Package: hypermodR (via r-universe)

November 4, 2026

Type Package

Title Hyperspectral Modelling and Validation

Version 0.0.0.9001

Description Interpretable and black-box hyperspectral modelling with explicit spectral, spatial and temporal data contracts, grouped validation, reproducible evaluation, optional isolated model runtimes, and an interactive research workbench. Model and instrument qualification is recorded separately from interface availability.

License MIT + file LICENSE

URL https://github.com/CTTIR/hypermodR

BugReports https://github.com/CTTIR/hypermodR/issues

Encoding UTF-8

Depends R (>= 4.3.0)

Imports cli, digest, generics, jsonlite, methods, stats, utils

Suggests zip, brms, cluster, dbscan, e1071, fastICA, glmnet, isotree, kernlab, lme4, MASS, mgcv, ordinal, pls, quadprog, ranger, survival, xgboost, hyperspectR, hyperannotatR, tivis.r, cuvis.r, openssl, bslib, callr, covr, httr2, knitr, lintr, magick, tiff, processx, rmarkdown, shiny, testthat (>= 3.2.0), withr

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

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Config/Needs/website CTTIR/themakR

Config/hypermodR/status development

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

Date/Publication 2026-11-04 18:38:00 UTC

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

RemoteRef HEAD

RemoteSha bfa086ac5117d4907e61711532bc88d98b9b2b48

Contents

augment.hm_fit . . . 4
glance.hm_fit . . . 4
hm_active_learning . . . 5
hm_admit_backend . . . 6
hm_annotation_contract . . . 6
hm_app . . . 7
hm_app_state . . . 7
hm_apply_calibration . . . 8
hm_as_dataset.cuvis_session . . . 8
hm_as_hsi_cube . . . 9
hm_backend_capabilities . . . 10
hm_backend_plan . . . 10
hm_backend_status . . . 11
hm_bake . . . 11
hm_benchmark_protocol . . . 12
hm_bootstrap . . . 13
hm_cache . . . 14
hm_cache_key . . . 14
hm_calibrate . . . 15
hm_cancel_job . . . 16
hm_check_backend . . . 16
hm_compare . . . 17
hm_coordinates . . . 18
hm_dataset . . . 18
hm_dual_stream_mhsi . . . 19
hm_endpoint_contract . . . 19
hm_evaluate . . . 20
hm_example . . . 21
hm_export_annotations . . . 21
hm_export_result . . . 22
hm_extract_native . . . 22
hm_fit . . . 23
hm_from_tivis . . . 25
hm_htc . . . 25
hm_hyperfree . . . 26
hm_hypersigma . . . 26
hm_hypervision . . . 27
hm_install_backend . . . 27
hm_instrument_profiles . . . 28
hm_job_status . . . 28

hm_model_spec . . . 29
hm_models . . . 29
hm_physics_kernel . . . 30
hm_pool . . . 31
hm_portable_sequence . . . 31
hm_predict . . . 32
hm_prediction . . . 33
hm_prep . . . 34
hm_prompt . . . 35
hm_provenance . . . 35
hm_read_block . . . 36
hm_read_run . . . 36
hm_recipe . . . 37
hm_reliability . . . 38
hm_render_hsi . . . 38
hm_resolve_backend . . . 39
hm_resolve_checkpoint . . . 40
hm_resources . . . 40
hm_save_run . . . 41
hm_sequence . . . 42
hm_spectra . . . 43
hm_spectral_library . . . 43
hm_spectralearth . . . 44
hm_spectralgpt . . . 44
hm_split . . . 45
hm_stage_annotations . . . 46
hm_start_backend . . . 47
hm_stop_backend . . . 48
hm_stream . . . 48
hm_stream_cancel . . . 49
hm_stream_push . . . 49
hm_submit_job . . . 50
hm_task . . . 51
hm_temporal_model . . . 51
hm_tivita_profile . . . 52
hm_tune . . . 53
hm_verify_provenance . . . 54
hm_verify_registry . . . 54
hm_vllm_chat . . . 55
hm_with_resources . . . 56
hm_worker_request . . . 57
hm_workflow . . . 57
HMDataset-class . . . 58
HMSequence-class . . . 58
tidy.hm_fit . . . 59

Index **60**

augment.hm_fit *Attach predictions to observation identities*

Description

Attach predictions to observation identities

Usage

```
## S3 method for class 'hm_fit'
augment(x, newdata, ...)
```

Arguments

x	An hm_fit.
newdata	Explicit dataset accepted by hm_predict().
...	Arguments passed to prediction, including optional managed workers.

Value

Observation metadata with .fitted, and .resid for scalar numeric outcomes. Matrix outputs remain one explicit list column.

glance.hm_fit *Inspect model and training scope*

Description

Inspect model and training scope

Usage

```
## S3 method for class 'hm_fit'
glance(x, ...)
```

Arguments

x	An hm_fit.
...	Currently unused.

Value

One row recording model, task, training count, input track and the absence of inherited scientific qualification.

`hm_active_learning` *Queue uncertain training-pool cases for annotation review*

Description

Queue uncertain training-pool cases for annotation review

Usage

```
hm_active_learning(
  predictions,
  pool,
  training_ids,
  heldout_ids,
  budget = 1L,
  seed = 1L,
  reference_revision,
  data,
  split
)
```

Arguments

`predictions`	Probabilistic `hm_prediction` aligned exactly to pool rows.
`pool`	Unlabelled data frame with observation_id and all task grouping columns.
`training_ids`	Character identities exactly matching the validated training partition.
`heldout_ids`	Character identities exactly matching the validated test partition.
`budget`	Positive integer number of pool cases to queue.
`seed`	Seed used only to break entropy ties.
`reference_revision`	Explicit immutable reference revision identifier.
`data`	Source data used to verify pool identities and grouping.
`split`	Validated grouped train/test split.

Details

Only training-access unlabelled rows may enter the pool. This creates a review queue; it does not acquire labels or modify annotations.

Value

List with an entropy-ranked pending-review queue, seed, budget, reference revision, run hash, annotation owner and labels_acquired=FALSE.

hm_admit_backend *Admit reviewed resolved metadata as an exact install plan*

Description

Admit reviewed resolved metadata as an exact install plan

Usage

```
hm_admit_backend(
  resolution,
  reviewed_terms,
  cache = tools::R_user_dir("hypermodR", "cache"),
  hardware_verified = FALSE,
  checkpoint = NULL,
  port = 8000L
)
```

Arguments

resolution	Result of hm_resolve_backend.
reviewed_terms	Full reviewed terms entries, preserving each package, version and artifact identity. Use the resolved entries after artifact notice review and set each reviewed flag to TRUE.
cache	External runtime cache.
hardware_verified	Whether exact driver/device ABI has been checked.
checkpoint	Optional exact local checkpoint artifact manifest for vLLM.
port	Unprivileged loopback server port.

Value

An hm_backend_plan bound to reviewed artifact terms and supplied device/checkpoint admission.

hm_annotation_contract *Bind an annotation project to its immutable export contract*

Description

Bind an annotation project to its immutable export contract

Usage

```
hm_annotation_contract(project, label_map)
```

Arguments

project	annot_project.
label_map	Named positive integer class codes; zero is background.

Value

Serializable exact manifest binding and label map.

---

hm_app *Open the local spectral modelling workbench*

Description

Import, configuration and submission are separate explicit actions. Optional runtimes are never installed on launch. Each connection owns its state/jobs.

Usage

```
hm_app(launch = TRUE, host = "127.0.0.1", port = NULL, ...)
```

Arguments

launch	Run the application; FALSE returns a shiny.appobj.
host	Listening interface, default loopback.
port	Optional listening port.
...	Arguments passed to shiny::runApp.

