

Package: hyperspectR (via r-universe)

October 14, 2026

Title Hyperspectral Imaging Analysis for Biomedical Applications

Version 0.2.0.9000

Description Read, process, analyze, and visualize hyperspectral imaging (HSI) data from Cubert snapshot cameras and standard ENVI/TIFF formats. Provides calibration workflows (dark correction, white reference normalization), spectral preprocessing (Savitzky-Golay smoothing, SNV, MSC, derivatives), dimensionless spectral indices, research hemoglobin fraction fitting, dimensionality reduction (PCA, MNF), pixel classification (SAM, SVM, random forest), and constrained spectral unmixing. Includes an interactive 'shiny' explorer, reproducible batch recipes and grouped evaluation tools. Results are research outputs; clinical or device-specific accuracy is not established by the included synthetic examples.

License MIT + file LICENSE

URL <https://github.com/cttir/hyperspectR>,
<https://cttir.github.io/hyperspectR/>

BugReports <https://github.com/cttir/hyperspectR/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Depends R (>= 4.1.0)

Imports rlang, cli, tibble, tidyr, ggplot2, stats, utils, grDevices,
graphics, tools, patchwork, viridisLite, scales

Suggests bslib, callr, shiny, dplyr, plotly, DT, terra, prospectr,
signal, nnls, uwot, e1071, ranger, cuvis.r, tivis.r, withr,
testthat (>= 3.0.0), knitr, rmarkdown, pkgdown, covr,
shinytest2

Additional_repositories <https://cttir.r-universe.dev>

VignetteBuilder knitr

Config/Needs/website CTTIR/themakR
Config/roxygen2/version 8.0.0
Config/pak/sysreqs libicu-dev
Repository <https://cttir.r-universe.dev>
Date/Publication 2026-10-14 10:37:51 UTC
RemoteUrl <https://github.com/CTTIR/hyperspectR>
RemoteRef HEAD
RemoteSha 642fc056cd43bee6e41aa4cd3b849a279a698b53

Contents

[.hsi_cube	3
as.data.frame.hsi_cube	4
as_tibble.hsi_cube	5
autoplot.hsi_cube	6
dim.hsi_cube	7
hs_absorbance	7
hs_band_ratio	8
hs_batch	8
hs_beer_lambert	9
hs_calibrate	10
hs_chromophore_data	11
hs_classify_rf	12
hs_classify_svm	13
hs_clinical_indices	14
hs_continuum_removal	14
hs_dark_correct	15
hs_derivative	16
hs_endmembers	16
hs_example_cube	17
hs_example_files	17
hs_export_png	18
hs_fix_bad_pixels	19
hs_group_summary	20
hs_grouped_evaluate	20
hs_mnf	21
hs_msc	22
hs_ndi	23
hs_npi	23
hs_pca	24
hs_plot_clinical	25
hs_plot_image	26
hs_plot_index	26
hs_plot_rgb	27
hs_plot_spectra	28

hs_predict	28
hs_process	29
hs_read_cube	29
hs_read_cubert	30
hs_read_envi	31
hs_read_tiff	32
hs_read_tivita	32
hs_recipe	33
hs_resample	34
hs_roi_polygon	34
hs_roi_rect	35
hs_roi_stats	36
hs_run_app	36
hs_sam	37
hs_simulate_cube	38
hs_smooth	39
hs_snv	40
hs_sto2	40
hs_thi	41
hs_transform	42
hs_twi	42
hs_umap	43
hs_unmix_nnl	44
hs_white_normalize	45
hs_write_envi	45
hs_write_tiff	46
hsi_cube	47
print.hsi_cube	48
scale_color_wavelength	48
summary.hsi_cube	49
theme_hsi	50

Index**51**

[.hsi_cube

*Subset an hsi_cube Object***Description**

Extract spatial and/or spectral subsets from a cube. Subsetting preserves the hsi_cube class.

Usage

```
## S3 method for class 'hsi_cube'
x[i, j, k, ...]
```

Arguments

x	An hsi_cube object.
i	Row indices (spatial).
j	Column indices (spatial).
k	Band indices (spectral).
...	Additional arguments (ignored).

Value

A subsetted [hsi_cube](#) object.

Examples

```
cube <- hs_example_cube()
sub <- cube[1:10, 1:10, 1:5]
dim(sub)
```

```
as.data.frame.hsi_cube
```

Convert hsi_cube to Data Frame

Description

Convert hsi_cube to Data Frame

Usage

```
## S3 method for class 'hsi_cube'
as.data.frame(x, ..., long = FALSE)
```

Arguments

x	An hsi_cube object.
...	Additional arguments (ignored).
long	Logical. If TRUE, returns long format with columns x, y, wavelength, value. If FALSE, returns wide format with columns x, y, and one column per band. Default FALSE.

Value

A data.frame.

Examples

```
cube <- hs_example_cube()
df <- as.data.frame(cube[1:3, 1:3, 1:3], long = TRUE)
head(df)
```

as_tibble.hsi_cube	<i>Convert hsi_cube to Tibble</i>
--------------------	-----------------------------------

Description

Convert hsi_cube to Tibble

Usage

```
## S3 method for class 'hsi_cube'
as_tibble(x, ..., long = TRUE, .name_repair = "unique")
```

Arguments

x	An hsi_cube object.
...	Additional arguments (ignored).
long	Logical. If TRUE (default), returns long format. See as.data.frame.hsi_cube() for details.
.name_repair	Name repair strategy (passed to tibble::as_tibble()).

Value

A [tibble::tibble](#).

Examples

```
cube <- hs_example_cube()
tb <- tibble::as_tibble(cube[1:3, 1:3, 1:3])
head(tb)
```

autoplot.hsi_cube	<i>Plot an hsi_cube Object</i>
-------------------	--------------------------------

Description

Creates a ggplot2 visualization of a hyperspectral cube. Supports RGB composite, single-band images, and spectral profile plots.

Usage

```
## S3 method for class 'hsi_cube'
autoplot(
  object,
  type = c("rgb", "band", "spectra"),
  band = NULL,
  wavelength = NULL,
  r = NULL,
  g = NULL,
  b = NULL,
  ...
)
```

Arguments

object	An hsi_cube object.
type	Character. Plot type: "rgb" (default), "band", or "spectra".
band	Integer. Band index for single-band display. Used when type = "band".
wavelength	Numeric. Wavelength (nm) for single-band display. Alternative to band.
r, g, b	Numeric. Wavelengths for RGB channels. Defaults: 640, 550, 460.
...	Additional arguments passed to internal plot functions.

Value

A [ggplot2::ggplot](#) object.

Examples

```
cube <- hs_example_cube()
ggplot2::autoplot(cube, type = "rgb")
```

dim.hsi_cube	<i>Get Dimensions of an hsi_cube</i>
--------------	--------------------------------------

Description

Get Dimensions of an hsi_cube

Usage

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

Arguments

x An [hsi_cube](#) object.

Value

Integer vector c(rows, cols, bands).

Examples

```
cube <- hs_example_cube()  
dim(cube)
```

hs_absorbance	<i>Convert Reflectance to Absorbance</i>
---------------	--

Description

Computes $A(\lambda) = -\log_{10}(R(\lambda))$. Values of R at or below zero are clamped to floor before log transformation.

Usage

```
hs_absorbance(cube, floor = 1e-06)
```

Arguments

cube An [hsi_cube](#) object with reflectance values.
floor Numeric. Minimum reflectance value to prevent log(0). Default 1e-6.

