Package: biocharkit (via r-universe)

July 21, 2026

Title Biochar Characterisation and Adsorption Data Analysis

Version 0.3.0

Description A toolkit for analysing biochar characterisation and batch adsorption experiments. Provides functions to parse structured sample identifiers encoding pyrolysis conditions, read raw FTIR and XRD instrument output, compute adsorption capacity and removal efficiency, fit adsorption isotherms following Langmuir (1918) <doi:10.1021/ja02242a004> and Sips (1948) <doi:10.1063/1.1746922> among other models, fit adsorption kinetics following Ho and McKay (1999) <doi:10.1016/S0032-9592(98)00112-5> and Chien and Clayton (1980) <doi:10.2136/sssaj1980.03615995004400020013x> among other models, fit batches of samples at once, compute van't Hoff thermodynamic parameters, baseline-correct and pick peaks in FTIR spectra, deconvolve XRD patterns into a crystallinity index, compute BET surface area following Brunauer, Emmett, and Teller (1938) <doi:10.1021/ja01269a023>, compute proximate and ultimate analysis summaries including directly from a thermogravimetric analysis (TGA) curve, compute a smoothed derivative thermogravimetric (DTG) curve and pick its decomposition peaks, fit non-isothermal decomposition kinetics from multi-heating-rate TGA data following Kissinger (1957) <doi:10.1021/ac60131a045>, build correlation matrices with p-values, and produce publication-style base-graphics figures including 600 dpi TIFF export. Built on base R ('stats', 'graphics', 'grDevices') so it has no dependency on packages that require external CRAN network access to install.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Depends R (>= 4.0)

Imports stats, utils, graphics, grDevices

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

NeedsCompilation no

Author Sukamal Sarkar [aut, cre]

Maintainer Sukamal Sarkar <sukamal.sarkar@gm.rkmvu.ac.in>

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

Date/Publication 2026-07-21 11:10:13 UTC

RemoteUrl https://github.com/cran/biocharkit

RemoteRef HEAD

RemoteSha 2012fd7ee53a20ed094dc4f83b8bf1f9f0442a07

Contents

`assign_dtg_peaks` *Assign DTG peak temperatures to decomposition stages*

Description

Matches a vector of observed DTG peak temperatures (e.g. from `find_dtg_peaks()`) against a stage-range reference table. Where a temperature falls inside more than one range (lignin decomposition overlaps the others, since it is broad and underlying), the first matching row of `reference` wins, so the default reference lists the narrower hemicellulose/cellulose ranges before the broad lignin range.

Usage

```
assign_dtg_peaks(
  peak_temperatures_C,
  reference = tga_decomposition_reference()
)
```

Arguments

`peak_temperatures_C`	Numeric vector of observed DTG peak temperatures (degrees C).
`reference`	A reference table as returned by `tga_decomposition_reference()`; defaults to that table.

Value

A data frame with columns `peak_temperature_C`, `stage`, `description`. Peaks matching no range get `NA` stage/description.

Examples

```
assign_dtg_peaks(c(80, 300, 350))
```

assign_ftir_peaks *Assign FTIR peak wavenumbers to functional groups*

Description

Matches a vector of observed peak positions (e.g. peak-picked maxima) against a band-assignment reference table.

Usage

```
assign_ftir_peaks(peaks, reference = ftir_band_reference())
```

Arguments

peaks	Numeric vector of observed peak wavenumbers (cm^-1).
reference	A reference table as returned by ftir_band_reference(); defaults to that table.

Value

A data.frame with columns peak_cm1, group, assignment. Peaks matching no range in the reference get NA group/assignment.

Examples

```
assign_ftir_peaks(c(3420, 2920, 1710, 1600, 1030))
```

baseline_correct *Baseline-correct an FTIR spectrum*

Description

Removes a simple estimated baseline from a spectrum before further analysis (e.g. functional_group_density() or find_ftir_peaks()). Two methods are provided: "linear" subtracts the straight line connecting the spectrum's first and last points (a common quick correction for a roughly flat, tilted baseline); "rolling_min" subtracts a rolling-minimum envelope, which follows a more uneven baseline at the cost of also shaving the base of broad peaks.

Usage

```
baseline_correct(spectrum, method = c("linear", "rolling_min"), window = 25)
```

Arguments

spectrum	A data frame with columns wavenumber_cm1, intensity (as returned by read_ftir_txt()).
method	"linear" (default) or "rolling_min".
window	For method = "rolling_min", the half-width (in number of points) of the rolling window. Default 25.

Value

A data frame in the same shape as spectrum, with intensity replaced by the baseline-corrected values (never negative; clipped at 0).

Examples

```
spec <- data.frame(wavenumber_cm1 = seq(4000, 400, by = -4),
                   intensity = 0.1 + runif(901, 0, 0.05))
corrected <- baseline_correct(spec)
```

---

bet_surface_area *BET surface area from multi-point gas adsorption data*

Description

Fits the linearised multi-point BET (Brunauer-Emmett-Teller) equation 1 / (Q * (P0/P - 1)) = 1/(Qm*C) + ((C-1)/(Qm*C)) * (P/P0) by ordinary least squares, then converts the BET monolayer capacity Qm to a specific surface area using the adsorbate's molecular cross-sectional area.

Usage

```
bet_surface_area(
  P_P0,
  Q,
  cross_sectional_area_nm2 = 0.162,
  molar_volume_cm3mol = 22414
)
```

Arguments

P_P0	Numeric vector of relative pressures (P/P0, dimensionless, between 0 and 1).
Q	Numeric vector of quantity adsorbed at each P_P0, in cm^3/g at STP (the standard gas-volumetric convention).
cross_sectional_area_nm2	Molecular cross-sectional area of the adsorbate, in nm^2. Default 0.162 (N2 at 77 K, the standard BET probe gas).
molar_volume_cm3mol	Molar volume of an ideal gas at STP, in cm^3/mol. Default 22414.

Details

Conventionally only points in the relative pressure range `P/P0` roughly 0.05-0.35 are used (where the BET model is valid); this function does not enforce that range for you, so filter `P_P0` and `Q` before calling if your data extends beyond it.

