Package: hyperannotatR (via r-universe)

November 3, 2026

Type Package

Title Region-of-Interest Annotation for Hyperspectral Data

Version 0.3.1

Description Annotate multiple layers of regions of interest in hyperspectral data cubes and their associated display rasters. Annotations are stored as validated simple-feature geometries in image pixel coordinates with explicit pyramid-level transforms, and can be rasterised to binary, labelled, or multi-class integer masks for downstream segmentation and classification workflows. A batch annotation application built on 'shiny' supports resumable sessions across image queues, live mask preview, and export to 'GeoJSON', 'QuPath', and 'TIFF' formats. A versioned partner contract exchanges hash-inventoried handoff directories, deep-learning training datasets and typed commands over an optional local control service.

License MIT + file LICENSE

URL https://github.com/CTTIR/hyperannotatR, https://cttir.github.io/hyperannotatR/

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

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports cli, digest, dplyr, fs, ggplot2, grDevices, jsonlite, rlang (>= 1.1.0), sf (>= 1.0.0), stars (>= 0.6.0), stats, tibble, tools, utils

Suggests bslib (>= 0.6.0), callr, chromote, covr, curl, htmltools, htmlwidgets, httpuv, knitr, later, magick, openssl, pkgload, png, RBioFormats, rmarkdown, shiny (>= 1.7.0), shinyjs, shinytest2, terra, testthat (>= 3.3.2), tiff, withr, xml2, zip

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/Needs/website CTTIR/themakR

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-11-03 12:05:15 UTC

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

RemoteRef HEAD

RemoteSha 2ac2309744e39420cb0aa16326af68c1ada0b080

Contents

at_add_layer . . . 5
at_add_roi . . . 6
at_annotate . . . 6
at_annotation_revision . . . 8
at_app . . . 8
at_backend_detect . . . 10
at_backend_get . . . 11
at_backend_list . . . 11
at_backend_register . . . 12
at_band_operations . . . 13
at_band_view . . . 14
at_bands . . . 15
at_batch_apply . . . 16
at_batch_check_geometry . . . 17
at_canvas . . . 17
at_canvas_fit . . . 19
at_canvas_proxy . . . 20
at_canvas_set_annotations . . . 20
at_canvas_set_band . . . 21
at_canvas_set_overlay . . . 22
at_canvas_set_selection . . . 23
at_canvas_set_tool . . . 23
at_check_containment . . . 24
at_check_geometry . . . 25
at_clamp . . . 26
at_close_image . . . 27
at_commit_qupflowr . . . 27
at_control_capabilities . . . 29
at_control_close . . . 30
at_control_command . . . 30
at_control_connect . . . 31
at_control_events . . . 32

at_control_request_status . . . 33
at_control_start . . . 34
at_control_state . . . 36
at_control_stop . . . 36
at_convert_values . . . 37
at_cubert_export_envi . . . 38
at_current . . . 39
at_dims . . . 39
at_example_image . . . 40
at_example_path . . . 41
at_example_project . . . 42
at_example_session . . . 42
at_export_all . . . 43
at_export_qupflowr . . . 44
at_extract . . . 46
at_extract_pixels . . . 47
at_extract_spectrum . . . 48
at_fix_geometry . . . 49
at_flip_y . . . 50
at_goto . . . 51
at_hsi_meta . . . 51
at_image_capabilities . . . 52
at_import_qupflowr . . . 53
at_inspect_saved . . . 54
at_inspect_source . . . 55
at_interop_capabilities . . . 56
at_interop_manifest . . . 57
at_is_pyramidal . . . 58
at_is_spectral . . . 58
at_layer . . . 59
at_layer_add . . . 60
at_layer_n . . . 61
at_layer_remove . . . 61
at_layer_rois . . . 62
at_layers . . . 62
at_load_project . . . 63
at_load_session . . . 64
at_manifest . . . 64
at_mask . . . 65
at_mask_agreement . . . 67
at_mask_boundary . . . 69
at_mask_derive . . . 69
at_mask_legend . . . 71
at_mask_preview . . . 71
at_mask_stack . . . 72
at_mask_stats . . . 73
at_meta . . . 74
at_n_bands . . . 74

at_n_levels . . . 75
at_next . . . 76
at_pixel_size . . . 76
at_plot . . . 77
at_plot_image . . . 78
at_plot_mask . . . 79
at_plot_overlay . . . 80
at_plot_project . . . 80
at_plot_spectrum . . . 81
at_plot_summary . . . 82
at_prev . . . 83
at_project . . . 83
at_read_geojson . . . 84
at_read_image . . . 85
at_read_mask . . . 86
at_read_npy . . . 88
at_read_qupath . . . 89
at_read_rois_csv . . . 90
at_read_stats . . . 91
at_relink_source . . . 92
at_remove_layer . . . 93
at_remove_roi . . . 94
at_resume . . . 94
at_roi_area . . . 95
at_roi_bbox . . . 96
at_roi_buffer . . . 96
at_roi_centroid . . . 97
at_roi_circle . . . 98
at_roi_contains . . . 99
at_roi_distance . . . 99
at_roi_ellipse . . . 100
at_roi_freehand . . . 101
at_roi_from_geojson . . . 102
at_roi_from_sf . . . 103
at_roi_overlaps . . . 104
at_roi_point . . . 104
at_roi_polygon . . . 105
at_roi_rect . . . 106
at_roi_rescale . . . 107
at_roi_ring . . . 108
at_roi_setops . . . 109
at_roi_simplify . . . 110
at_rois . . . 111
at_rois_overlap . . . 112
at_save_project . . . 113
at_save_session . . . 114
at_session . . . 115
at_session_status . . . 116

at_set_status . . . 116
at_snap . . . 117
at_stage_qupflowr . . . 118
at_style . . . 119
at_summary . . . 120
at_summary_table . . . 121
at_tile . . . 121
at_tile_source . . . 123
at_tivita_profile . . . 124
at_training_check . . . 125
at_training_export . . . 126
at_training_import . . . 128
at_transform . . . 129
at_validate . . . 130
at_wavelengths . . . 131
at_write_geojson . . . 132
at_write_mask . . . 133
at_write_masks . . . 134
at_write_npy . . . 135
at_write_qupath . . . 136
at_write_rois_csv . . . 138
atCanvasOutput . . . 139
hyperannotatR-classes . . . 139
renderAtCanvas . . . 140

Index **141**

`at_add_layer` *Add a layer to a project*

Description

Add a layer to a project

Usage

```
at_add_layer(project, layer, call = rlang::caller_env())
```

Arguments

`project`	An annot_project.
`layer`	An annot_layer with a name not already used in the project.
`call`	The calling environment, for error reporting.

