied, otherwise to the working directory under <code>emergence_plot.png</code> .
plot_file	Optional path of the exported figure, used with <code>plot = TRUE</code> : a path with an extension is the file itself (the format follows the extension — png, tiff and jpeg are supported), a path without one is a folder, created when missing, and the figure is written inside it; NULL (default) means the working directory under <code>emergence_plot.png</code> . The path written is returned as <code>plot_file</code> . The projection can still be drawn on screen at any time with <code>plot(fit)</code> on the returned <code>fit</code> .

show_hatch	Plot option, see plot.emergence .
plot_title, plot_sub, plot_xlab, plot_ylab	Plot options (title, subtitle, axis labels); NULL keeps the defaults of plot.emergence .
plot_family	Text font family, see plot.emergence (NULL keeps the default "serif" — Times New Roman on 'Windows'; Chinese characters are rendered through the device's font fallback, i.e. SimSun on Chinese 'Windows').
plot_width, plot_height, plot_units, plot_res	Physical size and resolution of the exported figure (only used with plot = TRUE), same semantics as in gdd_analyze : the composition is identical at every resolution, plot_res only adds pixels.
export	Logical; write the results document to disk? Default FALSE.
export_path	Output directory for the results document; created when missing. NULL (default) means getwd .
export_file	File name of the results document (.xlsx or .csv); NULL (default) means emergence_results.xlsx. Relative paths are resolved against export_path; absolute paths are used as-is. The parent directory is created when it does not exist. The written path is returned invisibly in the export_file component of the result.
...	Further arguments passed to emergence_calc (reserved for future options; keeps user code forward compatible).

Value

A list with components:

data	the survey table actually analysed
fit	the "emergence" object returned by emergence_calc — fit\$predictions (the quantile dates), fit\$table (cumulative development), fit\$interpolate (a closure for arbitrary quantiles); print / summary / plot / predict S3 methods are available
plot_file	the path of the written figure when plot = TRUE, otherwise NULL
export_file	the results-document path when export = TRUE, otherwise NULL

See Also

[emergence_read](#), [emergence_calc](#), [emergence_export](#), [emergence_export_plot](#)

Examples

```
f <- system.file("extdata", "emergence_example.csv",
  package = "insectecol")

## --- way 1 (recommended): read the file yourself, pass columns in ---
d <- read.csv(f)
out <- emergence_analyze(stage = d$stage, count = d$count,
  days = d$days, survey_date = "2026-03-20")
out$fit                                # three quantile dates
out$fit$table                          # cumulative development
```

```

predict(out$fit, c(0.25, 0.75))    # arbitrary quantiles

## --- way 2: pass the whole data frame ---
out2 <- emergence_analyze(data = d, survey_date = "2026-03-20")

## --- way 3: let the function read the file ---
out3 <- emergence_analyze(path = f, survey_date = "2026-03-20")

## --- hatch projection + png export + results workbook ---
## plot_file writes the projection png; export = TRUE writes the
## results workbook (quantile dates + cumulative development table).
## export_file accepts an absolute path (the parent directory is
## created when missing), so export_path is not needed here.
## plot_title / plot_xlab / plot_ylab accept custom labels; Chinese
## labels are rendered through the device's font fallback (SimSun
## on Chinese Windows)
out4 <- emergence_analyze(path = f, survey_date = "2026-03-20",
                          pre_ovip = 3, egg_days = 10,
                          plot = TRUE,
                          plot_file = file.path(tempdir(), "emergence.png"),
                          export = TRUE,
                          export_file = file.path(tempdir(),
                                                  "emergence_results.xlsx"))

out4$fit$predictions
out4$plot_file      # path of the written png
out4$export_file    # path of the written workbook

```

emergence_calc

Core Emergence-Period Calculation (Stage-Grading Method)

Description

Computes the emergence-period projection from one field survey of the population stage structure (the classic Chinese stage-grading method, *fen ling fen ji tui suan fa*). The stages are ordered by their days to eclosion (most developed first), the cumulative development share is built top-down and the eclosion dates of the requested quantiles — by default 16 (beginning), 50 i.e. the mean $\pm$ 1 SD of a normal emergence curve) — are obtained by linear interpolation of the days-to-eclosion axis, then anchored to the survey date. When *pre_ovip* / *egg_days* are supplied, the larval hatch dates are projected as well (eclosion + pre-oviposition period + egg duration).

Usage

```

emergence_calc(
  data,
  stage_col = NULL,
  count_col = NULL,
  percent_col = NULL,
  days_col = NULL,
  survey_date,

```

```

p = c(0.16, 0.5, 0.84),
labels = NULL,
pre_ovip = 0,
egg_days = 0
)

```

Arguments

data	A data.frame with a stage column (character), a count or percent column (numeric) and a days column: the average days from that stage to adult eclosion at the current temperature.
stage_col, count_col, percent_col, days_col	Column names; auto-detected by default (English and Chinese aliases, e.g. stage / count / days or their Chinese equivalents). Exactly one of count / percent is required.
survey_date	The survey date: a Date or a character string ("2026-03-20", "2026/3/20").
p	Numeric vector of emergence quantiles, default c(0.16, 0.5, 0.84).
labels	Optional labels of the quantiles (same length as p); NULL (default) uses 'Beginning (16%)' etc. for the default p, or '16%' -style labels otherwise.
pre_ovip	Pre-oviposition period in days (default 0 = not used).
egg_days	Egg duration in days (default 0 = not used).

Details

The cumulative shares are renormalised to sum to 1, so a quantile below the share of the most developed stage is the only possible extrapolation: it lies at or below that share, has partially eclosed before the survey, and is linearly extrapolated backwards from the outermost segment with a warning. Stages are sorted by days to eclosion, so the row order of the input is irrelevant; a stage sharing its days value with another stage is allowed but flagged (check the stage durations).

Value

A list of class "emergence":

survey_date	the survey Date
table	data.frame: stage, count (or percent), proportion, cumulative share, days to eclosion, projected eclosion date
predictions	data.frame: label, p, interpolated days (fractional, measured from the survey date), projected calendar date, and hatch_date when applicable
interpolate	closure function(p) returning the interpolated days for arbitrary quantiles (for further programming)
n, n_stages, pre_ovip, egg_days, p, labels	the inputs

See Also

[emergence_analyze](#) (the main entry point), [emergence_read](#), [emergence_export](#)

Examples

```
## Overwintering-generation survey, 40 individuals (Tianyang case):
d <- data.frame(
  stage = c("Pupal exuviae", paste("Pupa", 7:1), "Prepupa", "Larva 5"),
  count = c(2, 3, 5, 7, 7, 5, 3, 4, 2, 2),
  days = seq(0, 18, 2))      # days from that stage to eclosion
fit <- emergence_calc(d, survey_date = "2026-03-20")
fit      # three quantile dates
fit$table      # cumulative development
predict(fit, c(0.25, 0.75))      # arbitrary quantiles

## With the hatch projection (pre-oviposition 3 d + egg 10 d):
fit2 <- emergence_calc(d, survey_date = "2026-03-20",
  pre_ovip = 3, egg_days = 10)
fit2$predictions
```

emergence_export	<i>Save Emergence-Period Projection Results</i>
------------------	---

Description

Saves the quantile predictions (eclosion dates, and hatch dates when present) as CSV (UTF-8) or xlsx. The cumulative-development table behind the projection is written alongside by default.

Usage

```
emergence_export(x, file = "emergence_results.csv", include_stages = TRUE, ...)
```

Arguments

x	A "emergence" object returned by emergence_calc or emergence_analyze .
file	Output path; the format is chosen by the extension (.csv / .xlsx).
include_stages	Logical; whether to also write the cumulative-development table. Default TRUE.
...	Further arguments passed to write.csv (CSV mode).

Examples

```
f <- system.file("extdata", "emergence_example.csv",
  package = "insectecol")
fit <- emergence_calc(emergence_read(f), survey_date = "2026-03-20")
emergence_export(fit, tempfile(fileext = ".csv"))
```

emergence_export_plot *Export the Emergence-Period Plot as PNG*

Description

Draws the emergence-period projection of an "emergence" object on a png device ('ragg' when available, otherwise [png](#)) and writes it to disk - the standalone counterpart of `plot_file = in` [emergence_analyze](#), usable on an existing fit at any time. All plot options of `plot.emergence` are supported.

Usage

```
emergence_export_plot(  
  x,  
  file = "emergence_plot.png",  
  show_hatch = TRUE,  
  title = NULL,  
  sub = NULL,  
  xlab = NULL,  
  ylab = NULL,  
  family = NULL,  
  width = 10.67,  
  height = 6,  
  units = c("in", "cm", "px"),  
  res = 150,  
  ...  
)
```

Arguments

x	A "emergence" object returned by emergence_calc or emergence_analyze .
file	Output png path.
show_hatch, title, sub, xlab, ylab, family	Plot options, see plot.emergence ; NULL (default) keeps the function defaults.
width, height, units, res	Physical size and resolution of the png; the composition is identical at every resolution, res only adds pixels (same semantics as in emergence_analyze).
...	Further arguments passed to plot.emergence .

Value

Invisibly, file.

Examples

```
f <- system.file("extdata", "emergence_example.csv",
                 package = "insectecol")
fit <- emergence_calc(emergence_read(f), survey_date = "2026-03-20",
                     pre_ovip = 3, egg_days = 10)
emergence_export_plot(fit, tempfile(fileext = ".png"))
```

emergence_read	<i>Read a Stage-Structure Survey from a File or a Folder</i>
----------------	--

Description

Reads one field survey of the population stage structure for [emergence_calc](#): one row per stage, with a stage name column, a count (or percent) column and a days-to-eclosion column. The input may be a single csv file (delimiter auto-detected), a single xlsx/xls file (via the 'readxl' package) or a folder containing csv/xlsx files (batch mode; the files are combined and the source file name kept in a source_file column).

Usage

```
emergence_read(
  path,
  encoding = "UTF-8",
  header = TRUE,
  pattern = "\\.(csv|xlsx|xls)$"
)
```

Arguments

path	Path to a csv/xlsx file or to a folder (batch mode).
encoding	Text encoding of csv files, default "UTF-8". Use "GBK" for csv files saved from Chinese 'Excel' on 'Windows'.
header	Logical; whether the file(s) contain a header row. Default TRUE.
pattern	Regular expression selecting the files in batch mode; default: csv / xlsx / xls.

Value

A data.frame (single file), or the combined data.frame with an extra source_file column (folder input).

See Also

[emergence_calc](#), [emergence_analyze](#)

Examples

```
f <- system.file("extdata", "emergence_example.csv",
                 package = "insectecol")
d <- emergence_read(f)
head(d)
```

gdd_analyze

Analyse Temperature-Dependent Development (Main Function)

Description

Non-interactive, fully parameter-driven entry point for the degree-day module, in the same style as [lifeTable_analyze](#) and [lc50_analyze](#). It (1) obtains the long-format data — user-supplied column vectors (temp = d\$T, duration = d\$days, group = d\$stage), a whole data frame, or a csv/xlsx file / folder read via [gdd_read](#) —, (2) optionally validates the data with [gdd_check](#) (including the linear-range check), (3) fits the chosen model — or selects the best model per group by AICc with model = "auto" — via [gdd_calc](#) and (4) optionally draws the fitted curves with [gdd_plot](#) — in the same style as the [lc50_analyze](#) entry point of the bioassay module. With plot = TRUE the fitted-curves figure is always written to disk (plot_file, or the working directory under a default name); tabular export is handled separately by [gdd_export](#).

