Package: libscanR (via r-universe)

November 3, 2026

Type Package

Title Analysis and Visualization of Laser-Induced Breakdown Spectroscopy Data

Version 0.1.0.9001

Description Provides a vendor-agnostic pipeline for importing, preprocessing, calibrating, and visualizing Laser-Induced Breakdown Spectroscopy (LIBS) spectral data. Supports baseline correction, peak identification against user-supplied atomic emission line references, univariate and multivariate calibration, relative elemental amounts under an explicitly specified two-stage Saha-Boltzmann model, and calibration-based detection-limit estimates. Includes exploratory chemometric analysis (principal component analysis, partial least squares discriminant analysis, clustering), heuristic tissue classification, and spatial elemental mapping. Includes an interactive 'shiny' application for spectral exploration. Designed for biomedical tissue analysis, environmental monitoring, and materials characterization workflows.

License MIT + file LICENSE

URL https://github.com/cttir/libscanR, https://cttir.github.io/libscanR/

BugReports https://github.com/cttir/libscanR/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Depends R (>= 4.1.0)

Imports rlang, cli, tibble, jsonlite, Matrix, ggplot2, stats, utils, tools, grDevices, shiny (>= 1.6.0)

Suggests bslib (>= 0.5.0), plotly, DT, shinyWidgets, signal, pls, e1071, ranger, uwot, dbscan, withr, testthat (>= 3.0.0), knitr, rmarkdown, pkgdown, covr

VignetteBuilder knitr

LazyData false

Config/Needs/website CTTIR/themakR

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

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

Date/Publication 2026-11-03 13:22:01 UTC

RemoteUrl https://github.com/CTTIR/libscanR

RemoteRef HEAD

RemoteSha eccbb3e50dc6f2ace752a3ae10b388849298f910

Contents

[.libs_dataset . . . 4
[.libs_spectrum . . . 4
c.libs_spectrum . . . 5
dim.libs_dataset . . . 5
dim.libs_spectrum . . . 6
is_libs_calibration . . . 6
is_libs_dataset . . . 7
is_libs_spectrum . . . 7
length.libs_dataset . . . 8
length.libs_spectrum . . . 8
ls_average_shots . . . 9
ls_baseline . . . 9
ls_build_map . . . 10
ls_calibrate . . . 11
ls_calibration . . . 13
ls_classify . . . 14
ls_cluster . . . 15
ls_crop . . . 16
ls_dataset . . . 16
ls_demo_peaks . . . 17
ls_element_db . . . 18
ls_example_data . . . 19
ls_export_spectra . . . 20
ls_find_peaks . . . 21
ls_gate_optimize . . . 22
ls_identify_peaks . . . 22
ls_lod . . . 23
ls_loq . . . 24
ls_map_elements . . . 25
ls_normalize . . . 26
ls_pca . . . 27
ls_peak_area . . . 27

ls_plot_calibration 28
ls_plot_element_map 29
ls_plot_loadings 30
ls_plot_map 30
ls_plot_map_panel 31
ls_plot_overlay 32
ls_plot_pca 32
ls_plot_plsda 33
ls_plot_region 34
ls_plot_residuals 35
ls_plot_scree 35
ls_plot_spectrum 36
ls_plsda 37
ls_quantify 38
ls_read_aurora 38
ls_read_auto 39
ls_read_csv 40
ls_read_dir 40
ls_read_sciaps 41
ls_read_spectrum 42
ls_run_app 43
ls_saha_boltzmann 44
ls_simulate_spectrum 47
ls_smooth 48
ls_spectrum 49
ls_tissue_classify 50
ls_tissue_discriminate 51
ls_train_classifier 52
ls_write_csv 52
plot.libs_dataset 53
plot.libs_spectrum 54
predict.libs_calibration 54
print.libs_calibration 55
print.libs_dataset 55
print.libs_spectrum 56
scale_color_wavelength 56
summary.libs_calibration 57
summary.libs_dataset 57
summary.libs_spectrum 58
theme_libs 58

Index **59**

[.libs_dataset *Subset a LIBS Dataset*

Description

Subset a LIBS Dataset

Usage

```
## S3 method for class 'libs_dataset'
x[i]
```

Arguments

x	A libs_dataset object.
i	Integer or logical index of spectra to keep.

Value

A new libs_dataset.

[.libs_spectrum *Subset a LIBS Spectrum by Wavelength*

Description

Returns a new libs_spectrum with channels whose wavelengths fall in the supplied numeric vector's range.

Usage

```
## S3 method for class 'libs_spectrum'
x[i]
```

Arguments

x	A libs_spectrum object.
i	Numeric vector. Wavelengths are kept if they fall in [min(i), max(i)]. Can also be a logical or integer index into channels.

Value

A new libs_spectrum object.

c.libs_spectrum *Combine Multiple LIBS Spectra by Stacking Shots*

Description

Concatenates shots from multiple spectra. All spectra must share the same wavelength axis. Shared metadata are retained; source metadata are stored in `combined_metadata`. If every input has a baseline, the output stores its shot-weighted mean; otherwise the aggregate baseline is unknown.

Usage

```
## S3 method for class 'libs_spectrum'
c(...)
```

Arguments

`...`	Two or more `libs_spectrum` objects.

Value

A new `libs_spectrum` with combined shots.

dim.libs_dataset *Dimensions of a Dataset (spectra x channels)*

Description

Dimensions of a Dataset (spectra x channels)

Usage

```
## S3 method for class 'libs_dataset'
dim(x)
```

Arguments

`x`	A `libs_dataset` object.

Value

Integer vector of length 2.

dim.libs_spectrum *Dimensions of a LIBS Spectrum (shots x channels)*

Description

Dimensions of a LIBS Spectrum (shots x channels)

Usage

```
## S3 method for class 'libs_spectrum'
dim(x)
```

Arguments

x	A libs_spectrum object.

Value

Integer vector of length 2.

is_libs_calibration *Test whether an object is a libs_calibration*

Description

Test whether an object is a libs_calibration

Usage

```
is_libs_calibration(x)
```

Arguments

x	Object to test.

Value

Logical scalar.

is_libs_dataset *Test whether an object is a libs_dataset*

Description

Test whether an object is a libs_dataset

Usage

```
is_libs_dataset(x)
```

Arguments

x	Object to test.

Value

Logical scalar.

is_libs_spectrum *Test whether an object is a libs_spectrum*

Description

Test whether an object is a libs_spectrum

Usage

```
is_libs_spectrum(x)
```

Arguments

x	Object to test.

Value

Logical scalar.

length.libs_dataset *Number of Spectra in a Dataset*

Description

Number of Spectra in a Dataset

Usage

```
## S3 method for class 'libs_dataset'
length(x)
```

Arguments

x	A libs_dataset object.

Value

Integer scalar.

length.libs_spectrum *Number of Channels in a LIBS Spectrum*

Description

Number of Channels in a LIBS Spectrum

Usage

```
## S3 method for class 'libs_spectrum'
length(x)
```

Arguments

x	A libs_spectrum object.

Value

Integer scalar.

ls_average_shots *Average Replicate Shots*

Description

Averages shots within a `ls_spectrum()`.

Usage

```
ls_average_shots(x, method = "mean", trim = 0, remove_outliers = FALSE)
```

Arguments

`x`	A `ls_spectrum()` object.
`method`	Character. `"mean"` or `"median"`. Default `"mean"`.
`trim`	Numeric. Fraction trimmed from each end for trimmed mean. Default 0.
`remove_outliers`	Logical. Remove shots with deviation > 3 SD from the per-channel mean. Default `FALSE`.

Value

A `libs_spectrum` with a single averaged shot.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(), n_shots = 10, seed = 1)
ls_average_shots(spec, remove_outliers = TRUE)
```

ls_baseline *Baseline Correction*

Description

Removes continuum/baseline drift from LIBS spectra. Supported methods: `"snip"` (Statistics-sensitive Non-linear Iterative Peak-clipping, Morhac/Ryan), `"als"` (Asymmetric Least Squares smoothing, Eilers & Boelens), `"rolling_ball"` (morphological), `"linear"` (two-endpoint interpolation), or `"polynomial"` (low-order polynomial fit of baseline-only regions).

Usage

```
ls_baseline(
  x,
  method = "snip",
  iterations = 100,
  order = 3,
  lambda = 1e+05,
  p = 0.01,
  radius = 50,
  ...
)
```

Arguments

x	A ls_spectrum() or ls_dataset() object.
method	Character. One of "snip", "als", "rolling_ball", "linear", "polynomial". Default "snip".
iterations	Integer. Iterations for iterative methods (SNIP, ALS). Default 100.
order	Integer. Polynomial order for "polynomial". Default 3.
lambda	Numeric. Smoothness parameter for ALS. Default 1e5.
p	Numeric. Asymmetry parameter for ALS (0 < p < 1). Default 0.01.
radius	Integer. Rolling-ball radius in channels. Default 50.
...	Reserved for future use.

Value

Same class as input with baseline-corrected intensities; the estimated baseline is stored in the baseline element for inspection.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(), seed = 1)
corr <- ls_baseline(spec, method = "snip")
is.null(corr$baseline)
```

---

ls_build_map *Build a Spatial Elemental Map*

Description

Constructs a spatial map from LIBS line-scan or raster-scan data. The dataset's sample_info must include x_pos and y_pos columns. Coordinates must be finite numeric values and each pair must be unique; aggregate replicate positions explicitly before mapping them.