Value

A new annot_project with the layer added. ROIs in `layer` whose identifiers are missing, already used in the project, or repeated within the layer are re-minted; the first occurrence of each valid id is kept. The input is unchanged.

See Also

Other projects: `at_add_roi()`, `at_example_project()`, `at_layers()`, `at_project()`, `at_remove_layer()`, `at_remove_roi()`, `at_rois()`, `at_summary()`, `at_validate()`

`at_add_roi` *Add an ROI to a project layer*

Description

Add an ROI to a project layer

Usage

```
at_add_roi(project, layer, roi, call = rlang::caller_env())
```

Arguments

`project`	An annot_project.
`layer`	Single string name of the target layer.
`roi`	An annot_roi.
`call`	The calling environment, for error reporting.

Value

A new annot_project with the ROI added to the layer. A colliding ROI identifier is re-minted to keep identifiers unique across the project. The input is unchanged.

See Also

Other projects: `at_add_layer()`, `at_example_project()`, `at_layers()`, `at_project()`, `at_remove_layer()`, `at_remove_roi()`, `at_rois()`, `at_summary()`, `at_validate()`

`at_annotate` *Launch the batch annotation application*

Description

A launcher around `at_app()`: it builds the app object from explicit arguments and runs it. Nothing is exchanged through global options.

Usage

```
at_annotate(
  x = NULL,
  labels = character(),
  layers = NULL,
  out_dir = NULL,
  launch.browser = TRUE,
  port = NULL,
  ...,
  call = rlang::caller_env()
)
```

Arguments

x	What to annotate: an annot_session, annot_project, annot_image, a character vector of image paths, a directory of images, or NULL (opens the bundled example session).
labels	Character vector of the global label vocabulary.
layers	An annot_layer, a list of them, or NULL. Applied as a template to every image.
out_dir	Directory for autosave and exports. When NULL, a new folder inside R's temporary directory is used, which R deletes when it exits; a message says so at launch.
launch.browser	Passed to shiny::runApp().
port	Passed to shiny::runApp().
...	Passed to at_app() (control, read_only, view).
call	The calling environment, for error reporting.

Value

Invisible NULL. Launches a Shiny application; called for side effects.

See Also

at_app()

Other shiny: atCanvasOutput(), at_app(), at_canvas(), at_canvas_fit(), at_canvas_proxy(), at_canvas_set_annotations(), at_canvas_set_band(), at_canvas_set_overlay(), at_canvas_set_selection(), at_canvas_set_tool(), at_tile_source(), renderAtCanvas()

Examples

```
if (interactive()) {
  at_annotate(at_example_session(5))
}
```

at_annotation_revision *Annotation revision token*

Description

A content revision for annotations: the SHA-256 of the canonical record of every layer (name, labels, colours, z-order, visibility, lock) and ROI (id, layer, label, level, geometry WKB, source, lock, review status, plane and attributes). It changes whenever annotation content changes and is independent of timestamps, authors, file paths and pixel data. Partners pass it back as `expected_revision` to detect stale patches.

Usage

```
at_annotation_revision(x, call = rlang::caller_env())
```

Arguments

`x`	An annot_project, annot_layer, or annot_session (combining every materialised project and the queue statuses).
`call`	The calling environment, for error reporting.

Value

A single string `"sha256:<64 hex>"`.

See Also

Other interop: `at_commit_qupflowr()`, `at_export_qupflowr()`, `at_import_qupflowr()`, `at_interop_capabilities()`, `at_interop_manifest()`, `at_stage_qupflowr()`

Examples

```
proj <- at_project(at_example_image("cube"))
at_annotation_revision(proj)
```

at_app *Build the annotation app as a Shiny app object*

Description

Construct the batch annotation application from explicit inputs, without launching a browser or touching global options. The result can be run with `shiny::runApp()`, embedded in another Shiny application, or exercised in tests; `at_annotate()` is a launcher around it.

Usage

```
at_app(
  input = NULL,
  session = NULL,
  control = "off",
  labels = character(),
  layers = NULL,
  out_dir = NULL,
  read_only = FALSE,
  view = NULL,
  ...,
  call = rlang::caller_env()
)
```

Arguments

input	What to annotate: an annot_project, annot_image, a character vector of image paths, a directory of images, or NULL.
session	An annot_session to resume. Supply input or session, not both; with neither, the bundled example session is used.
control	"off" (default; no control service) or an at_control_handle from at_control_start() whose session the app then displays and synchronises with.
labels, layers, out_dir	Passed to at_session() when input is not a session.
read_only	Logical; disable all annotation edits (view and inspect only).
view	Optional initial band view, e.g. list(operation = "rgb", bands = c(30, 20, 10)).
...	Reserved; must be empty.
call	The calling environment, for error reporting.

Details

All Shiny sessions of one app object share its in-memory queue: reloading the page or opening a second tab continues from the latest edits instead of the session the app was built with. Edits are stored per queue entry; two tabs editing the same image keep the last edit.

Annotations autosave to the session's out_dir (unless the session was created with autosave = FALSE); the app warns when that folder is inside R's temporary directory, which is deleted when R exits. Before it first replaces an existing _session.rds that holds another queue or more ROIs, the app keeps a timestamped _session-*.previous.rds copy.

The Data page reads folder and handoff paths on the machine running R. The app is meant for local use. Set options(hyperannotatR.app_root = "/allowed/folder") to restrict canonical image paths and known ENVI/TIVITA companions to that folder. This is not an operating-system sandbox: custom readers, external references and concurrent filesystem changes need separate controls in a shared deployment.

Value

A `shiny.appobj`.

See Also

`at_annotate()`, `at_control_start()`

Other shiny: `atCanvasOutput()`, `at_annotate()`, `at_canvas()`, `at_canvas_fit()`, `at_canvas_proxy()`, `at_canvas_set_annotations()`, `at_canvas_set_band()`, `at_canvas_set_overlay()`, `at_canvas_set_selection()`, `at_canvas_set_tool()`, `at_tile_source()`, `renderAtCanvas()`

Examples

```
app <- at_app(input = at_example_path("cube"))
class(app)
## Not run:
shiny::runApp(app)

## End(Not run)
```

---

at_backend_detect *Detect the backend for a file*

Description

Detect the backend for a file

Usage

```
at_backend_detect(path, call = rlang::caller_env())
```

Arguments

`path`	Single string file path.
`call`	The calling environment, for error reporting.