Usage

```
gdd_analyze(
  temp = NULL,
  duration = NULL,
  group = NULL,
  data = NULL,
  path = NULL,
  temp_col = NULL,
  duration_col = NULL,
  by = NULL,
  model = c("linear", "logan", "lactin", "briere1", "briere2", "wang", "auto"),
  start = NULL,
  conf_level = 0.95,
  min_n = 3,
  maxiter = 1000,
  check = TRUE,
  encoding = "UTF-8",
  header = TRUE,
  temp_from_file = FALSE,
  pattern = "\\.(csv|xlsx|xls)$",
  plot = FALSE,
  plot_file = NULL,
  plot_group = NULL,
  show_C = TRUE,
  show_Topt = TRUE,
```

```

    plot_title = NULL,
    plot_sub = NULL,
    plot_xlab = NULL,
    plot_ylab = NULL,
    plot_family = NULL,
    plot_width = NULL,
    plot_height = NULL,
    plot_units = c("cm", "in", "px"),
    plot_res = 300,
    export = FALSE,
    export_path = NULL,
    export_file = NULL,
    ...
)

```

Arguments

temp, duration, group	User-supplied column vectors, e.g. temp = d\$T, duration = d\$days, group = d\$stage after d <- read.csv("XXX.csv"). This is the recommended entry when many data sets live in one file. group is optional (omit it to fit the overall model). When supplied, these vectors take precedence over data and path.
data	A data.frame in the long format required by gdd_calc (one row per observation, with a temperature and a duration column). Used when temp / duration are not supplied; takes precedence over path.
path	Optional; path to a csv/xlsx file or a folder (batch mode), read with gdd_read . Used only when neither the column vectors nor data are supplied.
temp_col, duration_col	Column names; auto-detected by default (ignored when the column vectors are supplied).
by	Grouping variable(s), e.g. "stage"; NULL fits the overall model (ignored when the group vector is supplied, which then serves as the grouping).
model	Single model name or "auto" (best per group by AICc); default "linear". See gdd_calc for the model list and the minimum number of temperature points per model.
start	Optional named list of starting values for a nonlinear model, e.g. list(a = 1e-4, T0 = 10, Tm = 35).
conf_level	Confidence level, default 0.95.
min_n	Minimum rows per group, default 3.
maxiter	Iteration limit passed to the nonlinear fitter.
check	Logical; whether to validate the data with gdd_check before fitting (default TRUE). The result is attached to the returned list; a rate decline at high temperature triggers a targeted warning.
encoding, header, temp_from_file, pattern	Reading options for gdd_read (only used when path is supplied).

<code>plot</code>	Logical; whether to write the fitted-curves figure to disk (default FALSE). With <code>plot = TRUE</code> the figure is always written, to <code>plot_file</code> when supplied, otherwise to the working directory under <code>gdd_plot.png</code> .
<code>plot_file</code>	Optional path of the exported figure, used with <code>plot = TRUE</code> : a path with an extension is the file itself (the format follows the extension — png, tiff and jpeg are supported), a path without one is a folder, created when missing, and the figure is written inside it; NULL (default) means the working directory under <code>gdd_plot.png</code> . The path written is returned as <code>plot_file</code> . The curves can still be drawn on screen at any time with <code>plot(fit)</code> on the returned fit.
<code>plot_group, show_C, show_Topt</code>	Plot options, see gdd_plot .
<code>plot_title</code>	Custom plot title; NULL = the automatic per-group caption (group + fitted statistics). A named vector is matched per group, e.g. <code>c(Egg = "egg", Pupa = "pupa")</code> .
<code>plot_sub</code>	Custom subtitle; NULL keeps the automatic statistics caption (as subtitle when <code>plot_title</code> is set).
<code>plot_xlab, plot_ylab</code>	Custom axis labels; NULL keeps the defaults of gdd_plot .
<code>plot_family</code>	Text font family, see gdd_plot (NULL keeps the default "serif" — Times New Roman on 'Windows'; Chinese characters are rendered through the device's font fallback, i.e. SimSun on Chinese 'Windows').
<code>plot_width, plot_height</code>	Physical size of the exported figure in <code>plot_units</code> (only used with <code>plot = TRUE</code>). NULL (default, for both) picks a canvas that gives the axes a panel with a height:width ratio of about 3:4: 12 x 10 cm for a single-panel figure, 15 x 12.2 cm when several groups are drawn. Because the size is physical, the composition is identical at every resolution — <code>plot_res</code> only adds pixels.
<code>plot_units</code>	Unit of <code>plot_width</code> / <code>plot_height</code> : "cm" (default), "in" or "px". Use "cm" / "in" for publication figures. With "px" the canvas is a fixed pixel count; the text size is compensated internally so that changing <code>plot_res</code> keeps the 300-dpi composition (only the recorded dpi metadata changes).
<code>plot_res</code>	Resolution (dpi) of the exported figure, default 300. Higher values add pixels (sharper print) without changing the layout or the physical size. E.g. an 8 cm-wide figure at journal quality: <code>plot_units = "cm", plot_width = 8, plot_res = 300</code> .
<code>export</code>	Logical; write the results document to disk? Default FALSE.
<code>export_path</code>	Output directory for the results document; created when missing. NULL (default) means getwd .
<code>export_file</code>	File name of the results document (.xlsx or .csv); NULL (default) means <code>gdd_results.xlsx</code> . Relative paths are resolved against <code>export_path</code> ; absolute paths are used as-is. The parent directory is created when it does not exist. The written path is returned invisibly in the <code>export_file</code> component of the result.
<code>...</code>	Further arguments passed to gdd_calc (reserved for future model options; keeps user code forward compatible).

Value

A list with components:

<code>data</code>	the long-format data actually analysed
<code>check</code>	the <code>gdd_check</code> result, or <code>NULL</code> when <code>check = FALSE</code>
<code>fit</code>	the "gdd" object returned by <code>gdd_calc</code> — <code>fit\$results</code> (summary table), <code>fit\$fits</code> (per-group details incl. coefficient tables), <code>fit\$comparison</code> (model comparison, "auto" mode); <code>print/summary/plot/predict</code> S3 methods are available
<code>plot_file</code>	the path of the written figure when <code>plot = TRUE</code> , otherwise <code>NULL</code>
<code>export_file</code>	the results-document path when <code>export = TRUE</code> , otherwise <code>NULL</code>

See Also

`gdd_read`, `gdd_check`, `gdd_calc`, `gdd_plot`, `gdd_predict`, `gdd_compare`, `gdd_export`, `gdd_export_plot`, `gdd_daily`

Examples

```
f <- system.file("extdata", "gdd_example.csv", package = "insectecol")

## --- way 1 (recommended): read the file yourself, pass columns in ---
## Typical when many data sets live in one csv: the user reads the
## file and picks the columns with $, exactly like lifeTable_analyze()
d <- read.csv(f)
out1 <- gdd_analyze(temp = d$temp, duration = d$duration, group = d$stage)
out1$fit$results      # C, K, SE and CI per stage
summary(out1$fit)     # detailed coefficient tables

## Without a grouping column: one overall model
out1b <- gdd_analyze(temp = d$temp, duration = d$duration)

## --- way 2: pass the whole data frame ---
out2 <- gdd_analyze(data = d, by = "stage")
gdd_predict(out2$fit, temp = c(20, 25), group = "Egg")

## --- way 3: let the function read the file ---
out3 <- gdd_analyze(path = f, by = "stage")

## --- AICc model selection + png export + custom labels ---
## plot_title / plot_xlab / plot_ylab accept custom labels; Chinese
## labels are rendered through the device's font fallback
## export = TRUE additionally writes the results workbook
## (C, K, SE, CI and the model comparison table) as xlsx/csv;
## export_file accepts an absolute path (the parent directory is
## created when missing), so export_path is not needed here

out4 <- gdd_analyze(temp = d$temp, duration = d$duration, group = d$stage,
                    model = "auto", plot = TRUE,
                    plot_file = file.path(tempdir(), "gdd.png"),
                    export = TRUE,
```

```

                                export_file = file.path(tempdir(), "gdd_results.xlsx"))
out4$fit$comparison      # full comparison table, best flag included
out4$plot_file           # path of the written png
out4$export_file         # path of the written workbook

```

gdd_calc	<i>Compute Effective Accumulated Temperature and Developmental Parameters (Linear or Nonlinear Models)</i>
----------	--

Description

Fits a temperature-dependent development model to constant- temperature data. By default the classic linear degree-day model

$$V = (T - C)/K$$

is fitted by ordinary least squares, yielding the developmental threshold temperature C (deg C) and the effective accumulated temperature K (degree-days). Nonlinear models describing the full response curve (optimum and upper threshold included) are available via `model`.

Usage

```

gdd_calc(
  data,
  temp_col = NULL,
  duration_col = NULL,
  by = NULL,
  model = c("linear", "logan", "lactin", "briere1", "briere2", "wang", "auto"),
  start = NULL,
  conf_level = 0.95,
  min_n = 3,
  maxiter = 1000
)

```

Arguments

<code>data</code>	A data.frame with at least a temperature column and a duration column.
<code>temp_col, duration_col</code>	Column names (auto-detected by default).
<code>by</code>	Grouping variable(s), e.g. "stage"; NULL fits the overall model.
<code>model</code>	Single model name or "auto"; default "linear".
<code>start</code>	Optional named list of starting values for a nonlinear model, e.g. <code>list(a = 1e-4, T0 = 10, Tm = 35)</code> .
<code>conf_level</code>	Confidence level, default 0.95.
<code>min_n</code>	Minimum rows per group, default 3.
<code>maxiter</code>	Iteration limit passed to <code>nls.control</code> .

Details

Available models:

- "linear" (default): $V = (T - C)/K$. Fitted by ordinary least squares in the direction $T = C + K \cdot V$, i.e. temperature regressed on the developmental rate: K is the slope, C the intercept, and the reported standard errors are the OLS standard errors of that slope and intercept. This is the classical effective-temperature-sum formulation ($K = N(T - C)$ with the duration $N = 1/V$). The opposite direction (V regressed on T , then $C = -a/b$, $K = 1/b$) gives slightly different values whenever the points do not lie exactly on a line. Only "linear" provides K .
- "logan": Logan-6 (Logan et al. 1976). Does NOT define a lower threshold; gives T_m and T_{opt} .
- "lactin": Lactin et al. (1995). $\lambda < 0$ lets the curve cross zero, so the lower threshold is derived numerically.
- "briere1": $V = aT(T - T_0)\sqrt{T_m - T}$ (Briere et al. 1999); T_0 is the lower threshold.
- "briere2": $V = aT(T - T_0)(T_m - T)^{1/m}$; m adds flexibility.
- "wang": Wang et al. (1982), 7 parameters — needs at least 9 temperature points.
- "auto": fits all candidates and selects the best per group by AICc; the full comparison table is stored in `object$comparison`.

Nonlinear fits use `nls` (port algorithm, bounded) with heuristic starting values and deterministic restarts (start entries not belonging to the fitted model are ignored). Standard errors are asymptotic; derived quantities (T_{opt} , V_{max} , numerically derived C) carry no SE. Each nonlinear model requires at least (number of parameters + 2) temperature points.

Value

A "gdd" object: results (summary table, one row per group; the selected model in "auto" mode), fits (per-group details), data (cleaned data), and comparison (model comparison table, "auto" only).

See Also

[`gdd_read()`], [`gdd_check()`], [`gdd_compare()`], [`gdd_predict()`], [`gdd_plot()`], [`gdd_export()`], [`gdd_daily()`]

Examples

```
# csv example shipped with the package (inst/extdata)
f <- system.file("extdata", "gdd_example.csv", package = "insectecol")
df <- gdd_read(f)
fit1 <- gdd_calc(df, by = "stage") # linear (default)

fit2 <- gdd_calc(df, by = "stage", model = "briere1") # nonlinear
fit3 <- gdd_calc(df, by = "stage", model = "auto") # AICc selection
```

gdd_check

*Check Developmental Data (Long Format) Before Fitting***Description**

Non-stopping validation of a long-format development data frame (one row per observation with temperature and duration columns), i.e. the layout required by [gdd_calc](#). Reports per-row problems (missing / non-numeric / non-positive values), a per-group summary (sample sizes, temperature coverage) and a linear-range check: if the highest mean developmental rate is not reached at the highest temperature, the data contain a high-temperature decline and the linear model must not be fitted to the full range.