Value

A Shiny application, invisibly when launched.

---

hm_app_state *Create an isolated workbench state*

Description

Create an isolated workbench state

Usage

```
hm_app_state()
```

Value

An independent session state environment.

hm_apply_calibration *Apply a frozen calibrator without learning from new outcomes*

Description

Apply a frozen calibrator without learning from new outcomes

Usage

```
hm_apply_calibration(calibration, predictions)
```

Arguments

calibration	Frozen hm_calibration.
predictions	Probabilistic hm_prediction with the identical class map.

Details

Does not learn from new outcomes. Abstention marks low confidence without deleting rows.

Value

An updated hm_prediction with scaled probabilities, maximum-probability labels, logical abstain vector and calibration provenance.

hm_as_dataset.cuvis_session *Convert a qualified companion object*

Description

Convert a qualified companion object

Usage

```
## S3 method for class 'cuvis_session'
hm_as_dataset(x, profile = NULL, ...)

## S3 method for class 'cuvis_measurement'
hm_as_dataset(x, profile = NULL, ...)

## S3 method for class 'annot_image'
hm_as_dataset(x, profile = NULL, observations = .hm_image_observations(), ...)

## S3 method for class 'annot_project'
```

```
hm_as_dataset(x, profile = NULL, ...)

hm_as_dataset(x, profile = NULL, ...)

## Default S3 method:
hm_as_dataset(x, profile = NULL, ...)

## S3 method for class 'hsi_cube'
hm_as_dataset(
  x,
  profile = NULL,
  observations = .hm_image_observations(),
  geometry = .hm_native_geometry(),
  ...
)
```

Arguments

x	Companion object.
profile	Explicit source profile where required.
...	Adapter-specific arguments.
observations	Explicit acquisition and biological identities.
geometry	Source spatial geometry.

Details

hyperspectR cube conversion preserves measured bands, masks and declared geometry. A cube exported by this package also retains selected-frame observations and band metadata; recorded content identity must still match on reimport. Explicit observations override restored identities. hyperannotatR conversion uses a bounded public tile and records manifest/revision identity. CUVIS session enumeration and SDK scaling remain blocked pending source-backed fixtures. Conversion does not infer calibration.

Value

HMDataset or HMSequence for an admitted class; unsupported classes and currently unqualified CUVIS sources raise typed errors.

---

hm_as_hsi_cube *Materialise one frame through hyperspectR*

Description

Materialise one frame through hyperspectR

Usage

```
hm_as_hsi_cube(x, acquisition_id, frame_id = NULL, ...)
```

Arguments

x	Spatial HMDataset.
acquisition_id	One acquisition ID.
frame_id	One frame ID for sequences; NULL for an image.
...	Reserved.

Value

An hsi_cube with explicit native band sorting map in metadata.

---

hm_backend_capabilities *Inspect local runtime capabilities without starting services*

Description

Inspect local runtime capabilities without starting services

Usage

```
hm_backend_capabilities()
```

Value

List with OS, architecture, R version, Python executable path, and explicit unestablished GPU qualification.

---

hm_backend_plan *Plan an isolated runtime installation*

Description

Plan an isolated runtime installation

Usage

```
hm_backend_plan(
  model_id,
  device = "auto",
  profile = NULL,
  cache = tools::R_user_dir("hypermodR", "cache"),
  capabilities = hm_backend_capabilities()
)
```

Arguments

model_id	Candidate or custom model identity.
device	Requested device.
profile	Exact custom artifact manifest; no upstream profile is inferred.
cache	External user cache.
capabilities	Local capability record.

Value

An hm_backend_plan containing the validated profile, capabilities, cache, artifact-byte budget, actions and immutable digest.

---

hm_backend_status *Inspect a managed process*

Description

Inspect a managed process

Usage

```
hm_backend_status(handle)
```

Arguments

handle	Owned process handle.

Value

List with PID, model, alive flag, exit status and plan digest.

---

hm_bake *Apply immutable training transformations*

Description

Apply immutable training transformations

Usage

```
hm_bake(recipe, newdata)
```

Arguments

recipe	Fitted recipe returned by hm_prep().
newdata	Numeric observation-by-feature matrix with the original input column count and ordering.

Details

Reuses frozen training transformations without learning from new data.

Value

Numeric transformed matrix; rows retain input order and columns follow the fitted steps.

---

hm_benchmark_protocol *Admit a bounded Spectralverse benchmark protocol without accessing data*

Description

Admit a bounded Spectralverse benchmark protocol without accessing data

Usage

```
hm_benchmark_protocol(protocol, comparison = NULL)
```

Arguments

protocol	List declaring version=1, program=Spectralverse, cohort, role, dataset_revision, reference_revision, license_admitted, person_grouping_verified, source_cases, roi_ids, input_track, context_budget, pretraining_overlap, prior_outcome_access, target_access and capabilities.
comparison	Optional second protocol requiring identical cohort/revisions/cases/ROIs/input/context and target access.

Details

Cohorts are HeiPor (primary_anatomy), xeno (development_species_perfusion) and ManU (human_admission). ManU requires verified person grouping. Each capability lane supplies model_id, task, platform, logical qualified and evidence. Admission is declarative and does not access data or execute a benchmark.

Value

List with validated protocol, content hash, target-role classification, capability-lane admission states, execution=not_run and scientific_qualification=FALSE.

`hm_bootstrap` *Paired whole-unit bootstrap of fixed predictions*

Description

Paired whole-unit bootstrap of fixed predictions

Usage

```
hm_bootstrap(
  evaluation,
  unit,
  draws = 200L,
  seed = 1L,
  level = 0.95,
  comparison = NULL
)
```

Arguments

`evaluation`	An `hm_evaluation` of fixed predictions.
`unit`	Reference column naming whole independent resampling units; connected biological/source components must not be split.
`draws`	Integer number of bootstrap replicates, at least two.
`seed`	Seed for reproducible unit sampling.
`level`	Interval coverage strictly between zero and one.
`comparison`	Optional exactly matched second evaluation for paired metric differences.

Details

Resamples complete units with replacement. Fitting and model-selection uncertainty are excluded. If any draw lacks valid metrics, intervals remain unavailable and status reports invalid draws.

Value

List with percentile interval, draw-by-metric resamples, row indices, sampled units, validity flags, invalid draw count, seed, level and status.

hm_cache *Create independent in-memory cached values*

Description

Create independent in-memory cached values

Usage

```
hm_cache()
```

Value

A list with get and set functions; get returns an independent copy.

hm_cache_key *Compute a complete cache identity*

Description

Compute a complete cache identity

Usage

```
hm_cache_key(data, state = list())
```

Arguments

data	Data, including all source values and band metadata.
state	Reader, calibration, model and preprocessing state.

Value

SHA-256 identity. Lazy sources include declared content hashes.

`hm_calibrate` *Freeze temperature scaling and abstention using an inner held-out partition*

Description

Freeze temperature scaling and abstention using an inner held-out partition

Usage

```
hm_calibrate(
  predictions,
  reference,
  split,
  data,
  temperature_grid = c(0.5, 1, 2),
  abstention = 0,
  fit,
  outer_split
)
```

Arguments

`predictions`	Probabilistic predictions from fit on the inner held-out calibration partition.
`reference`	Exact source observation rows for the calibration partition, in prediction order.
`split`	Validated inner train/test split partitioning outer training.
`data`	Original source data used for fitting and split validation.
`temperature_grid`	Positive finite candidate temperatures.
`abstention`	Confidence threshold in `[0,1]` frozen into the calibrator.
`fit`	Training-only `hm_fit` that generated the supplied predictions.
`outer_split`	Validated outer split whose training rows are partitioned by split.