Value

A list with elements:

`Qm_cm3g`	BET monolayer capacity (cm^3/g STP)
`C`	BET constant (related to adsorbate-adsorbent interaction energy)
`SBET_m2g`	Specific surface area (m^2/g)
`R2`	Coefficient of determination of the linear BET fit
`model`	The underlying `lm` fit object

Examples

```
P_P0 <- c(0.05, 0.10, 0.15, 0.20, 0.25, 0.30)
Q <- c(12.1, 15.8, 18.9, 21.6, 24.1, 26.8)
bet_surface_area(P_P0, Q)
```

`correlation_matrix` *Pairwise correlation matrix with p-values*

Description

Computes a pairwise correlation matrix (Spearman by default) across all numeric columns of a data frame, along with a matching matrix of p-values from `stats::cor.test()`. Implemented with base R only (no Hmisc dependency).

Usage

```
correlation_matrix(
  df,
  method = c("spearman", "pearson", "kendall"),
  use = "pairwise.complete.obs"
)
```

Arguments

`df`	A `data.frame` (non-numeric columns are dropped with a message).
`method`	Correlation method passed to `stats::cor.test()`: `"spearman"` (default), `"pearson"`, or `"kendall"`.
`use`	Passed to `stats::cor.test()` for handling of missing values via pairwise complete observations (always used here).

Value

A list with elements `estimate` (correlation coefficient matrix) and `p_value` (matching p-value matrix), both with row/column names equal to the numeric column names of `df`.

Examples

```
df <- data.frame(temp = c(300, 450, 600, 300, 450, 600),
                 EC = c(1.1, 1.8, 2.6, 1.0, 1.9, 2.5),
                 pH = c(7.1, 8.0, 9.2, 7.0, 8.1, 9.0))
correlation_matrix(df)
```

`find_dtg_peaks` *Automatically pick decomposition peaks from a DTG curve*

Description

Finds local maxima in `dtg$dtg` (weight-loss-rate peaks): a point is a candidate peak if it is the highest point within `window` indices on either side, following the same local-maxima-plus-prominence approach as `find_ftir_peaks()`. Peaks with non-positive `dtg` (baseline drift or mass gain, e.g. oxidation) are dropped.

Usage

```
find_dtg_peaks(dtg, window = 5, min_prominence = 0)
```

Arguments

`dtg`	A data frame with columns `temperature_C`, `dtg` (as returned by `tga_dtg()`).
`window`	Half-width (in points) used to test local maxima. Default `5`.
`min_prominence`	Minimum prominence (in %/degC) for a local maximum to be kept. Default `0` (keep all local maxima).

Value

A data frame with columns `peak_temperature_C`, `dtg_max`, `prominence`, sorted by decreasing `dtg_max`.

Examples

```
set.seed(1)
T <- seq(25, 800, by = 5)
W <- 100 - 5 * (T > 110) - 40 / (1 + exp(-(T - 340) / 20)) + rnorm(length(T), 0, 0.15)
tga <- data.frame(temperature_C = T, weight_pct = pmax(W, 20))
find_dtg_peaks(tga_dtg(tga))
```

find_ftir_peaks *Automatically pick peaks from a full FTIR spectrum*

Description

Finds local maxima in `spectrum$intensity`: a point is a candidate peak if it is the highest point within `window` indices on either side. Weak shoulders are then filtered out by a minimum prominence requirement (height above the higher of the two valleys flanking the peak). Intended as a quick first pass, not a substitute for visual inspection of the spectrum.

Usage

```
find_ftir_peaks(spectrum, window = 5, min_prominence = 0)
```

Arguments

`spectrum`	A data frame with columns `wavenumber_cm1`, `intensity`.
`window`	Half-width (in points) used to test local maxima. Default `5`.
`min_prominence`	Minimum prominence (in intensity units) for a local maximum to be kept. Default `0` (keep all local maxima).

Value

A data frame with columns `peak_cm1`, `intensity`, `prominence`, sorted by decreasing intensity.

Examples

```
wn <- seq(4000, 400, by = -2)
spec <- data.frame(
  wavenumber_cm1 = wn,
  intensity = exp(-((wn - 1710)^2) / (2 * 20^2)) +
              exp(-((wn - 2920)^2) / (2 * 30^2))
)
find_ftir_peaks(spec)
```

fit_confint *Confidence intervals for a fitted isotherm, kinetics, or van't Hoff model*

Description

Wraps `stats::confint()` around the underlying `nls`/`lm` fit object produced by this package's `fit_*()` functions, returning a tidy data frame rather than a bare matrix.

Usage

```
fit_confint(fit, level = 0.95)
```

Arguments

fit	A fit object returned by any of fit_langmuir(), fit_freundlich(), fit_temkin(), fit_dr(), fit_sips(), fit_pfo(), fit_pso(), fit_elovich(), fit_intraparticle(), or fit_vant_hoff().
level	Confidence level. Default 0.95.

Value

A data frame with columns parameter, estimate, lower, upper.

Examples

```
Ce <- c(2, 5, 10, 20, 35, 50, 70)
qe <- c(0.8, 1.6, 2.3, 3.0, 3.6, 4.0, 4.4)
fit <- fit_langmuir(Ce, qe)
fit_confint(fit)
```

---

fit_dr *Fit the Dubinin-Radushkevich (D-R) adsorption isotherm*

Description

Fits qe = qs * exp(-Kad * epsilon^2), where epsilon = R * temperature_K * log(1 + 1/Ce) is the Polanyi potential. Also returns the mean free energy of adsorption E = 1 / sqrt(2 * Kad) (kJ/mol), conventionally used to distinguish physisorption (E < 8 kJ/mol) from chemisorption (8-16 kJ/mol).

Usage

```
fit_dr(Ce, qe, temperature_K = 298.15, start = NULL)
```

Arguments

Ce	Numeric vector of equilibrium concentrations (mg/L).
qe	Numeric vector of equilibrium adsorption capacities (mg/g).
temperature_K	Absolute temperature at which the isotherm was measured, in Kelvin. Default 298.15 (25 degC).
start	Optional named list list(qs=, Kad=) of starting values.

Value

A list with elements qs (mg/g), Kad (mol^2/kJ^2), E (kJ/mol), R2, model.

Examples

```
Ce <- c(2, 5, 10, 20, 35, 50, 70)
qe <- c(0.8, 1.6, 2.3, 3.0, 3.6, 4.0, 4.4)
fit_dr(Ce, qe)
```

fit_elovich *Fit the Elovich kinetics model*

Description

Fits `qt = (1/beta) * log(1 + alpha * beta * t)`, where `alpha` is the initial sorption rate (mg/g/min) and `beta` relates to the extent of surface coverage / activation energy for chemisorption (g/mg).