Usage

```
gdd_check(data, temp_col = NULL, duration_col = NULL, by = NULL)
```

Arguments

`data` A data.frame with temperature and duration columns.
`temp_col, duration_col` Column names (auto-detected by default).
`by` Optional grouping variable(s), as in [gdd_calc](#).

Value

A list:

`valid` logical vector, one entry per row of data
`problems` data.frame (row, column, reason); empty if none
`group_summary` data.frame per group: n, n_temp, temperature and duration ranges
`linear_check` data.frame per group: temperature of the maximal mean rate, highest temperature, rate_declines flag
`data` the valid rows, standardised to columns temp / duration / rate / group (NULL if no valid row remains)

See Also

[gdd_read](#), [gdd_calc](#)

Examples

```
f <- system.file("extdata", "gdd_example.csv", package = "insectecol")
chk <- gdd_check(gdd_read(f), by = "stage")
chk$group_summary # per-group sample sizes and ranges
chk$linear_check  # is the maximal mean rate at the highest temperature?
```

gdd_compare

*Compare Temperature-Development Models***Description**

Fits all candidate models (linear + nonlinear) per group and returns a comparison table with R-squared, RMSE, AIC, AICc, BIC, delta-AICc and the best model (smallest AICc). Useful for reporting model selection tables in publications.

Usage

```
gdd_compare(
  data,
  temp_col = NULL,
  duration_col = NULL,
  by = NULL,
  models = c("linear", "logan", "lactin", "briere1", "briere2", "wang"),
  start = NULL,
  conf_level = 0.95,
  min_n = 3,
  maxiter = 1000
)
```

Arguments

`data`, `temp_col`, `duration_col`, `by`
Same as in `[gdd_calc()]`.

`models`
Character vector of candidate models; default is all six. "auto" is not allowed here (this function IS the comparison).

`start`, `conf_level`, `min_n`, `maxiter`
Same as in `[gdd_calc()]`.

Value

A data.frame with columns: `group`, `model`, `n`, `converged`, `k`, `R2`, `RMSE`, `AIC`, `AICc`, `BIC`, `note`, `delta_AICc`, `best`.

Examples

```
f <- system.file("extdata", "gdd_example.csv", package = "insectecol")
gdd_compare(gdd_read(f), by = "stage")
```

gdd_daily

*Daily Accumulation of Degree-Days (Field Application)***Description**

Accumulates daily degree-days above the developmental threshold temperature, useful for predicting field phenology (e.g. the date on which a stage completes its effective accumulated temperature K).

Usage

```
gdd_daily(Tmin, Tmax, C, method = c("avg", "triangle"))
```

Arguments

Tmin, Tmax	Numeric vectors of daily minimum / maximum temperatures (deg C).
C	Developmental threshold temperature (deg C), e.g. taken from a <code>gdd_calc()</code> fit: <code>fit\$fits\$Egg\$C</code> .
method	"avg" (default) uses the daily mean temperature; "triangle" uses the single-triangle method, which credits partial degree-days when the daily minimum lies below C.

Value

A list with daily, cumulative and total.

Examples

```
gdd_daily(c(8, 10, 12), c(20, 22, 25), C = 11)$total
```

gdd_export

*Save Degree-Day Analysis Results***Description**

Saves the analysis results as CSV (UTF-8) or xlsx. The summary table (results, one row per group) is always written; the model-comparison table (available when `model = "auto"` was used), per-group coefficient tables with confidence intervals, and the cleaned data can be included optionally.

Usage

```
gdd_export(
  x,
  file = "gdd_results.csv",
  include_data = FALSE,
  include_coefs = FALSE,
  include_comparison = TRUE,
  ...
)
```

Arguments

<code>x</code>	A "gdd" object returned by <code>[gdd_calc()]</code> .
<code>file</code>	Output path; the format is chosen by the extension (.csv / .xlsx).
<code>include_data</code>	Logical; whether to also write the cleaned data. Default FALSE.
<code>include_coefs</code>	Logical; whether to write the per-group coefficient tables (estimate, SE, t, p, CI). Default FALSE.
<code>include_comparison</code>	Logical; whether to write the model comparison table when it exists (only "auto" mode). Default TRUE.
<code>...</code>	Further arguments passed to <code>write.csv</code> (CSV mode).

Examples

```
f <- system.file("extdata", "gdd_example.csv", package = "insectecol")
fit <- gdd_calc(gdd_read(f), by = "stage")
gdd_export(fit, tempfile(fileext = ".csv"))
```

<code>gdd_export_plot</code>	<i>Export the Degree-Day Plot</i>
------------------------------	-----------------------------------

Description

Draws the degree-day figure of a "gdd" object on a file device ('ragg' when available, otherwise the matching [grDevices](#) device) and writes it to disk - the standalone counterpart of `plot_file = in_gdd_analyze`, usable on an existing fit at any time. All plot options of [gdd_plot](#) are supported.

Usage

```
gdd_export_plot(
  x,
  file = "gdd_plot.png",
  group = NULL,
  show_C = TRUE,
  show_Topt = TRUE,
  title = NULL,
```

```

    sub = NULL,
    xlab = NULL,
    ylab = NULL,
    family = NULL,
    width = NULL,
    height = NULL,
    units = c("cm", "in", "px"),
    res = 300,
    ...
  )

```

Arguments

<code>x</code>	A "gdd" object returned by <code>[gdd_calc()]</code> or <code>[gdd_analyze()]</code> .
<code>file</code>	Output path; the format follows the file extension (png, tiff and jpeg are supported; any other extension is written as png).
<code>group, show_C, show_Topt, title, sub, xlab, ylab, family</code>	Plot options, see gdd_plot ; NULL (default) keeps the function defaults.
<code>width, height, units, res</code>	Physical size and resolution of the figure. NULL (default, for both) picks a canvas that gives the axes a panel with a height:width ratio of about 3:4: 12 x 10 cm for a single-panel figure, 15 x 12.2 cm when several groups are drawn (wider so that the statistics caption fits at a larger font). The composition is identical at every resolution, res only adds pixels (same semantics as in gdd_analyze).
<code>...</code>	Further arguments passed to gdd_plot .

Value

Invisibly, file.

Examples

```

f <- system.file("extdata", "gdd_example.csv", package = "insectecol")
fit <- gdd_calc(gdd_read(f), by = "stage")
gdd_export_plot(fit, tempfile(fileext = ".png"),
               title = "Developmental rate vs temperature")

```

`gdd_plot`

Plot Developmental Rate vs Temperature with the Fitted Model

Description

Draws a scatter plot of the developmental rate against temperature with the fitted model. For the linear model the threshold temperature C is marked where the fitted line crosses $V = 0$; for nonlinear models the fitted curve is drawn and C / T_{opt} / the upper threshold are marked where defined.

Usage

```
gdd_plot(
  x,
  group = NULL,
  show_C = TRUE,
  show_Topt = TRUE,
  title = NULL,
  sub = NULL,
  family = "serif",
  xlab = expression("Temperature " * italic(T) * " (" * degree * "C)"),
  ylab = expression("Developmental rate " * italic(V) * " (d^-1 * ")"),
  ...
)
```

Arguments

<code>x</code>	A "gdd" object returned by <code>[gdd_calc()]</code> .
<code>group</code>	Character vector of group names to plot; default is all groups.
<code>show_C</code>	Logical; whether to mark the threshold temperature C. Default TRUE.
<code>show_Topt</code>	Logical; whether to mark the optimum temperature (nonlinear models only). Default TRUE.
<code>title</code>	Plot title. NULL (default) uses the automatic per-group caption (group name + C / K / R-squared for the linear model, group name + model label + R-squared / AIC for nonlinear models). A single character string is used for every plotted group; a named character vector is matched by group name (e.g. <code>c(Egg = "egg", Pupa = "pupa")</code>); use <code>""</code> to drop the title. When a custom title is given, the automatic statistics caption is moved to the sub line unless <code>sub</code> is also supplied.
<code>sub</code>	Plot subtitle (bottom line). NULL = the automatic statistics caption when <code>title</code> is customised, otherwise none.
<code>family</code>	Text font family used for the title, axis labels and tick labels. Default "serif" — a portable alias that maps to Times New Roman on 'Windows' (the journal standard) and to the system serif font elsewhere, and is valid on every device (including <code>pdf()</code>). Latin characters are rendered with this font; Chinese characters are rendered through the device's font fallback, which on Chinese 'Windows' is SimSun, so mixed English-Chinese titles work without extra settings. Set to <code>""</code> to use the device default, or pass an explicit family such as "Times New Roman" (widely available on 'Windows'; may be unknown to the <code>pdf()</code> device on other platforms).
<code>xlab, ylab</code>	Axis labels; character or plotmath expression. Defaults <code>expression("Temperature " * italic(T) * " (" * degree * "C)")</code> and <code>expression("Developmental rate " * italic(V) * " (d^-1 * ")")</code> — the variable letters are italic, as journal style requires.
<code>...</code>	Further graphical parameters passed to <code>plot</code> .

Examples

```
f <- system.file("extdata", "gdd_example.csv", package = "insectecol")
fit <- gdd_calc(gdd_read(f), by = "stage")
gdd_plot(fit, group = "Egg")

## Custom title, subtitle and axis labels (portable English example):
gdd_plot(fit, group = "Egg",
         title = "Developmental rate vs temperature",
         sub = "Constant-temperature rearing experiment, 2026")

## Chinese titles are rendered through the device's font fallback
## (SimSun on Chinese 'Windows'); note that the plain pdf() device
## cannot embed CJK glyphs on many platforms, so draw on a png or
## cairo device for Chinese titles.
```

gdd_predict

*Predict Developmental Duration at Given Temperatures***Description**

Evaluates the fitted model (linear or nonlinear) at the given temperatures and returns the predicted developmental rate and duration $D = 1/V$.

Usage

```
gdd_predict(object, temp, group = NULL)
```

Arguments

object	A "gdd" object returned by [gdd_calc()].
temp	Numeric vector of target temperatures (deg C).
group	Character vector of group names to predict; default is all groups.

Value

A data.frame with columns: group, temp, pred_rate, pred_duration, within_range.

Examples

```
f <- system.file("extdata", "gdd_example.csv", package = "insectecol")
fit <- gdd_calc(gdd_read(f), by = "stage")
gdd_predict(fit, temp = c(20, 24, 28), group = "Egg")
```

Description

Reads constant-temperature development data in long format (one row per observation, with a temperature column and a duration column; optional grouping columns such as life stage). The input may be:

- a single csv file (delimiter auto-detected),
- a single xls/xlsx file (via the 'readxl' package),
- a folder containing csv/xlsx files (batch mode), e.g. one file per temperature or per stage; the files are combined and the source file name is kept in a `source_file` column.

Usage

```
gdd_read(
  path,
  encoding = "UTF-8",
  header = TRUE,
  temp_from_file = FALSE,
  pattern = "\\.(csv|xlsx|xls)$"
)
```

Arguments

<code>path</code>	Path to a csv/xlsx file or to a folder (batch mode).
<code>encoding</code>	Text encoding of csv files, default "UTF-8". Use "GBK" for csv files saved from Chinese 'Excel' on 'Windows'.
<code>header</code>	Logical; whether the file(s) contain a header row. Default TRUE.
<code>temp_from_file</code>	Logical (batch mode only); if TRUE, the file name (extension stripped) is converted to a number and written to the temp column — convenient when one file per temperature is named e.g. "25.csv". Default FALSE.
<code>pattern</code>	Regular expression selecting the files in batch mode; default: csv / xlsx / xls.

Value

A data.frame (single file), or the combined data.frame with an extra `source_file` column (folder input).

See Also

[check_path_type](#) (path handling), [gdd_check](#), [gdd_calc](#)

Examples

```
# Single csv file shipped with the package (inst/extdata)
f <- system.file("extdata", "gdd_example.csv", package = "insectecol")
df <- gdd_read(f)
head(df)

# Batch mode: one csv per temperature, temperature from the file names
d <- system.file("extdata", "gdd_batch", package = "insectecol")
df2 <- gdd_read(d, temp_from_file = TRUE)
head(df2)
```

get_stage_names

*Extract the Developmental Stage Names from the Header***Description**

Extracts the names of the developmental stages from the header (first row) of a life table csv file. The header is truncated at the sex column, the ID column is dropped, and the two adult labels before the sex column are replaced by "Female" and "Male".