Details

Selects the first temperature minimizing calibration log loss. Requires full class support and verifies the supplied predictions against the fit; outer test outcomes cannot enter calibration.

Value

An `hm_calibration` containing selected temperature, abstention threshold, class map, fit/calibration identities, reference hash and grid scores.

hm_cancel_job *Cancel only an owned modelling job*

Description

Cancel only an owned modelling job

Usage

```
hm_cancel_job(job)
```

Arguments

job	Owned job handle.

Value

Confirmed job state, invisibly.

hm_check_backend *Inspect installation and separately supplied parity evidence*

Description

Inspect installation and separately supplied parity evidence

Usage

```
hm_check_backend(
  model_id,
  tests = c("availability", "parity"),
  backend = NULL,
  parity = NULL
)
```

Arguments

model_id	Model identity.
tests	Requested checks.
backend	Installed backend.
parity	Named evidence including plan_digest and passed.

Details

Availability verifies the declared installation path, containment, artifact sizes/hashes and any Python environment inventory. Parity is caller-supplied evidence bound to the plan digest, reported as reported_pass rather than independently executed here. No scientific qualification follows.

Value

List with model identity, availability status, reported parity status and scientific_qualification=FALSE.

---

hm_compare *Compare matched predictions using paired whole-unit resampling*

Description

Compare matched predictions using paired whole-unit resampling

Usage

```
hm_compare(
  evaluations,
  paired = TRUE,
  unit,
  draws = 200L,
  seed = 1L,
  family = NULL
)
```

Arguments

evaluations	List of exactly two matched hm_evaluation objects, ordered first minus second.
paired	Must be TRUE.
unit	Reference column identifying independent whole units.
draws	Integer number of paired bootstrap draws, at least two.
seed	Reproducible resampling seed.
family	Optional descriptive comparison-family declaration, retained without multiplicity adjustment.

Details

Requires identical cases, references, task, class map, estimand, hierarchy and metric names. This is descriptive inference without multiplicity-adjusted significance claims.

Value

An hm_evaluation list containing metric differences, paired bootstrap uncertainty, family and interpretation.

hm_coordinates *Map image coordinates through explicit geometry*

Description

Map image coordinates through explicit geometry

Usage

```
hm_coordinates(geometry, coordinates, inverse = FALSE)
```

Arguments

geometry	Geometry with origin and homogeneous transform.
coordinates	Two-column x,y matrix of source indices (1-based).
inverse	Map world coordinates back to source indices.

Value

A two-column numeric matrix.

hm_dataset *Construct a native spectral dataset*

Description

Construct a native spectral dataset

Usage

```
hm_dataset(values, observations, bands, geometry = NULL, provenance = list())
```

Arguments

values	Numeric observation-by-band matrix or y,x,wavelength array.
observations	Data frame with unique character observation_id; images also require image_id and acquisition_id.
bands	Band metadata with numeric wavelength in nm. Unknown wavelengths require availability='unavailable'.
geometry	Explicit origin and invertible homogeneous transform for images.
provenance	Reproducible source and calibration metadata.

Value

An HMDataset.

hm_dual_stream_mhsi *Declare the dual_stream_mhsi upstream candidate*

Description

Declare the dual_stream_mhsi upstream candidate

Usage

```
hm_dual_stream_mhsi(task, parameters = list())
```

Arguments

task	An hm_task.
parameters	Explicit mode, provenance and preprocessing declarations.

Value

An hm_model_spec; upstream execution remains admission-gated.

hm_endpoint_contract *Declare independent reference endpoints*

Description

Declare independent reference endpoints

Usage

```
hm_endpoint_contract(references)
```

Arguments

references	Data frame with observation_id, endpoint and value. Missing values remain missing.

Value

Validated declaration, without claiming source truth.

`hm_evaluate` *Evaluate predictions against an explicitly aligned reference*

Description

Evaluate predictions against an explicitly aligned reference

Usage

```
hm_evaluate(
  predictions,
  reference,
  estimand = "pooled",
  resampling = NULL,
  hierarchy = c("subject_id", "acquisition_id")
)

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

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

Arguments

`predictions`	An `hm_prediction`.
`reference`	Data frame with observation_id in exactly prediction order and task outcome/reference columns. Reconstruction targets are supplied as the numeric matrix attribute target. For predictions in fitted recipe coordinates, bake this target with the fitted recipe and set attribute target_space to predictions$provenance$output_space.
`estimand`	Either pooled or equal_class_subject_acquisition.
`resampling`	Optional named argument list passed to `hm_bootstrap()`.
`hierarchy`	Two column names giving biological unit then acquisition for hierarchical categorical accuracy.
`x`	An object returned by `hm_evaluate()`.
`...`	Additional arguments; forwarded to graphics for plotting, otherwise unused.
`object`	An object returned by `hm_evaluate()`.

Details

The hierarchical estimand averages observations within class/acquisition, acquisitions within class/subject, subjects within class, then classes equally. Missing class support produces unavailable macro metrics. Anomaly scores are not calibrated risks; reconstruction spectral angles are radians and refer to the declared target coordinates. Named matrix reference/output columns must match exactly; no reordering is inferred. Survival uses comparable event-time pairs. Evaluation does not fit models or repair reference alignment.

Value

An hm_evaluation list containing named numeric metrics, task-specific per-class support/confusion, predictions, reference, estimand, hierarchy and optional uncertainty.

The input object invisibly.

A list of descriptive fields.

---

hm_example *Generate an explicitly synthetic grouped spectral example*

Description

Generate an explicitly synthetic grouped spectral example

Usage

```
hm_example(seed = 42L)
```

Arguments

seed	Nonnegative integer seed.

Value

An HMDataset with synthetic labels, subject groups and eight bands.

---

hm_export_annotations *Export grouped annotations through hyperannotatR*

Description

Export grouped annotations through hyperannotatR

Usage

```
hm_export_annotations(x, destination, groups, ...)
```

Arguments

x	Annotation project or session.
destination	Export directory.
groups	Explicit biological grouping table accepted by at_training_export.
...	Additional public export arguments.

Value

Companion export including integrity checks.

hm_export_result *Export a reproducible workbench result*

Description

Export a reproducible workbench result

Usage

```
hm_export_result(result, path, include_private = FALSE)
```

Arguments

result	Completed fit, prediction, evaluation and configuration bundle.
path	New export directory, never an existing run.
include_private	Include fitted objects and original prediction identities. The default exports configuration, aggregate metrics and a reproduction script.

Details

Reference reproduction requires an explicitly supplied admitted backend and artifact_root; it retrains in an owned bounded job. Private reference exports include checked learned files. The generated read_private_fit() helper resolves their location after moving the export; use hm_predict with an owned matching backend_handle to verify and predict. Installed runtime receipts and cache paths are not portable exports. The destination is reserved before writing. Its completion marker appears only after reproduction code and any private artifacts are finalized. Interrupted exports retain a failed destination with partial contents for inspection; the destination cannot be read as a completed run.

Value

Verified export directory, invisibly.

hm_extract_native *Extract admitted native upstream image features*

Description

Extract admitted native upstream image features

Usage

```
hm_extract_native(
  model_id,
  cube,
  wavelengths,
  mask,
  admission,
  control = list()
)
```

Arguments

model_id	Named candidate identity (htc, hyperfree or hypersigma).
cube	Numeric full-frame HWC array; spectral channels remain measured.
wavelengths	Strictly increasing measured wavelengths in nm.
mask	Logical full-frame eligible-pixel mask.
admission	Explicit exact source, checkpoint and separate rights admission.
control	Runtime list with an owned backend_handle and optional timeout.

Details

Only htc, hyperfree and hypersigma extraction are admitted by this entry point. The worker must verify its adapter-specific exact source/checkpoint/input contract. Admission requires execute_source_acknowledged, code_terms_admitted and weight_terms_admitted flags. Extraction is not upstream downstream-task training or scientific qualification; other named architectures remain blocked.

Value

An hm_native_features list with worker-returned numeric features, identity metadata, qualified=FALSE and input digest. Feature shape is adapter-specific.

---

hm_fit *Fit a workflow using training indices only*

Description

Fit a workflow using training indices only

Usage