Usage

```
ls_build_map(dataset, element, line_nm, calibration = NULL, window_nm = NULL)
```

Arguments

dataset	A ls_dataset() object with spatial coordinates.
element	Character. Element label (for metadata).
line_nm	Numeric. Emission line wavelength (nm).
calibration	Optional ls_calibration(). If provided, intensities are converted to concentration. Default NULL.
window_nm	Optional integration window. NULL reuses the calibration's fitted window, or 1 nm for an uncalibrated map.

Value

A list with S3 class libs_map containing x, y, values, element, line_nm, unit, and a rasterized grid matrix when the points form a regular grid.

Examples

```
ds <- ls_example_data("spatial")
m <- ls_build_map(ds, "Ca", 393.37)
dim(m$grid)
```

---

ls_calibrate *Build Calibration Curve*

Description

Constructs a calibration model from standards with known concentrations.

Usage

```
ls_calibrate(
  dataset,
  element,
  line_nm,
  concentrations,
  method = "univariate",
  internal_std_nm = NULL,
  window_nm = 1,
  pls_window_nm = NULL,
  n_components = 5,
  unit = "ppm",
  verbose = TRUE
)
```

Arguments

dataset	A ls_dataset() containing standard spectra. Spectra order must correspond to concentrations.
element	Character. Target element symbol.
line_nm	Numeric. Emission line wavelength (nm).
concentrations	Numeric vector of known concentrations, one per spectrum in dataset.
method	Character. "univariate" (default), "internal_std", "pls", or "cf_libs".
internal_std_nm	Numeric. Internal standard line wavelength, required for method = "internal_std".
window_nm	Numeric. Peak integration window. Default 1.
pls_window_nm	Numeric vector of length 2. Spectral window for PLS. Default c(line_nm - 5, line_nm + 5).
n_components	Integer. Number of PLS components. Default 5.
unit	Character. Concentration unit. Default "ppm".
verbose	Logical. Default TRUE.

Details

Prediction preserves the fitted wavelength axis, ordered preprocessing, integration window and internal-standard ratio or PLS columns. Univariate predictions retain inverse concentration-on-response regression. Estimated 3-sigma LOD and 10-sigma LOQ use the residual standard deviation of a separate response-on-concentration fit divided by its absolute slope. These estimates require independent verification near the limits; they are not validated instrument limits. PLS reports training RMSE separately and leaves LOD/LOQ unavailable (NA); residual prediction error alone is not a detection limit.

Value

A ls_calibration() object.

Examples

```
ds <- ls_example_data("calibration")
conc <- ds$sample_info$concentration
cal <- ls_calibrate(ds, "Ca", 393.37, conc, verbose = FALSE)
print(cal)
```

`ls_calibration` *Create a LIBS Calibration Model*

Description

Constructs a `libs_calibration` S3 object holding a fitted calibration model plus the underlying training data and figures of merit.

Usage

```
ls_calibration(
  element,
  wavelength_nm,
  concentrations,
  intensities,
  model,
  method = "univariate",
  lod = NULL,
  loq = NULL,
  r_squared = NULL,
  unit = "ppm",
  feature_contract = NULL,
  response_model = NULL,
  lod_basis = NULL,
  training_rmse = NULL
)
```

Arguments

`element`	Character. Element symbol (e.g. "Ca", "Fe").
`wavelength_nm`	Numeric. Emission line wavelength used.
`concentrations`	Numeric vector. Known concentrations of standards.
`intensities`	Numeric vector. Measured peak intensities or areas (or ratios for `internal_std`). Same length as `concentrations`.
`model`	Fitted model object (`lm` with one plain numeric predictor and an intercept, or `mvr` for PLS). Transformed terms and offsets are rejected by univariate prediction.
`method`	Character. One of `"univariate"`, `"internal_std"`, `"pls"`, `"cf_libs"`. Default `"univariate"`.
`lod`	Numeric. Limit of detection (3-sigma). Default `NULL`.
`loq`	Numeric. Limit of quantification (10-sigma). Default `NULL`.
`r_squared`	Numeric. Coefficient of determination. Default `NULL`.
`unit`	Character. Concentration unit, e.g. `"ppm"`, `"wt_pct"`, `"mg_kg"`. Default `"ppm"`.
`feature_contract`	Optional fitted wavelength/preprocessing and extraction contract, normally populated by `ls_calibrate()`.

response_model	Optional response-on-concentration regression used to estimate univariate detection limits.
lod_basis	Character description of the limit estimate, or NULL.
training_rmse	Numeric training prediction error, or NULL. This is not a detection limit or a held-out estimate.

Value

A libs_calibration S3 object.

Examples

```
conc <- c(100, 500, 1000, 2500, 5000)
ints <- conc * 0.12 + stats::rnorm(5, 0, 5)
m <- stats::lm(conc ~ ints)
cal <- ls_calibration("Ca", 393.37, conc, ints, model = m,
                      r_squared = summary(m)$r.squared)
print(cal)
```

---

ls_classify *Classify Unknown Spectra*

Description

Applies a trained classifier (from ls_plsda() or ls_train_classifier()) to new spectra.

Usage

```
ls_classify(model, new_data)
```

Arguments

model	A libs_plsda or libs_classifier object.
new_data	A ls_dataset() of unknown spectra.

Value

A tibble::tibble() with columns sample_id, predicted_class, and score for PLS-DA (an unbounded discriminant score, not a probability), or probability for SVM/RF. The wavelength axis and ordered preprocessing must match the fitted contract. Legacy models without this contract must be refitted before prediction.

Examples

```
ds <- ls_example_data("tissue")

if (requireNamespace("e1071", quietly = TRUE)) {
  clf <- ls_train_classifier(ds[1:30], "tissue", method = "svm")
  ls_classify(clf, ds[31:50])
}
```

`ls_cluster` *Cluster LIBS Spectra*

Description

Unsupervised clustering via k-means, hierarchical, or DBSCAN (DBSCAN requires the `dbscan` package in Suggests).

Usage

```
ls_cluster(dataset, method = "kmeans", k = 3, ...)
```

Arguments

`dataset`	A `ls_dataset()` object.
`method`	Character. `"kmeans"`, `"hclust"`, or `"dbscan"`. Default `"kmeans"`.
`k`	Integer. Number of clusters for kmeans/hclust. Default 3.
`...`	Additional arguments passed to the underlying clustering function.

Value

An S3 object of class `libs_clusters` with elements `cluster`, `method`, `k`, `silhouette` (mean score excluding DBSCAN noise; singleton clusters contribute zero; `NA` when not computable), `sample_info`.

Examples

```
ds <- ls_example_data("tissue")
cl <- ls_cluster(ds, method = "kmeans", k = 5)
cl$cluster
```

ls_crop *Crop Spectral Range*

Description

Subsets a spectrum or dataset to a wavelength range.

Usage

```
ls_crop(x, min_nm = NULL, max_nm = NULL)
```

Arguments

x	A ls_spectrum() or ls_dataset() object.
min_nm	Numeric. Minimum wavelength (nm). Default NULL = no lower bound.
max_nm	Numeric. Maximum wavelength (nm). Default NULL = no upper bound.