Usage

```
get_stage_names(lt)
```

Arguments

lt A life_table object returned by [lifeTable_read](#).

Details

Stage names are taken from the header exactly as they are spelled in the csv file, so English headers produce English stage names and Chinese headers produce Chinese stage names. Only the two adult labels are always converted to "Female" and "Male". Internally the sex column is located by matching the header entry gender and the ID column by the header entry ID (both are fixed parts of the csv template).

Value

A character vector of stage names, e.g. c("Egg", "Larva", "Pupa", "Female", "Male"); used as column names by [calc_sxj](#) and as legend labels by [lifeTable_plot](#).

See Also

[lifeTable_read](#), [calc_sxj](#), [lifeTable_plot](#)

Examples

```
lt <- lifeTable_read(system.file("extdata", "lifetable_example.csv",
                                package = "insectecol"))

get_stage_names(lt)
```

lc50_analyze*Analyse Bioassay Data for LC Estimation (Main Function)*

Description

Non-interactive, fully parameter-driven entry point for the LC (lethal concentration) analysis. It (1) assembles the standardised data list from a data frame, a named list of data frames or three parallel vectors, (2) computes the LC estimates with the selected method(s) via [lc50_calculate](#) and (3) optionally builds the regression plot(s) in the style of [lc50_plot](#), optionally written to disk as png when `plot_file` is supplied. Tabular export is handled separately by [lc50_export](#).

Usage

```
lc50_analyze(  
  d = NULL,  
  concentration = NULL,  
  tested = NULL,  
  dead = NULL,  
  name = "bioassay",  
  path = NULL,  
  lc = 0.5,  
  method = "traditional",  
  plot = FALSE,  
  plot_file = NULL,  
  plot_width = 12,  
  plot_height = 8,  
  plot_units = "cm",  
  plot_res = 300,  
  plot_method = NULL,  
  font = "TNM",  
  unit = NULL,  
  shape = c("sigmoid", "linear"),  
  ci = TRUE,  
  ci_level = 0.95,  
  error_bar = TRUE,  
  move_thres = 0.5,  
  lc_ci = TRUE,  
  lc_p = TRUE,  
  lc_lab_gap = 0.35,  
  lc_lab_gap_right = 0.1,  
  lc_lab_dy = 0.1,  
  lc_lab_lh = 1.05,  
  export = FALSE,  
  export_path = NULL,  
  export_file = NULL  
)
```

Arguments

d	Optional; the bioassay data: a data frame with the columns Concentration, Tested and Dead (headers are matched loosely, as in lc50_read , so a header like "Concentration (mg/L)" works), a named list of such data frames (e.g. the return value of lc50_read), or NULL to build the data from the three vectors below.
concentration, tested, dead	Numeric vectors; the concentration, the number of insects tested and the number of dead insects, one entry per concentration group (replicates = repeated values). Used only when d is NULL.
name	Character; the data set name used in the results and the saved plot file names when d is a single data frame or the vectors are used (default "bioassay"); ignored for a named list input.
path	Path to a csv file (or a folder with a single csv) in the LC50 input format; read with lc50_read and used instead of d/concentration.
lc	Numeric; the lethal proportion (default 0.5 = LC50, e.g. 0.9 = LC90), passed to lc50_calculate .
method	Character; one or several of "traditional", "improved", "probit" or "all", passed to lc50_calculate .
plot	Logical; whether to build the regression plot(s) (default FALSE). The ggplot objects are also returned.
plot_file	Optional png path, used with plot = TRUE. A path with an extension is the file itself (one data set); a path without one is a folder, created when missing, and the figure(s) are written inside it as LC50_<name>.png; NULL (default) writes them to the working directory.
plot_width, plot_height, plot_units, plot_res	Physical size and resolution of the exported png (only used when plot_file is supplied); defaults 12 x 8 cm at 300 dpi.
plot_method	Character; which of the computed methods to plot (default NULL = the first method that succeeded). Ignored when plot = FALSE.
font, unit, shape, ci, ci_level, error_bar, move_thres, lc_ci, lc_p, lc_lab_gap, lc_lab_gap_right, lc_lab_dy, lc_lab_lh	Plot settings, passed to the internal plot engine exactly as in lc50_plot (unit = NULL means "mg/L", unit = "" shows no unit).
export	Logical; write the results workbook to disk? Default FALSE.
export_path	Output directory for the results workbook; created when missing. NULL (default) means getwd .
export_file	File name of the results workbook; NULL (default) means <name>_results.xlsx. Relative paths are resolved against export_path; absolute paths are used as-is. The parent directory is created when it does not exist. The written path is returned invisibly in the export_file component of the result.

Value

A list with elements data (the standardised data list, one data frame per data set), results (the list returned by `lc50_calculate`: results, summary_df, lc), plot (a named list of ggplot objects when plot = TRUE, otherwise NULL) and plot_file (the written path(s) when plot = TRUE, otherwise NULL) and export_file (the workbook path when export = TRUE, otherwise NULL).

See Also

`lc50_read`, `lc50_calculate`, `lc50_plot`, `lc50_export`, `lc50_export_plot`

Examples

```
## way 1: data frame straight from the package example csv
f <- system.file("extdata", "lc50_example.csv", package = "insectecol")
out1 <- lc50_analyze(lc50_read(f), method = "probit")
out1$results$summary_df

## way 2: three parallel vectors, no csv involved; all three methods
conc <- c(0, 1.5, 3, 6, 12, 24)
n    <- c(120, 60, 60, 60, 60, 60)
dead <- c(7, 9, 18, 32, 48, 57)
out2 <- lc50_analyze(concentration = conc, tested = n, dead = dead,
                     name = "trial1", method = "all")
out2$results$summary_df

## way 3: LC90, improved regression, plot on the linear axis
out3 <- lc50_analyze(concentration = conc, tested = n, dead = dead,
                     name = "trial1", lc = 0.9, method = "improved",
                     plot = TRUE, plot_method = "improved",
                     shape = "linear",
                     plot_file = file.path(tempdir(), "LC50_linear.png"))
invisible(out3$plot$trial1) # ggplot object: print(), customise, export
out3$plot_file             # the png that was written

## --- export: figure (png) and results workbook (xlsx) to disk ---
## plot_file writes the regression png; export = TRUE writes the
## workbook with the LC values, CI and the per-group details.
## export_file accepts an absolute path (the parent directory is
## created when missing); a relative path would be resolved against
## export_path (default getwd()). Writing the workbook takes a few
## seconds, so this last part is not run by default.

out4 <- lc50_analyze(concentration = conc, tested = n, dead = dead,
                     name = "trial1", method = "all",
                     plot = TRUE, plot_method = "probit",
                     plot_file = file.path(tempdir(), "trial1.png"),
                     export = TRUE,
                     export_file = file.path(tempdir(),
                                              "trial1_results.xlsx"))
out4$plot_file           # path of the written png
out4$export_file         # path of the written workbook (LC table + details)
```

lc50_calculate

*Batch Calculation of LC Values***Description**

Computes the LC estimates for every data set read by [lc50_read](#), using the selected estimation method(s), and returns both the detailed per-file results and a summary data frame.

Usage

```
lc50_calculate(lcd, lc = 0.5, method = "traditional")
```

Arguments

lcd	The named list returned by lc50_read .
lc	Numeric; the lethal proportion for which the concentration is estimated. The default 0.5 gives the LC50, 0.9 the LC90.
method	Character string or vector; the estimation method(s) to use: one or several of "traditional" (traditional linear regression, the default), "improved" (improved linear regression) and "probit" (probit analysis), case-insensitive, or "all" for all three in a single call. The full English method names are accepted as well.

Details

The selected method(s) are applied to every data set; the default is the traditional linear regression. The summary data frame gets one row per file and method, and the progress log reports the status of every method for every file. A file that fails (e.g. because too few valid concentrations remain after the Abbott correction) does not interrupt the batch: the error message is recorded in the Equation column of the summary data frame instead.

Value

A list with elements

results	nested list: one element per file, each holding one element per selected method with its result list (or the error message)
summary_df	data frame with one row per file and method: estimate, 95 goodness-of-fit
lc	the lethal proportion used

References

Finney, D. J. (1971) *Probit Analysis*, 3rd edition. Cambridge University Press, Cambridge.

See Also

[lc50_read](#), [lc50_traditional](#), [lc50_improved](#), [lc50_probit](#), [lc50_plot](#), [lc50_export](#)

Examples

```
f <- system.file("extdata", "lc50_example.csv", package = "insectecol")
res <- lc50_calculate(lc50_read(f), lc = 0.7)      # LC70
res$summary_df
lc50_calculate(lc50_read(f), method = "all")$summary_df  # all methods
```

lc50_export

*Export LC Results to 'Excel'***Description**

Writes the results of `lc50_calculate` into a multi-sheet 'Excel' workbook: a summary sheet with the LC estimates, confidence intervals and regression parameters of all files and methods, plus one detail sheet per file with the preprocessed data and the parameters of the successfully computed methods.

Usage

```
lc50_export(results, output_dir = NULL, filename = "LC50_results.xlsx")
```

Arguments

<code>results</code>	The result list returned by <code>lc50_calculate</code> .
<code>output_dir</code>	Character string; the output folder. If NULL (the default), a folder selection dialog is opened.
<code>filename</code>	Character string; the name of the output file (default "LC50_results.xlsx").

Details

The detail sheet of each file contains the preprocessed data (concentration, tested and dead insects, raw and corrected mortality, log concentration and probit) followed by a parameter block of every method that succeeded for that file. Files for which no method succeeded get an empty sheet.

Value

The full path of the exported xlsx file (invisibly).

See Also

[lc50_calculate](#), [lc50_plot](#)

Examples

```
## writes an xlsx workbook; that takes a few seconds, so it is not run
## by default (the example itself is what lc50_export() is for)
f <- system.file("extdata", "lc50_example.csv", package = "insectecol")
lc50_export(lc50_calculate(lc50_read(f)), output_dir = tempdir())
```

lc50_export_auto	<i>Save the LC Results of Every Data File (One xlsx per csv)</i>
------------------	--

Description

Reads the csv file(s) at path, computes the LC values with [lc50_calculate](#) and writes one 'Excel' workbook per csv file with [lc50_export](#) - named after the data file and placed next to the raw data.

Usage

```
lc50_export_auto(path = NULL, lc = 0.5, method = "traditional", suffix = NULL)
```

Arguments

path	Character string; a csv file or a folder with csv files, classified with check_path_type . NULL (the default) opens a folder selection dialog.
lc	Numeric; the lethal proportion, passed on to lc50_calculate (default 0.5 = LC50).
method	Character string; the estimation method, passed on to lc50_calculate (default "traditional").
suffix	Optional character string appended to the output file names, e.g. "_v2"; the default NULL adds nothing.

Details

Each csv file is processed completely on its own (read, calculate, export) in its own loop pass, so the outputs of different files can never mix. LB_48.csv produces LB_48.xlsx in the same folder; for a folder every csv file inside is processed the same way. A file that cannot be read is skipped with a message (a single-file input that cannot be read is an error); a file whose computation fails still gets its workbook with the error recorded in the summary sheet. Non-default settings are appended to the names so repeated runs do not overwrite each other: lc = 0.9 gives LB_48_LC90.xlsx, method = "probit" gives LB_48_probit.xlsx.

Value

The paths of the written xlsx files, invisibly.