```
hm_fit(workflow, data, split, control = list())

## S3 method for class 'hm_fit'
coef(object, ..., scale = c("fitted", "original"))
```

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

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

Arguments

workflow	An hm_workflow.
data	An HMDataset of native spectra, or list with numeric values matrix, observations data frame, band_map and optional provenance.
split	One explicit train/test fold, normally from hm_split(); indices are source row positions.
control	Engine-control list: seed defaults to one; weights may align to all source rows or training rows. Optional tuning contains candidates, metric and direction. Reference workers require backend_handle and a new artifact_path.
object	An hm_fit.
...	Arguments forwarded to the fitted engine coefficient method where supported.
scale	Either fitted coordinates or original measured-band coordinates.
x	An hm_fit.

Details

Preprocessing and engine fitting use training rows only. Connected biological/source identities cannot cross train/test; inner folds must remain inside outer training. Spatial images/sequences require explicit supported routes and are never silently flattened. R reference spectral training currently admits reference:cnn1d, reference:spectral_vit, reference:masked_autoencoder, reference:vae, reference:distillation, reference:multitask and reference:domain_adversarial with the appropriate task; worker-level tensor families have a separate protocol. A fitted model is not independent scientific qualification.

Fitted coefficient availability depends on the engine. Original-band mapping is restricted to lm, glm, glmnet, pls, pcr and bayes_conjugate with globally affine scale, pca, select, derivative, resample or reflectance recipes. Nonlinear/missingness-dependent recipes, aliased coefficients and incompatible dimensions raise typed errors. Coefficients describe a fitted coordinate system and do not establish biological causality.

Value

An hm_fit containing workflow, fitted recipe, engine, training indices, validated split, original band map, input radiometric unit and hash-bound provenance.

Engine coefficients with scale and link attributes. Original-scale output is a named intercept/band vector (or engine list of vectors), with band_map attached.

Print methods return the input invisibly.

Summary methods return model identity and training-row count.

hm_from_tivis *Import an explicitly profiled TIVITA measurement list*

Description

Import an explicitly profiled TIVITA measurement list

Usage

```
hm_from_tivis(
  x,
  profile,
  observations = .hm_image_observations(),
  assets = data.frame(),
  registration_id = NULL,
  mask = NULL
)
```

Arguments

x	Unclassed tivis_measurement result.
profile	Explicit hm_tivita_profile declaration.
observations	Explicit source/group identity.
assets	Asset inventory with role and registration_id columns.
registration_id	Expected registration identity.
mask	Optional logical y,x validity mask; NA remains unknown.

Value

HMDataset; no device calibration is inferred.

hm_htc *Declare the htc upstream candidate*

Description

Declare the htc upstream candidate

Usage

```
hm_htc(task, parameters = list())
```

Arguments

task	An hm_task.
parameters	Explicit mode, provenance and preprocessing declarations.

Value

An hm_model_spec; upstream execution remains admission-gated.

---

hm_hyperfree *Declare the hyperfree upstream candidate*

Description

Declare the hyperfree upstream candidate

Usage

```
hm_hyperfree(task, parameters = list())
```

Arguments

task	An hm_task.
parameters	Explicit mode, provenance and preprocessing declarations.

Value

An hm_model_spec; upstream execution remains admission-gated.

---

hm_hypersigma *Declare the hypersigma upstream candidate*

Description

Declare the hypersigma upstream candidate

Usage

```
hm_hypersigma(task, parameters = list())
```

Arguments

task	An hm_task.
parameters	Explicit mode, provenance and preprocessing declarations.

Value

An hm_model_spec; upstream execution remains admission-gated.

hm_hypervision *Declare the hypervision upstream candidate*

Description

Declare the hypervision upstream candidate

Usage

```
hm_hypervision(task, parameters = list())
```

Arguments

task	An hm_task.
parameters	Explicit mode, provenance and preprocessing declarations.

Value

An hm_model_spec; upstream execution remains admission-gated.

hm_install_backend *Install declared files into an immutable user runtime directory*

Description

Install declared files into an immutable user runtime directory

Usage

```
hm_install_backend(
  plan,
  accept_terms = FALSE,
  offline = FALSE,
  acknowledge_custom_code = FALSE
)
```

Arguments

plan	Resolved backend plan.
accept_terms	Explicit acceptance bound to this plan's terms and artifacts.
offline	Forbid network downloads.
acknowledge_custom_code	Explicit permission to run reviewed custom code.

Value

An hm_backend list with installation path, original plan and installed_unverified state.

hm_instrument_profiles *Inspect admitted instrument capabilities*

Description

Inspect admitted instrument capabilities

Usage

```
hm_instrument_profiles()
```

Value

A data frame. Synthetic profiles are not device qualification.

hm_job_status *Poll a modelling job*

Description

Poll a modelling job

Usage

```
hm_job_status(job)
```

Arguments

job	Owned job handle.

Value

State, elapsed time, result only after successful completion, and error.

hm_model_spec *Declare a model specification*

Description

Declare a model specification

Usage

```
hm_model_spec(model_id, task, parameters = list())
```

Arguments

model_id	Single model identifier from hm_models() or an explicitly admitted custom identity.
task	An hm_task declaration.
parameters	Named list of engine-specific parameters; interpreted when fitting.

Details

This constructor declares intent and checks known core task compatibility. It neither installs a backend nor qualifies a model scientifically.

Value

An hm_model_spec list containing model_id, task and parameters.

hm_models *List candidate model families*

Description

List candidate model families

Usage

```
hm_models(task = NULL, backend = NULL, include_unavailable = TRUE)
```

Arguments

task	Optional task filter.
backend	Optional backend filter.
include_unavailable	Include unqualified candidates.

Details

Registry visibility denotes candidate declaration or setup availability, not installed execution or independent parity. Reference families have separate spectral/tensor routes; not every listed identity is accepted by hm_fit(). All returned scientific qualification flags remain FALSE.

Value

List of model records with identity, supported task declarations, candidate backend, availability/admission status and scientific qualification.

`hm_physics_kernel` *Declare supplied physical forward and inverse kernels*

Description

Declare supplied physical forward and inverse kernels

Usage

```
hm_physics_kernel(forward, inverse, parameters = list(), source, units)
```

Arguments

`forward`	Function of concentrations and a parameter list, producing spectra.
`inverse`	Function of spectra and a parameter list, producing concentrations.
`parameters`	Serializable physical constants and assumptions.
`source`	Source identifier for the implemented physical model.
`units`	Named units for input and output quantities.

Value

A declared `hm_physics_kernel`. Supplying a function does not validate its physiological assumptions; independent standards remain necessary.

hm_pool *Explicitly pool all valid spatial pixels*

Description

Explicitly pool all valid spatial pixels

Usage

```
hm_pool(x, mask = NULL, method = "mean", selection = list())
```

Arguments

x	Spectral dataset accepted by hm_read_block().
mask	Optional logical array matching the block validity dimensions; no missing entries.
method	Pooling rule; currently only mean is supported.
selection	Named block selection passed to hm_read_block().

Details

All eligible finite pixels in the selected support contribute equally; an empty eligible support is rejected.

Value

List with a one-row band matrix values, band_map, and provenance including eligible pixel count, equal weights and mask hash.

hm_portable_sequence *Index a portable spectral sequence without repairing chronology*

Description

Index a portable spectral sequence without repairing chronology

Usage

```
hm_portable_sequence(
  frames,
  frame_index,
  observations,
  bands,
  geometry,
  provenance = list()
)
```

Arguments

frames	Numeric cubes or admitted RDS locators.
frame_index	Explicit frame table; time is source elapsed seconds, including missing or reversed values.
observations	Observation identities.
bands	Native band metadata.
geometry	Source grid.
provenance	Source evidence.

Value

HMSequence retaining original ordering and timing quality.

---

hm_predict *Predict with a fitted workflow*

Description

Predict with a fitted workflow

Usage

```
hm_predict(object, newdata, ...)