Value

An [hsi_cube](#) object with absorbance values.

Examples

```
cube <- hs_example_cube()
abs_cube <- hs_absorbance(cube)
```

hs_band_ratio	<i>Compute an Unscaled Spectral Band Ratio</i>
---------------	--

Description

A scene-independent, dimensionless ratio of mean spectral values. It is not calibrated oxygen saturation, perfusion, concentration or water content.

Usage

```
hs_band_ratio(cube, numerator, denominator)
```

Arguments

cube An [hsi_cube](#) object.
 numerator, denominator Numeric scalar wavelength or two-element range in nm.

Value

A spatial numeric matrix. Zero denominators and invalid pixels are NA.

hs_batch	<i>Run and Resume a Manifest of Hyperspectral Recordings</i>
----------	--

Description

Each recording is isolated: failures are recorded without discarding successful results. Resume requires matching input checksums, recipe, analysis parameters, package implementation, R version and dependency versions. Input and output paths are local; no network or hardware acquisition is performed.

Usage

```
hs_batch(
  manifest,
  recipe = hs_recipe(),
  out_dir,
  analysis = c("summary", "beer_lambert", "pca"),
  analysis_args = list(),
  resume = TRUE
)
```

Arguments

manifest	Data frame with distinct id and path columns. Optional reader_args list-column provides per-recording reader arguments.
recipe	A recipe from hs_recipe() .
out_dir	Directory for per-recording RDS files and run-summary.csv.
analysis	One of "summary", "beer_lambert", or "pca".
analysis_args	Named list of arguments for the analysis function.
resume	Logical. Reuse results only if their fingerprint matches.

Value

Data frame of recording identifiers, statuses, result files and errors.

hs_beer_lambert	<i>Beer-Lambert Chromophore Fitting</i>
-----------------	---

Description

Fits decadic absorbance to reference extinction coefficients with NNLS. Without a known optical pathlength, coefficients are concentration-pathlength products. The hemoglobin fraction is a research model estimate; diffuse tissue scattering and camera response can bias it.

Usage

```
hs_beer_lambert(
  cube,
  chromophores = c("Hb02", "Hb"),
  wavelength_range = c(500, 600),
  input = c("auto", "reflectance", "absorbance"),
  pathlength_cm = NULL,
  response = c("point", "gaussian"),
  keep_residuals = FALSE
)
```

Arguments

cube	An hsi_cube object with an explicit reflectance or absorbance domain.
chromophores	Character vector. Default c("Hb02", "Hb").
wavelength_range	Numeric vector of length 2. Fitting range. Default c(500, 600) (Hb Q-band region for best contrast).
input	"auto" reads the declared processing mode, or explicitly specify "reflectance" or "absorbance". Values never determine the domain.
pathlength_cm	Positive scalar optical pathlength, or NULL (unknown).
response	"point" (default) samples band centers; "gaussian" integrates reference coefficients using cube FWHM as Gaussian widths.
keep_residuals	Logical. Retain spectral residuals. Default FALSE.

Value

A list with class "hsi_chromophore_fit":

coefficients Named matrices of concentration-pathlength products (mol/L * cm).

concentrations Named matrices in mol/L when pathlength is supplied; otherwise NULL.

sto2 Matrix of oxygen saturation = $\text{HbO}_2 / (\text{HbO}_2 + \text{Hb}) * 100$.

total_hb Sum of HbO₂ and Hb concentration-pathlength products.

rmse Matrix of fit residuals.

Examples

```
cube <- hs_example_cube()
fit <- hs_beer_lambert(cube)
range(fit$sto2, na.rm = TRUE)
```

hs_calibrate

Calibrate Raw HSI Data to Reflectance

Description

Applies dark current subtraction and white reference normalization: $R(x, y, \lambda) = (\text{raw} - \text{dark}) / (\text{white} - \text{dark})$. Values are clamped to $[\emptyset, 1]$ unless `clamp = FALSE`.

Usage

```
hs_calibrate(cube, dark, white, clamp = TRUE, saturation = NULL)
```

Arguments

cube	An hsi_cube object with raw data.
dark	An hsi_cube object or 3D array representing the dark reference (lens cap measurement).
white	An hsi_cube object or 3D array representing the white reference (Spectralon panel measurement).
clamp	Logical. Clamp output to $[\emptyset, 1]$. Default TRUE.
saturation	Optional finite saturation threshold in raw data units. Defaults to <code>metadata\$saturation_value</code> when present; otherwise saturation remains an unresolved QC check.

Value

An [hsi_cube](#) object with reflectance values.

Examples

```
cube <- hs_simulate_cube(rows = 10, cols = 10, noise_sd = 0)
dark <- hsi_cube(array(0.01, dim(cube$data)), cube$wavelengths)
white <- hsi_cube(array(0.95, dim(cube$data)), cube$wavelengths)
cal <- hs_calibrate(cube, dark, white)
range(cal$data)
```

hs_chromophore_data *Get Chromophore Extinction Coefficient Spectra*

Description

Returns a pinned tabulation of decadic molar extinction coefficients for hemoglobin, compiled by Prahl and distributed with MNE-Python v1.10.2. Synthetic shapes are available separately for simulation only.

Usage

```
hs_chromophore_data(
  chromophore = c("HbO2", "Hb"),
  wavelength_range = c(400, 1000),
  source = c("reference", "synthetic")
)
```

Arguments

chromophore	Character vector. One or more of: "HbO2" (oxyhemoglobin), "Hb" (deoxyhemoglobin), "water", "melanin", "metHb" (methemoglobin). Default c("HbO2", "Hb").
wavelength_range	Numeric vector of length 2. Wavelength range in nm. Default c(400, 1000).
source	"reference" (default) supports HbO2 and Hb; "synthetic" returns the legacy analytical shapes, unsuitable for quantitative fitting.

Value

A [tibble::tibble](#) with columns wavelength (nm) and one column per requested chromophore (extinction coefficient in $\text{cm}^{-1} / (\text{mol/L})$).

Examples

```
hb_data <- hs_chromophore_data()
head(hb_data)
```

hs_classify_rf*Random Forest Pixel Classification*

Description

Trains a random forest on labeled pixels and classifies all pixels. Requires the `ranger` package.

Usage

```
hs_classify_rf(
  cube,
  training_labels,
  training_mask = NULL,
  num.trees = 500L,
  mtry = NULL,
  ...
)
```

Arguments

<code>cube</code>	An hsi_cube object.
<code>training_labels</code>	Character or factor matrix with class labels. NA for unlabeled pixels.
<code>training_mask</code>	Logical matrix. Default NULL.
<code>num.trees</code>	Integer. Number of trees. Default 500.
<code>mtry</code>	Integer. Variables per split. Default NULL (auto).
<code>...</code>	Additional arguments passed to ranger::ranger() .

Value

A list with class "hsi_classification" containing `class_map`.

Examples

```
# Requires ranger package
cube <- hs_example_cube()
labels <- matrix(NA_character_, 30, 30)
labels[1:10, 1:10] <- "class_a"
labels[20:30, 20:30] <- "class_b"
if (requireNamespace("ranger", quietly = TRUE)) {
  result <- hs_classify_rf(cube, labels)
}
```

hs_classify_svm	<i>SVM Pixel Classification</i>
-----------------	---------------------------------

Description

Trains a Support Vector Machine on labeled pixels and classifies all pixels. Requires the `e1071` package.