Value

Same class as input, cropped.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(), seed = 1)
ls_crop(spec, 380, 450)
```

ls_dataset *Create a LIBS Dataset*

Description

Constructs a libs_dataset: a collection of ls_spectrum() objects sharing a common wavelength axis, plus a sample-info table.

Usage

```
ls_dataset(spectra, sample_info = NULL)
```

Arguments

spectra	List of libs_spectrum objects. All must share the same wavelength axis (exact channel values). Resample deliberately before combining measurements on different axes; no channels are relabelled.
sample_info	Optional data.frame or tibble with one row per spectrum. Must contain column sample_id matching metadata$sample_id in each spectrum. Additional columns (e.g. material, group, concentration, x_pos, y_pos) are optional. If NULL, built from spectrum metadata.

Value

A libs_dataset S3 object.

Examples

```
specs <- lapply(1:3, function(i) {
 ls_simulate_spectrum(line_data = ls_demo_peaks(), elements = c(Ca = 1000 * i, Na = 500), seed = i)
})
specs <- Map(function(s, id) {
  s$metadata$sample_id <- paste0("s", id)
  s
}, specs, seq_along(specs))
ds <- ls_dataset(specs)
print(ds)
```

---

ls_demo_peaks *Fictional Peaks for Software Demonstrations*

Description

Returns invented peak strengths at the package's existing teaching and heuristic integration windows. Component names are labels used by the example scenarios, not validated elemental identifications. This table is not derived from an atomic database: it contains no transition rates, energy levels or statistical weights, and must not be used as a reference for experimental spectra. It is accepted only as explicit simulation input.

Usage

```
ls_demo_peaks()
```

Value

A tibble with component, wavelength_nm and relative_strength. Every strength is the arbitrary multiplier 0.001.

Examples

```
peaks <- ls_demo_peaks()
spec <- ls_simulate_spectrum(seed = 1, line_data = peaks)
ls_plot_spectrum(spec)
```

`ls_element_db` *Read a User-Supplied Atomic Line Reference*

Description

Reads a local CSV or RDS table supplied by the user. No atomic reference compilation is bundled and this function does not access the network. Configure a file with `options(libscanR.element_db = "/path/lines.csv")` or the environment variable `LIBSCANR_ELEMENT_DB`. An explicit `path` takes precedence over the option, which takes precedence over the variable.

Usage

```
ls_element_db(
  elements = NULL,
  range_nm = c(190, 900),
  min_aki = 0,
  ionization = c(1L, 2L),
  persistent_only = FALSE,
  path = NULL
)
```

Arguments

`elements`	Character vector of element symbols, or `NULL` for all.
`range_nm`	Two ordered finite wavelength limits, or `NULL` for all.
`min_aki`	Minimum transition probability in units of 10^8 s^-1. Missing rates are excluded only when this is positive.
`ionization`	Ionization states: 1 = neutral, 2 = singly ionized, or `NULL` for all supplied positive integer stages.
`persistent_only`	Keep only rows explicitly marked `persistent`.
`path`	Local CSV/RDS file. Default `NULL` uses the configured path.

Details

The table must contain `element`, `ionization`, `wavelength_nm`, `aki` (transition probability in units of 10^8 s^-1), `wavelength_medium` (`"air"` or `"vacuum"`) and a nonempty `source` attribution per row. Missing transition rates remain missing. Optional `ei_ev`, `ek_ev` and `persistent` columns default to missing energies and `FALSE`; no persistent line designation is inferred. Other columns and provenance are retained. The installed package provides `scripts/import_nist.py` for converting a separately downloaded NIST ASD CSV; see the README for exact settings.

Value

A `tibble::tibble()` retaining the reference columns. Physical fields required by `ls_saha_boltzmann()` must be supplied and independently checked; loading a reference does not establish its scientific validity or reuse rights.

Examples

```
# After downloading and importing a reference as described in the README:
# options(libscanR.element_db = "local-data/atomic-lines.csv")
# ls_element_db(elements = "Ca")
```

`ls_example_data` *Generate Example LIBS Dataset*

Description

Creates a synthetic `libs_dataset` suitable for demonstrating package features without an atomic reference download. All scenarios use explicit fictional peaks from `ls_demo_peaks()`; component names and tissue labels are teaching labels, not measured concentrations or biological ground truth. Scenarios:

Usage

```
ls_example_data(
  scenario = c("tissue", "calibration", "spatial", "all"),
  seed = 42,
  n_channels = 1024
)
```

Arguments

`scenario`	Character. One of `"tissue"`, `"calibration"`, `"spatial"`, `"all"`. Default `"tissue"`.
`seed`	Integer. Random seed. Default 42.
`n_channels`	Integer. Number of spectral channels. Default 1024 (reduced from 2048 for faster examples).

Details

- `"tissue"` — 50 spectra from 5 tissue types (bone, liver, kidney, muscle, fat), 10 per type, with deliberately different synthetic profiles.
- `"calibration"` — 27 spectra of Ca standards (9 concentrations x 3 replicates) with constant matrix (Na, K, Mg).
- `"spatial"` — 400 spectra on a 20x20 grid simulating a tissue cross-section with Ca/Fe gradients and a Zn-enriched hotspot.
- `"all"` — named list of the above.

Value

A `libs_dataset` object (or named list for `"all"`).

Examples

```
ds <- ls_example_data("tissue")
print(ds)
```

`ls_export_spectra` *Export a LIBS Dataset to a Directory*

Description

Writes each spectrum in a dataset as a separate CSV file, plus a `sample_info.csv` metadata table. IDs must be unique portable filenames. Existing output is left intact if `overwrite = FALSE`; if any target exists, the whole export is skipped. All files are staged before replacement, with rollback on ordinary write failures. Concurrent writers and power-loss recovery are not supported.

Usage

```
ls_export_spectra(dataset, dir, overwrite = FALSE)
```

Arguments

`dataset`	A `libs_dataset` object.
`dir`	Character. Output directory (created if missing).
`overwrite`	Logical. Overwrite existing files. Default `FALSE`.

Value

Invisibly returns `dir`.

Examples

```
ds <- ls_example_data("tissue")[1:3]
tmp <- tempfile()
ls_export_spectra(ds, tmp)
list.files(tmp)
unlink(tmp, recursive = TRUE)
```

`ls_find_peaks` *Find Emission Peaks*

Description

Detects peaks in a LIBS spectrum using local maxima detection with prominence and signal-to-noise filtering.

Usage

```
ls_find_peaks(
  x,
  snr_threshold = 3,
  min_prominence = 0.01,
  min_distance_nm = 0.5
)
```

Arguments

`x`	A `ls_spectrum()` object.
`snr_threshold`	Numeric. Minimum signal-to-noise ratio. Default 3.
`min_prominence`	Numeric. Minimum peak prominence as a fraction of maximum intensity (0-1). Default 0.01.
`min_distance_nm`	Numeric. Minimum distance between peaks (nm). Default 0.5.

Value

A `tibble::tibble()` with columns `wavelength_nm`, `intensity`, `snr`, `prominence`, `fwhm_nm`, `area`, sorted by intensity descending.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(),
                             elements = c(Ca = 5000, Na = 1000), seed = 1)
peaks <- ls_find_peaks(spec, snr_threshold = 5)
head(peaks)
```

ls_gate_optimize *Gate Delay Optimization*

Description

Analyzes SNR and signal-to-background ratios across spectra acquired at different gate delays to recommend optimal timing for a target line.

Usage

```
ls_gate_optimize(spectra, element, line_nm, window_nm = 1)
```

Arguments

spectra	List of ls_spectrum() objects. Each spectrum's metadata should contain gate_delay_us.
element	Character. Target element symbol.
line_nm	Numeric. Target emission line wavelength (nm).
window_nm	Numeric. Window around the line. Default 1.

Value

A tibble::tibble() with columns gate_delay_us, snr, sbr, peak_intensity, with attribute "recommended" giving the row index of the highest SNR.

Examples

```
specs <- lapply(c(0.5, 1, 2, 5, 10), function(g) {
  s <- ls_simulate_spectrum(line_data = ls_demo_peaks(), seed = round(g * 10))
  s$metadata$gate_delay_us <- g
  s
})
ls_gate_optimize(specs, "Ca", 393.37)
```

ls_identify_peaks *Identify Peaks Against the NIST Line Database*

Description

Matches detected peaks to known atomic emission lines. Each peak is assigned its nearest wavelength candidate from the user-configured reference. No atomic reference is bundled; configure ls_element_db() first. This is a candidate identification, not proof of elemental presence or resolution of blended transitions. Check the measured air/vacuum convention.

Usage

```
ls_identify_peaks(
  peaks,
  elements = NULL,
  tolerance_nm = 0.2,
  ionization = c(1L, 2L)
)
```

Arguments

peaks	A tibble from ls_find_peaks().
elements	Character vector. Restrict search. Default NULL = all.
tolerance_nm	Numeric. Maximum wavelength deviation. Default 0.2.
ionization	Integer vector of ionization states. Default c(1L, 2L).