## S3 method for class 'hm_fit'
hm_predict(object, newdata, ...)

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

Arguments

object	An hm_fit.
newdata	Native spectral data in the same form as hm_fit data, with exactly matching measured band semantics.
...	Engine prediction controls; reference workers require control with the owned backend_handle.

Details

Applies the frozen training recipe. Measured wavelength/unit/fwhm/native/canonical band metadata present during training must match exactly. Prediction does not fit preprocessing or use new outcomes. The declared radiometric unit must also match; calibration identities may differ between acquisitions. Older serialized fits without this unit contract must be refitted. Reconstruction after preprocessing is in fitted recipe coordinates, recorded as provenance$output_space. Evaluate it against a target baked with that recipe and the matching target_space reference attribute, never against untransformed spectra.

Value

An hm_prediction in newdata observation order, with task-specific values and optional class probabilities.

---

hm_prediction *Validate model predictions*

Description

Validate model predictions

Usage

```
hm_prediction(
  values,
  task,
  observation_id,
  probabilities = NULL,
  class_map = NULL,
  provenance = list()
)

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

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

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

Arguments

values	Prediction vector, or observation-by-output matrix for multilabel, unmixing and reconstruction.
task	An hm_task.
observation_id	Unique nonmissing character identifiers in prediction row order.
probabilities	Optional numeric observation-by-class matrix in [0,1], with columns exactly matching class_map; rows sum to one except for multilabel tasks.
class_map	Exact character class/label order, required for categorical and multilabel predictions.
provenance	List of source, fit and transformation provenance.
x	An object returned by hm_prediction().
...	Additional arguments; forwarded to graphics for plotting, otherwise unused.
object	An object returned by hm_prediction().

Details

Categorical labels must belong to the declared map. Unmixing outputs are nonnegative and sum to one unless task reference explicitly declares nonnegative-only abundances. This validation does not establish predictive accuracy or calibrated probabilities. Numeric scalar tasks (including survival risk) require finite numeric vectors; matrix tasks require finite numeric matrices. Named matrix outputs retain column identity.

Value

An `hm_prediction` list retaining values, task, identities, probabilities, class map and provenance.

The input object invisibly.

A list of descriptive fields.

The graphics plotting return value, invisibly.

`hm_prep` *Fit preprocessing on training spectra*

Description

Fit preprocessing on training spectra

Usage

```
hm_prep(recipe, training, wavelength = NULL)
```

Arguments

`recipe`	Unfitted `hm_recipe`.
`training`	Numeric observation-by-feature training matrix.
`wavelength`	Optional measured wavelength vector in nm, aligned to input columns; required by wavelength-dependent steps.

Details

Call only on the training partition. Selection, derivatives, resampling and PCA can change output dimensions; resampling marks the track as adapted.

Value

A fitted `hm_recipe` containing learned step state, input/output wavelength grids, feature count and training hash.

hm_prompt *Declare auxiliary language-model prompt provenance*

Description

Declare auxiliary language-model prompt provenance

Usage

```
hm_prompt(text, origin, rendering = NULL, timestamps = NULL, images = NULL)
```

Arguments

text	Prompt text.
origin	Origin of instructions, independent of held-out outcomes.
rendering	Rendering manifest for image content.
timestamps	Frame timestamps retained in order.
images	Optional PNG/JPEG data URLs from an explicitly manifested rendering.

Value

An hm_prompt list retaining text, origin, rendering, timestamps, optional images, auxiliary_only flag and content digest.

hm_provenance *Capture explicit reproducibility identities*

Description

Capture explicit reproducibility identities

Usage

```
hm_provenance(objects = list(), files = character(), resources = NULL)
```

Arguments

objects	Named list of R objects whose serialized identities are hashed.
files	Character vector of existing file paths whose bytes are SHA-256 hashed.
resources	Optional resource declaration retained in the receipt.

Value

An hm_provenance receipt with object/file hashes, R version, platform and resources.

hm_read_block *Read a bounded spatial block*

Description

Read a bounded spatial block

Usage

```
hm_read_block(x, selection = list())

## S4 method for signature 'HMDataset'
hm_read_block(x, selection = list())
```

Arguments

x	Spatial HMDataset or HMSequence.
selection	Named acquisition_id, frame_id, rows, cols and bands selectors. Out-of-grid rows/cols explicitly request invalid NA padding.

Details

Rows and cols are unique integer source-grid indices; out-of-grid values request NA padding marked invalid. Bands are valid positive 1-based indices. Select exactly one acquisition; sequence frames retain requested order. Variable grids must be read one frame at a time. Validity requires every selected band finite and any declared spatial mask TRUE. The byte check bounds planned output allocation, not total resident memory; RDS sources have a separate bounded deserialization admission check.

Value

Named list with a numeric four-dimensional values array (y, x, wavelength, time), a logical three-dimensional valid array (y, x, time), observations, band_map, geometry and provenance. Default output budget is 256 MiB (option hypermodR.block_bytes).

hm_read_run *Read and verify a completed run*

Description

Read and verify a completed run

Usage

```
hm_read_run(path)
```

Arguments

path	Run directory created by hm_save_run().

Details

Requires matching complete receipt, object SHA-256 and COMPLETE marker before reading the object. Read only trusted RDS sources.

Value

The deserialized original R object.

---

hm_recipe *Declare preprocessing steps*

Description

Declare preprocessing steps

Usage

```
hm_recipe(steps = list())
```

Arguments

steps	List of step lists, each with a type and its required controls; see Details.

Details

Steps execute in list order. Learned quantities are estimated only by hm_prep() on its supplied training matrix. Supported types are impute (training column means), scale (training means and standard deviations), pca (optional integer components), select (explicit unique integer bands), normalize (row Euclidean norm), snv (row mean/standard deviation), msc (training mean reference), reflectance (explicit dark and white band vectors), derivative (adjacent finite differences per nm), resample (explicit wavelength grid within measured support), and continuum (positive upper concave envelope division). Missing values must be imputed before steps requiring finite data.

Value

An unfitted hm_recipe with native input track.

hm_reliability *Describe confidence bins and coverage-risk without claiming OOD validation*

Description

Describe confidence bins and coverage-risk without claiming OOD validation

Usage

```
hm_reliability(predictions, reference, breaks = seq(0, 1, length.out = 11))
```

Arguments

predictions	Categorical hm_prediction with probabilities.
reference	Exactly aligned reference data frame accepted by hm_evaluate().
breaks	Strictly increasing numeric confidence-bin boundaries from zero to one.

Details

Confidence is the maximum class probability. Empty bins or zero retained coverage have unavailable accuracy/risk. Confidence does not constitute a validated OOD detector.

Value

List of bin counts/mean confidence/accuracy, coverage-risk table, evaluation scores and an OOD interpretation.

hm_render_hsi *Render a declared three-band auxiliary image*

Description

Render a declared three-band auxiliary image

Usage