Usage

```
hs_classify_svm(
  cube,
  training_labels,
  training_mask = NULL,
  kernel = "radial",
  cost = 10,
  gamma = NULL,
  ...
)
```

Arguments

<code>cube</code>	An hsi_cube object.
<code>training_labels</code>	Character or factor matrix (same spatial dims as <code>cube</code>) with class labels. NA for unlabeled pixels.
<code>training_mask</code>	Logical matrix. Alternative to NA in labels. Default NULL.
<code>kernel</code>	Character. SVM kernel: "radial" (default), "linear", "polynomial".
<code>cost</code>	Numeric. Cost parameter. Default 10.
<code>gamma</code>	Numeric. Gamma parameter. Default NULL (auto).
<code>...</code>	Additional arguments passed to e1071::svm() .

Value

A list with class "hsi_classification" containing `class_map`.

Examples

```
# Requires e1071 package
cube <- hs_example_cube()
labels <- matrix(NA_character_, 30, 30)
labels[1:10, 1:10] <- "class_a"
labels[20:30, 20:30] <- "class_b"
if (requireNamespace("e1071", quietly = TRUE)) {
  result <- hs_classify_svm(cube, labels)
}
```

hs_clinical_indices	<i>Compute All Available Clinical Indices</i>
---------------------	---

Description

Convenience function that computes StO₂, NPI, THI, and TWI (if wavelength range permits) and returns them as a named list of matrices.

Usage

```
hs_clinical_indices(cube)
```

Arguments

cube	An hsi_cube object with reflectance data.
------	---

Value

A named list of numeric matrices: sto2, np_i, thi, twi (TWI may be NA matrix if wavelengths are insufficient).

Examples

```
cube <- hs_example_cube()
indices <- hs_clinical_indices(cube)
names(indices)
```

hs_continuum_removal	<i>Continuum Removal</i>
----------------------	--------------------------

Description

Removes the spectral continuum (convex hull) from each spectrum. Useful for enhancing absorption features and normalizing baseline variations.

Usage

```
hs_continuum_removal(cube, method = c("division", "subtraction"))
```

Arguments

cube	An hsi_cube object.
method	Character. "division" (default) divides by the continuum, "subtraction" subtracts the continuum.

Value

An [hsi_cube](#) object with continuum-removed spectra.

Examples

```
cube <- hs_example_cube()
cr <- hs_continuum_removal(cube, method = "division")
```

hs_dark_correct	<i>Apply Dark Current Correction</i>
-----------------	--------------------------------------

Description

Subtracts a dark reference from the cube data.

Usage

```
hs_dark_correct(cube, dark)
```

Arguments

cube	An hsi_cube object.
dark	An hsi_cube object or 3D array representing the dark reference.

Value

An [hsi_cube](#) object with dark-corrected values.

Examples

```
cube <- hs_simulate_cube(rows = 10, cols = 10, noise_sd = 0)
dark <- hsi_cube(array(0.01, dim(cube$data)), cube$wavelengths)
corrected <- hs_dark_correct(cube, dark)
```

hs_derivative	<i>Spectral Derivative</i>
---------------	----------------------------

Description

Computes the spectral derivative using Savitzky-Golay differentiation. Shorthand for [hs_smooth\(\)](#) with `deriv > 0`.

Usage

```
hs_derivative(cube, order = 1L, window = 5L)
```

Arguments

cube	An hsi_cube object.
order	Integer. Derivative order. Default 1.
window	Integer (odd). SG window size. Default 5.

Value

An [hsi_cube](#) object with derivative spectra.

Examples

```
cube <- hs_example_cube()
d1 <- hs_derivative(cube, order = 1)
```

hs_endmembers	<i>Extract Endmember Spectra from a Cube</i>
---------------	--

Description

Extracts endmember spectra from specified pixel locations.

Usage

```
hs_endmembers(cube, pixels, labels = NULL)
```

Arguments

cube	An hsi_cube object.
pixels	Data.frame or matrix with columns x (col) and y (row).
labels	Character vector of class labels for each pixel. Default NULL (auto-generated).

Value

A named matrix with endmember spectra (rows = endmembers, cols = bands).

Examples

```
cube <- hs_example_cube()
pixels <- data.frame(x = c(5, 25), y = c(5, 25))
em <- hs_endmembers(cube, pixels, labels = c("region_1", "region_2"))
dim(em)
```

hs_example_cube	<i>Get Example HSI Cube</i>
-----------------	-----------------------------

Description

Generates a small pre-configured synthetic cube for quick examples and documentation. The cube simulates a 30x30 pixel tissue scene with four regions of different synthetic spectra. Region names do not identify tissue diagnoses, and the example does not establish quantitative accuracy.

Usage

```
hs_example_cube()
```

Value

An [hsi_cube](#) object (30 x 30 x 61 bands, 430-910 nm).

Examples

```
cube <- hs_example_cube()
print(cube)
dim(cube)
```

hs_example_files	<i>Write Example ENVI Files to a Temporary Directory</i>
------------------	--

Description

Writes a minimal ENVI header + binary pair for testing I/O functions.

Usage

```
hs_example_files(dir = tempdir())
```

Arguments

dir Character. Directory to write to. Default `tempdir()`.

Value

Character. Path to the written .hdr file (invisibly).

Examples

```
hdr_path <- hs_example_files()
file.exists(hdr_path)
```

hs_export_png	<i>Export a Single-Band Image as PNG</i>
---------------	--

Description

Exports a single band or index map from an HSI cube as a PNG image.

Usage

```
hs_export_png(
  data,
  path,
  palette = "viridis",
  range = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data Numeric matrix to export (rows x cols).

path Character. Output file path (should end in .png).

palette Character. Color palette: "viridis", "magma", "inferno", "plasma", "grey". Default "viridis".

range Numeric vector of length 2. Data range for color mapping. Default NULL (auto from data range).

width Integer. Image width in pixels. Default NULL (matches data cols).

height Integer. Image height in pixels. Default NULL (matches data rows).

Value

Invisible path to the written file.

Examples

```
cube <- hs_example_cube()
mat <- cube$data[, , 30]
path <- file.path(tempdir(), "band30.png")
hs_export_png(mat, path)
```

hs_fix_bad_pixels	<i>Detect and Correct Bad Pixels</i>
-------------------	--------------------------------------

Description

Identifies dead/hot pixels by statistical deviation from spatial neighborhood median/MAD and replaces with neighborhood interpolation. Detection uses the immutable image; replacement excludes every detected defect. Defects occupying most of a neighborhood may not be distinguishable.

Usage

```
hs_fix_bad_pixels(cube, threshold = 3, method = "median")
```

Arguments

cube	An hsi_cube object.
threshold	Numeric. Robust standard deviation (scaled MAD) threshold. Default 3.
method	Character. Replacement method: "mean" or "median" of spatial neighbors. Default "median".

Value

An [hsi_cube](#) object with corrected pixels.

Examples

```
cube <- hs_example_cube()
# Introduce a hot pixel
cube$data[15, 15, ] <- 999
fixed <- hs_fix_bad_pixels(cube, threshold = 3)
```

hs_group_summary	<i>Summarize Repeated ROI Measurements with Group-Level Uncertainty</i>
------------------	---

Description

First averages observations within each independent group, then computes an equally weighted mean across groups. Bootstrap samples resample whole groups. Pixel or repeated-recording counts do not inflate the independent sample size.

Usage