See Also

[lc50_export_plot_auto](#) for the matching figure export, [lc50_export](#) for a custom output location

Examples

```
## writes an xlsx workbook; that takes a few seconds, so it is not run
## by default
f <- system.file("extdata", "lc50_example.csv", package = "insectecol")
tmp <- file.path(tempdir(), "lc50_example.csv")
file.copy(f, tmp, overwrite = TRUE)
lc50_export_auto(tmp)           # -> <tempdir>/bioassay.xlsx
```

lc50_export_plot	<i>Save LC50 Plots</i>
------------------	------------------------

Description

Saves one plot or a list of plots from `lc50_plot`, like `ggsave(path, plot, device = "tiff", width = 12, height = 8, dpi = 300, units = "cm", bg = "white")` but with the dpi handling of `lc50_plot` applied. The same plot object can be written at any dpi without being re-created.

Usage

```
lc50_export_plot(
  plot,
  path = NULL,
  device = "tiff",
  width = 12,
  height = 8,
  dpi = 300,
  units = "cm",
  bg = "white",
  ...
)
```

Arguments

<code>plot</code>	A ggplot or a (named) list of ggplots.
<code>path</code>	Output file (single plot) or folder (list of plots, or a path without extension); NULL (default) opens a folder selection dialog.
<code>device, width, height, units, bg</code>	Passed on to <code>ggsave</code> (defaults "tiff", 12, 8, "cm", "white").
<code>dpi</code>	Resolution of the written file (default 300).
<code>...</code>	Further arguments passed on to <code>ggsave</code> .

Value

Path(s) of the written file(s), invisibly.

See Also

[lc50_plot](#), [lc50_export](#)

Examples

```
f <- system.file("extdata", "lc50_example.csv", package = "insectecol")
plots <- lc50_plot(lc50_calculate(lc50_read(f)))
lc50_export_plot(plots$bioassay, file.path(tempdir(), "LC50_demo.tiff"))
```

lc50_export_plot_auto *Save the LC Figure of Every Data File (One image per csv)*

Description

Reads the csv file(s) at path, computes the LC values, draws the regression plot of [lc50_plot](#) for every file and saves it with [lc50_export_plot](#) - named after the data file and placed next to the raw data.

Usage

```
lc50_export_plot_auto(
  path = NULL,
  lc = 0.5,
  method = "traditional",
  suffix = NULL,
  device = "tiff",
  dpi = 600,
  width = 12,
  height = 8,
  units = "cm",
  bg = "white",
  font = "TNM",
  unit = NULL,
  shape = c("sigmoid", "linear"),
  ci = TRUE,
  ci_level = 0.95,
  error_bar = TRUE,
  move_thres = 0.5,
  lc_ci = TRUE,
  lc_p = TRUE,
  lc_lab_gap = 0.35,
  lc_lab_gap_right = 0.1,
  lc_lab_dy = 0.1,
  lc_lab_lh = 1.05,
  preview = FALSE
)
```

Arguments

path, lc, method, suffix
 Same as [lc50_export_auto](#).

device, width, height, dpi, units, bg
 Figure settings, passed on to [lc50_export_plot](#) (defaults "tiff", 12 x 8 cm, 600 dpi, white background).

font, unit, shape, ci, ci_level, error_bar, move_thres, lc_ci, lc_p, lc_lab_gap, lc_lab_gap_right, lc_lab_dy, lc_lab_lh
 Plot settings, passed on to [lc50_plot](#) unchanged.

preview
 Logical (default FALSE); also print every figure on the screen.

Details

The pipeline and the file-naming rules are the same as in [lc50_export_auto](#), plus `_linear` for `shape = "linear"`: `LB_48.csv` gives `LB_48.tiff`, `lc = 0.9`, `method = "probit"` gives `LB_48_LC90_probit.tiff`. The two functions are fully independent: each reads and computes the data on its own, so they can be called alone or in any order. A file whose computation fails gets no figure. With `method = "all"` the figure shows the first method that succeeded.

Value

Invisibly a list with elements `plots` (named list of the ggplot objects, e.g. for `print()` or `lc50_export_plot(plots, ...)`) and `files` (paths of the written images).

See Also

[lc50_export_auto](#), [lc50_plot](#), [lc50_export_plot](#)

Examples

```
f <- system.file("extdata", "lc50_example.csv", package = "insectecol")
tmp <- file.path(tempdir(), "lc50_example.csv")
file.copy(f, tmp, overwrite = TRUE)
lc50_export_plot_auto(tmp)           # -> <tempdir>/bioassay.tiff
```

lc50_plot

LC50 Regression Plots

Description

Plots every data set: observed points, the fitted curve of the computed method with its pointwise confidence band, and dashed reference lines marking the LC estimate, which is itself marked by a circle where it lies on the fitted curve. By default the concentration axis is on a log10 scale, which gives the classical symmetric S-shaped curve; `shape = "linear"` restores the original linear axis.

Usage

```
lc50_plot(
  results,
  save_path = NULL,
  font = "TNM",
  width = 7,
  height = 6,
  dpi = 300,
  unit = NULL,
  shape = c("sigmoid", "linear"),
  ci = TRUE,
  ci_level = 0.95,
  error_bar = TRUE,
  move_thres = 0.5,
  method = NULL,
  lc_ci = TRUE,
  lc_p = TRUE,
  lc_lab_gap = 0.35,
  lc_lab_gap_right = 0.1,
  lc_lab_dy = 0.1,
  lc_lab_lh = 1.05
)
```

Arguments

<code>results</code>	Result list of <code>lc50_calculate</code> .
<code>save_path</code>	Folder for the png files; NULL (default) displays the plots only.
<code>font</code>	Font family (default "TNM").
<code>width, height</code>	Figure size in inches (default 7 x 6).
<code>dpi</code>	Resolution of the saved files (default 300); at any dpi the figures keep the physical size they have at 300 dpi.
<code>unit</code>	Unit of the concentration (e.g. "mg/L"), used in the LC label and the x-axis title. NULL (the default) is treated as "mg/L"; pass "" to show no unit at all.
<code>shape</code>	"sigmoid" (default): log10 concentration axis, the symmetric S-shaped dose-response curve. "linear": the original linear concentration axis.
<code>ci</code>	Logical (default TRUE): draw the pointwise confidence band of the fitted curve.
<code>ci_level</code>	Confidence level of the curve band and of the replicate error bars (default 0.95).
<code>error_bar</code>	Logical (default TRUE): replicate rows of the same concentration are pooled to a single point, the Abbott-corrected $\text{sum(Dead)} / \text{sum(Tested)}$ (equal to the replicate mean when the replicate groups are of equal size), with a Wilson score interval at <code>ci_level</code> as the error bar, clipped to [0, 1]. FALSE draws every raw row as a plain point (the previous behaviour).
<code>move_thres</code>	Numeric (default 0.5). A dashed reference line that does not land on a regular tick normally gets an extra tick whose value is labelled next to the axis like a regular tick. If the LC position is at most <code>move_thres</code> regular tick spacings away

from the nearest tick (measured on the display axis, i.e. log10 concentrations for `shape = "sigmoid"`), that label would overlap the neighbouring tick label, so the value is drawn inside the panel instead: the concentration just above the x axis to the right of the vertical dashed line, the mortality just right of the y axis above the horizontal dashed line (each flips to the other side of its dashed line when it would not fit). `0` disables the move; with evenly spaced ticks `0.5` moves every value that is not midway between two ticks.

<code>method</code>	Character scalar, which methods to plot: a subset of <code>c("traditional", "improved", "probit")</code> , or <code>"all"</code> (default) for every method present in the results object.
<code>lc_ci</code>	Logical (default <code>TRUE</code>): show the 95 interval of the LC estimate as a second line of the LC reference label, e.g. $(0.98-1.55)$ below $LC_{50} = 1.23$ mg/L. <code>FALSE</code> omits the line.
<code>lc_p</code>	Logical (default <code>TRUE</code>): append the chi-square goodness-of-fit result (chi-square statistic and P value) as an additional line of the LC reference label. <code>FALSE</code> omits the line.
<code>lc_lab_gap</code>	Clearance between the vertical reference line and the near edge of the LC label when the label sits LEFT of the line, in text widths of the label itself (default <code>0.35</code>). Larger pushes the label further away from the line; smaller moves it towards it.
<code>lc_lab_gap_right</code>	The same clearance when the label sits RIGHT of the line (default <code>0.1</code> , smaller than <code>lc_lab_gap</code> because the label then hangs below the crossing, where a smaller gap keeps it closer to the reference line).
<code>lc_lab_dy</code>	Clearance between the LC label block and the LC crossing, in y-axis units (default <code>0.1</code>): the distance from the crossing to the edge of the block that faces it. The block is placed in the diagonal quadrant the fitted curve never enters (above the crossing when the label sits left of the vertical reference line, below it when the label sits right), anchored by that facing edge, so adding lines or changing <code>lc_lab_lh</code> grows the block away from the crossing and never onto the dashed reference line. Larger moves the whole block further from it.
<code>lc_lab_lh</code>	Line spacing of the LC label in multiples of its font size (<code>1</code> = single spacing, default <code>1.05</code>). The lines are spaced evenly whichever of them <code>lc_ci</code> / <code>lc_p</code> switches on. The spacing is measured in the nominal size the label was calibrated at (25 pt on the 300-dpi reference device), while the text itself renders at a third of that, so the default leaves a visibly loose line pitch that also clears the sub- and superscripts of the plotmath lines (LC_{50} , χ^2).

Details

Replicates of the same concentration are pooled and drawn as the Abbott-corrected pooled mortality with Wilson score intervals. The LC reference label is centred around the crossing of the two dashed reference lines. Because the sigmoid only ever passes through the lower-left and upper-right quadrants around that crossing, the label is placed in one of the two free ones: above the crossing when it sits left of the vertical reference line, below it when it sits right, at a clearance of `lc_lab_dy`. If a replicate error bar reaches into the label block, the block is shifted vertically by the smallest amount that restores a clearance of about one line height from the bar end, preferring the direction that keeps it on its own side of the crossing. The extra x-axis value always sits to the right of the

vertical dashed line (it moves to the left only when it would run off the right panel edge); when the label block dips into the value’s strip, the block is raised clear of it instead.

Value

Named list of ggplot objects (invisibly).

File names

Saved files are named after the data set plus suffixes for every non-default setting that changes the look (`_linear` for `shape = "linear"`, `_noband` for `ci = FALSE`, `_nobar` for `error_bar = FALSE`, `_nolcCI` for `lc_ci = FALSE`, `_nochi` for `lc_p = FALSE`, and the selected method names), so plots saved to one folder can never overwrite each other.

See Also

[lc50_export](#), [lc50_export_plot](#)

Examples

```
f <- system.file("extdata", "lc50_example.csv", package = "insectecol")
res <- lc50_calculate(lc50_read(f))
plots <- lc50_plot(res, save_path = tempdir())
plots <- lc50_plot(res, shape = "linear", save_path = tempdir()) # original axis
plots <- lc50_plot(res, ci = FALSE, error_bar = FALSE,
                  save_path = tempdir()) # bare version
```

lc50_read	<i>Read Bioassay Data for LC Estimation</i>
-----------	---

Description

Reads the raw csv files of a dose-response bioassay (one file per insecticide, population or similar) and returns a list of standardised data frames that the LC functions of the package work with.

Usage

```
lc50_read(path = NULL)
```

Arguments

path	Character string; the data path: a folder (all csv files inside are read) or a single csv file. If NULL (the default), a folder selection dialog is opened.
------	---

Details

Each csv file must contain one row per concentration with three required columns: the concentration, the number of insects tested and the number of dead insects. The column names are matched loosely against the fixed keywords of the csv template, so headers with additional text such as units (e.g. a concentration header with "(mg/L)" appended) are recognised as well. Rows with a concentration of zero are treated as the control group and are used for the Abbott correction during the analysis.

The data are standardised and validated while reading: rows with non-numeric or missing entries are dropped, the number of tested insects must be positive, the number of dead insects must lie between zero and the number of tested insects, at least one concentration greater than zero must be present, and the rows are sorted by increasing concentration. The file encoding is detected automatically (UTF-8 with BOM and GBK are tried), so files written by both English and Chinese versions of 'Excel' can be read.

Value

A named list with one data frame per csv file; the list elements are named after the files (without extension) and each data frame has the columns Concentration, Tested and Dead.