```
hm_render_hsi(cube, wavelengths, bands, limits, source_id, timestamp = NULL)
```

Arguments

cube	Finite height by width by channel numeric array.
wavelengths	Measured wavelength vector.
bands	Three distinct channel indices.
limits	Explicit common intensity range used for clipping and scaling.
source_id	Source provenance identity.
timestamp	Optional acquisition time.

Value

An hm_rendering list with a height-by-width-by-3 image scaled to [0,1] and a hash-bound rendering manifest.

`hm_resolve_backend` *Resolve official binary metadata into an exact reviewable runtime manifest*

Description

This explicit network operation resolves metadata only. It does not download model weights or install packages. Published terms and inherited distribution notices must be reviewed before converting the result into an install plan.

Usage

```
hm_resolve_backend(
  backend = c("reference_cpu", "vllm_linux_cuda"),
  model_id,
  python,
  uv = unname(Sys.which("uv")),
 directory = tempfile("resolution-", tmpdir = tools::R_user_dir("hypermodR", "cache"))
)
```

Arguments

backend	reference_cpu. The former vllm_linux_cuda setup route is blocked pending a reviewed replacement for its retired vulnerable profile.
model_id	Exact model identity or reference family.
python	Explicit existing Python 3.11 or 3.12 executable.
uv	Explicit uv resolver executable (version 0.11.26).
directory	External persistent resolution directory.

Value

An hm_backend_resolution containing exact artifact profile, package metadata, external directory, hard gates and pending qualification.

hm_resolve_checkpoint *Resolve an immutable public Hugging Face checkpoint without downloading weights*

Description

Resolve an immutable public Hugging Face checkpoint without downloading weights

Usage

```
hm_resolve_checkpoint(
  repository,
  revision,
 directory = tempfile("checkpoint-", tmpdir = tools::R_user_dir("hypermodR", "cache"))
)
```

Arguments

repository	Public model repository owner/name.
revision	Exact 40-character commit SHA.
directory	New external metadata staging directory.

Value

List containing repository/revision, architecture, local checkpoint directory name, artifact hashes/sizes/URLs, unreviewed terms and pending qualification.

hm_resources *Declare bounded execution resources*

Description

Declare bounded execution resources

Usage

```
hm_resources(
  cpus = 1L,
  threads = 1L,
  memory_mb = 1024,
  device = "cpu",
  timeout = 60,
  network = FALSE
)
```

Arguments

cpus	Positive integer logical-CPU count selected from inherited allowed affinity.
threads	Positive integer numerical thread budget, no larger than cpus.
memory_mb	Positive finite per-process virtual address-space limit in MiB for bounded child execution.
device	Single device identifier. General bounded children require cpu. A separately admitted CUDA backend may select an explicit cuda:N device, subject to inherited visibility and verified hardware ABI.
timeout	Positive finite time budget in seconds.
network	Logical network-use policy declaration; network access is not isolated by this launcher.

Details

Bounded child routes currently require Linux taskset, prlimit and timeout. They apply inherited-affinity CPU subsets, numerical thread environment settings before startup, per-process RLIMIT_AS and an external elapsed timeout. Memory limits concern virtual address space, not aggregate descendant RSS; no additional CPU quota or network isolation is supplied. CUDA selection is admitted only by the backend launcher; hm_with_resources remains CPU-only. GPU memory is not limited, and admitting a device does not establish live GPU or scientific qualification. This is not a security sandbox.

Value

An hm_resources list of the validated declarations.

---

hm_save_run *Save an immutable run with an atomic completion receipt*

Description

Save an immutable run with an atomic completion receipt

Usage

```
hm_save_run(object, path)
```

Arguments

object	R object to serialize and verify by readback.
path	New run-directory path whose parent already exists.

Details

Reserves a new directory, writes object.rds and a hash-bound receipt, and atomically finalizes COMPLETE. Existing paths are rejected. A failed write leaves a failed receipt for inspection; completion markers are removed.

Value

Normalized run-directory path invisibly.

`hm_sequence` *Construct a spectral sequence*

Description

Construct a spectral sequence

Usage

```
hm_sequence(
  frames,
  frame_index,
  observations,
  bands,
  geometry,
  provenance = list()
)
```

Arguments

`frames`	List of native y,x,wavelength arrays or declarative RDS sources.
`frame_index`	Frame metadata with character frame_id, sequence_id, acquisition_id and numeric source time in elapsed seconds (NA allowed). timing_quality marks ordered, unavailable, duplicate or reversed source timing; source order is retained.
`observations`	Observation metadata linked by frame_id.
`bands`	Band metadata in nm.
`geometry`	Explicit spatial geometry.
`provenance`	Source and timing declarations.

Value

An HMSequence.

hm_spectra *Read native spectra without spatial reduction*

Description

Read native spectra without spatial reduction

Usage

```
hm_spectra(x, selection = list())

## S4 method for signature 'HMDataset'
hm_spectra(x, selection = list())
```

Arguments

x	HMDataset containing native spectra.
selection	Named list with character observation_id and positive bands.

Details

Only native spectral matrices are admitted; images require explicit spatial reduction. Bands are unique positive 1-based indices in requested order.

Value

Named list with numeric observation-by-selected-band values matrix, selected observations in request order, band_map and source provenance.

hm_spectral_library *Declare an extinction or response library*

Description

Values must be supplied from measured or otherwise justified sources. This constructor records a declaration; it cannot verify the physical reference.

Usage

```
hm_spectral_library(values, wavelength, units, source, license)
```

Arguments

values	Numeric wavelength-by-component matrix.
wavelength	Strictly increasing wavelength centres in nm.
units	Character description of coefficient units.
source	Character source citation or persistent identifier.
license	Character declaration of reuse terms.

Value

An immutable-by-identity hm_spectral_library list.

---

hm_spectralearth *Declare the spectralearth upstream candidate*

Description

Declare the spectralearth upstream candidate

Usage

```
hm_spectralearth(task, parameters = list())
```

Arguments

task	An hm_task.
parameters	Explicit mode, provenance and preprocessing declarations.

Value

An hm_model_spec; upstream execution remains admission-gated.

---

hm_spectralgpt *Declare the spectralgpt upstream candidate*

Description

Declare the spectralgpt upstream candidate

Usage

```
hm_spectralgpt(task, parameters = list())
```

Arguments

task	An hm_task.
parameters	Explicit mode, provenance and preprocessing declarations.

Value

An hm_model_spec; upstream execution remains admission-gated.

---

hm_split *Plan grouped resampling*

Description

Plan grouped resampling

Usage

```
hm_split(
  data,
  task,
  design = "nested_grouped",
  seed = 1L,
  folds = 5L,
  inner_folds = 3L,
  ...
)
```

Arguments

data	Observation data frame, HMDataset, or list containing observations.
task	An hm_task declaring grouping columns.
design	One of grouped, nested_grouped or temporal.
seed	Seed for reproducible grouped assignment.
folds	Integer outer-fold count, at least two.
inner_folds	Integer inner-fold count for nested_grouped, at least two.
...	Temporal controls: time is the numeric timestamp-column name (default timestamp), cutoff is required, and nonnegative look_back/embargo use the same units as time.

Details

Groups connected through any task grouping column or known biological/source identities stay together; overlapping source windows are connected. Grouped assignment balances component counts rather than stratifying outcomes and rejects outer categorical folds without full class support. Temporal training precedes cutoff minus embargo and look-back; groups exposed at/after cutoff are removed entirely from training. Source times are not repaired or reinterpreted. Temporal results retain the selected time-column identity and its observations; fitting rejects a stale clock or altered partition. Regenerate the temporal split after changing timestamps.

Value

An `hm_split`. Grouped designs contain folds (train/test row indices and optional inner folds), component membership, observation identities and hash. Temporal designs contain train/test/excluded indices, cutoff, look-back and embargo.

`hm_stage_annotations` *Stage a bound integer prediction for human review*

Description

Stage a bound integer prediction for human review

Usage