```
hs_group_summary(
  data,
  value = "value",
  group = "subject_id",
  bootstrap = 1000L,
  seed = 1L
)
```

Arguments

data	Data frame containing a numeric outcome and a group identifier.
value	Column name containing the outcome.
group	Column name identifying independent subjects or sampling units.
bootstrap	Number of bootstrap samples. Default 1000.
seed	Random seed. Default 1.

Value

List containing group means, independent group count, overall estimate, percentile interval and number of excluded nonfinite observations.

hs_grouped_evaluate	<i>Evaluate Classification with Entire Groups Held Out</i>
---------------------	--

Description

Splits by subject (or another independent sampling unit), fits preprocessing only on training spectra, and predicts held-out groups. Hyperparameters must be chosen before evaluation; this function does not tune on validation data. Group accuracy is conditional on valid predictions; prediction coverage and unevaluated group counts are reported separately. Per-class recall treats withheld predictions as missed labels. Accuracy intervals resample whole groups, not pixels. These intervals summarize out-of-fold predictions and do not include uncertainty from refitting models.

Usage

```

hs_grouped_evaluate(
  spectra,
  wavelengths,
  labels,
  groups,
  folds = NULL,
  n_folds = 5L,
  method = c("svm", "rf"),
  recipe = hs_recipe(),
  model_args = list(),
  seed = 1L,
  bootstrap = 1000L
)

```

Arguments

spectra	Numeric matrix, observations in rows and wavelengths in columns.
wavelengths	Numeric band centers in nm.
labels	Character class labels for each observation.
groups	Independent group identifiers, usually subject identifiers.
folds	Optional named integer vector mapping every group to a fold.
n_folds	Number of folds when no mapping is supplied. Default 5.
method	Classifier, "svm" or "rf".
recipe	Processing recipe. Learned references are fitted inside each fold.
model_args	Named list of fixed classifier parameters.
seed	Random seed for fold assignment and model fitting.
bootstrap	Number of group bootstrap samples for the accuracy interval.

Value

List of out-of-fold predictions, group metrics, confusion matrix, group-mean accuracy and interval, fold assignments and fitted recipes.

hs_mnf

Minimum Noise Fraction Transform

Description

Orders components by signal-to-noise ratio rather than variance. Estimates noise covariance from spatial first-differences.

Usage

```
hs_mnf(cube, n_components = 5L)
```

Arguments

cube An [hsi_cube](#) object.

n_components Integer. Number of components. Default 5.

Value

A list with class "hsi_mnf" (same structure as [hs_pca\(\)](#)).

Examples

```
cube <- hs_example_cube()
mnf <- hs_mnf(cube, n_components = 3)
dim(mnf$scores)
```

hs_msc

Multiplicative Scatter Correction

Description

Corrects spectra for multiplicative and additive scatter effects by regressing each spectrum against a reference spectrum (default: mean spectrum).

Usage

```
hs_msc(cube, reference = NULL)
```

Arguments

cube An [hsi_cube](#) object.

reference Numeric vector. Reference spectrum. Default NULL = mean spectrum of the cube.

Value

An [hsi_cube](#) object with MSC-corrected spectra.

Examples

```
cube <- hs_example_cube()
msc_cube <- hs_msc(cube)
```

hs_ndi	<i>Compute Normalized Difference Index (General Purpose)</i>
--------	--

Description

$NDI = (R_{band1} - R_{band2}) / (R_{band1} + R_{band2})$. A flexible building block for any two-band ratio index.

Usage

```
hs_ndi(cube, band1, band2)
```

Arguments

cube	An hsi_cube object.
band1	Numeric. Center wavelength or range <code>c(min, max)</code> for first band.
band2	Numeric. Center wavelength or range <code>c(min, max)</code> for second band.

Value

A numeric matrix, in $[-1, 1]$ for nonnegative input spectra. Zero denominators are NA; transformed signed spectra can exceed these bounds.

Examples

```
cube <- hs_example_cube()
ndi <- hs_ndi(cube, band1 = 540, band2 = 660)
range(ndi, na.rm = TRUE)
```

hs_npi	<i>Compute Near-Infrared Perfusion Index (NPI)</i>
--------	--

Description

Legacy name for a dimensionless NIR band ratio. It is not a validated perfusion measurement or an implementation of a vendor index.

Usage

```
hs_npi(cube, band1 = c(655, 735), band2 = c(825, 910))
```

Arguments

cube	An hsi_cube object with reflectance data.
band1	Numeric vector of length 2. Default <code>c(655, 735)</code> .
band2	Numeric vector of length 2. Default <code>c(825, 910)</code> .

Value

A numeric matrix of unscaled dimensionless band ratios.

Examples

```
cube <- hs_example_cube()
npi <- hs_npi(cube)
```

hs_pca

PCA on Hyperspectral Cube

Description

Performs Principal Component Analysis on the spectral dimension of an HSI cube. Reduces the spectral bands to a smaller number of orthogonal components ordered by explained variance.

Usage

```
hs_pca(cube, n_components = 5L, center = TRUE, scale = FALSE)
```

Arguments

cube	An hsi_cube object.
n_components	Integer. Number of components to retain. Default 5.
center	Logical. Center bands before PCA. Default TRUE.
scale	Logical. Scale bands to unit variance. Default FALSE.

Value

A list with class "hsi_pca":

scores 3D array (rows x cols x n_components) of component score maps.

loadings Matrix (bands x n_components) of spectral loadings.

variance_explained Numeric vector of proportion of variance per component.

center Centering vector used (or FALSE).

scale Scaling vector used (or FALSE).

wavelengths Wavelength vector from input cube.

Examples

```
cube <- hs_example_cube()
pca <- hs_pca(cube, n_components = 3)
pca$variance_explained
```

hs_plot_clinical	<i>Research HSI Panel Display</i>
------------------	-----------------------------------

Description

Displays RGB, fitted hemoglobin fraction and dimensionless band ratios. These research outputs are not equivalent to vendor clinical indices.

Usage

```
hs_plot_clinical(  
  cube,  
  indices = c("sto2", "npi", "thi"),  
  mask_background = TRUE,  
  threshold = 0.05,  
  ncol = NULL  
)
```

Arguments

cube	An hsi_cube object with reflectance data.
indices	Character vector. Which indices to display. Default <code>c("sto2", "npi", "thi")</code> . TWI is included only if wavelengths permit.
mask_background	Logical. Mask non-tissue pixels. Default TRUE.
threshold	Numeric. Masking threshold on mean reflectance. Default 0.05.
ncol	Integer. Number of panel columns. Default NULL (auto).

Value

A patchwork composite [ggplot2::ggplot](#) object.

Examples

```
cube <- hs_example_cube()  
hs_plot_clinical(cube)
```

hs_plot_image	<i>Plot Single-Band Spatial Image</i>
---------------	---------------------------------------

Description

Displays a single spectral band as a spatial image using a pseudocolor palette.

Usage

```
hs_plot_image(cube, wavelength = NULL, band = NULL, palette = "viridis")
```

Arguments

cube	An hsi_cube object.
wavelength	Numeric. Center wavelength to display (nearest band selected). Default NULL.
band	Integer. Band index. Alternative to wavelength. Default NULL.
palette	Character. Color palette name. Default "viridis".

Value

A [ggplot2::ggplot](#) object using [ggplot2::geom_raster\(\)](#).

Examples

```
cube <- hs_example_cube()
hs_plot_image(cube, wavelength = 550)
```

hs_plot_index	<i>Plot Index Map with Clinical Color Scale</i>
---------------	---

Description

Displays a single tissue index as a pseudocolor spatial map with color scales selected for research visualization.

Usage