Value

The name of the first available backend (in priority order) that can read `path`. Errors informatively if none can.

See Also

Other backends: `at_backend_get()`, `at_backend_list()`, `at_backend_register()`, `at_close_image()`, `at_cubert_export_envi()`, `at_image_capabilities()`, `at_read_image()`, `at_tile()`, `at_tivita_profile()`

at_backend_get *Get a registered backend*

Description

Get a registered backend

Usage

```
at_backend_get(name, call = rlang::caller_env())
```

Arguments

name	Single string backend name.
call	The calling environment, for error reporting.

Value

The annot_backend object.

See Also

Other backends: at_backend_detect(), at_backend_list(), at_backend_register(), at_close_image(), at_cubert_export_envi(), at_image_capabilities(), at_read_image(), at_tile(), at_tivita_profile()

at_backend_list *List registered image backends*

Description

List registered image backends

Usage

```
at_backend_list(call = rlang::caller_env())
```

Arguments

call	The calling environment, for error reporting.

Value

A tibble::tibble with columns name (character), description (character), extensions (character, comma-separated), and available (logical), plus metadata_only and window_read (logical; NA when unknown). A 0-row tibble with these columns when nothing is registered.

See Also

Other backends: at_backend_detect(), at_backend_get(), at_backend_register(), at_close_image(), at_cubert_export_envi(), at_image_capabilities(), at_read_image(), at_tile(), at_tivita_profile()

Examples

```
at_backend_list()
```

`at_backend_register` *Register an image backend*

Description

Backends are how hyperannotatR reads image formats. Each supplies functions to read an image's metadata, fetch tiles, detect whether it can read a file, and report whether its optional dependencies are installed.

Usage

```
at_backend_register(
  name,
  read_fn,
  tile_fn,
  detect_fn,
  available_fn,
  description = "",
  extensions = character(),
  call = rlang::caller_env(),
 capabilities = list(metadata_only = NA, window_read = NA, samples = "unqualified", io =
    "unknown"),
  close_fn = NULL
)
```

Arguments

`name`	Single string backend name (unique).
`read_fn`	`function(path, ...)` returning an annot_image.
`tile_fn`	`function(img, level, xrange, yrange, bands)` returning a numeric array `[y, x, band]`.
`detect_fn`	`function(path)` returning `TRUE` if the backend can read the file.
`available_fn`	`function()` returning `TRUE` if the backend's dependencies are installed.
`description`	Single string human-readable description.
`extensions`	Character vector of file extensions the backend handles.
`call`	The calling environment, for error reporting.
`capabilities`	Additive list declaring metadata_only, window_read, samples and io. Omitted capabilities are unknown. Readers can override these per image in meta$capabilities.
`close_fn`	Optional function(img) releasing runtime resources.

Value

The registered annot_backend, invisibly.

See Also

at_backend_list(), at_read_image()

Other backends: at_backend_detect(), at_backend_get(), at_backend_list(), at_close_image(), at_cubert_export_envi(), at_image_capabilities(), at_read_image(), at_tile(), at_tivita_profile()

---

at_band_operations *Registered band operations*

Description

List the display band operations the app and the control service accept. Derived views are display products: they carry their parent digest, band list, operation and parameters, and are never written back as analysis input.

Usage

```
at_band_operations()
```

Value

A tibble::tibble with columns operation, n_bands (integer; NA means one or more), params (comma-separated parameter names) and description.

See Also

Other hsi: at_band_view(), at_convert_values(), at_cubert_export_envi(), at_hsi_meta(), at_read_stats(), at_tivita_profile()

Examples

```
at_band_operations()
```

`at_band_view` *Compute a registered band view*

Description

Evaluate one of `at_band_operations()` on a window of an image. The result is a display product: a `[y, x, k]` array (`k = 3` for `"rgb"`, otherwise 1) with provenance attributes, which hyperannotatR never uses as analysis input.

Usage

```
at_band_view(
  img,
  operation = "rgb",
  bands = NULL,
  params = list(),
  level = 0L,
  xrange = NULL,
  yrange = NULL,
  call = rlang::caller_env()
)
```

Arguments

`img`	An annot_image.
`operation`	A registered operation name.
`bands`	Band indices (defaults depend on the operation).
`params`	Named list of operation parameters.
`level`, `xrange`, `yrange`	Passed to `at_tile()`.
`call`	The calling environment, for error reporting.

Value

A numeric array with attributes `operation`, `bands`, `wavelengths`, `params`, `parent_digest`, `transform_digest`, `level` and `display_product = TRUE`.

See Also

Other hsi: `at_band_operations()`, `at_convert_values()`, `at_cubert_export_envi()`, `at_hsi_meta()`, `at_read_stats()`, `at_tivita_profile()`

Examples

```
cube <- at_example_image("cube")
v <- at_band_view(cube, "normalized_difference", bands = c(30, 10))
attr(v, "operation")
```

at_bands *Band (channel) table*

Description

Band (channel) table

Usage

```
at_bands(x, call = rlang::caller_env())
```

Arguments

x	An annot_image.
call	The calling environment, for error reporting.

Value

A tibble::tibble with one row per band and the columns:

`index` Band index, 1-based (integer).

`name` Band name (character); a default `"Band k"` when unnamed.

`wavelength` Centre wavelength (double); `NA` when not spectral or unknown for that band. Never interpolated.

`unit` Declared spectral coordinate unit (character); `NA` when not spectral or when the unit is unknown. Scalar metadata applies to all bands; per-band units are preserved without conversion.

`fwhm` Full width at half maximum in the wavelength unit (double); `NA` when the source does not declare it.

`order` Position of the band in file/manifest order (integer).

`wavelength_status` `"ok"`, `"missing"`, `"duplicate"` or `"non_numeric"` (character).

Always has `at_n_bands(x)` rows. The first four columns are unchanged from hyperannotatR 0.1; the remaining columns were added in 0.2.0.