Value

A tibble with all peak columns plus: element, ionization, nist_wavelength_nm, nist_aki, deviation_nm, confidence (0-1), n_candidates, and wavelength_medium. The legacy confidence name denotes only 1 - abs(deviation_nm) / tolerance_nm, not a probability.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(),
                             elements = c(Ca = 5000, Na = 1000), seed = 1)
pk <- ls_find_peaks(spec, snr_threshold = 5)
id <- ls_identify_peaks(pk, elements = c("Ca", "Na"))
head(id)
```

---

ls_lod *Limit of Detection (3-sigma)*

Description

Calculates LOD using the 3-sigma criterion.

Usage

```
ls_lod(calibration, blank = NULL, window_nm = NULL)
```

Arguments

calibration	A ls_calibration() object.
blank	Optional ls_spectrum() of a blank sample. If NULL (the default), LOD is recomputed from the residual standard error of the response-on-concentration calibration model. A supplied blank must contain at least two replicate shots on the fitted axis with the same preprocessing; its standard deviation is computed from the same integrated response (or internal-standard ratio) for each shot, not variation across wavelengths.
window_nm	Optional window; NULL uses the fitted window. A different value is rejected.

Value

Estimated numeric LOD in the calibration's concentration unit; NA for PLS. This 3-sigma estimate requires independent confirmation near the detection limit and is not an instrument qualification.

Examples

```
ds <- ls_example_data("calibration")
conc <- ds$sample_info$concentration
cal <- ls_calibrate(ds, "Ca", 393.37, conc, verbose = FALSE)
ls_lod(cal)
```

---

ls_loq *Limit of Quantification (10-sigma)*

Description

Calculates LOQ using the 10-sigma criterion.

Usage

```
ls_loq(calibration, blank = NULL, window_nm = NULL)
```

Arguments

calibration	A ls_calibration() object.
blank	Optional ls_spectrum() of a blank sample.
window_nm	Optional window; NULL uses the fitted window.

Value

Numeric LOQ in the calibration's concentration unit.

Examples

```
ds <- ls_example_data("calibration")
conc <- ds$sample_info$concentration
cal <- ls_calibrate(ds, "Ca", 393.37, conc, verbose = FALSE)
ls_loq(cal)
```

`ls_map_elements` *Build Maps for Multiple Elements*

Description

Build Maps for Multiple Elements

Usage

```
ls_map_elements(
  dataset,
  elements,
  lines_nm,
  calibrations = NULL,
  window_nm = NULL
)
```

Arguments

`dataset`	A `ls_dataset()` object with spatial coordinates.
`elements`	Character vector. Elements to map.
`lines_nm`	Named numeric vector: wavelength per element (e.g. `c(Ca = 393.37, Fe = 371.99)`).
`calibrations`	Named list of `ls_calibration()` objects. Default `NULL`.
`window_nm`	Optional integration window. `NULL` reuses each calibration's fitted window, or 1 nm for uncalibrated maps.

Value

A named list of `libs_map` objects.

Examples

```
ds <- ls_example_data("spatial")
ls_map_elements(ds, c("Ca", "Fe"),
                c(Ca = 393.37, Fe = 371.99))
```

`ls_normalize` *Normalize Spectra*

Description

Normalizes intensity values. Methods:

- `"total"` — divide each shot by total intensity
- `"max"` — divide by per-shot maximum
- `"snv"` — Standard Normal Variate: `(x - mean) / sd`
- `"internal_std"` — divide by intensity at a reference line
- `"area"` — divide by integrated (trapezoidal) area

Usage

```
ls_normalize(x, method = "total", ref_wavelength = NULL, ref_window = 1)
```

Arguments

`x`	A `ls_spectrum()` or `ls_dataset()` object.
`method`	Character. Normalization method. Default `"total"`.
`ref_wavelength`	Numeric. Required for `method = "internal_std"`.
`ref_window`	Numeric. Window width (nm) around `ref_wavelength`. Default 1.

Value

Same class as input with normalized intensities. A zero or non-finite divisor is an error; a constant SNV shot becomes zero. Previous removed baselines are retained in `metadata$baseline_history` and cleared from `baseline`, because per-shot scaling changes their units.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(), seed = 1)
spec_n <- ls_normalize(spec, method = "total")
sum(spec_n$intensity[1, ])  # ~1
```

ls_pca *Principal Component Analysis of LIBS Spectra*

Description

Performs PCA on the intensity matrix of a `ls_dataset()`.

Usage

```
ls_pca(dataset, n_components = 10, scale = TRUE, center = TRUE)
```

Arguments

`dataset`	A `ls_dataset()` object.
`n_components`	Integer. Number of PCs to retain. Default 10.
`scale`	Logical. Scale variables to unit variance. Default `TRUE`.
`center`	Logical. Center variables. Default `TRUE`.

Value

A list with S3 class `libs_pca` containing: `scores` (matrix: spectra x PCs), `loadings` (matrix: channels x PCs), `sdev`, `variance_explained` (fraction), `cumulative_variance`, `wavelength`, `sample_info`, `pca_obj`.

Examples

```
ds <- ls_example_data("tissue")
pca <- ls_pca(ds, n_components = 5)
pca$variance_explained
```

ls_peak_area *Calculate Peak Area*

Description

Integrates peak area at a given emission line, with a local baseline subtracted (linear interpolation between window edges).

Usage

```
ls_peak_area(x, center_nm, window_nm = 1, method = "trapezoidal")
```

Arguments

x	A ls_spectrum() object.
center_nm	Numeric. Peak center wavelength (nm).
window_nm	Numeric. Integration window width. Default 1. To support coarsely sampled input, the effective window is at least five median channel spacings, with up to five nearest channels used at edges. This does not resolve overlapping lines; inspect instrument resolution and the integration region before interpreting an area as an element.
method	Character. "trapezoidal" or "gaussian_fit". Default "trapezoidal". A failed, non-converged or non-physical Gaussian fit emits a warning and returns the trapezoidal area instead.

Value

Numeric. Baseline-subtracted integrated peak area.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(),
                             elements = c(Ca = 5000), seed = 1)
ls_peak_area(spec, 393.37)
```

---

ls_plot_calibration *Plot a Calibration Curve*

Description

Plot a Calibration Curve

Usage

```
ls_plot_calibration(
  calibration,
  show_lod = TRUE,
  show_loq = TRUE,
  show_prediction_interval = TRUE
)
```

Arguments

calibration	A ls_calibration() object. PLS shows fitted training concentrations against known concentrations, not a one-feature curve.
show_lod	Logical. Show LOD as a horizontal dashed line. Default TRUE.
show_loq	Logical. Show LOQ as a horizontal dashed line. Default TRUE.
show_prediction_interval	Logical. Show 95% prediction band. Default TRUE. For univariate/internal-standard fits this is the fitted model's interval for a new observation, not a confidence band for its mean. PLS plots do not supply a prediction interval.

Value

A ggplot2 object.

Examples

```
ds <- ls_example_data("calibration")
conc <- ds$sample_info$concentration
cal <- ls_calibrate(ds, "Ca", 393.37, conc, verbose = FALSE)
ls_plot_calibration(cal)
```

---

ls_plot_element_map *Plot an Element Map (alias)*

Description

Convenience alias for `ls_plot_map()`.

Usage

```
ls_plot_element_map(map, ...)
```

Arguments

map	A libs_map object.
...	Passed to `ls_plot_map()`.

Value

A ggplot2 object.

Examples

```
ds <- ls_example_data("spatial")
ls_plot_element_map(ls_build_map(ds, "Ca", 393.37))
```

ls_plot_loadings *Plot PCA Loadings*

Description

Plots the loadings for a selected principal component across the full wavelength axis, optionally annotating the top contributing channels.

Usage

```
ls_plot_loadings(pca, pc = 1, n_top = 10)
```

Arguments

pca	A libs_pca object.
pc	Integer. Which PC's loadings to show. Default 1.
n_top	Integer. Number of top-absolute contributors to annotate. Default 10.

Value

A ggplot2 object.

Examples

```
ds <- ls_example_data("tissue")
pca <- ls_pca(ds, n_components = 4)
ls_plot_loadings(pca, pc = 1, n_top = 5)
```

ls_plot_map *Plot an Elemental Map*

Description

Plot an Elemental Map

Usage

```
ls_plot_map(map, color_scale = "viridis", title = NULL)
```

Arguments

map	A libs_map object from ls_build_map().
color_scale	Character. "viridis", "magma", "plasma", "inferno", "hot", "jet". Default "viridis".
title	Character. Plot title. Default: element symbol.

Value

A ggplot2 object.

Examples

```
ds <- ls_example_data("spatial")
m <- ls_build_map(ds, "Ca", 393.37)
ls_plot_map(m)
```

`ls_plot_map_panel` *Plot Multi-Element Map Panel*

Description

Plots multiple elemental maps as a faceted panel.

Usage

```
ls_plot_map_panel(maps, ncol = 3)
```

Arguments

`maps`	Named list of `libs_map` objects.
`ncol`	Integer. Facet columns. Default 3.

Value

A ggplot2 object.

Examples

```
ds <- ls_example_data("spatial")
ms <- ls_map_elements(ds, c("Ca", "Fe", "Zn"),
                      c(Ca = 393.37, Fe = 371.99, Zn = 213.86))
ls_plot_map_panel(ms)
```

ls_plot_overlay *Overlay Multiple Spectra*

Description

Plots multiple spectra on the same axes, optionally colored by a grouping variable.

Usage