```
hs_plot_index(
  index_matrix,
  title = "",
  palette = "sto2",
  range = NULL,
  mask = NULL
)
```

Arguments

index_matrix	Numeric matrix (rows x cols) from an index function.
title	Character. Map title (e.g., "StO2 (%)"). Default "".
palette	Character. "sto2" (blue-red diverging), "perfusion" (viridis), "hemoglobin" (magma), "water" (mako). Default "sto2".
range	Numeric vector of length 2. Display range. Default NULL (data range).
mask	Logical matrix. Pixels to mask (FALSE = masked). Default NULL.

Value

A `ggplot2::ggplot` object.

Examples

```
cube <- hs_example_cube()
sto2 <- hs_sto2(cube)
hs_plot_index(sto2, title = "Fitted Hb fraction (%)", palette = "sto2")
```

hs_plot_rgb

Synthesize RGB Image from Spectral Cube

Description

Creates a pseudo-color RGB composite by mapping three wavelength bands to the red, green, and blue channels.

Usage

```
hs_plot_rgb(cube, r = 640, g = 550, b = 460, stretch = "linear")
```

Arguments

cube	An hsi_cube object.
r, g, b	Numeric. Center wavelengths for R, G, B channels. Defaults: r=640, g=550, b=460.
stretch	Character. Histogram stretch: "linear" (default), "none".

Value

A `ggplot2::ggplot` object.

Examples

```
cube <- hs_example_cube()
hs_plot_rgb(cube)
```

hs_plot_spectra	<i>Plot Spectral Profiles</i>
-----------------	-------------------------------

Description

Displays spectral profiles from an HSI cube. Can show mean spectrum, random pixel spectra, or spectra from specific pixel locations.

Usage

```
hs_plot_spectra(cube, pixels = "mean", n = 100L, show_sd = TRUE)
```

Arguments

cube	An hsi_cube object.
pixels	Character or data.frame. "mean" for spatial mean, "random" for random pixel sample, or a data.frame with columns x and y. Default "mean".
n	Integer. Number of random spectra if pixels = "random". Default 100.
show_sd	Logical. Show mean +/- SD ribbon. Default TRUE when pixels = "mean".

Value

A [ggplot2::ggplot](#) object.

Examples

```
cube <- hs_example_cube()
hs_plot_spectra(cube)
```

hs_predict	<i>Predict Classes in a New Cube Using a Fitted Classifier</i>
------------	--

Description

Predict Classes in a New Cube Using a Fitted Classifier

Usage

```
hs_predict(model, cube)
```

Arguments

model	Classification result from hs_classify_svm() or hs_classify_rf() .
cube	New hsi_cube on the same wavelength grid and preprocessing domain.

Value

Character matrix of predicted classes; invalid pixels are NA.

hs_process	<i>Execute a Processing Recipe</i>
------------	------------------------------------

Description

Execute a Processing Recipe

Usage

```
hs_process(cube, recipe = hs_recipe(), learn = TRUE)
```

Arguments

cube	An hsi_cube object.
recipe	A recipe from hs_recipe() .
learn	Logical. Allow fitting an unspecified MSC reference on this cube. Set FALSE for validation or prediction data.

Value

List containing the processed cube and reusable fitted recipe.

hs_read_cube	<i>Read a Hyperspectral Data Cube from Any Supported Format</i>
--------------	---

Description

Auto-detects format from file extension and dispatches to the appropriate reader. Supported formats: ENVI (.hdr), TIFF (.tif/.tiff), Cubert (.cu3s).

Usage

```
hs_read_cube(path, ...)
```

Arguments

path	Path to the hyperspectral data file.
...	Additional arguments passed to the format-specific reader.

Value

An [hsi_cube](#) object.

Examples

```
hdr_path <- hs_example_files()
cube <- hs_read_cube(hdr_path, verbose = FALSE)
dim(cube)
```

hs_read_cubert	<i>Read a Cubert .cu3s Session File</i>
----------------	---

Description

Reads Cubert session files using the `cuvis.r` package, an R wrapper for the Cubert CUVIS C SDK (see <https://github.com/CTTIR/cuvis.r>).

Usage

```
hs_read_cubert(
  path,
  index = 1L,
  mode = c("reflectance", "spectral_radiance", "dark_subtract", "raw"),
  settings_dir = NULL,
  verbose = TRUE
)
```

Arguments

<code>path</code>	Path to .cu3s Cubert session file.
<code>index</code>	Integer. Measurement index within session (1-based). Default 1.
<code>mode</code>	Character. Processing mode: "reflectance" (default), "spectral_radiance", "dark_subtract", "raw".
<code>settings_dir</code>	Character or NULL. Path to the CUVIS settings directory (e.g., "C:/ProgramData/cuvis"). If NULL (default), uses the CUVIS_SETTINGS environment variable or a temporary directory.
<code>verbose</code>	Logical. Print progress messages. Default TRUE.

Details

The processing context automatically loads dark/white references embedded in the session file, so separate reference files are typically not needed.

Cubert reflectance values are stored as uint16 scaled by 10000 (i.e. 10000 = 100% reflectance). This function automatically converts to fractional reflectance (0-1) when mode = "reflectance".

Value

An `hsi_cube` object. Reflectance data is in 0-1 range.

Examples

```
# Requires cuvis.r package and Cubert CUVIS SDK
# cube <- hs_read_cubert("path/to/session.cu3s")
```

hs_read_envi

*Read an ENVI Hyperspectral Image File***Description**

Reads ENVI format hyperspectral data consisting of a .hdr header file and a binary data file. Supports BSQ, BIL, and BIP interleave formats.

Usage

```
hs_read_envi(
  path,
  backend = "auto",
  bands = NULL,
  extent = NULL,
  verbose = TRUE
)
```

Arguments

path	Path to the ENVI header file (.hdr) or binary file.
backend	Character. "auto" (default) uses the deterministic built-in reader. "builtin" forces pure-R reader. "terra" forces terra (errors if not installed).
bands	Integer vector of band indices to read. Default NULL = all.
extent	Numeric vector c(row_start, row_end, col_start, col_end) for spatial subset. Default NULL = full image.
verbose	Logical. Print progress messages. Default TRUE.

Value

An [hsi_cube](#) object.

Examples

```
hdr_path <- hs_example_files()
cube <- hs_read_envi(hdr_path, verbose = FALSE)
dim(cube)
```

hs_read_tiff	<i>Read a Multi-Channel TIFF File</i>
--------------	---------------------------------------

Description

Reads a multi-band TIFF file as a hyperspectral cube. Requires the terra package for TIFF reading.

Usage

```
hs_read_tiff(path, wavelengths, fwhm = NULL, verbose = TRUE)
```

Arguments

path	Path to multi-channel TIFF file.
wavelengths	Numeric vector of wavelengths (required for TIFF files as they lack spectral metadata).
fwhm	Numeric vector of FWHM values. Default NULL.
verbose	Logical. Print progress messages. Default TRUE.

Value

An [hsi_cube](#) object.

Examples

```
# Requires terra package
# cube <- hs_read_tiff("path/to/image.tif", wavelengths = seq(430, 910, by = 8))
```

hs_read_tivita	<i>Read a TIVITA Suite Recording</i>
----------------	--------------------------------------

Description

Reads a Diaspective Vision TIVITA *_SpecCube.dat recording through the **tivis.r** package and returns it as an [hsi_cube](#).

Usage