See Also

Other images: `at_close_image()`, `at_dims()`, `at_example_image()`, `at_hsi_meta()`, `at_image_capabilities()`, `at_is_pyramidal()`, `at_is_spectral()`, `at_meta()`, `at_n_bands()`, `at_n_levels()`, `at_pixel_size()`, `at_read_image()`, `at_tile()`, `at_wavelengths()`, `hyperannotatR-classes`

`at_batch_apply` *Apply a function to every project in a session*

Description

Apply a function to every project in a session

Usage

```
at_batch_apply(
  session,
  fn,
  ...,
  scope = "all",
  progress = TRUE,
  call = rlang::caller_env()
)
```

Arguments

`session`	An annot_session.
`fn`	A function `fn(project, image, idx, ...)` returning an annot_project.
`...`	Passed to `fn`.
`scope`	`"all"` (default), `"complete"`, `"current"`, or `"flagged"`.
`progress`	Logical; show a progress bar. Default `TRUE`.
`call`	The calling environment, for error reporting.

Value

The annot_session with modified projects. An image whose project cannot be materialised, or for which `fn` errors or does not return an annot_project, keeps its previous project; the failures are reported in one warning and attached as the `"batch_errors"` attribute, a tibble::tibble with columns `idx`, `image` and `message`.

See Also

Other batch: `at_batch_check_geometry()`, `at_export_all()`, `at_summary_table()`

at_batch_check_geometry *Check geometry across a whole session*

Description

Run `at_check_geometry()` on every materialised project in the session and stack the reports. Like `at_check_geometry()` (and unlike the assertion `at_validate()`), this returns a report rather than throwing.

Usage

```
at_batch_check_geometry(session, call = rlang::caller_env())
```

Arguments

`session`	An annot_session.
`call`	The calling environment, for error reporting.

Value

A tibble::tibble with columns `image`, `roi_id`, `issue`, and `severity`. A 0-row tibble when all geometry is valid.

See Also

`at_check_geometry()`

Other batch: `at_batch_apply()`, `at_export_all()`, `at_summary_table()`

at_canvas *An annotation canvas widget*

Description

The canvas shows one embedded, downsampled image with the project's ROIs drawn in their label colours (hidden layers are left out; locked layers cannot be erased or moved on the canvas). It pans with a drag and zooms with the mouse wheel or a two-finger pinch. When no image can be shown, it says why instead of keeping a previous image.

Usage

```
at_canvas(
  img,
  project = NULL,
  tool = "pan",
  width = NULL,
  height = NULL,
  elementId = NULL,
  options = list(),
  view = NULL,
  call = rlang::caller_env()
)
```

Arguments

`img`	An annot_image to display.
`project`	Optional annot_project whose ROIs are shown initially.
`tool`	Initial tool: one of `"pan"`, `"rect"`, `"polygon"`, `"freehand"`, `"circle"`, `"point"`, `"line"` (two clicks for start and end), `"reflection"` (request a reflection proposal), `"edit"`, `"erase"`, `"probe"` (emit a pixel position for spectra) or `"select"` (emit the clicked ROI id).
`width, height`	Widget dimensions.
`elementId`	Optional element id.
`options`	A list of extra options passed to the widget: `creationTarget` (a list with `layer`, `label` and `locked`; drawing is refused on a locked target), `selection` (ROI ids to highlight), `pointRadius` (display radius in CSS pixels, default 4; saved with new points as `attributes$point_radius`, without changing mask membership), `circleRadius` (fixed radius in level-0 image pixels; 0/default draws by dragging), and `suggestion` (an unsaved GeoJSON feature shown dashed).
`view`	Optional band view passed to `at_tile_source()`.
`call`	The calling environment, for error reporting.

Value

An `htmlwidget` object.

Shiny inputs

In Shiny, the canvas with output id `id` sends these inputs. Each value is a list `entry_id`, `revision`, `payload`: the identity rendered when the gesture began (the project's `meta$entry_id` and `meta$annotation_revision`, or the element id and `0` for a project without them) and the event data. A render or `at_canvas_set_annotations()` with a different identity cancels unfinished gestures, and stale events are never sent.

`<id>_created` A drawn GeoJSON feature in level-0 pixels; `target` holds `options$creationTarget` (`layer`, `label`) as it was when the drawing began. A click with the rectangle or circle tool is not a shape.

`<id>_edited` `roi_id`, `layer`, `label` and the moved `geometry` (sent only after an actual drag).

`<id>_erased` The erased ROI id.

`<id>_selected` The ids of the selected ROIs (possibly none).

`<id>_probed` `x`, `y`: the probed level-0 pixel position.

`<id>_reflection_seed` `x`, `y` and `target`: a clicked pixel for a reflection proposal. It does not create an ROI.

`<id>_viewport` `zoom`, `center_x`, `center_y` and bounds (`xmin`, `ymin`, `xmax`, `ymax` in level-0 pixels) after a pan or zoom.

`<id>_ready` `ready` and `error`: whether the image is displayed.

A drawn or moved shape is shown dashed until new annotations arrive, and disappears after a few seconds if none do, so the canvas does not keep showing an ROI the app did not accept.

See Also

`at_canvas_proxy()`, `at_tile_source()`

Other shiny: `atCanvasOutput()`, `at_annotate()`, `at_app()`, `at_canvas_fit()`, `at_canvas_proxy()`, `at_canvas_set_annotations()`, `at_canvas_set_band()`, `at_canvas_set_overlay()`, `at_canvas_set_selection()`, `at_canvas_set_tool()`, `at_tile_source()`, `renderAtCanvas()`

Examples

```
at_canvas(at_example_image("tissue"))
```

---

at_canvas_fit *Zoom the canvas to a bounding box*

Description

Zoom the canvas to a bounding box

Usage

```
at_canvas_fit(proxy, bbox = NULL, call = rlang::caller_env())
```

Arguments

`proxy`	An `atcanvas_proxy`.
`bbox`	Optional `c(xmin, ymin, xmax, ymax)` in level-0 pixels; fits the whole image when `NULL`.
`call`	The calling environment, for error reporting.

Value

The proxy, invisibly.

See Also

Other shiny: atCanvasOutput(), at_annotate(), at_app(), at_canvas(), at_canvas_proxy(), at_canvas_set_annotations(), at_canvas_set_band(), at_canvas_set_overlay(), at_canvas_set_selection(), at_canvas_set_tool(), at_tile_source(), renderAtCanvas()

---

at_canvas_proxy *A proxy for an existing annotation canvas*

Description

Mirrors the leafletProxy() pattern: update a live canvas without re-rendering the whole widget. Inside a Shiny module, pass the module's (un-namespaced) output id; it is namespaced with the module session, as leaflet::leafletProxy() does.