References

Abbott, W. S. (1925) A method of computing the effectiveness of an insecticide. *Journal of Economic Entomology* 18(2), 265-267.

See Also

[lc50_calculate](#) for the analysis workflow, [check_path_type](#) for the path handling.

Examples

```
f <- system.file("extdata", "lc50_example.csv", package = "insectecol")
lcd <- lc50_read(f)
lcd$bioassay
if (interactive()) lcd <- lc50_read() # interactive folder dialog
```

lifeTable_analyze	Analyse a Life Table from User-Supplied Columns (Main Function)
-------------------	---

Description

Non-interactive, fully parameter-driven entry point for the age-stage, two-sex life table analysis. It (1) builds a `life_table` object from column vectors of an already loaded data frame (e.g. after `data <- read.csv("XXX.csv")`, or accepts a ready `life_table` object), (2) computes the life table parameters and (3) optionally draws the age-stage survival curves with customisable title, axis titles and legend labels, written to disk as png (to the working directory when `plot_file` is not given). Tabular export is handled separately by [lifeTable_export](#).

Usage

```

lifeTable_analyze(
  lt = NULL,
  path = NULL,
  stages = NULL,
  adult_days = NULL,
  sex = NULL,
  oviposition = NULL,
  stage_names = NULL,
  file_name = NULL,
  check = TRUE,
  fecundity = TRUE,
  bootstrap = FALSE,
  B = 1e+05,
  seed = NULL,
  plot = FALSE,
  title = NULL,
  x_title = "Age(days)",
  y_title = expression("Age-Stage Survival Rate (" * italic(S)[italic(xj)] * ")"),
  legend_labels = NULL,
  dpi = 300,
  plot_file = NULL,
  plot_width = 12,
  plot_height = 8,
  plot_units = "cm",
  plot_res = 300,
  export = FALSE,
  export_path = NULL,
  export_file = NULL
)

```

Arguments

lt	Optional; an existing life_table object (from lifeTable_read or lifeTable_build). If NULL (default), the object is built from stages, adult_days, sex and oviposition.
path	Path to a csv file; read with lifeTable_read and used instead of lt/stages.
stages, adult_days, sex, oviposition, stage_names, check	Passed to lifeTable_build (ignored when lt is supplied).
file_name	Character; data set name, used as the default plot title and as the base name of the exported png and xlsx. NULL (default) keeps the name stored in the life table object (for a csv input, the file name without extension); an explicit value overrides it in every input mode.
fecundity	Logical; whether to compute the reproduction-related parameters (F, F_xj, m_x, R0, r, lambda, T). FALSE skips them entirely - oviposition is then not required at all and may be left NULL.
bootstrap	Logical; whether to estimate the standard errors and percentile confidence intervals of all scalar parameters with the bootstrap technique of 'TWOSEX-

	MSChart' via lifeTable_bootstrap (default FALSE). The result is attached as results\$boot and is exported by lifeTable_export as an extra worksheet.
B	Integer; number of bootstrap replicates (only used when bootstrap = TRUE). The 'TWOSEX-MSChart' standard is 100000 (the default).
seed	Integer; seed of the bootstrap random number generator (only used when bootstrap = TRUE); NULL uses the current R session state.
plot	Logical; whether to draw the age-stage survival curves (default FALSE). The returned ggplot object can be printed, customised further or passed to lifeTable_export .
title	Character; plot title. NULL = file_name.
x_title, y_title	Character or plotmath expression; axis titles. Defaults "Age(days)" and expression("Age-Stage Survival Rate (<i>"*italic(S)[italic(xj)]*"</i>)") (S with an italic xj subscript).
legend_labels	Character vector; legend labels, one per stage (immature stages + Female + Male), e.g. c("Egg", "1st instar", "Pupa", "Female", "Male"). NULL (default) = the stage names of the data (Egg, 1st instar, 2nd instar, ..., Female, Male).
dpi	Numeric; resolution used for scaling the text of the plot (default 300).
plot_file	Optional png path, used with plot = TRUE. A path with an extension is the file itself; a path without one is a folder, created when missing, and the figure is written inside it; NULL (default) writes the figure to the working directory under <file_name>_plot.png. The path written is returned as plot_file.
plot_width, plot_height, plot_units, plot_res	Physical size and resolution of the exported png (only used when plot_file is supplied); defaults 12 x 8 cm at 300 dpi.
export	Logical; write the results workbook to disk? Default FALSE.
export_path	Output directory for the results workbook; created when missing. NULL (default) means getwd .
export_file	File name of the results workbook; NULL (default) means <file_name>_results.xlsx. Relative paths are resolved against export_path; absolute paths are used as-is. The parent directory is created when it does not exist. The written path is returned invisibly in the export_file component of the result.

Value

A list with components lt (the life_table object), results (the list returned by [lifeTable_calculate_all](#); additionally containing boot, the [lifeTable_bootstrap](#) result, when bootstrap = TRUE), plot (the ggplot object when plot = TRUE, otherwise NULL), plot_file (the png path when plot = TRUE, otherwise NULL) and export_file (the workbook path when export = TRUE, otherwise NULL).

See Also

[lifeTable_build](#), [lifeTable_calculate_all](#), [lifeTable_bootstrap](#), [lifeTable_plot](#), [lifeTable_export](#)


```

                                "Female", "Male"))
invisible(out3$plot)      # print or further customise the ggplot object
out3$plot_file           # the png that was written

## --- way 4: export the figure (png) and the results workbook (xlsx) ---
## plot_file writes the survival-curve png; export = TRUE writes the
## workbook with all result tables (Summary, s_xj, l_x, e_x, e_xj, ...).
## export_file accepts an absolute path (the parent directory is
## created when missing); a relative path would be resolved against
## export_path (default getwd()). Writing the workbook takes well over
## 5 seconds, so this last part is not run by default.

out4 <- lifeTable_analyze(stages = d[2:8], adult_days = d$Adult,
                          sex = d$gender, oviposition = d[, 11:17],
                          file_name = "Example - export",
                          plot = TRUE,
                          plot_file = file.path(tempdir(), "sxj.png"),
                          export = TRUE,
                          export_file = file.path(tempdir(),
                                                    "life_table_results.xlsx"))

out4$plot_file           # path of the written png
out4$export_file         # path of the written workbook (all result tables)

```

lifeTable_bootstrap *Bootstrap Standard Errors of Life Table Parameters*

Description

Estimates the standard errors and percentile confidence intervals of all scalar life table parameters with the bootstrap technique used by 'TWOSEX-MSChart': complete individual records (stage durations, adult days, sex and daily oviposition) are resampled with replacement B times, and every parameter is recomputed from each resampled cohort. Individuals that died before the adult stage are resampled as well - they carry their mortality and zero fecundity into the replicates.

Usage

```
lifeTable_bootstrap(lt, B = 1e+05, seed = NULL, conf.level = 0.95)
```

Arguments

lt	A life_table object returned by lifeTable_read or lifeTable_build .
B	Integer; number of bootstrap replicates. The published 'TWOSEX-MSChart' standard is 100000 (the default); smaller values run faster but give rougher standard errors.
seed	Integer; seed for the random number generator. Set it to make the results exactly reproducible; NULL (default) uses the current R session state. The session state is restored when the function exits.
conf.level	Numeric; confidence level of the percentile intervals (default 0.95).

Details

The parameters covered are the mean developmental time of every immature stage (averaged over the individuals that entered the stage, including those that died during it), the female and male adult longevity, and - when oviposition data are present - the mean fecundity F , the net reproductive rate R_0 , the intrinsic rate of increase r , the finite rate of increase λ and the mean generation time T . The intrinsic rate is solved per replicate from the Euler-Lotka equation with the same age convention as `calc_r`, but with a widened search interval so that declining cohorts ($R_0 < 1$, i.e. $r < 0$) are also handled. Replicates in which no female is drawn have NA fecundity, and replicates without any offspring have NA r , λ and T ; such replicates are excluded from the means, standard errors and intervals of the affected parameters (column `Valid_B` shows how many remained). Curve-type outputs (`s_xj`, `l_x`, `m_x`, `e_x`) are not bootstrapped.

Attach the result to an analysis to have it exported by `lifeTable_export`: `results$boot <- lifeTable_bootstrap(lt)`.

Value

An object of class `life_table_boot`: a list with

<code>summary</code>	data frame; one row per parameter with the original point estimate, the bootstrap mean, the bootstrap standard error and the percentile confidence interval
<code>boot</code>	numeric matrix; the raw $B \times p$ replicate values
<code>B</code> , <code>conf.level</code> , <code>N</code> , <code>parameters</code>	the settings used

References

Meyer, J. S., Ingersoll, C. G., McDonald, L. L. and Boyce, M. S. (1986) Estimating uncertainty in population growth rates: jackknife vs. bootstrap. *Ecological Modelling* 29, 251-271.

Efron, B. and Tibshirani, R. J. (1993) *An Introduction to the Bootstrap*. New York: Chapman and Hall.

Chi, H., You, M. S., Atlihan, R., Smith, C. L., Kavousi, A., Ozgokce, M. S., Guncan, A. and Tuan, S. J. (2020) Age-stage, two-sex life table: an introduction to theory, data analysis, and application. *Entomologia Generalis* 40(2), 103-124.

See Also

`lifeTable_boot_test` for the paired bootstrap test between two cohorts, `lifeTable_calculate_all` for the point estimates.

Examples

```
f <- system.file("extdata", "lifetable_example.csv", package = "insectecol")
lt <- lifeTable_read(f)

## B = 100000 is the recommended setting for publications; a smaller
## B is used here so that the example runs fast
bt <- lifeTable_bootstrap(lt, B = 2000, seed = 1)
bt$summary
```

lifeTable_boot_test	<i>Paired Bootstrap Test Between Two Life Tables</i>
---------------------	--

Description

Compares the life table parameters of two cohorts with the paired bootstrap test used by 'TWOSEX-MSChart': both cohorts are resampled independently B times, the differences $d = \text{parameter}(\text{group } 1) - \text{parameter}(\text{group } 2)$ are formed replicate by replicate, and the 95 percent percentile interval of the differences is inspected. Following the convention of the life table literature, the difference is considered significant when the confidence interval of the difference does not include zero.

Usage

```
lifeTable_boot_test(lt1, lt2, B = 1e+05, seed = NULL, conf.level = 0.95)
```

Arguments

lt1, lt2	life_table objects (e.g. two treatments or two host plants) returned by lifeTable_read or lifeTable_build .
B	Integer; number of bootstrap replicates per group (default 100000, the 'TWOSEX-MSChart' standard).
seed	Integer; seed for the random number generator; the session state is restored when the function exits.
conf.level	Numeric; confidence level of the intervals of the differences (default 0.95).

Details

The two cohorts are resampled independently, so they do not need to have the same number of individuals. The parameters compared are those of [lifeTable_bootstrap](#) that occur in BOTH cohorts; if the stage structures differ, only the common parameters are compared and a warning is issued. A bootstrap p-value is reported in addition to the interval test: $2 * \min(P(d \leq 0), P(d \geq 0))$ over the valid replicates. Replicates in which a parameter is undefined in either group (e.g. no female drawn) are dropped from the comparison.

Value

A data frame with one row per compared parameter:

Parameter	parameter name (as in lifeTable_bootstrap)
Boot_mean_1, Boot_mean_2	bootstrap means of the two groups
Diff	mean of the bootstrap differences (group 1 minus group 2)
CI_low, CI_high	percentile interval of the differences
Significant	logical; TRUE when the interval excludes zero
P_bootstrap	two-sided bootstrap p-value

The attribute groups contains the file names of the two cohorts.

References

- Meyer, J. S., Ingersoll, C. G., McDonald, L. L. and Boyce, M. S. (1986) Estimating uncertainty in population growth rates: jackknife vs. bootstrap. *Ecological Modelling* 29, 251-271.
- Chi, H., You, M. S., Atlihan, R., Smith, C. L., Kavousi, A., Ozgokce, M. S., Guncan, A. and Tuan, S. J. (2020) Age-stage, two-sex life table: an introduction to theory, data analysis, and application. *Entomologia Generalis* 40(2), 103-124.