```
hs_read_tivita(path, bands = NULL, verbose = TRUE)
```

Arguments

path	Path to a *_SpecCube.dat file.
bands	Optional integer vector of band indices to read. NULL (default) reads all bands.
verbose	Logical. Print progress messages. Default TRUE.

Details

Unlike the Cubert reader, this needs no vendor SDK: the TIVITA container is a plain binary format and **tivis.r** is pure R.

Acquisition metadata from the Suite's *_meta.log is attached to the cube when present, along with the paths of the parameter images the Suite exported beside the recording (RGB rendering, oxygenation, NIR perfusion, THI, TWI). Those are vendor outputs for comparison, not independent physiological ground truth or equivalents of this package's research indices.

Value

An [hsi_cube](#) object. Values are calibrated reflectance, normally within $[0, 1]$ but not clamped, since specular regions legitimately exceed it.

See Also

[hs_read_cube\(\)](#) for extension-based dispatch, [hs_read_cubert\(\)](#) for Cubert session files.

Examples

```
# Requires the tivis.r package
# cube <- hs_read_tivita("2019_11_25_13_29_24_SpecCube.dat")
```

hs_recipe

Define a Reproducible Processing Recipe

Description

Recipes contain ordered operations and their explicit parameters. A fitted MSC reference is stored in the returned recipe, allowing reuse on held-out data.

Usage

```
hs_recipe(steps = list(), seed = 1L)
```

Arguments

steps	List of steps, each with method and an optional args list. Methods: smooth, snv, msc, derivative, absorbance, resample, continuum_removal, calibrate, dark_correct, white_normalize, fix_bad_pixels.
seed	Integer random seed recorded with the recipe.

Value

An hsi_recipe object.

hs_resample

Resample Spectra to New Wavelength Grid

Description

Interpolates the spectral data to a new set of wavelength positions.

Usage

```
hs_resample(cube, target_wavelengths, method = "linear")
```

Arguments

cube An [hsi_cube](#) object.

target_wavelengths Numeric vector. Target wavelength positions in nm.

method Character. Interpolation method: "linear" (default), "spline".

Value

An [hsi_cube](#) object with resampled spectra.

Examples

```
cube <- hs_example_cube()
resampled <- hs_resample(cube, target_wavelengths = seq(450, 900, by = 10))
length(resampled$wavelengths)
```

hs_roi_polygon

Define Polygon ROI

Description

Creates a polygon region of interest using point-in-polygon testing.

Usage

```
hs_roi_polygon(cube, vertices)
```

Arguments

cube An [hsi_cube](#) object.

vertices Matrix with columns x and y (polygon vertices in pixel coordinates).

Value

A list with class "hsi_roi" containing the mask.

Examples

```
cube <- hs_example_cube()
verts <- matrix(c(5, 5, 15, 15, 5, 20, 5, 20), ncol = 2,
               dimnames = list(NULL, c("x", "y")))
roi <- hs_roi_polygon(cube, verts)
```

hs_roi_rect	<i>Define Rectangular ROI</i>
-------------	-------------------------------

Description

Creates a rectangular region of interest on an HSI cube.

Usage

```
hs_roi_rect(cube, x_range, y_range)
```

Arguments

cube An [hsi_cube](#) object.

x_range Numeric vector of length 2 (column indices, min and max).

y_range Numeric vector of length 2 (row indices, min and max).

Value

A list with class "hsi_roi" containing the mask and bounds.

Examples

```
cube <- hs_example_cube()
roi <- hs_roi_rect(cube, x_range = c(5, 15), y_range = c(5, 15))
sum(roi$mask)
```

hs_roi_stats	<i>Compute ROI Statistics</i>
--------------	-------------------------------

Description

Computes per-band spectral statistics within a region of interest.

Usage

```
hs_roi_stats(cube, roi)
```

Arguments

cube	An hsi_cube object.
roi	An hsi_roi object (from hs_roi_rect() or hs_roi_polygon()), or a logical mask matrix.

Value

A [tibble::tibble](#) with columns wavelength, mean, sd, median, min, max, n_pixels, n_roi_pixels, valid_fraction, and n_repaired_pixels. Counts and fractions are per band; SD describes spatial variation, not uncertainty of an independent-group mean.

Examples

```
cube <- hs_example_cube()
roi <- hs_roi_rect(cube, x_range = c(5, 15), y_range = c(5, 15))
stats <- hs_roi_stats(cube, roi)
head(stats)
```

hs_run_app	<i>Launch Interactive Hyperspectral Image Explorer</i>
------------	--

Description

Opens a Shiny application for interactive exploration of hyperspectral cubes. Provides band selection, spectral profiling, research index mapping, cancellable background processing and analysis, and export tools across six tabs.

Usage

```
hs_run_app(
  cube = NULL,
  port = NULL,
  launch.browser = TRUE,
  max_upload_mb = 256
)
```

Arguments

cube	An hsi_cube object to explore. If NULL (default), the app starts with the example cube.
port	Integer. Port for Shiny server. Default NULL (auto).
launch.browser	Logical. Open in browser. Default TRUE.
max_upload_mb	Positive upload limit in MiB. Default 256. Uploaded cubes and background workers must also fit available memory.

Value

Invisible NULL. Launches a Shiny application.

Examples

```
cube <- hs_example_cube()
if (interactive()) {
  hs_run_app(cube)
}
```

hs_sam

Spectral Angle Mapper Classification

Description

Classifies each pixel by computing the spectral angle to a set of reference endmember spectra and assigning the class with the smallest angle.

Usage

```
hs_sam(cube, endmembers, threshold = 0.1)
```

Arguments

cube	An hsi_cube object.
endmembers	Named matrix or data.frame. Rows = endmembers, cols = bands. Row names become class labels.
threshold	Numeric. Maximum angle (radians) for assignment. Pixels exceeding this are classified as "unclassified". Default 0.1.

Value

A list with class "hsi_classification":

class_map Character matrix (rows x cols) of class labels.

angle_map 3D array of angles to each endmember.

endmembers The endmember matrix used.

Examples

```
cube <- hs_example_cube()
# Define endmembers from specific pixels
em <- rbind(
  healthy = cube$data[5, 5, ],
  ischemic = cube$data[25, 5, ]
)
result <- hs_sam(cube, em, threshold = 0.5)
table(result$class_map)
```

hs_simulate_cube	<i>Generate a Synthetic HSI Cube for Testing and Examples</i>
------------------	---

Description

Creates a realistic synthetic tissue hyperspectral cube with known ground truth. Simulates a tissue scene with regions of varying oxygenation, hemoglobin concentration, and a background (non-tissue) region.

Usage

```
hs_simulate_cube(
  rows = 50L,
  cols = 50L,
  wavelengths = seq(430, 910, by = 8),
  n_regions = 4L,
  sto2_range = c(0.3, 0.95),
  noise_sd = 0.01,
  seed = 42L
)
```

Arguments

rows	Integer. Spatial rows. Default 50.
cols	Integer. Spatial columns. Default 50.
wavelengths	Numeric vector. Wavelength grid in nm. Default: seq(430, 910, by = 8) matching Cubert Ultras X MR.
n_regions	Integer. Number of distinct tissue regions. Default 4.
sto2_range	Numeric vector of length 2. Synthetic mixture fractions (0 to 1), not validated tissue oxygenation measurements. Default c(0.3, 0.95).
noise_sd	Numeric. Gaussian noise standard deviation. Default 0.01.
seed	Integer. Random seed for reproducibility. Default 42.

Value

An [hsi_cube](#) object with known synthetic region_map. Legacy sto2_ground_truth metadata records simulator parameters only; the generated spectra are not an independent reference for quantitative tissue validation.

Examples