```
ls_plot_overlay(dataset, color_by = NULL, normalize = FALSE, alpha = 0.5)
```

Arguments

dataset	A ls_dataset() object.
color_by	Character. Column in sample_info for color mapping. Default NULL.
normalize	Logical. Divide each spectrum by its positive maximum. Default FALSE.
alpha	Numeric. Line alpha. Default 0.5.

Value

A ggplot2 object.

Examples

```
ds <- ls_example_data("tissue")
ls_plot_overlay(ds[1:10], color_by = "tissue")
```

ls_plot_pca *Plot PCA Scores*

Description

Plot PCA Scores

Usage

```
ls_plot_pca(pca, pc_x = 1, pc_y = 2, color_by = NULL, ellipses = TRUE)
```

Arguments

pca	A libs_pca object from ls_pca().
pc_x	Integer. PC for x-axis. Default 1.
pc_y	Integer. PC for y-axis. Default 2.
color_by	Character. Sample-info column. Numeric values use a continuous colour scale; other values define discrete groups. Default NULL.
ellipses	Logical. Draw nominal 95% normal-theory data ellipses for discrete groups with at least four non-collinear points. They describe score dispersion, not confidence intervals for a group mean. Default TRUE.

Value

A ggplot2 object.

Examples

```
ds <- ls_example_data("tissue")
pca <- ls_pca(ds, n_components = 4)
ls_plot_pca(pca, color_by = "tissue")
```

---

ls_plot_plsda *Plot PLS-DA Results*

Description

Plot PLS-DA Results

Usage

```
ls_plot_plsda(plsda, type = "scores")
```

Arguments

plsda	A libs_plsda object from ls_plsda().
type	Character. "scores" (LV1 vs LV2), "confusion" (heatmap), or "vip" (VIP scores spectrum). Default "scores".

Value

A ggplot2 object.

Examples

```
ds <- ls_example_data("tissue")

if (requireNamespace("pls", quietly = TRUE)) {
  plsda <- ls_plsda(ds, "tissue", n_components = 3, validation = "none")
  ls_plot_plsda(plsda, type = "scores")
}
```

`ls_plot_region` *Plot a Spectral Region in Detail*

Description

Zooms into a specified wavelength range and annotates peaks when a user reference is configured. With default `elements = NULL` and no reference, the spectrum remains visible and a subtitle reports unavailable annotations. Explicitly requesting element labels requires a configured reference.

Usage

```
ls_plot_region(x, min_nm, max_nm, elements = NULL, snr_threshold = 5)
```

Arguments

`x`	A `ls_spectrum()` or `ls_dataset()` object.
`min_nm`	Numeric. Minimum wavelength (nm).
`max_nm`	Numeric. Maximum wavelength (nm).
`elements`	Optional character vector. Element symbols to annotate. Default `NULL` (auto-detect top peaks).
`snr_threshold`	Numeric. SNR threshold for peak detection. Default 5.

Value

A ggplot2 object.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(), seed = 1)
ls_plot_region(spec, 380, 410)
```

ls_plot_residuals *Plot Calibration Residuals*

Description

Standard regression diagnostic: residuals vs fitted.

Usage

```
ls_plot_residuals(calibration)
```

Arguments

calibration	A ls_calibration() object.

Value

A ggplot2 object.

Examples

```
ds <- ls_example_data("calibration")
conc <- ds$sample_info$concentration
cal <- ls_calibrate(ds, "Ca", 393.37, conc, verbose = FALSE)
ls_plot_residuals(cal)
```

ls_plot_scree *Plot PCA Variance (Scree Plot)*

Description

Plot PCA Variance (Scree Plot)

Usage

```
ls_plot_scree(pca)
```

Arguments

pca	A libs_pca object.

Value

A ggplot2 object.

Examples

```
ds <- ls_example_data("tissue")
pca <- ls_pca(ds, n_components = 6)
ls_plot_scree(pca)
```

`ls_plot_spectrum` *Plot a LIBS Spectrum*

Description

Generic spectrum plotter. Returns a ggplot2 object.

Usage

```
ls_plot_spectrum(
  x,
  show_peaks = FALSE,
  show_elements = NULL,
  normalize = FALSE,
  snr_threshold = 5,
  ...
)
```

Arguments

`x`	A `ls_spectrum()` object.
`show_peaks`	Logical. Annotate detected peaks. Default `FALSE`.
`show_elements`	Character vector of element symbols to highlight as dashed vertical lines from the configured reference. All matching rows in the displayed range are used; no persistent-line status is inferred. Default `NULL`.
`normalize`	Logical. Divide intensity by its positive maximum before plotting. Default `FALSE`.
`snr_threshold`	Numeric. SNR threshold for peak annotation when `show_peaks = TRUE`. Default 5.
`...`	Additional arguments (currently unused).

Value

A ggplot2 object.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(), seed = 1)
ls_plot_spectrum(spec)
# Element overlays require a configured reference; see ls_element_db().
```

`ls_plsda` *PLS-DA Classification*

Description

Partial Least Squares Discriminant Analysis for class prediction.

Usage

```
ls_plsda(dataset, grouping, n_components = 5, validation = "CV")
```

Arguments

`dataset`	A `ls_dataset()` object.
`grouping`	Character. Column name in `sample_info` with class labels.
`n_components`	Integer. Number of PLS latent variables. Default 5.
`validation`	Character. `"CV"` (10-fold), `"LOO"`, or `"none"`. Default `"CV"`. Accuracy and predictions use held-out predictions for CV/LOO and fitted predictions only for `"none"`. Components are capped at the centered feature rank and training-sample bound of every fold. Every training fold must retain at least two classes. These sample-level folds do not establish patient/site-level generalization.

Value

An S3 object of class `libs_plsda` with elements `model`, `predictions` (tibble with predicted/observed class), `confusion_matrix`, `accuracy`, `class_labels`, `n_components`, `sample_info`, `feature_contract`, and `evaluation_basis` (`"cross_validation"`, `"leave_one_out"`, or `"training"`).

Examples

```
ds <- ls_example_data("tissue")

if (requireNamespace("pls", quietly = TRUE)) {
  plsda <- ls_plsda(ds, "tissue", n_components = 3, validation = "none")
  plsda$accuracy
}
```

ls_quantify *Quantify Element Concentration*

Description

Applies a calibration model using its saved wavelength axis, preprocessing and extraction contract. Legacy models without this contract must be refitted. Unavailable detection-limit flags are NA, not FALSE.

Usage

```
ls_quantify(calibration, x, window_nm = NULL)
```

Arguments

calibration	A ls_calibration() object.
x	A ls_spectrum() or ls_dataset() object.
window_nm	Optional numeric integration window. NULL reuses the fitted window; a different supplied value is rejected.

Value

A tibble::tibble() with columns sample_id, element, concentration, unit, below_lod, below_loq.

Examples

```
ds <- ls_example_data("calibration")
conc <- ds$sample_info$concentration
cal <- ls_calibrate(ds, "Ca", 393.37, conc, verbose = FALSE)
ls_quantify(cal, ds)
```

ls_read_aurora *Read Applied Spectra / Aurora Data*

Description

Parses output from Applied Spectra J200 and Aurora LIBS systems. Applied Spectra files commonly have wavelength in the first column and shots in subsequent columns, optionally with a few header lines.

Usage

```
ls_read_aurora(path, verbose = TRUE)
```

Arguments

path	Character. Path to export file (.csv or .asc).
verbose	Logical. Default TRUE.

Value

A libs_spectrum object.

Examples

```
tmp <- tempfile(fileext = ".csv")
writeLines(c(
  "Applied Spectra J200",
  "Model,J200",
  "wl,shot1,shot2",
  "200.0,50,52",
  "200.5,55,58",
  "201.0,52,50"
), tmp)
spec <- ls_read_aurora(tmp, verbose = FALSE)
unlink(tmp)
```

---

ls_read_auto *Auto-Detect and Read LIBS Data*

Description

Attempts to identify the file format (generic CSV, SciAps, Applied Spectra) and dispatches to the correct reader. When path is a directory, delegates to ls_read_dir().

Usage

```
ls_read_auto(path, verbose = TRUE)
```

Arguments

path	Character. Path to spectrum file or directory.
verbose	Logical. Default TRUE.

Value

A libs_spectrum or libs_dataset object.

Examples

```
tmp <- tempfile(fileext = ".csv")
utils::write.csv(data.frame(wavelength = seq(200, 300, 1),
                            intensity = 1:101),
                 tmp, row.names = FALSE)
spec <- ls_read_auto(tmp, verbose = FALSE)
unlink(tmp)
```

---

ls_read_csv *Read a LIBS Spectrum from CSV (Convenience Wrapper)*

Description

Read a LIBS Spectrum from CSV (Convenience Wrapper)

Usage