Usage

```
at_canvas_proxy(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

id	The canvas output id.
session	The Shiny session.

Value

An atcanvas_proxy object.

See Also

Other shiny: atCanvasOutput(), at_annotate(), at_app(), at_canvas(), at_canvas_fit(), at_canvas_set_annotations(), at_canvas_set_band(), at_canvas_set_overlay(), at_canvas_set_selection(), at_canvas_set_tool(), at_tile_source(), renderAtCanvas()

---

at_canvas_set_annotations *Replace the annotations on a canvas*

Description

Replace the annotations on a canvas

Usage

```
at_canvas_set_annotations(
  proxy,
  project,
  layer = NULL,
  call = rlang::caller_env()
)
```

Arguments

proxy	An atcanvas_proxy.
project	An annot_project.
layer	Optional layer filter.
call	The calling environment, for error reporting.

Value

The proxy, invisibly.

See Also

Other shiny: atCanvasOutput(), at_annotate(), at_app(), at_canvas(), at_canvas_fit(), at_canvas_proxy(), at_canvas_set_band(), at_canvas_set_overlay(), at_canvas_set_selection(), at_canvas_set_tool(), at_tile_source(), renderAtCanvas()

---

at_canvas_set_band *Set the displayed band or RGB triple*

Description

Retained for API compatibility; it has no visible effect on the built-in canvas. The canvas shows one embedded image rendered in R, and cannot recompute bands in the browser. To change the displayed bands, re-render the widget with a view (see at_canvas() and at_band_view()), as the annotation app does.

Usage

```
at_canvas_set_band(proxy, band, call = rlang::caller_env())
```

Arguments

proxy	An atcanvas_proxy.
band	A band index or a length-3 vector of band indices.
call	The calling environment, for error reporting.

Value

The proxy, invisibly.

See Also

Other shiny: atCanvasOutput(), at_annotate(), at_app(), at_canvas(), at_canvas_fit(), at_canvas_proxy(), at_canvas_set_annotations(), at_canvas_set_overlay(), at_canvas_set_selection(), at_canvas_set_tool(), at_tile_source(), renderAtCanvas()

---

at_canvas_set_overlay *Overlay a mask on the canvas*

Description

Overlay a mask on the canvas

Usage

```
at_canvas_set_overlay(proxy, mask, alpha = 0.5, call = rlang::caller_env())
```

Arguments

proxy	An atcanvas_proxy.
mask	An annot_mask.
alpha	Overlay opacity. Default 0.5.
call	The calling environment, for error reporting.

Value

The proxy, invisibly.

See Also

Other shiny: atCanvasOutput(), at_annotate(), at_app(), at_canvas(), at_canvas_fit(), at_canvas_proxy(), at_canvas_set_annotations(), at_canvas_set_band(), at_canvas_set_selection(), at_canvas_set_tool(), at_tile_source(), renderAtCanvas()

at_canvas_set_selection *Highlight selected ROIs on a canvas*

Description

Highlight selected ROIs on a canvas

Usage

```
at_canvas_set_selection(
  proxy,
  roi_ids = character(),
  call = rlang::caller_env()
)
```

Arguments

`proxy`	An `atcanvas_proxy`.
`roi_ids`	Character vector of ROI ids to highlight (empty clears).
`call`	The calling environment, for error reporting.

Value

The proxy, invisibly.

See Also

Other shiny: `atCanvasOutput()`, `at_annotate()`, `at_app()`, `at_canvas()`, `at_canvas_fit()`, `at_canvas_proxy()`, `at_canvas_set_annotations()`, `at_canvas_set_band()`, `at_canvas_set_overlay()`, `at_canvas_set_tool()`, `at_tile_source()`, `renderAtCanvas()`

at_canvas_set_tool *Set the active drawing tool*

Description

Set the active drawing tool

Usage

```
at_canvas_set_tool(proxy, tool, call = rlang::caller_env())
```

Arguments

proxy	An atcanvas_proxy.
tool	One of the canvas tools.
call	The calling environment, for error reporting.

Value

The proxy, invisibly.

See Also

Other shiny: atCanvasOutput(), at_annotate(), at_app(), at_canvas(), at_canvas_fit(), at_canvas_proxy(), at_canvas_set_annotations(), at_canvas_set_band(), at_canvas_set_overlay(), at_canvas_set_selection(), at_tile_source(), renderAtCanvas()

---

at_check_containment *Check cross-layer containment*

Description

Report ROIs in a constrained layer that fall outside their container region. A layer declares the constraint through its metadata, e.g. at_layer("state", within = list(layer = "anatomy", label = "wound")); then every ROI in state must be spatially covered by the union of the anatomy layer's wound ROIs. This enforces the annotation rule that a class is painted only inside its region of interest. The report mirrors the shape of at_check_geometry().

Usage

```
at_check_containment(project, tol = 0, call = rlang::caller_env())
```

Arguments

project	An annot_project.
tol	Non-negative slack, in level-0 pixels, by which the container is grown before testing, to tolerate sub-pixel tracing overhang. Default 0.
call	The calling environment, for error reporting.

Value

A tibble::tibble with columns roi_id, layer, issue, severity ("warning" for an ROI outside its container, "error" for a missing/empty container), and fixable (always FALSE). A 0-row tibble when every constrained ROI is contained, or when no layer declares a within constraint.

See Also

at_check_geometry(), at_validate()

Other geometry: at_check_geometry(), at_clamp(), at_fix_geometry(), at_flip_y(), at_roi_buffer(), at_roi_contains(), at_roi_distance(), at_roi_overlaps(), at_roi_rescale(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_rois_overlap(), at_snap(), at_transform()

Examples

```
anatomy <- at_layer_add(at_layer("anatomy", labels = "wound"),
                        at_roi_rect(2, 2, 8, 8, label = "wound"))
state <- at_layer("state", labels = "injury",
                  within = list(layer = "anatomy", label = "wound"))
state <- at_layer_add(state, at_roi_rect(0, 0, 4, 4, label = "injury"))
at_check_containment(at_project(at_example_image("cube"), list(anatomy, state)))
```

---

at_check_geometry *Check the geometry of every ROI in a project*

Description

Check the geometry of every ROI in a project

Usage