```
cube <- hs_simulate_cube(rows = 20, cols = 20)
dim(cube)
```

hs_smooth

Savitzky-Golay Spectral Smoothing

Description

Applies a Savitzky-Golay filter along the spectral dimension. Optionally computes derivatives per nanometer on a uniform wavelength grid. Only full-window center bands are returned. Invalid spectra remain missing.

Usage

```
hs_smooth(
  cube,
  window = 5L,
  poly = 2L,
  deriv = 0L,
  backend = c("builtin", "prospectr", "signal")
)
```

Arguments

cube	An hsi_cube object.
window	Integer (odd). Filter window size in bands. Default 5.
poly	Integer. Polynomial order. Must be less than window. Default 2.
deriv	Integer. Derivative order (0 = smoothing only, 1 = first derivative, 2 = second derivative). Default 0.
backend	Character. Explicit backend: "builtin" (default), "prospectr", or "signal". All use the same band support and units.

Value

An [hsi_cube](#) object with smoothed/differentiated spectra.

Examples

```
cube <- hs_example_cube()
smoothed <- hs_smooth(cube, window = 5, poly = 2)
dim(smoothed)
```

hs_snv

Standard Normal Variate Correction

Description

Normalizes each spectrum to zero mean and unit variance. Useful for reducing multiplicative scatter effects and baseline variation.

Usage

```
hs_snv(cube)
```

Arguments

cube An [hsi_cube](#) object.

Value

An [hsi_cube](#) object with SNV-corrected spectra.

Examples

```
cube <- hs_example_cube()
snv_cube <- hs_snv(cube)
```

hs_sto2

Estimate the Oxygenated Hemoglobin Fraction for Research

Description

Returns the fitted HbO2 fraction from [hs_beer_lambert\(\)](#). This is a research model estimate and requires empirical validation for tissue use.

Usage

```
hs_sto2(
  cube,
  band1 = c(500, 650),
  band2 = c(700, 815),
  method = "beer_lambert"
)
```

Arguments

cube	An hsi_cube object with reflectance data.
band1, band2	Retired ratio-method arguments. Supplying either raises a migration error; use hs_band_ratio() or the fitting range in hs_beer_lambert() .
method	"beer_lambert" (default). The former "ratio" method is rejected; use hs_band_ratio() for a relative band ratio.

Value

A numeric matrix (rows x cols) with values 0-100 representing estimated tissue oxygen saturation percentage. Returns NA for masked pixels.

Examples

```
cube <- hs_example_cube()
sto2 <- hs_sto2(cube)
range(sto2, na.rm = TRUE)
```

hs_thi

Compute Tissue Hemoglobin Index (THI)

Description

Legacy name for a dimensionless reference/Hb-band reflectance ratio. It is not a hemoglobin concentration measurement.

Usage

```
hs_thi(cube, band1 = c(530, 590), band2 = c(785, 825))
```

Arguments

cube	An hsi_cube object.
band1	Numeric vector of length 2. Default c(530, 590) (Hb Q-bands).
band2	Numeric vector of length 2. Default c(785, 825) (reference).

Value

A numeric matrix of unscaled dimensionless band ratios.

Examples

```
cube <- hs_example_cube()
thi <- hs_thi(cube)
```

hs_transform	<i>Apply a Fitted Spectral Transform to Another Cube</i>
--------------	--

Description

Applies saved PCA/MNF centering, scaling and loadings without refitting.

Usage

```
hs_transform(model, cube)
```

Arguments

model	Result from hs_pca() or hs_mnf() .
cube	New hsi_cube with the same wavelength grid.

Value

Score array with spatial dimensions of the new cube; invalid pixels are NA.

hs_twi	<i>Compute Tissue Water Index (TWI)</i>
--------	---

Description

Legacy name for a dimensionless NIR reflectance ratio. The default 830-910 nm bands do not include the 970 nm water peak and do not measure water content.

Usage

```
hs_twi(cube, numerator = c(880, 910), denominator = c(830, 870))
```

Arguments

cube	An hsi_cube object.
numerator	Numeric vector of length 2. Default c(880, 910) (adapted for Cubert range).
denominator	Numeric vector of length 2. Default c(830, 870).

Value

A numeric matrix of unscaled dimensionless ratios, or an NA matrix with a warning if required wavelengths are unavailable.

Examples

```
cube <- hs_example_cube()
twi <- hs_twi(cube)
```

hs_umap

*UMAP Embedding of Spectral Data***Description**

Computes a UMAP (Uniform Manifold Approximation and Projection) embedding of the spectral data. Requires the uwot package.

Usage

```
hs_umap(
  cube,
  n_components = 2L,
  n_neighbors = 15L,
  min_dist = 0.1,
  pca_pre = 20L,
  seed = 1L
)
```

Arguments

cube	An hsi_cube object.
n_components	Integer. UMAP dimensions. Default 2.
n_neighbors	Integer. Default 15.
min_dist	Numeric. Default 0.1.
pca_pre	Integer or NULL. PCA pre-reduction dimensionality. Default 20.
seed	Integer random seed. Default 1; the caller's random state is restored.

Value

A list with class "hsi_umap" containing embedding matrix and spatial coordinates.

Examples

```
# Requires uwot package
cube <- hs_example_cube()
if (requireNamespace("uwot", quietly = TRUE)) {
  umap_result <- hs_umap(cube, n_components = 2)
}
```

hs_unmix_nnl

*Linear Spectral Unmixing via NNLS***Description**

Solves the linear mixing model per pixel using non-negative least squares.

Usage

```
hs_unmix_nnl(
  cube,
  endmembers,
  sum_to_one = FALSE,
  backend = c("builtin", "nnl"),
  keep_residuals = TRUE,
  chunk_size = 10000L
)
```

Arguments

cube	An hsi_cube object (reflectance or absorbance).
endmembers	Numeric matrix. Columns = endmember spectra. Rows = bands. Column names become abundance map labels.
sum_to_one	Logical. Apply sum-to-one constraint. Default FALSE.
backend	Explicit solver: "builtin" (default) or "nnl".
keep_residuals	Logical. Retain the full residual cube.
chunk_size	Positive integer. Number of pixels per processing block.

Value

A list with class "hsi_unmix":

abundances 3D array (rows x cols x n_endmembers) of abundance maps.

residuals 3D array of reconstruction residuals.

rmse Numeric matrix of per-pixel RMSE.

endmember_names Character vector.

Examples

```
cube <- hs_example_cube()
# Create simple endmembers
em <- cbind(
  tissue = cube$data[5, 5, ],
  background = cube$data[25, 25, ]
)
result <- hs_unmix_nnl(cube, em)
dim(result$abundances)
```

hs_white_normalize	<i>Apply White Reference Normalization</i>
--------------------	--

Description

Normalizes the cube by a white reference, optionally with dark correction.

Usage

```
hs_white_normalize(cube, white, dark = NULL)
```

Arguments

cube	An hsi_cube object.
white	An hsi_cube object or 3D array representing the white reference.
dark	An hsi_cube object or 3D array representing the dark reference. Default NULL (no dark subtraction).

Value

An [hsi_cube](#) object with normalized values.

Examples

```
cube <- hs_simulate_cube(rows = 10, cols = 10, noise_sd = 0)
white <- hsi_cube(array(0.95, dim(cube$data)), cube$wavelengths)
norm <- hs_white_normalize(cube, white)
```

hs_write_envi	<i>Write an HSI Cube to ENVI Format</i>
---------------	---

Description

Writes an [hsi_cube](#) object as an ENVI header (.hdr) and binary (.dat) file pair.

Usage