Usage

```
fit_elovich(t, qt, start = NULL)
```

Arguments

`t`	Numeric vector of contact times.
`qt`	Numeric vector of adsorption capacity at time `t` (mg/g).
`start`	Optional named list `list(alpha=, beta=)` of starting values.

Value

A list with elements `alpha`, `beta`, `R2`, `model`.

Examples

```
t <- c(5, 15, 30, 60, 120, 240)
qt <- c(1.2, 2.1, 2.9, 3.6, 4.0, 4.2)
fit_elovich(t, qt)
```

fit_freundlich *Fit the Freundlich adsorption isotherm*

Description

Fits `qe = Kf * Ce^(1/n)` by nonlinear least squares, with a fallback to the linearised form `log(qe) = log(Kf) + (1/n) * log(Ce)` fit by OLS if the nonlinear fit fails to converge.

Usage

```
fit_freundlich(Ce, qe, start = NULL)
```

Arguments

`Ce`	Numeric vector of equilibrium concentrations (mg/L).
`qe`	Numeric vector of equilibrium adsorption capacities (mg/g).
`start`	Optional named list `list(Kf=, n=)` of starting values.

Value

A list with elements `Kf`, `n`, `R2`, `method`, `model` (see `fit_langmuir()` for details of the structure).

Examples

```
Ce <- c(2, 5, 10, 20, 35, 50)
qe <- c(0.8, 1.6, 2.6, 3.6, 4.2, 4.5)
fit_freundlich(Ce, qe)
```

---

fit_intraparticle *Fit the intraparticle diffusion (Weber-Morris) kinetics model*

Description

Fits the linear form `qt = kid * sqrt(t) + C`, where `kid` is the intraparticle diffusion rate constant (mg/(g min^0.5)) and `C` is related to boundary-layer thickness (`C = 0` would indicate intraparticle diffusion is the sole rate-limiting step).

Usage

```
fit_intraparticle(t, qt)
```

Arguments

`t`	Numeric vector of contact times.
`qt`	Numeric vector of adsorption capacity at time `t` (mg/g).

Value

A list with elements `kid`, `C`, `R2`, `model` (an `lm` fit on `qt ~ sqrt(t)`).

Examples

```
t <- c(5, 15, 30, 60, 120, 240)
qt <- c(1.2, 2.1, 2.9, 3.6, 4.0, 4.2)
fit_intraparticle(t, qt)
```

`fit_isotherm_batch` *Fit an adsorption isotherm separately for each group in a data frame*

Description

Splits `df` by `group_col` (e.g. a sample ID column) and fits the chosen isotherm model independently within each group, returning one row of parameters per group. Groups that fail to fit (e.g. too few points, non-convergence) are skipped with a warning rather than aborting the whole batch.

Usage

```
fit_isotherm_batch(
  df,
  group_col,
  ce_col,
  qe_col,
  model = c("langmuir", "freundlich", "temkin", "dr", "sips"),
  temperature_K = 298.15
)
```

Arguments

`df`	A data frame containing at least `group_col`, `ce_col`, and `qe_col`.
`group_col`	Name of the grouping column (e.g. sample ID).
`ce_col, qe_col`	Column names for equilibrium concentration (mg/L) and adsorption capacity (mg/g).
`model`	One of `"langmuir"`, `"freundlich"`, `"temkin"`, `"dr"`, `"sips"`.
`temperature_K`	Only used when `model = "dr"`; see `fit_dr()`.

Value

A data frame with one row per successfully fitted group, columns `group_col` plus the model's parameter columns and `R2` (column names match what the corresponding single-sample `fit_*()` function returns).

Examples

```
df <- data.frame(
  id = rep(c("A", "B"), each = 6),
  Ce = rep(c(2, 5, 10, 20, 35, 50), 2),
  qe = c(0.8, 1.6, 2.3, 3.0, 3.6, 4.0,   # sample A
         0.5, 1.0, 1.6, 2.2, 2.7, 3.0)   # sample B
)
fit_isotherm_batch(df, "id", "Ce", "qe", model = "langmuir")
```

`fit_kinetics_batch` *Fit adsorption kinetics separately for each group in a data frame*

Description

Splits `df` by `group_col` (e.g. a sample ID column) and fits the chosen kinetics model independently within each group, returning one row of parameters per group. Groups that fail to fit are skipped with a warning rather than aborting the whole batch.

Usage

```
fit_kinetics_batch(
  df,
  group_col,
  t_col,
  qt_col,
  model = c("pfo", "pso", "elovich", "intraparticle")
)
```

Arguments

`df`	A data frame containing at least `group_col`, `t_col`, and `qt_col`.
`group_col`	Name of the grouping column (e.g. sample ID).
`t_col`, `qt_col`	Column names for contact time and adsorption capacity at time t (mg/g).
`model`	One of `"pfo"`, `"pso"`, `"elovich"`, `"intraparticle"`.

Value

A data frame with one row per successfully fitted group.

Examples

```
df <- data.frame(
  id = rep(c("A", "B"), each = 6),
  t = rep(c(5, 15, 30, 60, 120, 240), 2),
  qt = c(1.2, 2.1, 2.9, 3.6, 4.0, 4.2,
         0.8, 1.5, 2.0, 2.5, 2.8, 3.0)
)
fit_kinetics_batch(df, "id", "t", "qt", model = "pso")
```

`fit_langmuir` *Fit the Langmuir adsorption isotherm*

Description

Fits `qe = (Qmax * KL * Ce) / (1 + KL * Ce)` by nonlinear least squares. If nonlinear fitting fails to converge (common with few points or noisy data), falls back to the linearised form `Ce/qe = Ce/Qmax + 1/(KL*Qmax)` fit by ordinary least squares, and flags the result accordingly.

Usage

```
fit_langmuir(Ce, qe, start = NULL)
```

Arguments

`Ce`	Numeric vector of equilibrium concentrations (mg/L).
`qe`	Numeric vector of equilibrium adsorption capacities (mg/g).
`start`	Optional named list of starting values `list(Qmax=, KL=)` for the nonlinear fit. If `NULL`, sensible defaults are derived from the data.