See Also

[lifeTable_bootstrap](#) for the standard errors of a single cohort.

Examples

```
f <- system.file("extdata", "lifetable_example.csv", package = "insectecol")
lt1 <- lifeTable_read(f)

## compare the full cohort with its first half (demo only - a real
## comparison would use two different treatments)
lt2 <- lt1
lt2$data <- lt1$data[1:12, ]
rownames(lt2$data) <- NULL
lt2$file_name <- "Example (first half)"

## B = 100000 is the recommended setting for publications; a smaller
## B is used here so that the example runs fast
lifeTable_boot_test(lt1, lt2, B = 2000, seed = 1)
```

lifeTable_build

Build a Life Table Object from User-Supplied Columns

Description

Assembles a `life_table` object (the same structure returned by [lifeTable_read](#)) from individual column vectors already loaded into the R session, e.g. after `data <- read.csv("XXX.csv")`. This is the entry point for analysing data that do not come from a package-conform csv file.

Usage

```
lifeTable_build(
  stages,
  adult_days,
  sex,
  oviposition = NULL,
  stage_names = NULL,
  file_name = "life_table",
  check = TRUE
)
```

Arguments

stages	Stage-duration columns, one column per immature stage: a data frame, a matrix or a list of equal-length vectors (column j = days spent in immature stage j; blank/NA for stages not reached). If it is a named data frame/list, the names are used as stage names.
adult_days	Numeric vector; adult survival days (NA for individuals that died before the adult stage).
sex	Character/factor vector; F, M or N (died before adult).
oviposition	Optional; daily oviposition records, one column per day: a data frame, a matrix (rows = individuals in the same order as sex) or a single vector (one column). NULL if the reproduction-related parameters should not be computed (see fecundity in lifeTable_calculate_all).
stage_names	Character vector of stage names (length = number of columns of stages). NULL (default) uses the names of stages if it has non-empty names, otherwise default_stage_names .
file_name	Character; data set name (default plot title, base name of the exported xlsx).
check	Logical; validate the data with lifeTable_check (default TRUE).

Value

A life_table object, ready for all calc_*, lifeTable_plot and lifeTable_export functions.

Examples

```
## The raw example data shipped with the package
f <- system.file("extdata", "lifetable_example.csv", package = "insectecol")
## ^^ change to the actual package name
d <- read.csv(f)

## 1) Standard build: column-range subset of stage columns + adult days
##   + sex + oviposition columns (positional indexing is robust to
##   the space-containing headers like "1st instar")
lt1 <- lifeTable_build(d[2:8], adult_days = d$Adult, sex = d$gender,
                      oviposition = d[, 11:17], file_name = "Example")
names(lt1)      # components of the life_table object
head(lt1$df)    # wide table: ID + stages + Adult + gender + oviposition

## 2) Survival analysis only: omit oviposition entirely. Legal since
##   the data checker skips the oviposition check when the table ends
##   at the sex column (use fecundity = FALSE in the analysis).
lt2 <- lifeTable_build(d[2:8], adult_days = d$Adult, sex = d$gender)

## 3) Named list: the list names become the stage names
lt3 <- lifeTable_build(list(Egg = d[[2]], "1st instar" = d[[3]],
                          "2nd instar" = d[[4]], "3rd instar" = d[[5]],
                          "4th instar" = d[[6]], Prepupa = d[[7]],
                          Pupa = d[[8]]),
                      adult_days = d$Adult, sex = d$gender,
                      oviposition = d[, 11:17])
```

```
## 4) Friendly stage names via stage_names: exactly one per IMMATURE
##   stage. The adult labels "Female" and "Male" are appended
##   automatically and must NOT be included.
lt4 <- lifeTable_build(d[2:8], adult_days = d$Adult, sex = d$gender,
                      oviposition = d[, 11:17],
                      stage_names = c("Egg", "L1", "L2", "L3", "L4",
                                       "Prepupa", "Pupa"))

## 5) A common mistake, handled gracefully: stage_names wrongly
##   including the adult labels. The extra two entries are dropped
##   with a warning (only a WARNING - the build still succeeds).
lt5 <- lifeTable_build(d[2:8], adult_days = d$Adult, sex = d$gender,
                      oviposition = d[, 11:17],
                      stage_names = c("Egg", "L1", "L2", "L3", "L4",
                                       "Prepupa", "Pupa",
                                       "Female", "Male"))

## 6) Skip the consistency check (e.g. oviposition columns
##   deliberately shorter than the adult life span)
lt6 <- lifeTable_build(d[2:8], adult_days = d$Adult, sex = d$gender,
                      oviposition = d[, 11:17], check = FALSE)
```

lifeTable_calculate *Batch Analysis of Life Table Data*

Description

Runs the complete workflow (reading, validation, calculation, plotting and exporting) for every csv file in a folder, or for a single csv file. Each data set gets its own 'Excel' workbook; in addition an all.xlsx with the summary of all files is created. Files that fail (e.g. because of data errors) are skipped and reported at the end without interrupting the remaining files.

Usage

```
lifeTable_calculate(
  path,
  output_path = NULL,
  plot = TRUE,
  keep_tiff = FALSE,
  dpi = 300,
  bootstrap = FALSE,
  B = 1e+05,
  seed = NULL
)
```

Arguments

path	Character; the data path: a folder (all csv files inside are analysed) or a single csv file. The type of the path is determined by check_path_type().
------	---

output_path	Character; the export folder. Defaults to the parent folder of the csv file (single-file mode) or the data folder itself (folder mode).
plot	Logical; whether the age-stage survival curves are generated and embedded into the workbooks (default TRUE).
keep_tiff	Logical; whether to keep the standalone tiff files (default FALSE).
dpi	Numeric; resolution of the exported images (default 300).
bootstrap	Logical; whether to estimate the standard errors and percentile confidence intervals of all scalar parameters of every file with lifeTable_bootstrap (default FALSE). Each workbook then contains an extra worksheet with the bootstrap results and the summary workbook all.xlsx gains one _SE column per population parameter.
B	Integer; number of bootstrap replicates per file (only used when bootstrap = TRUE). The 'TWOSEX-MSChart' standard is 100000 (the default).
seed	Integer; base seed of the bootstrap random number generator (only used when bootstrap = TRUE); file p is analysed with seed seed + p. NULL uses the current R session state.

Value

A summary data frame with one row per successfully analysed file (population parameters as columns, plus their bootstrap standard errors as _SE columns when bootstrap = TRUE); the attribute error_files contains the names of the files that failed.

See Also

[lifeTable_read](#), [lifeTable_calculate_all](#), [lifeTable_bootstrap](#), [lifeTable_plot](#), [lifeTable_export](#)

Examples

```
## reads the example file, bootstraps it and writes an xlsx workbook
## plus a figure; that takes well over 5 seconds, so it is not run by
## default
f <- system.file("extdata", "lifetable_example.csv", package = "insectecol")
lifeTable_calculate(f, output_path = file.path(tempdir(), "insectecol-demo"))
```

lifeTable_calculate_all

Calculate All Parameters of One Life Table

Description

Convenience wrapper that computes all life table parameters of one data set in a single call: cohort size, mean fecundity, the age-stage survival rates, the age-specific rates and the derived population parameters (R_0 , r , λ , T).

Usage

```
lifeTable_calculate_all(lt, fecundity = TRUE)
```

Arguments

lt	A life_table object returned by lifeTable_read .
fecundity	Logical; whether to compute the reproduction-related parameters (F, F_xj, m_x, R0, r, lambda, T). FALSE skips them entirely; no oviposition data are then required.

Details

The intermediate results are passed on internally, so nothing is computed twice: s_xj first, then l_x and the life expectancies (e_x and e_xj), then F_xj and m_x, then r and R0, and finally lambda = exp(r) and T = log(R0) / r.

Value

A named list with elements N, F, sxj, lx, fxj, mx, ex, exj, R0, r, lambda and T With fecundity = FALSE (or when no oviposition data are supplied), fxj and mx are NULL and F, R0, r, lambda, T are NA_real_.

See Also

The individual calc_* functions; [lifeTable_calculate](#) for the batch workflow.

Examples

```
f <- system.file("extdata", "lifetable_example.csv", package = "insectecol")
lt <- lifeTable_read(f)
results <- lifeTable_calculate_all(lt)
results$R0
lifeTable_calculate_all(lt, fecundity = FALSE)$N
```

lifeTable_check

Validate a Life Table Data Set

Description

Checks the age data (illegal characters, negative values, illegal gaps) and verifies for every female that the number of her oviposition records matches her recorded survival days. If a problem is found, the function stops and reports the exact row and column positions of all offending cells.

Usage

```
lifeTable_check(lt)
```

Arguments

lt A life_table object returned by [lifeTable_read](#).

Details

Two kinds of problems are detected:

- **age data errors:** cells in the stage-duration and survival-day columns that contain illegal characters or negative values, or illegal gaps within a sequence of values;
- **oviposition errors:** for every female row, the number of non-empty daily fecundity cells must equal the adult survival days recorded in column $n - 1$.

The check runs automatically inside [lifeTable_read](#) unless `check = FALSE` is used there.

Value

`invisible(TRUE)` if no error is found; otherwise the function stops with a detailed error message.

See Also

[lifeTable_read](#)

lifeTable_export	<i>Save the Results of One Life Table Analysis</i>
------------------	--

Description

Writes all results of one data set into a multi-sheet 'Excel' workbook (<file name>_out.xlsx): the population parameters, the age-stage survival rates, the age-specific rates and, optionally, the survival curve plot.

Usage

```
lifeTable_export(
  lt,
  results,
  output_path = getwd(),
  plot = NULL,
  keep_tiff = FALSE,
  dpi = 300,
  filename = NULL
)
```

Arguments

<code>lt</code>	A <code>life_table</code> object returned by <code>lifeTable_read</code> .
<code>results</code>	The result list returned by <code>lifeTable_calculate_all</code> .
<code>output_path</code>	Character; folder the workbook is written to. Defaults to the current working directory.
<code>plot</code>	A ggplot object (usually from <code>lifeTable_plot</code>); if NULL (default) no image is exported.
<code>keep_tiff</code>	Keep the standalone tiff file next to the workbook when a plot is embedded (default FALSE, i.e. the tiff is deleted after being inserted).
<code>dpi</code>	Resolution of the embedded plot image.
<code>filename</code>	File name of the workbook; NULL (default) means <code><file_name>_out.xlsx</code> . A missing <code>.xlsx</code> extension is added automatically.

Details

The tiff is written through `lifeTable_plot`'s device handling, which writes bitmaps with 'ragg' so that Chinese labels fall back to the system CJK font per glyph while digits and symbols stay in Times New Roman. The figure therefore looks the same in every R session, no matter what 'showtext' settings are left over in the session. If the reproduction-related parameters were skipped (`fecundity = FALSE` in `lifeTable_calculate_all`), the corresponding values in the Summary sheet are NA and the sheets "Female fecundity (F_xj)" and "Age-specific fecundity (m_x)" are omitted. If bootstrap results are attached (`results$boot <- lifeTable_bootstrap(lt)`; done automatically by `lifeTable_analyze` and `lifeTable_calculate` when `bootstrap = TRUE`), an extra worksheet "Bootstrap (SE & CI)" with the parameter table (original estimate, bootstrap mean, standard error, percentile confidence interval) is written.

Value

The path of the exported xlsx file (invisibly).

See Also

`lifeTable_calculate`, `lifeTable_plot`

Examples

```
## writes an xlsx workbook plus a tiff; that takes a few seconds, so it
## is not run by default
f <- system.file("extdata", "lifetable_example.csv", package = "insectecol")
lt <- lifeTable_read(f)
results <- lifeTable_calculate_all(lt)
lifeTable_export(lt, results, tempdir())
```

lifeTable_plot	<i>Age-Stage Survival Rate Curves</i>
----------------	---------------------------------------

Description

Draws the age-stage survival rate $s(x,j)$ of every developmental stage (including the female and male adults) against age in days, in the style of the classical 'TWOSEX-MSChart' plots.