```
ls_read_csv(path, ...)
```

Arguments

path	Character. Path to CSV file.
...	Additional arguments passed to ls_read_spectrum().

Value

A libs_spectrum object.

---

ls_read_dir *Read All Spectra from a Directory*

Description

Batch-imports spectrum files from a directory into a libs_dataset.

Usage

```
ls_read_dir(
  dir,
  pattern = "\\.(csv|tsv|txt)$",
  recursive = FALSE,
  sample_info = NULL,
  ...
)
```

Arguments

dir	Character. Directory path.
pattern	Character. File pattern (regex). Default "\\.(csv|tsv|txt)$".
recursive	Logical. Recurse into subdirectories. Default FALSE.
sample_info	data.frame/tibble. Optional sample metadata table. Must have a sample_id column matching all of the filenames without extension. Default NULL.
...	Additional arguments passed to ls_read_spectrum().

Value

A libs_dataset object.

Examples

```
dir <- tempfile()
dir.create(dir)
for (i in 1:3) {
  wl <- seq(200, 900, length.out = 100)
  int <- exp(-((wl - 393)^2) / 2) + stats::rnorm(100, 0, 0.01)
  utils::write.csv(data.frame(w = wl, i = int),
                   file.path(dir, paste0("s", i, ".csv")),
                   row.names = FALSE)
}
ds <- ls_read_dir(dir, verbose = FALSE)
unlink(dir, recursive = TRUE)
```

---

ls_read_sciaps *Read SciAps LIBS Data*

Description

Parses output files from SciAps handheld LIBS analyzers (Z-series). The SciAps export format typically includes metadata lines preceded by # or a keyword header, followed by a wavelength/intensity table.

Usage

```
ls_read_sciaps(path, verbose = TRUE)
```

Arguments

path	Character. Path to SciAps export file.
verbose	Logical. Emit messages. Default TRUE.

Value

A `libs_spectrum` object with SciAps-specific metadata.

Examples

```
# See vignette("getting-started") for realistic usage; this parser
# is best exercised against actual SciAps export files.
tmp <- tempfile(fileext = ".csv")
writeLines(c(
  "# SciAps Z-903",
  "# Serial: 12345",
  "# Gate Delay (us): 2.5",
  "wavelength,intensity",
  "200.0,50",
  "200.5,55",
  "201.0,52"
), tmp)
spec <- ls_read_sciaps(tmp, verbose = FALSE)
unlink(tmp)
```

`ls_read_spectrum` *Read a LIBS Spectrum from File*

Description

Reads a single spectrum file. Supports CSV, TSV, and TXT formats where columns represent wavelength and intensity.

Usage

```
ls_read_spectrum(
  path,
  format = "auto",
  wavelength_col = 1,
  intensity_col = 2,
  skip = 0,
  metadata = list(),
  verbose = TRUE
)
```

Arguments

`path`	Character. Path to spectrum file.
`format`	Character. One of `"auto"` (detect from extension), `"csv"`, `"tsv"`, `"txt"`. Default `"auto"`.
`wavelength_col`	Column index or name for wavelength values. Default 1.

intensity_col	Column index/indices or name(s) for intensity values. Multiple columns are interpreted as multiple shots. Default 2.
skip	Integer. Header lines to skip before the data table. Default 0.
metadata	Named list of additional metadata. Default `list()`.
verbose	Logical. Emit import messages. Default `TRUE`.

Value

A `libs_spectrum` object.

Examples

```
tmp <- tempfile(fileext = ".csv")
wl <- seq(200, 900, length.out = 100)
int <- exp(-((wl - 393)^2) / 2)
utils::write.csv(data.frame(wavelength = wl, intensity = int),
                 tmp, row.names = FALSE)
spec <- ls_read_spectrum(tmp, verbose = FALSE)
unlink(tmp)
```

---

ls_run_app *Launch the libscanR Shiny Application*

Description

Starts an interactive Shiny application for exploring, preprocessing, and analyzing LIBS spectral data. The app provides six tabs: Import, Preprocessing, Peaks, Calibration, Chemometrics, and Export.

Usage

```
ls_run_app(data = NULL, port = NULL, launch.browser = TRUE)
```

Arguments

data	Optional. A `ls_spectrum()`, `ls_dataset()`, or file path to preload. Objects and files are validated before starting the app. Default `NULL` (start with upload interface).
port	Integer. Port for the app. Default `NULL` (auto-select).
launch.browser	Logical. Open browser on launch. Default `TRUE`.

Details

Upload batches are admitted together: invalid files leave the previously loaded data intact. Original upload names identify their source. New data and successful preprocessing invalidate downstream analysis results. Preprocessing controls take effect only after Apply; a failed operation retains the last successful data, and Reset restores raw data.

The complete ZIP download contains one CSV per spectrum and, for a dataset, `sample_info.csv`. Spectrum CSV files retain all shots, baseline and supported metadata through `ls_write_csv()`. ZIP creation requires an external `zip` command on `PATH` (or configured through `R_ZIPCMD`), including on Windows. The separate wide dataset CSV contains mean intensities only. Calibration CSV files summarize figures of merit rather than storing reusable prediction models. Load a labelled `ls_dataset()` through `data` for analyses requiring concentration or class columns in `sample_info`.

Example datasets use explicitly fictional teaching peaks and concentrations, not an atomic reference or experimental measurements. Peak detection and raw plots work without a reference. Optional element annotations require a user-supplied file configured for `ls_element_db()`; missing or invalid references leave raw detections available and report the annotation error. Peak and calibration CSVs carry `data_origin = "explicit_synthetic_peaks"` for wholly fictional inputs; mixed calibration inputs carry `"includes_explicit_synthetic_peaks"`. Peak exports follow the first spectrum actually used for detection.

The previous application data option is restored when the app stops or startup fails. Each browser session keeps its own working data and results.

Value

Invisible `NULL`. Launches a Shiny application as a side effect.

Examples

```
if (interactive()) {
  ls_run_app()
}
```

---

ls_saha_boltzmann *Relative Elemental Amounts under a Two-Stage LTE Model*

Description

Estimates a common Boltzmann temperature and relative elemental number fractions using explicit atomic and instrumental inputs. The Saha equation accounts for the neutral/singly-ionized balance. This research calculation requires optically thin emission, local thermodynamic equilibrium (LTE), a common sampled plasma, stoichiometric ablation and corrected spectral response. It does not establish those assumptions from an observed spectrum.

Usage

```
ls_saha_boltzmann(
  x,
  elements,
  lines_nm,
  electron_density = NULL,
  window_nm = 0.5,
  verbose = TRUE,
  atomic_data = NULL,
  partition_functions = NULL,
  ionization_energies_ev = NULL,
  temperature_k = NULL,
  intensity_scale = c("energy", "photon"),
  wavelength_medium = NULL,
  tolerance_nm = 0.01
)
```

Arguments

x	A ls_spectrum() with resolved, baseline-corrected lines.
elements	Unique character vector of elements to include.
lines_nm	Named list of measured line wavelengths per element.
electron_density	Independently estimated electron density in cm^-3. Required; no density is assumed.
window_nm	Integration width in nm. Default 0.5. Effective integration windows, including the coarse-channel expansion of ls_peak_area(), must be disjoint, wholly inside the measured wavelength range and contain at least three measured channels each.
verbose	Logical. Report the temperature and scope of the result.
atomic_data	A data.frame with one unambiguous record per selected transition: element, ionization (1 = neutral, 2 = singly ionized), wavelength_nm, wavelength_medium ("air" or "vacuum"), wavelength_vacuum_nm, A_s (transition rate in s^-1), g_upper (statistical weight), energy_upper_ev (relative to that ion's ground state), response (measured signal per relative emitted intensity), and source (the transition reference). These data must be supplied explicitly; the bundled historical line compilation is insufficient for this calculation. For vacuum records the two wavelength columns must agree within 1e-8 + 1e-8 * wavelength_nm nm (numerical rounding only).
partition_functions	Named list, one entry per element. Each entry is a list with neutral and ionized, each either a function of temperature in kelvin returning the partition function, or an explicitly chosen positive constant approximation. Use the same statistical-weight convention as g_upper and include the relevant energy levels.

`ionization_energies_ev`
: Named positive numeric vector of first ionization energies in eV, one per element.

`temperature_k`
: Optional independently estimated common temperature in kelvin. If NULL, estimate it from a shared slope with a separate intercept per element/ion stage. Estimation requires a full-rank fit and positive residual degrees of freedom. Non-negative slopes are rejected.

`intensity_scale`
: `"energy"` (default) or `"photon"`, describing the response-corrected integrated intensities. Energy intensities require the vacuum-wavelength factor; photon counts do not.

`wavelength_medium`
: Medium of the measured axis. If NULL, read `x$metadata$wavelength_medium`; an unspecified medium is an error.

`tolerance_nm`
: Maximum difference between requested and reference wavelengths. Default 0.01. Multiple possible transitions are an error.

Details

Earlier versions silently used unit statistical weights, omitted partition functions and ignored electron density. Those outputs cannot be reused as quantitative reference results. Supply verified physical inputs and refit. A low residual does not verify LTE, optical thinness, absence of self-absorption, spectral resolution, detector response or matrix-independent ablation.

Value

A tibble with `element`, `temperature_k`, `concentration_rel`, `electron_density`, `n_lines`, `r_squared` and `log_population_sd`. Fractions are normalized only over the supplied elements, under a model that excludes molecules and charge states above +1; they are neither mass fractions nor absolute sample concentrations. `log_population_sd` measures line/stage disagreement on the log population scale, not calibrated uncertainty. Attributes `line_diagnostics`, `assumptions`, and `temperature_source` retain the calculation's inputs and limitations.

References

NIST Atomic Spectra Database, spectral lines and Saha/LTE documentation: https://physics.nist.gov/PhysRefData/ASD/Html/lineshelp.html.

Examples