Value

A list with elements:

`Qmax`	Maximum adsorption capacity (mg/g)
`KL`	Langmuir affinity constant (L/mg)
`R2`	Coefficient of determination
`method`	`"nonlinear"` or `"linearised"`
`model`	The underlying `nls` or `lm` fit object

Examples

```
Ce <- c(2, 5, 10, 20, 35, 50)
qe <- c(0.8, 1.6, 2.6, 3.6, 4.2, 4.5)
fit_langmuir(Ce, qe)
```

`fit_pfo` *Fit pseudo-first-order adsorption kinetics*

Description

Fits `qt = qe * (1 - exp(-k1 * t))` by nonlinear least squares.

Usage

```
fit_pfo(t, qt, start = NULL)
```

Arguments

`t`	Numeric vector of contact times.
`qt`	Numeric vector of adsorption capacity at time `t` (mg/g).
`start`	Optional named list `list(qe=, k1=)` of starting values.

Value

A list with elements `qe`, `k1`, `R2`, `model` (an `nls` fit).

Examples

```
t <- c(5, 15, 30, 60, 120, 240)
qt <- c(1.2, 2.1, 2.9, 3.6, 4.0, 4.2)
fit_pfo(t, qt)
```

`fit_pso` *Fits* `qt = (k2 * qe^2 * t) / (1 + k2 * qe * t)` *by nonlinear least squares.*

Description

Fits `qt = (k2 * qe^2 * t) / (1 + k2 * qe * t)` by nonlinear least squares.

Usage

```
fit_pso(t, qt, start = NULL)
```

Arguments

`t`	Numeric vector of contact times.
`qt`	Numeric vector of adsorption capacity at time `t` (mg/g).
`start`	Optional named list `list(qe=, k2=)` of starting values.

Value

A list with elements qe, k2, R2, model (an nls fit).

Examples

```
t <- c(5, 15, 30, 60, 120, 240)
qt <- c(1.2, 2.1, 2.9, 3.6, 4.0, 4.2)
fit_pso(t, qt)
```

---

fit_sips *Fit the Sips (Langmuir-Freundlich) adsorption isotherm*

Description

Fits qe = (Qmax * Ks * Ce^(1/n)) / (1 + Ks * Ce^(1/n)), a three- parameter hybrid that reduces to Langmuir as n -> 1 and to Freundlich at low Ce.

Usage

```
fit_sips(Ce, qe, start = NULL)
```

Arguments

Ce	Numeric vector of equilibrium concentrations (mg/L).
qe	Numeric vector of equilibrium adsorption capacities (mg/g).
start	Optional named list list(Qmax=, Ks=, n=) of starting values.

Value

A list with elements Qmax (mg/g), Ks, n, R2, model.

Examples

```
Ce <- c(2, 5, 10, 20, 35, 50, 70)
qe <- c(0.8, 1.6, 2.3, 3.0, 3.6, 4.0, 4.4)
fit_sips(Ce, qe)
```

fit_temkin *Fit the Temkin adsorption isotherm*

Description

Fits `qe = B * log(KT * Ce)`, where `B = RT/b` is the Temkin constant related to the heat of sorption and `KT` is the equilibrium binding constant (L/mg).

Usage

```
fit_temkin(Ce, qe, start = NULL)
```

Arguments

`Ce`	Numeric vector of equilibrium concentrations (mg/L).
`qe`	Numeric vector of equilibrium adsorption capacities (mg/g).
`start`	Optional named list `list(KT=, B=)` of starting values.

Value

A list with elements `KT`, `B`, `R2`, `model` (an `nls` fit).

Examples

```
Ce <- c(2, 5, 10, 20, 35, 50, 70)
qe <- c(0.8, 1.6, 2.3, 3.0, 3.6, 4.0, 4.4)
fit_temkin(Ce, qe)
```

fit_vant_hoff *Van't Hoff thermodynamic analysis of adsorption*

Description

Computes standard enthalpy (`deltaH`) and entropy (`deltaS`) of adsorption from the linear van't Hoff relationship `log(Kc) = deltaS/R - deltaH/(R * temperature_K)`, fit by ordinary least squares across temperatures. Standard Gibbs free energy `deltaG = deltaH - temperature_K * deltaS` is then computed at each supplied temperature.

Usage

```
fit_vant_hoff(temperature_K, Kc)
```

Arguments

temperature_K	Numeric vector of absolute temperatures (Kelvin) at which adsorption was measured.
Kc	Numeric vector of dimensionless equilibrium distribution coefficients (e.g. qe / Ce at a fixed initial concentration) at each temperature in temperature_K.

Value

A list with elements:

deltaH_kJmol	Standard enthalpy of adsorption (kJ/mol)
deltaS_Jmolk	Standard entropy of adsorption (J/(mol K))
table	A data frame with one row per input temperature: temperature_K, Kc, deltaG_kJmol
R2	Coefficient of determination of the linear van't Hoff fit
model	The underlying lm fit object

Examples

```
temperature_K <- c(298, 308, 318, 328)
Kc <- c(1.8, 2.3, 2.9, 3.4)
fit_vant_hoff(temperature_K, Kc)
```

---

ftir_band_reference *Reference table of common FTIR functional-group band assignments*

Description

Returns a data.frame of commonly cited FTIR wavenumber ranges and their associated functional-group assignments for biochar/carbon materials, based on standard published band assignments. Intended as a starting reference for assign_ftir_peaks(); users should adjust ranges to match their own literature basis where needed.

Usage

```
ftir_band_reference()
```

Value

A data.frame with columns group, range_low_cm1, range_high_cm1, assignment.

functional_group_density

Functional group density from a full FTIR spectrum

Description

Computes a simple proxy for functional-group density: the integrated (trapezoidal) absorbance/intensity area within each wavenumber region of a reference table, over a full spectrum (not just picked peaks).

Usage

```
functional_group_density(spectrum, reference = ftir_band_reference())
```

Arguments

spectrum	A data.frame with columns wavenumber_cm1, intensity (as returned by read_ftir_txt()).
reference	A reference table as returned by ftir_band_reference(); defaults to that table.

Value

A data.frame with columns group, range_low_cm1, range_high_cm1, area (trapezoidal integral of intensity over the region) and mean_intensity.

parse_sbc_id

Parse structured biochar sample identifiers

Description

Parses sample IDs of the form <PREFIX><TEMPERATURE>-<TIME> (e.g. "SBC450-30") into their pyrolysis condition components. This matches naming conventions used for factorial pyrolysis designs where a feedstock prefix is followed by pyrolysis temperature (degrees C) and residence time (minutes), separated by a hyphen.