Usage

```
lifeTable_plot(
  lt,
  sxj = NULL,
  title = NULL,
  x_title = "Age(days)",
  y_title = expression("Age-Stage Survival Rate (" * italic(S)[italic(xj)] * ")"),
  legend_labels = NULL,
  dpi = 300
)
```

Arguments

lt	A life_table object returned by lifeTable_read .
sxj	Optional; the result of calc_sxj . Supplying it avoids recomputing the age-stage survival rates.
title	Character; plot title. Defaults to the name of the csv file.
x_title	Character; x axis title (default "Age(days)").
y_title	Character or plotmath expression; y axis title (default expression("Age-Stage Survival Rate ("*italic(S)[italic(xj)]*")"), i.e. S with an italic xj subscript).
legend_labels	Character vector; legend labels, one per stage (immature stages plus Female and Male), e.g. c("Egg", "1st instar", "Pupa", "Female", "Male"). NULL (default) uses the stage names of the data.
dpi	Numeric; resolution (default 300). Only influences the scaling of the graphical elements (title, axis labels, legend), so that the plot looks identical at 300 and 600 dpi.

Details

To keep the figure clean, each stage is drawn only over the age window in which it actually occurs (extended by two days on both sides). Immature stages are drawn as coloured dots connected by a thin line; females are grey and males black, both with diamond points. By default all text of the figure is in English; title, axis titles and legend labels can be customised.

Text sizes are true typographic points and do not depend on the resolution, so the figure can be saved with [ggsave](#) as it is. In a label that mixes the two scripts, the Chinese characters are set in

the system CJK font (SimSun on Windows, Songti on macOS) while the digits, symbols and Latin words next to them stay in Times New Roman; a label that is entirely Chinese uses the CJK font for all of it.

Value

A ggplot object that can be customised further or saved with [ggsave](#).

See Also

[calc_sxj](#), [lifeTable_export](#)

Examples

```
f <- system.file("extdata", "lifetable_example.csv", package = "insectecol")
p <- lifeTable_plot(lifeTable_read(f))
```

lifeTable_read	<i>Read an Age-Stage, Two-Sex Life Table from a csv File</i>
----------------	--

Description

Reads a raw csv file containing the individual daily records of an age-stage, two-sex life table experiment and returns a `life_table` object, the central data structure that all other functions of the package work with.

Usage

```
lifeTable_read(path, check = TRUE)
```

Arguments

path	Character string; path to the csv file.
check	Logical; if TRUE (the default) the data are validated by lifeTable_check immediately after reading. Set to FALSE to skip validation.

Details

The csv file contains one row per individual. If the sex column is located at column `n`, the layout must be:

- column 1: individual ID (header ID),
- columns 2 to `n - 2`: number of days spent in each developmental stage (egg, larva, ..., pupa),
- column `n - 1`: adult survival days,
- column `n`: sex, coded F (female), M (male) or N (died before the adult stage),
- columns `n + 1` onward: daily oviposition of the females (one column per day; cells after the death of a female stay empty).

The first line of the file must contain the stage names as column headers, including the ID column (ID) and the sex column (gender). The sex column is located automatically by scanning the first data row for the markers F/M/N. The file encoding is detected automatically, so UTF-8 and GBK files are both supported.

Value

A `life_table` object; a named list with components

<code>data</code>	data frame with the raw csv content
<code>file_name</code>	file name without extension
<code>n</code>	column index of the sex column
<code>n_1</code>	$n + 1$, the first column of the oviposition data
<code>n_2</code>	$n - 2$, the column index of the pupal stage
<code>header</code>	the first row of the file (stage names), as character
<code>encoding</code>	the detected file encoding
<code>path</code>	the full path of the csv file

References

Chi, H. and Liu, H. (1985) Two new methods for study of insect population ecology. *Bull. Inst. Zool. Acad. Sin* 24(2), 225-240.

Chi, H. (1988) Life-table analysis incorporating both sexes and variable development rates among individuals. *Environmental Entomology* 17(1), 26-34.

See Also

[lifeTable_check](#) for data validation, [lifeTable_calculate](#) for the complete analysis workflow.

Examples

```
f <- system.file("extdata", "lifetable_example.csv", package = "insectecol")
lt <- lifeTable_read(f)
head(lt$data[, 1:5])
```

plot.emergence

Plot an Emergence-Period Projection

Description

Draws the cumulative development curve of the survey (stage shares accumulated from the most developed stage downwards) against the projected eclosion dates, marks the quantile crossings (16 projection exists, arrows from each eclosion date to the corresponding hatch date. The quantile dates are listed in a legend box on the right-hand side of the figure (one row per quantile, with a true arrow glyph instead of the character combination "->").

Usage

```
## S3 method for class 'emergence'
plot(
  x,
  show_hatch = TRUE,
  title = NULL,
  sub = NULL,
  family = "serif",
  cex = 2,
  lwd = 2,
  legend_right = TRUE,
  xlab = "Projected eclosion date",
  ylab = "Cumulative development (%)",
  ...
)
```

Arguments

<code>x</code>	A "emergence" object returned by emergence_calc or emergence_analyze .
<code>show_hatch</code>	Logical (default TRUE); whether to draw the hatch arrows when a hatch projection exists.
<code>title</code>	Plot title; NULL (default) uses the automatic caption (survey date and sample size). Use "" to drop the title.
<code>sub</code>	Plot footnote; NULL (default) shows the hatch parameters when <code>show_hatch</code> is active (drawn below the x-axis title). Use "" to drop it.
<code>family</code>	Text font family. Default "serif" — a portable alias that maps to Times New Roman on 'Windows' and to the system serif font elsewhere; Chinese characters are rendered through the device's font fallback (SimSun on Chinese 'Windows'). Set to "" for the device default.
<code>cex</code>	Overall text-size multiplier. Default 2: the exported figure is meant to be placed at half the text width of a manuscript, where the labels, ticks and title then appear at about the size of the body text (12 pt). Set to 1 for full-screen viewing; the margins scale with <code>cex</code> automatically.
<code>lwd</code>	Overall line-width multiplier. Default 2.
<code>legend_right</code>	Logical (default TRUE); whether the quantile legend is placed outside the panel on the right-hand side (TRUE) or in the top-left corner of the panel (FALSE).
<code>xlab, ylab</code>	Axis labels.
<code>...</code>	Further graphical parameters passed to <code>plot</code> .

Examples

```
f <- system.file("extdata", "emergence_example.csv",
  package = "insectecol")
fit <- emergence_calc(emergence_read(f), survey_date = "2026-03-20",
  pre_ovip = 3, egg_days = 10)
plot(fit)
```

```
## the projection is fully customisable (title, axis labels, font
## family); Chinese labels are rendered through the device's font
## fallback (SimSun on Chinese 'Windows'):
plot(fit, title = "Stage-grading projection",
     xlab = "Projected eclosion date",
     ylab = "Cumulative development (%)")
```

predict.emergence	<i>Quantiles of an Emergence-Period Projection</i>
-------------------	--

Description

Returns the projected eclosion (or hatch) dates of arbitrary emergence quantiles.

Usage

```
## S3 method for class 'emergence'
predict(object, p, event = c("eclosion", "hatch"), ...)
```

Arguments

object	A "emergence" object returned by emergence_calc or emergence_analyze .
p	Optional numeric vector of quantiles (0-1). The default (missing p) returns the stored predictions of the original call.
event	"eclosion" (default) or "hatch": which event to date.
...	Further arguments (unused).

Value

A data.frame with columns p, days (fractional days after the survey date) and date.

Examples

```
d <- data.frame(stage = c("Pupa 2", "Pupa 1", "Prepupa"),
               count = c(3, 4, 3), days = c(12, 14, 16))
fit <- emergence_calc(d, survey_date = "2026-03-20",
                    pre_ovip = 3, egg_days = 10)
predict(fit) # the stored 16/50/84 predictions
predict(fit, c(0.25, 0.75)) # arbitrary eclosion quantiles
predict(fit, c(0.25, 0.75), event = "hatch")
```

print.emergence	<i>Print an Emergence-Period Projection</i>
-----------------	---

Description

Prints the projected eclosion dates (and hatch dates, when requested) of the emergence quantiles.

Usage

```
## S3 method for class 'emergence'
print(x, ...)
```

Arguments

x	A "emergence" object returned by emergence_calc or emergence_analyze .
...	Further arguments (unused).

summary.emergence	<i>Summarise an Emergence-Period Projection</i>
-------------------	---

Description

Prints the full cumulative-development table behind the projection and the quantile predictions.

Usage

```
## S3 method for class 'emergence'
summary(object, ...)
```

Arguments

object	A "emergence" object returned by emergence_calc or emergence_analyze .
...	Further arguments (unused).

Value

Invisibly, a list with table (the cumulative development) and predictions (the quantile dates).

Index

calc_ex, [4](#)
calc_exj, [3](#)
calc_r, [46](#)
calc_sxj, [4](#), [28](#), [55](#), [56](#)
check_path_type, [4](#), [27](#), [34](#), [41](#)

default_stage_names, [5](#), [49](#)

emergence_analyze, [6](#), [10–13](#), [58–60](#)
emergence_calc, [6–8](#), [9](#), [11–13](#), [58–60](#)
emergence_export, [6](#), [8](#), [10](#), [11](#)
emergence_export_plot, [8](#), [12](#)
emergence_read, [6–8](#), [10](#), [13](#)

gdd_analyze, [6](#), [8](#), [14](#), [23](#), [24](#)
gdd_calc, [14–17](#), [18](#), [20](#), [27](#)
gdd_check, [14](#), [15](#), [17](#), [20](#), [27](#)
gdd_compare, [17](#), [21](#)
gdd_daily, [17](#), [22](#)
gdd_export, [14](#), [17](#), [22](#)
gdd_export_plot, [17](#), [23](#)
gdd_plot, [14](#), [16](#), [17](#), [23](#), [24](#), [24](#)
gdd_predict, [17](#), [26](#)
gdd_read, [14](#), [15](#), [17](#), [20](#), [27](#)
get_stage_names, [5](#), [28](#)
getwd, [8](#), [16](#), [30](#), [43](#)
ggsave, [55](#), [56](#)
grDevices, [23](#)

lc50_analyze, [6](#), [14](#), [29](#)
lc50_calculate, [29–31](#), [32](#), [33](#), [34](#), [38](#), [41](#)
lc50_export, [29](#), [31](#), [32](#), [33](#), [34](#), [36](#), [40](#)
lc50_export_auto, [34](#), [37](#)
lc50_export_plot, [31](#), [35](#), [36](#), [37](#), [40](#)
lc50_export_plot_auto, [34](#), [36](#)
lc50_improved, [32](#)
lc50_plot, [29–33](#), [35–37](#), [37](#)
lc50_probit, [32](#)
lc50_read, [5](#), [30–32](#), [40](#)
lc50_traditional, [32](#)

lifeTable_analyze, [14](#), [41](#), [54](#)
lifeTable_boot_test, [46](#), [47](#)
lifeTable_bootstrap, [43](#), [45](#), [47](#), [48](#), [51](#)
lifeTable_build, [42](#), [43](#), [45](#), [47](#), [48](#)
lifeTable_calculate, [50](#), [52](#), [54](#), [57](#)
lifeTable_calculate_all, [43](#), [46](#), [49](#), [51](#),
[51](#), [54](#)
lifeTable_check, [49](#), [52](#), [56](#), [57](#)
lifeTable_export, [41](#), [43](#), [46](#), [51](#), [53](#), [56](#)
lifeTable_plot, [28](#), [43](#), [51](#), [54](#), [55](#)
lifeTable_read, [3](#), [28](#), [42](#), [45](#), [47](#), [48](#), [51–55](#),
[56](#)

nls, [19](#)
nls.control, [18](#)

plot.emergence, [8](#), [12](#), [57](#)
png, [12](#)
predict.emergence, [59](#)
print.emergence, [60](#)

summary.emergence, [60](#)