```
# A complete, explicitly synthetic forward-model example:
system.file("examples", "saha-boltzmann-synthetic.R", package = "libscanR")
```

`ls_simulate_spectrum` *Simulate a LIBS Spectrum*

Description

Generates a demonstration LIBS spectrum with configurable elemental composition, noise, and continuum background. Emission lines are drawn from the user-configured atomic reference (`ls_element_db()`) and modelled as Gaussian peaks. Peak widths are at least 2.5 channel spacings so coarse demonstrations remain visible; this is not an instrument response model or validation of quantitative elemental concentrations or tissue identity.

Usage

```
ls_simulate_spectrum(
  elements = c(Ca = 5000, Fe = 200, Na = 1000),
  wavelength_range = c(200, 900),
  n_channels = 2048,
  n_shots = 10,
  noise_level = 0.02,
  continuum_level = 100,
  shot_rsd = 0.1,
  fwhm_nm = 0.1,
  seed = NULL,
  line_data = NULL
)
```

Arguments

`elements`	Named numeric vector. Element concentrations (arbitrary units, interpreted as relative intensity scaling). Default `c(Ca = 5000, Fe = 200, Na = 1000)`.
`wavelength_range`	Numeric vector of length 2. Range in nm. Default `c(200, 900)`.
`n_channels`	Integer. Number of spectral channels. Default 2048.
`n_shots`	Integer. Number of replicate shots. Default 10.
`noise_level`	Numeric. Relative noise level (fraction of max intensity). Default 0.02.
`continuum_level`	Numeric. Continuum background level. Default 100.
`shot_rsd`	Numeric. Relative standard deviation of shot-to-shot intensity variation (0-1). Default 0.1.
`fwhm_nm`	Numeric. Typical line FWHM in nm. Default 0.1.
`seed`	Optional integer random seed for reproducibility.
`line_data`	Optional explicit teaching peak table, with `component`, `wavelength_nm` and `relative_strength` columns. Strengths are arbitrary intensity multipliers, not atomic transition probabilities. Use `ls_demo_peaks()` for a fictional fixture. `NULL` uses the configured atomic reference and retains the transition-rate scaling convention.

Value

A ls_spectrum() object.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(),
                             elements = c(Ca = 1000, Na = 500), seed = 1)
print(spec)
```

`ls_smooth` *Smooth Spectra*

Description

Applies spectral smoothing. Methods: `"savgol"` (Savitzky-Golay, needs the `signal` package when available, otherwise uses local polynomial least squares with an intercept), `"moving_avg"`, `"gaussian"`, `"median"`.

Usage

```
ls_smooth(x, method = "savgol", window = 11, poly_order = 3)
```

Arguments

`x`	A ls_spectrum() or ls_dataset() object.
`method`	Character. Default `"savgol"`.
`window`	Integer. Window size in channels (odd preferred). Default 11.
`poly_order`	Integer. Polynomial order for Savitzky-Golay. Default 3.

Value

Same class as input with smoothed intensities.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(), seed = 1)
sm <- ls_smooth(spec, method = "moving_avg", window = 9)
```

`ls_spectrum` *Create a LIBS Spectrum Object*

Description

Constructs a `libs_spectrum` S3 object from a wavelength axis and one or more intensity vectors (shots).

Usage

```
ls_spectrum(wavelength, intensity, metadata = list(), baseline = NULL)
```

Arguments

`wavelength`	Numeric vector of wavelengths in nanometers. Must be contain at least two distinct, finite, positive values. Channels and the optional baseline are sorted together in ascending wavelength order.
`intensity`	Numeric vector (single shot) or matrix (multiple shots) of emission intensities. If matrix: rows = shots, columns = wavelength channels, `ncol(intensity)` must equal `length(wavelength)`. All values must be finite; signed baseline-corrected intensities are supported.
`metadata`	Named list of metadata (e.g. `sample_id`, `material`, `gate_delay_us`, `integration_time_us`, `laser_energy_mj`, `atmosphere`, `date`). Default `list()`.
`baseline`	Optional numeric vector of baseline values corresponding to each wavelength channel (populated by `ls_baseline()`). Default `NULL`.

Value

An object of class `libs_spectrum`: a named list with elements `wavelength`, `intensity` (matrix: shots x channels), `metadata`, `baseline`, `n_shots`, `n_channels`, and `range_nm`.

Examples

```
wl <- seq(200, 900, length.out = 512)
int <- exp(-((wl - 393)^2) / 2) + stats::rnorm(512, 0, 0.05)
spec <- ls_spectrum(wl, int, metadata = list(sample_id = "demo"))
print(spec)
```

`ls_tissue_classify` *Tissue Classification via LIBS*

Description

Classifies tissue types based on LIBS spectral signatures. Implements three methods:

- `"ratio"` — an unvalidated heuristic based on integrated Ca, Na, K, Fe and C line responses. It is a research illustration, not a calibrated tissue probability or clinical classifier. All required lines must be covered and the response denominators must be positive; otherwise the row is unclassified.
- `"plsda"` — fits a PLS-DA model on the supplied labelled `reference` dataset and predicts class for `x`.
- `"svm"` — fits an SVM on `reference` and predicts class.

Usage

```
ls_tissue_classify(
  x,
  method = "ratio",
  reference = NULL,
  group_col = "tissue",
  verbose = TRUE
)
```

Arguments

`x`	A `ls_spectrum()` or `ls_dataset()` object (unknowns).
`method`	Character. `"ratio"`, `"plsda"`, or `"svm"`. Default `"ratio"`.
`reference`	A `ls_dataset()` with labelled tissue reference spectra (needed for `"plsda"` and `"svm"`).
`group_col`	Character. Column in reference `sample_info` with tissue labels. Default `"tissue"`.
`verbose`	Logical. Default `TRUE`.

Value

A `tibble::tibble()` with columns `sample_id`, `predicted_tissue`, `score`, `score_type`, and `status`. The former `confidence` column is replaced to distinguish relative heuristic scores, unrestricted PLS-DA scores and SVM probability estimates. None establishes tissue validity.

Examples

```
ds <- ls_example_data("tissue")
ls_tissue_classify(ds[1:3])
```

`ls_tissue_discriminate` *Tissue Discrimination Analysis*

Description

Identifies emission channels that best discriminate two tissue types via per-wavelength t-tests with Benjamini-Hochberg FDR correction, plus log2 fold-change. Element annotations are nearest wavelength candidates, not established elemental identities; `candidate_count` records ambiguity. Without a configured atomic reference, statistical results remain available, with missing element annotations and `annotation_status = "reference_unavailable"`. A configured but unreadable or invalid reference raises an error.

Usage

```
ls_tissue_discriminate(
  dataset,
  group_col,
  group_a,
  group_b,
  method = "t_test",
  tolerance_nm = 0.2
)
```

Arguments

`dataset`	A `ls_dataset()` with tissue labels.
`group_col`	Character. Label column in `sample_info`.
`group_a`	Character. First group label.
`group_b`	Character. Second group label.
`method`	Character. `"t_test"` (default) or `"fold_change"`.
`tolerance_nm`	Numeric. Matching tolerance when annotating elements. Default 0.2.

Value

A `tibble::tibble()` with columns `wavelength_nm`, `mean_a`, `mean_b`, `p_value`, `fold_change`, `fdr`, `significant`, `element`.

Examples

```
ds <- ls_example_data("tissue")
head(ls_tissue_discriminate(ds, "tissue", "bone", "muscle"))
```

ls_train_classifier *Train a Classifier*

Description

Trains SVM or Random Forest on spectral intensity features.

Usage