```
at_check_geometry(project, call = rlang::caller_env())
```

Arguments

project	An annot_project.
call	The calling environment, for error reporting.

Value

A tibble::tibble with columns roi_id (character), issue (character), severity (character, "error"/"warning"), and fixable (logical). Detected issues: self-intersection, invalid geometry, zero area, degenerate rings, out-of-bounds vertices, duplicate consecutive vertices, ring orientation, NA coordinates, empty geometry, and a pyramid level the image does not have (level_out_of_range). A 0-row tibble when all geometry is valid.

See Also

at_fix_geometry(), at_validate()

Other geometry: at_check_containment(), at_clamp(), at_fix_geometry(), at_flip_y(), at_roi_buffer(), at_roi_contains(), at_roi_distance(), at_roi_overlaps(), at_roi_rescale(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_rois_overlap(), at_snap(), at_transform()

Examples

```
at_check_geometry(at_example_project())
```

`at_clamp` *Clip an ROI to image bounds*

Description

Clip an ROI to image bounds

Usage

```
at_clamp(roi, dims, call = rlang::caller_env())
```

Arguments

`roi`	An annot_roi.
`dims`	Integer `c(width, height)` bounds in pixels.
`call`	The calling environment, for error reporting.

Value

An annot_roi clipped to `[0, width] x [0, height]`. Warns if the clip empties the geometry.

See Also

Other geometry: `at_check_containment()`, `at_check_geometry()`, `at_fix_geometry()`, `at_flip_y()`, `at_roi_buffer()`, `at_roi_contains()`, `at_roi_distance()`, `at_roi_overlaps()`, `at_roi_rescale()`, `at_roi_ring()`, `at_roi_setops`, `at_roi_simplify()`, `at_rois_overlap()`, `at_snap()`, `at_transform()`

Examples

```
at_clamp(at_roi_rect(-5, -5, 20, 20, label = "a"), dims = c(10, 10))
```

at_close_image *Release an image handle*

Description

Drop an image's runtime pixel handle, calling the backend's `close_fn` when it declares one.

Usage

```
at_close_image(x)
```

Arguments

`x`	An annot_image or annot_project.

Value

The image or project with its runtime handle released; assign the result. Later tile requests reopen the image from its source. Releasing a retained pixel snapshot loses those pixels when the source has changed or cannot be reopened. Built-in window readers open a connection per read and hold no persistent connection.

See Also

Other images: `at_bands()`, `at_dims()`, `at_example_image()`, `at_hsi_meta()`, `at_image_capabilities()`, `at_is_pyramidal()`, `at_is_spectral()`, `at_meta()`, `at_n_bands()`, `at_n_levels()`, `at_pixel_size()`, `at_read_image()`, `at_tile()`, `at_wavelengths()`, `hyperannotatR-classes`

Other backends: `at_backend_detect()`, `at_backend_get()`, `at_backend_list()`, `at_backend_register()`, `at_cubert_export_envi()`, `at_image_capabilities()`, `at_read_image()`, `at_tile()`, `at_tivita_profile()`

Examples

```
cube <- at_close_image(at_example_image("cube"))
dim(at_tile(cube, xrange = c(1, 4), yrange = c(1, 4), bands = 1))
```

at_commit_qupflowr *Commit a staged partner patch*

Description

Apply an at_stage_qupflowr() patch. Conflicting operations are never applied. The commit is idempotent: repeating it with the same idempotency_key and the same patch returns the original receipt without applying anything again, while the same key with a different patch aborts with IDEMPOTENCY_CONFLICT. When the patch targets a saved project file (or destination names one), the file is re-read and its revision re-checked immediately before writing; an existing file is only replaced with overwrite = TRUE. A <file>.commit.json receipt records the key across R sessions. The project and receipt are staged together; a failed publication restores both previous files, or reports retained recovery backups if restoration also fails. Writers to the same destination must be serialised; this is not a cross-file transaction against process or machine failure. A replay aborts with REVISION_CONFLICT if the saved project has changed since that receipt was written. Reusing a cached key for a different destination aborts with IDEMPOTENCY_CONFLICT.

Usage

```
at_commit_qupflowr(
  x,
  idempotency_key = NULL,
  destination = NULL,
  overwrite = FALSE,
  call = rlang::caller_env()
)
```

Arguments

x	An at_staged_patch.
idempotency_key	Optional client key; defaults to one derived from the patch id.
destination	Optional project .rds path to write the committed project to; defaults to the patch's target_path when staged from a file.
overwrite	Logical; allow replacing an existing project file.
call	The calling environment, for error reporting.

Value

An at_commit_receipt: a list with state = "committed", patch_id, idempotency_key, previous_revision, new_revision, applied and conflicts (operation tibbles), project, written (path or NA), committed_at and replayed.

See Also

at_stage_qupflowr()

Other interop: at_annotation_revision(), at_export_qupflowr(), at_import_qupflowr(), at_interop_capabilities(), at_interop_manifest(), at_stage_qupflowr()

Examples

```
proj <- at_example_project()
partner <- at_layer_add(at_layer("regions"), at_roi_rect(10, 10, 40, 40, label = "tumour",
                                                         id = "partner-1"))
rc <- at_commit_qupflowr(at_stage_qupflowr(proj, partner))
rc$new_revision
```

---

at_control_capabilities

Control protocol capabilities

Description

Describe the `annotatr-control-v1` protocol this hyperannotatR version speaks, without starting anything: endpoints, typed operations, limits, security rules and the digests of the shipped JSON Schemas. A client compares this (or the `/v1/handshake` response) with its own expectations before sending commands.

Usage

```
at_control_capabilities()
```

Value

A list with `protocol`, `protocol_version`, `annotatr_version`, `status` (see `at_interop_capabilities()`), `endpoints`, `operations`, `limits`, `security`, `schemas` (file name -> SHA-256) and `digest`.

See Also

`at_control_start()`, `at_control_command()`

Other control: `at_control_close()`, `at_control_command()`, `at_control_connect()`, `at_control_events()`, `at_control_request_status()`, `at_control_start()`, `at_control_state()`, `at_control_stop()`

Examples

```
cc <- at_control_capabilities()
cc$operations
```

at_control_close *Close a control service connection*

Description

Close the control service a handle refers to. For a local handle this is `at_control_stop()`; for a remote handle it sends `POST /v1/close`, after which that service stops accepting requests while its host process and any app keep running. Neither path terminates a process. A close is not revision-checked, so concurrent edits in an attached app never make it fail; unsaved app edits stay in the app.

Usage