```
hm_stage_annotations(predictions, project, label_map, contract, model)
```

Arguments

`predictions`	Integer y,x mask. NA is unknown; zero is background.
`project`	Destination annot_project.
`label_map`	Named class codes.
`contract`	hm_annotation_contract captured before prediction.
`model`	Model name, version and sha256 provenance.

Value

Companion staged patch with unknown mask receipt; never committed.

`hm_start_backend` *Start a process belonging to an installed runtime*

Description

Start a process belonging to an installed runtime

Usage

```
hm_start_backend(
  backend,
  model = backend$plan$model_id,
  resources = hm_resources(),
  host = "127.0.0.1"
)
```

Arguments

`backend`	Installed backend receipt.
`model`	Expected model identity.
`resources`	An `hm_resources` CPU declaration, or explicit cuda:N for a CUDA plan with verified hardware ABI; threads must be one.
`host`	Bind address, restricted to loopback.

Details

Starts the declared verified command through the Linux bounded child launcher described in hm_resources with one numerical thread. CUDA plans require explicit cuda:N selection, verified hardware ABI and a selected device within inherited CUDA_VISIBLE_DEVICES when set. CUDA memory is not limited by this launcher; CPU memory is per-process virtual address space, not aggregate RSS. Network remains a policy declaration. Admission is not live GPU qualification. Handles are valid only in their owning R session.

Value

Mutable owned `hm_backend_handle` environment containing process, PID, owner PID, model identity and plan digest.

hm_stop_backend *Stop only an owned runtime process*

Description

Stop only an owned runtime process

Usage

```
hm_stop_backend(handle)
```

Arguments

handle	Owned process handle.

Value

Final backend-status list invisibly.

hm_stream *Bounded causal frame streaming*

Description

Bounded causal frame streaming

Usage

```
hm_stream(predict_frame, max_frames = 256L)
```

Arguments

predict_frame	Function accepting a frame and its timestamp.
max_frames	Maximum frames before explicit reset.

Value

Mutable stream handle, with per-frame outputs and latency diagnostics.

hm_stream_cancel *Cancel a frame stream*

Description

Cancel a frame stream

Usage

```
hm_stream_cancel(stream)
```

Arguments

stream	An hm_stream.

Value

The cancelled stream invisibly.

hm_stream_push *Push ordered frames into a stream*

Description

Push ordered frames into a stream

Usage

```
hm_stream_push(stream, frames, timestamps, sequence_id)
```

Arguments

stream	An hm_stream.
frames	List of native frames; geometry may vary.
timestamps	Strictly increasing finite elapsed seconds.
sequence_id	Explicit recording identity.

Value

Outputs in source order, with timestamp and latency metadata.

`hm_submit_job` *Submit an isolated, cancellable modelling job*

Description

Submit an isolated, cancellable modelling job

Usage

```
hm_submit_job(
  workflow,
  data,
  split,
  control = list(),
  resources = hm_resources()
)
```

Arguments

`workflow`	Validated modelling workflow.
`data`	Spectral dataset.
`split`	One explicit grouped train/test fold.
`control`	Fitting controls.
`resources`	Execution resource declaration.

Details

Input object size is checked before launching a bounded Linux CPU child. The launcher applies an inherited-affinity subset, per-process virtual address-space limit, numerical thread settings and external elapsed timeout; see hm_resources for limits and network policy. Poll hm_job_status() to collect completion or failure. Handles belong to the creating R session.

Value

An owned job handle; poll with `hm_job_status()`.

hm_task *Declare a modelling task*

Description

Declare a modelling task

Usage

```
hm_task(outcome, type, unit, groups, reference = NULL)
```

Arguments

outcome	Single outcome-column name, or NULL for clustering, anomaly, reconstruction or unmixing.
type	One of classification, regression, ordinal, multilabel, segmentation, unmixing, reconstruction, clustering, anomaly, survival or temporal.
unit	Observation unit: spectrum, pixel, roi, image, sequence or subject.
groups	Nonempty character vector of biological/source grouping column names.
reference	Optional reference declaration; survival uses time/status column names, unmixing may declare abundance_constraint, and temporal tasks may declare horizon.

Details

Segmentation requires pixel/image units; temporal tasks require sequence/subject units. Declaring a task does not establish an executable backend for every task/model combination.

Value

An hm_task list recording outcome, type, unit, unique grouping columns and reference.

hm_temporal_model *Declare a causal temporal reference*

Description

Declare a causal temporal reference

Usage

```
hm_temporal_model(family, task, parameters = list())
```

Arguments

family	Framewise, timestamp_mean, temporal_cnn, lstm, gru, transformer or state_space.
task	An hm_task.
parameters	Explicit sequence and training declarations.

Value

An hm_model_spec with a separate package reference identity.

---

hm_tivita_profile *Declare an explicit TIVITA array profile*

Description

Declare an explicit TIVITA array profile

Usage

```
hm_tivita_profile(
  wavelengths = NULL,
  device = "unknown",
  software = "unknown",
  evidence = "unavailable"
)
```

Arguments

wavelengths	Independently established wavelengths in nm; NULL preserves unknown.
device	Device declaration, not qualification.
software	Export software declaration.
evidence	Evidence identifier.

Value

Serializable profile list.

hm_tune *Select parameters using only grouped inner training folds*

Description

Select parameters using only grouped inner training folds

Usage

```
hm_tune(
  workflow,
  data,
  split,
  candidates,
  control = list(),
  metric = NULL,
  direction = c("minimize", "maximize")
)
```

Arguments

workflow	An hm_workflow.
data	Source data accepted by hm_fit().
split	Validated outer split containing inner grouped folds; their test rows must cover outer training exactly once.
candidates	Nonempty list of complete model-parameter lists.
control	Fit-control list; resources must declare one CPU/thread. Optional max_trials must cover all candidates and seed controls reproducibility.
metric	Metric name returned by hm_evaluate(), or NULL for categorical log_loss and otherwise rmse.
direction	Whether smaller or larger mean inner-fold scores are preferred.

Details

Search is sequential inside one bounded child, so the declared elapsed timeout covers the entire search and final refit. The first tied candidate wins. Outer test outcomes are excluded from selection.

Value

An outer-training hm_fit refitted with the winning parameters and a tuning ledger containing fold scores, identities, seeds and protocol hashes.

hm_verify_provenance *Verify source identities against a provenance receipt*

Description

Verify source identities against a provenance receipt

Usage

```
hm_verify_provenance(receipt, objects = list())
```

Arguments

receipt	An hm_provenance receipt.
objects	Named objects corresponding exactly to those originally recorded.

Details

Rehashes the recorded file paths and supplied objects. Does not compare current R/platform metadata or establish source truth.

Value

TRUE invisibly on matching hashes; otherwise raises an integrity error.

hm_verify_registry *Verify a signed registry release*

Description

Verify a signed registry release

Usage

```
hm_verify_registry(
  path,
  signature,
  key,
  trusted_keys,
  revoked_keys = character(),
  minimum_version = 1L,
  now = Sys.time()
)
```

Arguments

path	Registry JSON file.
signature	Detached SHA256 RSA signature file.
key	PEM public key file provisioned by the maintainer.
trusted_keys	Allowlisted SHA256 hashes of public key files.
revoked_keys	Revoked key hashes.
minimum_version	Minimum allowed integer release sequence.
now	Verification time.

Value

An hm_registry_verification list with verified flag, file digest, trusted key fingerprint and release sequence.

---

hm_vllm_chat *Query an explicitly configured OpenAI-compatible vLLM endpoint*

Description

Query an explicitly configured OpenAI-compatible vLLM endpoint

Usage