Usage

```
parse_sbc_id(id, prefix_pattern = "[A-Za-z]+")
```

Arguments

id	Character vector of sample identifiers, e.g. c("SBC300-15", "SBC450-30", "SBC600-60").
prefix_pattern	Regex describing the non-numeric feedstock prefix. Defaults to one or more letters ("[A-Za-z]+").

Value

A `data.frame` with columns `id`, `feedstock`, `temperature_C`, `residence_time_min`. Rows that do not match the expected pattern have `NA` in the parsed columns and trigger a warning listing the offending IDs, rather than failing the whole batch.

Examples

```
parse_sbc_id(c("SBC300-15", "SBC450-30", "SBC600-60"))
```

`pct_mass_change` *Percentage mass loss between two mass measurements*

Description

Simple gravimetric helper: `100 * (initial_mass - final_mass) / initial_mass`. Useful for computing moisture % or volatile matter % from raw balance readings before passing them into `proximate_analysis()`.

Usage

```
pct_mass_change(initial_mass, final_mass)
```

Arguments

`initial_mass`, `final_mass`
: Numeric vectors of mass measurements (any consistent unit, e.g. g).

Value

Numeric vector of percentage mass loss.

Examples

```
pct_mass_change(initial_mass = 5.000, final_mass = 4.812)  # e.g. moisture %
```

plot_ftir_spectrum *Plot an FTIR spectrum*

Description

Line plot of an FTIR spectrum in conventional display orientation (wavenumber decreasing left to right).

Usage

```
plot_ftir_spectrum(spectrum, main = "FTIR spectrum", ...)
```

Arguments

spectrum	A data.frame with columns wavenumber_cm1, intensity (as returned by read_ftir_txt()).
main	Plot title.
...	Further arguments passed to graphics::plot().

Value

Invisibly, NULL. Called for its side effect of drawing a plot on the current graphics device.

plot_isotherm *Plot an adsorption isotherm fit*

Description

Base-graphics scatter of observed (Ce, qe) points with the fitted isotherm curve overlaid.

Usage

```
plot_isotherm(fit, Ce, qe, model = NULL, main = "Adsorption isotherm", ...)
```

Arguments

fit	A fit object from fit_langmuir(), fit_freundlich(), fit_temkin(), fit_dr(), or fit_sips().
Ce, qe	Original data used for the fit.
model	Which model fit represents: "langmuir", "freundlich", "temkin", "dr", or "sips". If NULL, inferred from the names of fit where unambiguous (Langmuir vs Freundlich only, for backwards compatibility) — for the other models, model must be supplied explicitly.
main	Plot title.
...	Further arguments passed to graphics::plot().

Value

Invisibly, NULL. Called for its side effect of drawing a plot on the current graphics device.

`plot_kinetics` *Plot a kinetics fit*

Description

Base-graphics scatter of observed (t, qt) points with the fitted kinetics curve overlaid.

Usage

```
plot_kinetics(
  fit,
  t,
  qt,
  model = c("pfo", "pso", "elovich", "intraparticle"),
  main = "Adsorption kinetics",
  ...
)
```

Arguments

`fit`	A fit object from `fit_pfo()`, `fit_pso()`, `fit_elovich()`, or `fit_intraparticle()`.
`t, qt`	Original data used for the fit.
`model`	Which model `fit` represents: `"pfo"`, `"pso"`, `"elovich"`, or `"intraparticle"`.
`main`	Plot title.
`...`	Further arguments passed to `graphics::plot()`.

Value

Invisibly, NULL. Called for its side effect of drawing a plot on the current graphics device.

`plot_tga` *Plot a TG/DTG curve*

Description

Base-graphics dual-axis plot: the thermogravimetric (TG) weight-loss curve on the left axis, with the derivative (DTG) curve overlaid as a dashed line on a right axis. If `dtg` is not supplied, it is computed internally via `tga_dtg()`.

Usage

```
plot_tga(
  tga,
  dtg = NULL,
  temperature_col = "temperature_C",
  weight_col = "weight_pct",
  main = "TGA/DTG curve",
  ...
)
```

Arguments

tga	A data frame with columns temperature_col, weight_col (as returned by read_tga_txt()).
dtg	Optional pre-computed DTG data frame from tga_dtg(); if NULL (default), computed internally from tga.
temperature_col, weight_col	Column names in tga. Defaults "temperature_C", "weight_pct".
main	Plot title.
...	Further arguments passed to the TG graphics::plot() call.

Value

Invisibly, NULL. Called for its side effect of drawing a plot on the current graphics device.

Examples

```
set.seed(1)
T <- seq(25, 800, by = 5)
W <- 100 - 5 * (T > 110) - 40 / (1 + exp(-(T - 340) / 20)) + rnorm(length(T), 0, 0.15)
tga <- data.frame(temperature_C = T, weight_pct = pmax(W, 20))
plot_tga(tga)
```

---

proximate_analysis *Proximate analysis summary (moisture, volatile matter, ash, fixed carbon)*

Description

Combines independently measured moisture %, volatile matter %, and ash % (e.g. each computed via pct_mass_change() from its own sub-sample, following standard gravimetric proximate analysis protocols) into a full proximate analysis table, computing fixed carbon by difference: FC% = 100 - moisture% - VM% - ash%.

Usage

```
proximate_analysis(moisture_pct, volatile_matter_pct, ash_pct)
```

Arguments

`moisture_pct`, `volatile_matter_pct`, `ash_pct`
: Numeric vectors (recycled to a common length) of each percentage, on an as-received mass basis.

Value

A data frame with columns `moisture_pct`, `VM_pct`, `ash_pct`, `fixed_carbon_pct`. A warning is issued (not an error) for any row where the components sum to more than 100%, since that indicates a likely measurement or transcription issue rather than a computable negative fixed-carbon value.

Examples

```
proximate_analysis(moisture_pct = 3.2, volatile_matter_pct = 28.5, ash_pct = 12.1)
```

`qe_batch` *Batch adsorption capacity (qe)*

Description

Computes the equilibrium adsorption capacity for a batch adsorption experiment: `qe = (C0 - Ce) * V / m`.