```
hs_write_envi(cube, path, interleave = "bsq", data_type = 4L, verbose = TRUE)
```

Arguments

cube	An hsi_cube object.
path	Character. Output path without extension. Files .hdr and .dat will be created.
interleave	Character. Data interleave format: "bsq" (default), "bil", or "bip".
data_type	Integer. ENVI data type code. Default 4 (float32).
verbose	Logical. Print progress. Default TRUE.

Value

Invisible character vector of written file paths.

Examples

```
cube <- hs_example_cube()
dir <- tempdir()
paths <- hs_write_envi(cube, file.path(dir, "test_cube"), verbose = FALSE)
file.exists(paths)
```

hs_write_tiff

Write an HSI Cube to Multi-Band TIFF

Description

Writes an [hsi_cube](#) object as a multi-band GeoTIFF file. Requires the terra package.

Usage

```
hs_write_tiff(cube, path, verbose = TRUE)
```

Arguments

cube	An hsi_cube object.
path	Character. Output path (should end in .tif).
verbose	Logical. Print progress. Default TRUE.

Value

Invisible path to the written file.

Examples

```
# Requires terra package
# cube <- hs_example_cube()
# hs_write_tiff(cube, file.path(tempdir(), "test.tif"))
```

hsi_cube	Create an HSI Cube Object
----------	---------------------------

Description

Constructs a hyperspectral imaging data cube as an S3 object. The cube stores a 3D array of spectral data (rows x columns x bands) along with wavelength metadata, spatial mask, and additional metadata.

Usage

```
hsi_cube(data, wavelengths, fwhm = NULL, metadata = list(), mask = NULL)
```

Arguments

data	Numeric 3D array with dimensions (rows, cols, bands).
wavelengths	Numeric vector of band center wavelengths in nanometers. Length must match <code>dim(data)[3]</code> .
fwhm	Numeric vector of full-width-at-half-maximum values per band in nm. Default NULL (unknown). If scalar, recycled to all bands.
metadata	Named list of metadata (camera model, integration time, processing mode, acquisition timestamp, etc.). Default empty list.
mask	Logical matrix matching spatial dimensions. TRUE = valid pixel. Default NULL (all pixels valid).

Value

An `hsi_cube` S3 object (a named list with class "hsi_cube").

Examples

```
# Create a small synthetic cube
data <- array(runif(10 * 10 * 5), dim = c(10, 10, 5))
wavelengths <- c(500, 550, 600, 650, 700)
cube <- hsi_cube(data, wavelengths)
print(cube)
```

```
print.hsi_cube
```

Print an hsi_cube Object

Description

Displays a compact summary of the hyperspectral cube.

Usage

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

Arguments

`x` An [hsi_cube](#) object.

`...` Additional arguments (ignored).

Value

Invisible x.

Examples

```
cube <- hs_example_cube()
print(cube)
```

```
scale_color_wavelength
```

Wavelength-to-Color Scale for Spectral Plots

Description

Maps wavelength values to approximate visible light colors for spectral line coloring. Returns a ggplot2 continuous color scale.

Usage

```
scale_color_wavelength(...)

scale_colour_wavelength(...)
```

Arguments

`...` Additional arguments passed to [ggplot2::scale_color_gradientn\(\)](#).

Value

A [ggplot2::Scale](#) object.

Examples

```
cube <- hs_example_cube()
library(ggplot2)
df <- data.frame(
  wavelength = cube$wavelengths,
  value = cube$data[15, ]
)
ggplot(df, aes(wavelength, value, color = wavelength)) +
  geom_line() +
  scale_color_wavelength()
```

summary.hsi_cube	<i>Summarize an hsi_cube Object</i>
------------------	-------------------------------------

Description

Returns a named list of cube statistics including per-band means and standard deviations.

Usage

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

Arguments

object	An hsi_cube object.
...	Additional arguments (ignored).

Value

A named list with elements dimensions, wavelength_range, n_bands, data_range, band_means, band_sds, n_valid_pixels, and metadata.

Examples

```
cube <- hs_example_cube()
s <- summary(cube)
s$dimensions
```

`theme_hsi`*Minimalist HSI Theme*

Description

A clean ggplot2 theme designed for hyperspectral image display. Uses minimal styling with a white background.

Usage

```
theme_hsi(base_size = 11)
```

Arguments

`base_size` Numeric. Base font size. Default 11.

Value

A `ggplot2::theme` object.

Examples

```
library(ggplot2)
ggplot(data.frame(x = 1:10, y = 1:10), aes(x, y)) +
  geom_point() +
  theme_hsi()
```

Index

[.hsi_cube, 3

as.data.frame.hsi_cube, 4

as.data.frame.hsi_cube(), 5

as_tibble.hsi_cube, 5

autoplot.hsi_cube, 6

dim.hsi_cube, 7

e1071::svm(), 13

ggplot2::geom_raster(), 26

ggplot2::ggplot, 6, 25–28

ggplot2::Scale, 49

ggplot2::scale_color_gradientn(), 48

ggplot2::theme, 50

hs_absorbance, 7

hs_band_ratio, 8

hs_band_ratio(), 41

hs_batch, 8

hs_beer_lambert, 9

hs_beer_lambert(), 40, 41

hs_calibrate, 10

hs_chromophore_data, 11

hs_classify_rf, 12

hs_classify_rf(), 28

hs_classify_svm, 13

hs_classify_svm(), 28

hs_clinical_indices, 14

hs_continuum_removal, 14

hs_dark_correct, 15

hs_derivative, 16

hs_endmembers, 16

hs_example_cube, 17

hs_example_files, 17

hs_export_png, 18

hs_fix_bad_pixels, 19

hs_group_summary, 20

hs_grouped_evaluate, 20

hs_mnf, 21

hs_mnf(), 42

hs_msc, 22

hs_ndi, 23

hs_npi, 23

hs_pca, 24

hs_pca(), 22, 42

hs_plot_clinical, 25

hs_plot_image, 26

hs_plot_index, 26

hs_plot_rgb, 27

hs_plot_spectra, 28

hs_predict, 28

hs_process, 29

hs_read_cube, 29

hs_read_cube(), 33

hs_read_cubert, 30

hs_read_cubert(), 33

hs_read_envi, 31

hs_read_tiff, 32

hs_read_tivita, 32

hs_recipe, 33

hs_recipe(), 9, 29

hs_resample, 34

hs_roi_polygon, 34

hs_roi_polygon(), 36

hs_roi_rect, 35

hs_roi_rect(), 36

hs_roi_stats, 36

hs_run_app, 36

hs_sam, 37

hs_simulate_cube, 38

hs_smooth, 39

hs_smooth(), 16

hs_snv, 40

hs_sto2, 40

hs_thi, 41

hs_transform, 42

hs_twi, 42

hs_umap, 43

hs_unmix_nnlsl, 44
hs_white_normalize, 45
hs_write_envi, 45
hs_write_tiff, 46
hsi_cube, 4–10, 12–17, 19, 22–37, 39–46, 47,
48, 49

print.hsi_cube, 48

ranger::ranger(), 12

scale_color_wavelength, 48
scale_colour_wavelength
 (scale_color_wavelength), 48
summary.hsi_cube, 49

tempdir(), 18
theme_hsi, 50
tibble::as_tibble(), 5
tibble::tibble, 5, 11, 36