```
hm_vllm_chat(
  endpoint,
  model,
  prompt,
  allow_remote = FALSE,
  disclosure = NULL,
  token = NULL,
  timeout = 30,
  max_bytes = 1024^2,
  generation = list(temperature = 0, max_tokens = 256L),
  .transport = .hm_vllm_transport
)
```

Arguments

endpoint	Explicit server base URL.
model	Exact served model identity.
prompt	A provenance-bearing prompt.
allow_remote	Explicit transmission consent for non-loopback endpoints.
disclosure	Description of the data transmitted to a remote endpoint.

token	Optional authentication secret, excluded from receipts and errors.
timeout	Maximum seconds.
max_bytes	Maximum request/response bytes.
generation	Deterministic generation settings.
.transport	Internal transport override for protocol testing.

Details

The endpoint must include the API base path expected before /chat/completions (for example /v1). Remote transmission requires HTTPS, allow_remote=TRUE and disclosure. Stock native HSI architectures are rejected. Responses must identify the exact served model, contain one choice and finish with stop. Text is auxiliary output with no calibrated probability interpretation.

Value

List with response text, exact model, prompt digest/origin, rendering/timestamps, generation settings, remote/disclosure fields and auxiliary-only, uncalibrated semantics.

---

hm_with_resources *Run an R expression in a bounded child process*

Description

Run an R expression in a bounded child process

Usage

```
hm_with_resources(resources, code)
```

Arguments

resources	An hm_resources declaration.
code	Unevaluated expression evaluated in a bounded child with its referenced serializable caller bindings.

Details

Evaluation occurs in a fresh child, so side effects on ordinary parent R objects are not propagated. Use serializable inputs and results; live handles are unsuitable. Linux child limits are described in hm_resources. Nested evaluation inside an already bounded child must reuse the identical outer budget. Unsupported platforms/devices fail admission rather than executing without limits.

Value

The serialized value of code returned from the child.

hm_worker_request *Exchange bounded typed arrays with an owned worker*

Description

Exchange bounded typed arrays with an owned worker

Usage

```
hm_worker_request(
  handle,
  operation = "handshake",
  array = NULL,
  channels = NULL,
  timeout = 30,
  max_bytes = 16 * 1024^2,
  metadata = list()
)
```

Arguments

handle	Managed process handle.
operation	One of handshake, predict, extract, fit or cancel.
array	Numeric array; dimensions and channel metadata are explicit.
channels	Expected measured channel identifiers.
timeout	Maximum seconds.
max_bytes	Maximum request or response bytes.
metadata	Additional JSON metadata.

Value

Decoded worker response list; any array payload is reconstructed as a finite numeric R array in declared dimensions. Cancellation returns backend status invisibly.

hm_workflow *Assemble a modelling workflow*

Description

Assemble a modelling workflow

Usage

```
hm_workflow(task, recipe, model)
```

Arguments

task	An hm_task.
recipe	An unfitted hm_recipe.
model	An hm_model_spec whose task is identical to task.

Details

Assembly validates compatible declarations; fitting and backend admission occur separately.

Value

An hm_workflow list containing the three declarations.

HMDataset-class *Validated spectral data container*

Description

Validated spectral data container

Slots

sources Native arrays or declarative RDS locators, never live handles.

observations Identity and observation metadata.

bands Native band mapping; wavelengths are in nm.

geometry Explicit spatial transform; empty for native spectra.

provenance Source, radiometry and calibration declarations.

data_kind One of spectra, image or sequence.

assets Distinct asset-role metadata.

HMSequence-class *Validated time-indexed spectral sequence*

Description

Validated time-indexed spectral sequence

Slots

frame_index Explicit frame identity and elapsed seconds; gaps are retained.

`tidy.hm_fit` *Summarize fitted coefficients in a standard table*

Description

Summarize fitted coefficients in a standard table

Usage

```
## S3 method for class 'hm_fit'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An `hm_fit`.
`conf.int`	Include conditional confidence or posterior intervals.
`conf.level`	Interval level strictly between zero and one.
`...`	Additional arguments; currently unused.

Details

Frequentist uncertainty is implemented for LM and GLM. Other coefficient engines retain unavailable uncertainty as NA. Intervals do not include model selection, preprocessing selection or biological transport uncertainty. Use `coef(x, scale = "original")` for an admitted affine map.

Value

A data frame with `term`, `estimate`, `std.error`, `statistic`, `p.value`, and optional `conf.low` and `conf.high`. Estimates use the fitted recipe coordinates. Bayesian rows are explicitly posterior summaries.

Index

augment.hm_fit, 4

coef.hm_fit *(*hm_fit*)*, 23

glance.hm_fit, 4

hm_active_learning, 5
hm_admit_backend, 6
hm_annotation_contract, 6
hm_app, 7
hm_app_state, 7
hm_apply_calibration, 8
hm_as_dataset
*(*hm_as_dataset.cuvis_session*)*, 8
hm_as_dataset.cuvis_session, 8
hm_as_hsi_cube, 9
hm_backend_capabilities, 10
hm_backend_plan, 10
hm_backend_status, 11
hm_bake, 11
hm_benchmark_protocol, 12
hm_bootstrap, 13
hm_cache, 14
hm_cache_key, 14
hm_calibrate, 15
hm_cancel_job, 16
hm_check_backend, 16
hm_compare, 17
hm_coordinates, 18
hm_dataset, 18
hm_dual_stream_mhsi, 19
hm_endpoint_contract, 19
hm_evaluate, 20
hm_example, 21
hm_export_annotations, 21
hm_export_result, 22
hm_extract_native, 22
hm_fit, 23
hm_from_tivis, 25
hm_htc, 25
hm_hyperfree, 26
hm_hypersigma, 26
hm_hypervision, 27
hm_install_backend, 27
hm_instrument_profiles, 28
hm_job_status, 28
hm_job_status(), *50*
hm_model_spec, 29
hm_models, 29
hm_physics_kernel, 30
hm_pool, 31
hm_portable_sequence, 31
hm_predict, 32
hm_prediction, 33
hm_prep, 34
hm_prompt, 35
hm_provenance, 35
hm_read_block, 36
hm_read_block,HMDataset-method
*(*hm_read_block*)*, 36
hm_read_run, 36
hm_recipe, 37
hm_reliability, 38
hm_render_hsi, 38
hm_resolve_backend, 39
hm_resolve_checkpoint, 40
hm_resources, 40
hm_save_run, 41
hm_sequence, 42
hm_spectra, 43
hm_spectra,HMDataset-method
*(*hm_spectra*)*, 43
hm_spectral_library, 43
hm_spectralearth, 44
hm_spectralgpt, 44
hm_split, 45
hm_stage_annotations, 46
hm_start_backend, 47

hm_stop_backend, 48
hm_stream, 48
hm_stream_cancel, 49
hm_stream_push, 49
hm_submit_job, 50
hm_task, 51
hm_temporal_model, 51
hm_tivita_profile, 52
hm_tune, 53
hm_verify_provenance, 54
hm_verify_registry, 54
hm_vllm_chat, 55
hm_with_resources, 56
hm_worker_request, 57
hm_workflow, 57
HMDataset-class, 58
HMSequence-class, 58

plot.hm_prediction *(*hm_prediction*)*, 33
predict.hm_fit *(*hm_predict*)*, 32
print.hm_evaluation *(*hm_evaluate*)*, 20
print.hm_fit *(*hm_fit*)*, 23
print.hm_prediction *(*hm_prediction*)*, 33

summary.hm_evaluation *(*hm_evaluate*)*, 20
summary.hm_fit *(*hm_fit*)*, 23
summary.hm_prediction *(*hm_prediction*)*, 33

tidy.hm_fit, 59