Usage

```
qe_batch(C0, Ce, V, m)
```

Arguments

`C0`	Initial solute concentration (mg/L).
`Ce`	Equilibrium solute concentration (mg/L).
`V`	Solution volume (L).
`m`	Adsorbent (biochar) mass (g).

Value

Numeric vector of adsorption capacities in mg/g. Recycled element-wise across `C0`, `Ce`, `V`, `m`.

Examples

```
qe_batch(C0 = 50, Ce = 12.5, V = 0.05, m = 0.1)
```

read_ftir_txt *Read a two-column FTIR spectrum export*

Description

Reads plain-text FTIR exports (e.g. from PerkinElmer ATR-FTIR software) consisting of two numeric columns (wavenumber and absorbance/transmittance), possibly preceded by header/metadata lines and using either comma, tab, or whitespace as a delimiter. Uses base `readLines()`/`strsplit()` rather than a heavier parser so that malformed or partially-exported files degrade gracefully (bad lines are skipped and reported) instead of aborting the whole read.

Usage

```
read_ftir_txt(
  path,
  skip_pattern = "^\\s*-?[0-9]",
  col_names = c("wavenumber_cm1", "intensity")
)
```

Arguments

`path`	Path to the text file.
`skip_pattern`	Regex used to identify non-data header lines to skip. Defaults to lines that don't start with an optional minus sign and a digit.
`col_names`	Column names to assign, default `c("wavenumber_cm1", "intensity")`.

Value

A `data.frame` with the two numeric columns, sorted by `wavenumber_cm1` descending (conventional FTIR display order), with a `attr(., "skipped_lines")` attribute recording line numbers that could not be parsed.

read_tga_txt *Read a raw thermogravimetric analysis (TGA) export*

Description

Reads plain-text TGA exports (e.g. from TA Instruments, Netzsch, or Shimadzu software) consisting of two or more numeric columns, typically temperature, time, and weight percent, possibly preceded by header/metadata lines and using comma, tab, or whitespace as a delimiter. Uses the same defensive `readLines()`/`strsplit()` approach as `read_ftir_txt()` so malformed or partially-exported lines are skipped and reported rather than aborting the whole read.

Usage

```
read_tga_txt(
  path,
  skip_pattern = "^\\s*-?[0-9]",
  col_names = c("temperature_C", "time_min", "weight_pct")
)
```

Arguments

path	Path to the text file.
skip_pattern	Regex used to identify non-data header lines to skip. Defaults to lines that don't start with an optional minus sign and a digit.
col_names	Column names to assign, in the order they appear in the file. Default `c("temperature_C", "time_min", "weight_pct")`. Must have at least 2 entries; a two-column file of just temperature and weight percent works by passing `col_names = c("temperature_C", "weight_pct")`.

Value

A `data.frame` with one column per entry in `col_names`, sorted by the first column (temperature) ascending, with a `attr(., "skipped_lines")` attribute recording line numbers that could not be parsed.

Examples

```
tmp <- tempfile(fileext = ".txt")
writeLines(c("Temp(C)\tTime(min)\tWeight(%)",
             "25\t0\t100.0", "50\t2\t99.8", "100\t6\t98.1"), tmp)
read_tga_txt(tmp)
unlink(tmp)
```

---

read_xrd_txt *Read a two-column XRD pattern export*

Description

Reads plain-text XRD exports (e.g. from Rigaku SmartLab, Cu K-alpha) consisting of two-theta and intensity columns. Same defensive `readLines()`/`strsplit()` approach as `read_ftir_txt()`.

Usage

```
read_xrd_txt(
  path,
  skip_pattern = "^\\s*-?[0-9]",
  col_names = c("two_theta", "intensity")
)
```

Arguments

path	Path to the text file.
skip_pattern	Regex used to identify non-data header lines to skip. Defaults to lines that don't start with an optional minus sign and a digit.
col_names	Column names to assign, default c("two_theta", "intensity").

Value

A data.frame with columns two_theta, intensity, sorted by two_theta ascending.

---

removal_efficiency *Percentage removal efficiency*

Description

Computes percentage removal: 100 * (C0 - Ce) / C0.

Usage

```
removal_efficiency(C0, Ce)
```

Arguments

C0	Initial solute concentration (mg/L).
Ce	Equilibrium solute concentration (mg/L).

Value

Numeric vector of removal efficiencies (%).

Examples

```
removal_efficiency(C0 = 50, Ce = 12.5)
```

save_tiff *Save a base-graphics plot as a 600 dpi TIFF*

Description

Wraps a plotting expression in a grDevices::tiff() device at publication resolution (600 dpi by default) with LZW compression, matching a common journal figure requirement. The font family defaults to "serif", which maps to Liberation Serif on most Linux systems (a metric-compatible substitute for Times New Roman).

Usage

```
save_tiff(expr, filename, width = 6, height = 5, dpi = 600, family = "serif")
```

Arguments

expr	An expression (typically a call to one of the plot_*() functions in this package, or any base-graphics plotting code) to evaluate with the device open. Pass as { ... } for multi-line code.
filename	Output file path, should end in .tif/.tiff.
width, height	Figure size in inches.
dpi	Resolution in dots per inch. Default 600.
family	Font family for the device. Default "serif".

Value

Invisibly, the filename.

Examples

```
Ce <- c(2, 5, 10, 20, 35, 50, 70)
qe <- c(0.8, 1.6, 2.3, 3.0, 3.6, 4.0, 4.4)
path <- tempfile(fileext = ".tif")
save_tiff(plot_isotherm(fit_langmuir(Ce, qe), Ce, qe), filename = path)
file.exists(path)
unlink(path)
```

`sbc_example` *Synthetic example biochar dataset*

Description

A small, randomly generated dataset in the shape of a 3x3 factorial pyrolysis design (temperature x residence time), used only to demonstrate and test package functions in examples and tests.

Usage

```
sbc_example
```

Format

A data frame with 9 rows and 7 columns:

id Synthetic sample identifier, e.g. `"SBC450-30"`

temperature_C Pyrolysis temperature (degrees C)

residence_time_min Pyrolysis residence time (minutes)

pH Synthetic pH value

EC_dS_m Synthetic electrical conductivity (dS/m)

Ce_mgL Synthetic equilibrium solute concentration (mg/L)

qe_mgg Synthetic adsorption capacity (mg/g)

Details

This is synthetic data, not experimental measurements. Values were generated with a fixed random seed to follow a plausible but entirely artificial trend, and must not be cited, plotted alongside real results, or otherwise treated as real biochar characterisation data. See `data-raw/generate_example_data.R` for the generating code.