```
at_control_close(handle, call = rlang::caller_env())
```

Arguments

`handle`	An `at_control_handle`.
`call`	The calling environment, for error reporting.

Value

The close response (remote) or `TRUE` (local), invisibly.

See Also

Other control: `at_control_capabilities()`, `at_control_command()`, `at_control_connect()`, `at_control_events()`, `at_control_request_status()`, `at_control_start()`, `at_control_state()`, `at_control_stop()`

at_control_command *Send a typed control command*

Description

Send one `annotatr-control-v1` operation. The command is a list with `operation` (one of `at_control_capabilities()$c` `payload` (a list) and `expected_revision` (the `state_revision` the command is based on). `request_id` and `idempotency_key` are generated when absent and `content_sha256` is always computed from the canonical payload. A stale revision raises an `at_conflict_error` (`REVISION_CONFLICT`); re-sending the same command with the same idempotency key returns the original result (`replayed = TRUE` in the response) instead of applying it twice. The service keeps the most recent 1000 request records; a replay of an older key raises `IDEMPOTENCY_EXPIRED` instead of executing again, and a large stored response (for example a mask preview PNG) is replayed in compact form with `response_truncated = TRUE`. `close` is the only operation that does not check `expected_revision`. Errors carry the same condition class as the corresponding local call.

Usage

```
at_control_command(handle, command, call = rlang::caller_env())
```

Arguments

handle	An at_control_handle.
command	A list with operation, payload, expected_revision and optionally request_id, idempotency_key and extensions.
call	The calling environment, for error reporting.

Value

The response data (a list) with the response envelope in attr(, "response").

See Also

at_control_request_status()

Other control: at_control_capabilities(), at_control_close(), at_control_connect(), at_control_events(), at_control_request_status(), at_control_start(), at_control_state(), at_control_stop()

Examples

```
sess <- at_session(rep(at_example_path("cube"), 3))
h <- at_control_start(sess, serve = FALSE)
st <- at_control_state(h)
at_control_command(h, list(operation = "context.goto", payload = list(queue_index = 2),
                           expected_revision = st$state_revision))
at_control_stop(h)
```

---

at_control_connect *Connect to a running control service*

Description

Build a remote handle for a control service started in another R process, from its lifecycle manifest and token file. The handshake is checked before the handle is returned: protocol, major version and instance id must match the manifest.

Usage

```
at_control_connect(manifest, token_file, call = rlang::caller_env())
```

Arguments

manifest	Path of the annotatr-control-<instance_id>.json manifest.
token_file	Path of the token file passed to at_control_start().
call	The calling environment, for error reporting.

Value

An at_control_handle of kind "remote".

See Also

Other control: at_control_capabilities(), at_control_close(), at_control_command(), at_control_events(), at_control_request_status(), at_control_start(), at_control_state(), at_control_stop()

---

at_control_events *Read control events after a cursor*

Description

Read control events after a cursor

Usage

```
at_control_events(
  handle,
  after = NULL,
  limit = 100L,
  call = rlang::caller_env()
)
```

Arguments

handle	An at_control_handle.
after	Optional event cursor ("<instance_id>:<seq>"); NULL returns the retained buffer. A cursor from another instance or older than the buffer yields resync_required = TRUE.
limit	Maximum number of events to return (a positive integer; the service caps a page at 500).
call	The calling environment, for error reporting.

Value

A list with events, next_cursor, resync_required, has_more and state_revision.

See Also

Other control: at_control_capabilities(), at_control_close(), at_control_command(), at_control_connect(), at_control_request_status(), at_control_start(), at_control_state(), at_control_stop()

`at_control_request_status` *Resolve the outcome of a control request*

Description

After a lost response, ask the service what happened to a request id. An unknown id raises `REQUEST_NOT_FOUND` and has no side effect.

Usage

```
at_control_request_status(handle, request_id, call = rlang::caller_env())
```

Arguments

`handle`	An `at_control_handle`.
`request_id`	The `request_id` of the earlier command.
`call`	The calling environment, for error reporting.

Value

The request record: `request_id`, `operation`, `status` (`"completed"` or `"failed"`), `http_status`, timestamps, `extensions` and the stored `response`.

See Also

Other control: at_control_capabilities(), at_control_close(), at_control_command(), at_control_connect(), at_control_events(), at_control_start(), at_control_state(), at_control_stop()

`at_control_start` *Start the local control service*

Description

Start an `annotatr-control-v1` service for a session so that a partner process (for example qupflowR) can read state and events and send typed commands. Nothing is started unless this function is called with `control = "loopback"`. The service binds only to `127.0.0.1`, requires a random 256-bit bearer token that expires after `ttl_seconds`, rejects browser-origin requests, enforces JSON size and path limits, and resolves file references only below `root`. There is no endpoint that evaluates code.

Usage

```
at_control_start(
  session,
  control = "loopback",
  host = "127.0.0.1",
  port = 0L,
  ttl_seconds = 900L,
  root = NULL,
  token_file = NULL,
  read_only = FALSE,
  max_body_bytes = 1048576L,
  serve = TRUE,
  call = rlang::caller_env()
)
```

Arguments

`session`	What to control: an annot_session (or anything `at_annotate()` accepts).
`control`	Must be `"loopback"`.
`host`	Must be `"127.0.0.1"`.
`port`	Port; `0` picks a random free port.
`ttl_seconds`	Token lifetime in seconds (1 to 86400).
`root`	Directory for payload references, exports, training exports and the lifecycle manifest; defaults to `<session out_dir>/control`.
`token_file`	Optional path of a new file to receive the token (for a client process). It must not exist yet. The token needs a cryptographically secure random source (`/dev/urandom`, or the **openssl** package where there is none, as on Windows); without one the service is not started (`CAPABILITY_UNAVAILABLE`).
`read_only`	Logical; refuse annotation mutations.
`max_body_bytes`	Maximum request body size.

serve	Logical; start the HTTP listener. With FALSE only in-process calls through the handle are possible (useful in tests).
call	The calling environment, for error reporting.

Details

The token is never written into URLs, logs, downloads, manifests or RDS files. In-process calls use the returned handle; a separate client process reads the token from token_file, which is created exclusively with owner-only permissions (on Windows, file modes are not enforced; keep the file in a directory only the user can read). A sanitised lifecycle manifest annotatr-control-<instance_id>.json is written to root at start and rewritten when the service stops. If startup fails, its listener, expiry timer, registry entries and newly created token file are removed.

The service stops itself when the token expires: either when the expiry timer runs or, if a request arrives first (for example after the computer slept), on that request. Error messages sent to clients never contain absolute local paths. Request bodies whose declared Content-Length exceeds max_body_bytes, and requests that fail the Host, Origin or token checks, are refused before the body is read (a chunked body without Content-Length is buffered by httpuv and then refused).

Two options bound work done for clients: hyperannotatR.control_mask_max_pixels (default 4096^2) is the largest mask mask.preview builds (without a level it picks the finest pyramid level within the budget), and hyperannotatR.control_hash_max_bytes (default 256 MiB) is the largest queue entry whose content id GET /v1/state computes before the entry is visited (larger entries report image_id: null until then).

Value

An at_control_handle (a list with kind = "local", instance_id, session_id, host, port, pid, manifest, expires_at); its print method never shows the token.

See Also

at_control_stop(), at_control_command(), at_app()

Other control: at_control_capabilities(), at_control_close(), at_control_command(), at_control_connect(), at_control_events(), at_control_request_status(), at_control_state(), at_control_stop()

Examples