```
ls_train_classifier(dataset, grouping, method = "svm", ...)
```

Arguments

dataset	A ls_dataset() with labelled samples.
grouping	Character. Column name in sample_info with class labels.
method	Character. "svm" or "rf". Default "svm".
...	Additional arguments passed to the underlying trainer.

Value

An S3 object of class libs_classifier, including predictions, a square confusion matrix, the fitted feature_contract, and evaluation_basis: "training" for SVM or "out_of_bag" for RF. SVM evaluation uses the same probability-based class decision as ls_classify(). Training accuracy is not an independent estimate of generalization.

Examples

```
ds <- ls_example_data("tissue")

if (requireNamespace("e1071", quietly = TRUE)) {
  clf <- ls_train_classifier(ds, "tissue", method = "svm")
  clf$accuracy
}
```

ls_write_csv *Write a LIBS Spectrum to CSV*

Description

Exports a libs_spectrum to a CSV file with a wavelength_nm column plus one column per shot.

Usage

```
ls_write_csv(x, path, include_metadata = TRUE)
```

Arguments

x	A libs_spectrum object.
path	Character. Output file path.
include_metadata	Logical. Prepend metadata as comment lines. Default TRUE. JSON comments preserve scalar values, arrays, nested lists, and the baseline. Named vectors become JSON objects (lists on import); R-specific classes and attributes are not preserved.

Value

Invisibly returns path.

Examples

```
spec <- ls_simulate_spectrum(line_data = ls_demo_peaks(), seed = 1)
tmp <- tempfile(fileext = ".csv")
ls_write_csv(spec, tmp)
unlink(tmp)
```

---

plot.libs_dataset *Plot a LIBS Dataset*

Description

Plot a LIBS Dataset

Usage

```
## S3 method for class 'libs_dataset'
plot(x, y, ...)
```

Arguments

x	A libs_dataset object.
y	Ignored.
...	Passed to ls_plot_overlay().

Value

A ggplot object.

plot.libs_spectrum *Plot a LIBS Spectrum*

Description

Plot a LIBS Spectrum

Usage

```
## S3 method for class 'libs_spectrum'
plot(x, y, ...)
```

Arguments

x	A libs_spectrum object.
y	Ignored.
...	Arguments passed to ls_plot_spectrum().

Value

A ggplot object.

predict.libs_calibration *Predict Concentrations from a Calibration Model*

Description

Predict Concentrations from a Calibration Model

Usage

```
## S3 method for class 'libs_calibration'
predict(object, newdata, ...)
```

Arguments

object	A libs_calibration object.
newdata	A finite numeric vector of already integrated responses (ratios for internal-standard calibration), or a matrix with the fitted PLS columns in their trained order. This low-level method cannot verify wavelength labels; use ls_quantify() for spectra with checked feature identity.
...	Unused.

Value

Numeric vector of predicted concentrations.

print.libs_calibration *Print a LIBS Calibration*

Description

Print a LIBS Calibration

Usage

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

Arguments

x	A libs_calibration object.
...	Unused.

Value

Invisibly x.

print.libs_dataset *Print a LIBS Dataset*

Description

Print a LIBS Dataset

Usage

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

Arguments

x	A libs_dataset object.
...	Unused.

Value

Invisibly returns x.

print.libs_spectrum *Print a LIBS Spectrum*

Description

Print a LIBS Spectrum

Usage

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

Arguments

x	A libs_spectrum object.
...	Unused.

Value

Invisibly returns x.

scale_color_wavelength *Wavelength-Based Color Scale*

Description

Maps wavelength values (nm) to approximate visible-light colors for intuitive spectral visualization. Below 380 nm: UV purple; above 780 nm: deep red.

Usage

```
scale_color_wavelength(name = "Wavelength (nm)", ...)
```

Arguments

name	Character. Legend title. Default "Wavelength (nm)".
...	Additional arguments passed to ggplot2::scale_color_gradientn().

Value

A ggplot2 scale object.

Examples

```
df <- data.frame(x = 1:5, y = 1:5, wl = c(200, 400, 550, 700, 850))
ggplot2::ggplot(df, ggplot2::aes(x, y, colour = wl)) +
  ggplot2::geom_point(size = 5) + scale_color_wavelength()
```

summary.libs_calibration *Summary of a LIBS Calibration*

Description

Summary of a LIBS Calibration

Usage

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

Arguments

`object`	A `libs_calibration` object.
`...`	Unused.

Value

Invisibly a list of key figures of merit.

summary.libs_dataset *Summary of a LIBS Dataset*

Description

Summary of a LIBS Dataset

Usage

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

Arguments

`object`	A `libs_dataset` object.
`...`	Unused.

Value

Invisibly returns a list with per-group statistics.

summary.libs_spectrum *Summary of a LIBS Spectrum*

Description

Summary of a LIBS Spectrum

Usage

```
## S3 method for class 'libs_spectrum'
summary(object, n_peaks = 5, ...)
```

Arguments

object	A libs_spectrum object.
n_peaks	Integer. Number of top peaks to show. Default 5.
...	Unused.

Value

Invisibly returns a list with summary statistics.

theme_libs *LIBS Theme for ggplot2*

Description

A clean publication-ready theme for LIBS spectral plots.

Usage

```
theme_libs(base_size = 12)
```

Arguments

base_size	Numeric. Base font size in points. Default 12.

Value

A ggplot2 theme object.

Examples

```
ggplot2::ggplot(mtcars, ggplot2::aes(mpg, wt)) +
  ggplot2::geom_point() + theme_libs()
```

Index

[.libs_dataset, 4
[.libs_spectrum, 4

c.libs_spectrum, 5

dim.libs_dataset, 5
dim.libs_spectrum, 6

ggplot2::scale_color_gradientn(), *56*

is_libs_calibration, 6
is_libs_dataset, 7
is_libs_spectrum, 7

length.libs_dataset, 8
length.libs_spectrum, 8
ls_average_shots, 9
ls_baseline, 9
ls_baseline(), *49*
ls_build_map, 10
ls_build_map(), *30*
ls_calibrate, 11
ls_calibrate(), *13*
ls_calibration, 13
ls_calibration(), *11, 12*, *24, 25*, *28*, *35*, *38*
ls_classify, 14
ls_classify(), *52*
ls_cluster, 15
ls_crop, 16
ls_dataset, 16
ls_dataset(), *10–12*, *14–16*, *25–27*, *32*, *34*, *37, 38*, *43, 44*, *48*, *50–52*
ls_demo_peaks, 17
ls_demo_peaks(), *19*, *47*
ls_element_db, 18
ls_element_db(), *22*, *44*, *47*
ls_example_data, 19
ls_export_spectra, 20
ls_find_peaks, 21
ls_find_peaks(), *23*
ls_gate_optimize, 22
ls_identify_peaks, 22
ls_lod, 23
ls_loq, 24
ls_map_elements, 25
ls_normalize, 26
ls_pca, 27
ls_pca(), *33*
ls_peak_area, 27
ls_peak_area(), *45*
ls_plot_calibration, 28
ls_plot_element_map, 29
ls_plot_loadings, 30
ls_plot_map, 30
ls_plot_map(), *29*
ls_plot_map_panel, 31
ls_plot_overlay, 32
ls_plot_overlay(), *53*
ls_plot_pca, 32
ls_plot_plsda, 33
ls_plot_region, 34
ls_plot_residuals, 35
ls_plot_scree, 35
ls_plot_spectrum, 36
ls_plot_spectrum(), *54*
ls_plsda, 37
ls_plsda(), *14*, *33*
ls_quantify, 38
ls_quantify(), *54*
ls_read_aurora, 38
ls_read_auto, 39
ls_read_csv, 40
ls_read_dir, 40
ls_read_dir(), *39*
ls_read_sciaps, 41
ls_read_spectrum, 42
ls_read_spectrum(), *40, 41*
ls_run_app, 43
ls_saha_boltzmann, 44
ls_saha_boltzmann(), *18*

ls_simulate_spectrum, 47
ls_smooth, 48
ls_spectrum, 49
ls_spectrum(), *9, 10*, *16*, *21, 22*, *24*, *26*, *28*, *34*, *36*, *38*, *43*, *45*, *48*, *50*
ls_tissue_classify, 50
ls_tissue_discriminate, 51
ls_train_classifier, 52
ls_train_classifier(), *14*
ls_write_csv, 52
ls_write_csv(), *44*

plot.libs_dataset, 53
plot.libs_spectrum, 54
predict.libs_calibration, 54
print.libs_calibration, 55
print.libs_dataset, 55
print.libs_spectrum, 56

scale_color_wavelength, 56
summary.libs_calibration, 57
summary.libs_dataset, 57
summary.libs_spectrum, 58

theme_libs, 58
tibble::tibble(), *14*, *18*, *21, 22*, *38*, *50, 51*