Source

Randomly generated; see `data-raw/generate_example_data.R`.

tga_decomposition_reference

Reference table of typical lignocellulosic decomposition temperature ranges

Description

Returns a `data.frame` of commonly cited temperature ranges for the main decomposition stages seen in biomass/biochar-feedstock TGA curves, based on standard published ranges. Intended as a starting reference for `assign_dtg_peaks()`; users should adjust ranges to match their own literature basis and feedstock where needed, the same way `ftir_band_reference()` is a starting point rather than a fixed truth.

Usage

```
tga_decomposition_reference()
```

Value

A `data.frame` with columns `stage`, `range_low_C`, `range_high_C`, `description`.

tga_dtg

Compute a smoothed derivative thermogravimetric (DTG) curve

Description

Fits a smoothing spline to the weight-loss curve and differentiates it analytically (`stats::smooth.spline()` plus its `deriv = 1` predict method), rather than taking a raw finite difference of noisy instrument data. Sign convention: `dtg` is positive during weight loss (`-dW/dT`), so decomposition stages show up as positive peaks, matching how DTG curves are conventionally plotted.

Usage

```
tga_dtg(
  tga,
  temperature_col = "temperature_C",
  weight_col = "weight_pct",
  spar = NULL
)
```

Arguments

tga	A data frame with columns temperature_col, weight_col (e.g. as returned by read_tga_txt()).
temperature_col, weight_col	Column names. Defaults "temperature_C", "weight_pct".
spar	Smoothing parameter passed to stats::smooth.spline(). NULL (default) lets it choose by generalized cross-validation.

Value

A data frame with columns temperature_C, weight_pct, weight_pct_smooth, dtg (percent per degree C), sorted by temperature_C ascending.

Examples

```
set.seed(1)
T <- seq(25, 800, by = 5)
W <- 100 - 5 * (T > 110) - 40 / (1 + exp(-(T - 340) / 20)) + rnorm(length(T), 0, 0.15)
tga <- data.frame(temperature_C = T, weight_pct = pmax(W, 20))
dtg <- tga_dtg(tga)
```

---

tga_kinetics_kissinger

Kissinger non-isothermal kinetics from multi-heating-rate DTG peaks

---

Description

Fits the Kissinger (1957) method for a single decomposition step observed at several heating rates: ln(beta/Tp^2) = ln(A*R/Ea) - Ea/(R*Tp), a linear regression of ln(beta/Tp^2) against 1/Tp. Requires the DTG peak temperature of the *same* decomposition stage (e.g. matched via assign_dtg_peaks()) from runs at several different heating rates on the same sample — not a single TGA curve.

Usage

```
tga_kinetics_kissinger(heating_rate_Kmin, T_peak_C)
```

Arguments

heating_rate_Kmin	Numeric vector of heating rates (degrees C/min), one per run.
T_peak_C	Numeric vector of DTG peak temperatures (degrees C) for the same decomposition stage, one per run, matched to heating_rate_Kmin.

Details

Because the fit is a plain stats::lm() with an element named model, fit_confint() works on the result directly, the same as on any isotherm/kinetics fit in this package.

Value

A list with elements `Ea_kJmol` (activation energy), `A_min1` (pre-exponential factor, per minute, consistent with `heating_rate_Kmin` in degC/min), `R2`, `model` (the underlying `lm` fit).

Examples

```
# Synthetic: true Ea = 150 kJ/mol, illustrating the expected trend
# (peak temperature rises with heating rate).
beta <- c(5, 10, 15, 20)
Tp_C <- c(320, 335, 344, 351)
tga_kinetics_kissinger(beta, Tp_C)
```

---

tga_normalize *Normalize raw TGA mass readings to percent of initial mass*

Description

Some instrument exports report raw sample mass (e.g. mg) rather than weight percent. This rescales a mass column to percent of the first (initial) reading, the form expected by `tga_dtg()`, `tga_stages()`, and `plot_tga()`.

Usage

```
tga_normalize(tga, mass_col = "weight_mg", out_col = "weight_pct")
```

Arguments

`tga`	A data frame containing `mass_col`.
`mass_col`	Name of the raw mass column. Default `"weight_mg"`.
`out_col`	Name to give the new percent column. Default `"weight_pct"`.

Value

`tga` with an added/overwritten `out_col` column.

Examples

```
tga <- data.frame(temperature_C = c(25, 100, 500), weight_mg = c(8.2, 8.1, 5.4))
tga_normalize(tga)
```

`tga_stages` *Proximate analysis (moisture, VM, ash, fixed carbon) from a TGA curve*

Description

Reads moisture, volatile matter, and ash percentages directly off a single TGA weight-loss curve at three temperature breakpoints, then hands them to `proximate_analysis()` for fixed-carbon-by-difference (so the 100%-sum check and warning behaviour is identical either way, whether the three percentages came from separate gravimetric sub-samples or from one continuous TGA run).

Usage

```
tga_stages(
  tga,
  temperature_col = "temperature_C",
  weight_col = "weight_pct",
  moisture_end = 110,
  vm_end = 650
)
```

Arguments

`tga`	A data frame with columns `temperature_col`, `weight_col`.
`temperature_col, weight_col`	Column names. Defaults `"temperature_C"`, `"weight_pct"`.
`moisture_end`	Temperature (degrees C) marking the end of the drying/moisture-loss stage. Default `110`.
`vm_end`	Temperature (degrees C) marking the end of the volatile matter (devolatilization) stage. Default `650`.

Details

This assumes `weight_col` is expressed as percent of the original as-received mass (starting near 100%) and that the run proceeded far enough for the residual weight at the final temperature to represent ash (i.e. an oxidative burn-off, per protocols such as ASTM E1131). If your run stayed under inert atmosphere throughout, the residual at the final temperature is fixed carbon + ash combined, not ash alone; in that case treat `ash_pct` in the output with caution, or supply a separately measured ash percentage to `proximate_analysis()` directly instead.

Value

A data frame as returned by `proximate_analysis()` (`moisture_pct`, `VM_pct`, `ash_pct`, `fixed_carbon_pct`), with two extra columns `moisture_end_C`, `vm_end_C` recording the breakpoints used.

Examples

```
set.seed(1)
T <- seq(25, 800, by = 5)
W <- 100 - 5 * (T > 110) - 40 / (1 + exp(-(T - 340) / 20)) + rnorm(length(T), 0, 0.15)
tga <- data.frame(temperature_C = T, weight_pct = pmax(W, 20))
tga_stages(tga)
```

`tga_stages_batch` *Proximate analysis from TGA curves, separately for each sample in a data frame*

Description

Splits `df` by `group_col` (e.g. a sample ID column) and applies `tga_stages()` independently within each group, returning one row per group. Groups that fail (e.g. too few points, breakpoints outside the temperature range) are skipped with a warning rather than aborting the whole batch, mirroring `fit_isotherm_batch()` and `fit_kinetics_batch()`.

Usage

```
tga_stages_batch(
  df,
  group_col,
  temperature_col = "temperature_C",
  weight_col = "weight_pct",
  moisture_end = 110,
  vm_end = 650
)
```

Arguments

`df`	A long-format data frame containing at least `group_col`, `temperature_col`, `weight_col`.
`group_col`	Name of the grouping column (e.g. sample ID).
`temperature_col, weight_col`	Column names. Defaults `"temperature_C"`, `"weight_pct"`.
`moisture_end`	Temperature (degrees C) marking the end of the drying/moisture-loss stage. Default `110`.
`vm_end`	Temperature (degrees C) marking the end of the volatile matter (devolatilization) stage. Default `650`.

Value

A data frame with one row per successfully processed group, `group_col` plus the columns returned by `tga_stages()`.

Examples

```
set.seed(1)
mk <- function(id, vm_center) {
  T <- seq(25, 800, by = 10)
  W <- 100 - 5 * (T > 110) - 40 / (1 + exp(-(T - vm_center) / 20)) + rnorm(length(T), 0, 0.15)
  data.frame(id = id, temperature_C = T, weight_pct = pmax(W, 20))
}
df <- rbind(mk("SBC300-30", 300), mk("SBC450-30", 340), mk("SBC600-30", 380))
tga_stages_batch(df, "id")
```

---

ultimate_ratios *Atomic ratios from ultimate (CHNS) elemental analysis*

Description

Converts elemental mass percentages (from a CHNS/CHNSO analyser) to atomic ratios (H/C, O/C, and optionally N/C), the standard inputs for a Van Krevelen diagram. Uses standard atomic weights (C 12.011, H 1.008, O 15.999, N 14.007).

Usage

```
ultimate_ratios(C_pct, H_pct, O_pct, N_pct = NULL)
```

Arguments

`C_pct`, `H_pct`, `O_pct`	Numeric vectors of carbon, hydrogen, and oxygen mass percentages.
`N_pct`	Optional numeric vector of nitrogen mass percentage; if supplied, `N/C` is also returned.

Details

Note: `O_pct` is very often obtained by difference (`100 - C - H - N - S - ash`) rather than measured directly; supply whichever value your own workflow produced.

Value

A data frame with columns `HC_ratio`, `OC_ratio`, and (if `N_pct` supplied) `NC_ratio`.

Examples

```
ultimate_ratios(C_pct = 65.2, H_pct = 2.8, O_pct = 18.4, N_pct = 1.1)
```

xrd_crystallinity_index *Peak-area crystallinity index from an XRD pattern*

Description

Computes a simple crystallinity index as the ratio of crystalline peak area to total (crystalline + amorphous) area, following the common peak-area deconvolution approach used for carbon materials: `CI = A_crystalline / (A_crystalline + A_amorphous)`.

Usage

```
xrd_crystallinity_index(area_crystalline, area_amorphous)
```

Arguments

`area_crystalline`	Numeric: integrated area of crystalline peak(s).
`area_amorphous`	Numeric: integrated area of the amorphous "hump".

Details

This function does not perform peak fitting/deconvolution itself; it expects the user to have already separated crystalline vs. amorphous contributions (e.g. via baseline subtraction and peak integration in their preferred XRD software), and simply computes the ratio from the resulting areas. This keeps the function honest about not fabricating peak-fitting results it cannot verify.

Value

Numeric crystallinity index between 0 and 1.

Examples

```
xrd_crystallinity_index(area_crystalline = 1200, area_amorphous = 800)
```

xrd_deconvolve *Simplified multi-Gaussian XRD peak deconvolution*

Description

Fits a sum of Gaussian peaks to an XRD pattern at user-supplied approximate peak centers (2theta), by simultaneous nonlinear least squares. From the fitted peak areas and the total pattern area, also computes a crystallinity index (see `xrd_crystallinity_index()`) by treating everything not accounted for by the fitted crystalline peaks as the amorphous contribution.

Usage

```
xrd_deconvolve(two_theta, intensity, peak_centers, start_width = 1)
```

Arguments

two_theta, intensity	Numeric vectors: the diffraction angle (degrees 2theta) and intensity of the pattern. Should already be baseline-corrected (e.g. via an approach analogous to baseline_correct()) so that intensity represents peak signal above a flat zero.
peak_centers	Numeric vector of approximate 2theta positions for the crystalline peaks to fit.
start_width	Starting value (degrees 2theta) for each peak's Gaussian width. Default 1.

Details

This is a practical, simplified tool intended for a handful of well-separated peaks on an already baseline-corrected pattern — it is not a substitute for Rietveld refinement or dedicated diffraction software, and may fail to converge for patterns with many strongly overlapping peaks (in which case, try narrowing peak_centers to the most prominent peaks, or refine starting values manually).

Value

A list with elements:

peaks	A data frame with one row per peak: center, height, width, area
crystallinity_index	sum(peaks$area) / total pattern area
R2	Coefficient of determination of the overall multi-Gaussian fit
model	The underlying nls fit object

Examples

```
set.seed(1)
two_theta <- seq(10, 40, by = 0.1)
intensity <- 300 * exp(-((two_theta - 22)^2) / (2 * 0.8^2)) +
             150 * exp(-((two_theta - 29)^2) / (2 * 0.6^2)) +
             rnorm(length(two_theta), 0, 3)
intensity <- pmax(0, intensity)
xrd_deconvolve(two_theta, intensity, peak_centers = c(22, 29))
```

Index