```
sess <- at_session(rep(at_example_path("cube"), 2))
h <- at_control_start(sess, serve = FALSE)
at_control_state(h)$state_revision
at_control_stop(h)
```

at_control_state *Read the state of a control service*

Description

Read the state of a control service

Usage

```
at_control_state(handle, call = rlang::caller_env())
```

Arguments

handle	An at_control_handle.
call	The calling environment, for error reporting.

Value

The state snapshot (a list) with state_revision, event_cursor, queue, current (image, layers, ROI summaries, annotation revision), view, selection, staged, last_commit and status fields.

See Also

Other control: at_control_capabilities(), at_control_close(), at_control_command(), at_control_connect(), at_control_events(), at_control_request_status(), at_control_start(), at_control_stop()

at_control_stop *Stop a local control service*

Description

Stop a service started by at_control_start() in this R process: close the listener, invalidate its token and rewrite its lifecycle manifest as stopped. A handle from another process (or a remote handle from at_control_connect()) is refused with an at_auth_error (NOT_OWNER); no process is ever terminated.

Usage

```
at_control_stop(handle, call = rlang::caller_env())
```

Arguments

handle	An at_control_handle of kind "local".
call	The calling environment, for error reporting.

Value

TRUE invisibly when the service was stopped, FALSE if it had already stopped.

See Also

Other control: at_control_capabilities(), at_control_close(), at_control_command(), at_control_connect(), at_control_events(), at_control_request_status(), at_control_start(), at_control_state()

---

at_convert_values *Convert spectral values between declared units*

Description

The only supported ways to change value semantics. Raw counts become reflectance through an explicit dark/white calibration, and reflectance becomes (pseudo-)absorbance as -log10(reflectance). Anything else, or a conversion without the calibration it needs, is refused with a classified error instead of being approximated.

Usage

```
at_convert_values(
  x,
  from = NULL,
  to = c("reflectance", "absorbance"),
  calibration = NULL,
  call = rlang::caller_env()
)
```

Arguments

x	A numeric vector, matrix or [y, x, band] array of values.
from	The unit of x: one of "raw", "reflectance", "radiance", "intensity", "absorbance". Taken from attr(x, "value_unit") when NULL.
to	Target unit: "reflectance" or "absorbance".
calibration	For raw -> reflectance, a list with finite numeric white and dark reference vectors: scalars, or one value per band. Factors, complex values and non-finite references are refused. Per-band references apply along the band axis of a [y, x, band] array, the columns of a pixel-by-band matrix, or the elements of a single spectrum vector; any other length is refused.
call	The calling environment, for error reporting.

Value

x converted, with attribute value_unit set to to and conversion recording the operation. Reflectance values <= 0 give NA absorbance; a zero white-minus-dark denominator gives NA.

See Also

Other hsi: at_band_operations(), at_band_view(), at_cubert_export_envi(), at_hsi_meta(), at_read_stats(), at_tivita_profile()

Examples

```
at_convert_values(c(0.5, 0.1), from = "reflectance", to = "absorbance")
at_convert_values(c(100, 550, 1000), from = "raw", to = "reflectance",
                  calibration = list(white = 1000, dark = 100))
```

---

at_cubert_export_envi *Export a Cubert measurement to a portable ENVI cube*

Description

The portable fallback for Cubert data: the CUVIS SDK itself writes an ENVI export, which hyperannotatR (and tools without the SDK) can then read through the `envi` backend. Requires `cuvis.r` and a usable CUVIS SDK.

Usage

```
at_cubert_export_envi(path, dir, measurement = 1L, call = rlang::caller_env())
```

Arguments

`path`	Path to a `.cu3s` session file.
`dir`	Output directory for the SDK's ENVI export (created if needed).
`measurement`	1-based measurement index within the session.
`call`	The calling environment, for error reporting.

Value

The files written into `dir` (character), invisibly.

See Also

Other backends: at_backend_detect(), at_backend_get(), at_backend_list(), at_backend_register(), at_close_image(), at_image_capabilities(), at_read_image(), at_tile(), at_tivita_profile()

Other hsi: at_band_operations(), at_band_view(), at_convert_values(), at_hsi_meta(), at_read_stats(), at_tivita_profile()

Examples

```
## Not run:
at_cubert_export_envi("measurement.cu3s", tempfile("envi-"))

## End(Not run)
```

at_current *The current project*

Description

Return the project for the image at the cursor, materialising it from the image (and applying the session's layer template) if it has not been visited. A stored project whose pixel handle was not saved is reopened from its recorded source (see `at_inspect_source()` when that is impossible).

Usage

```
at_current(session, call = rlang::caller_env())
```

Arguments

`session`	An annot_session.
`call`	The calling environment, for error reporting.

Value

An annot_project.

See Also

Other sessions: `at_example_session()`, `at_goto()`, `at_manifest()`, `at_next()`, `at_prev()`, `at_resume()`, `at_session()`, `at_session_status()`, `at_set_status()`

at_dims *Image dimensions at a pyramid level*

Description

Return the pixel dimensions `c(width, height)` of an annot_image at a given pyramid level.

Usage

```
at_dims(x, level = 0L, call = rlang::caller_env())

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

Arguments

`x`	An annot_image.
`level`	Integer pyramid level, `0` for full resolution. Must be in `0:(at_n_levels(x) - 1)`.
`call`	The calling environment, for error reporting.

Details

dim() on an annot_image returns c(width, height, bands) at level 0. Note the order: width first, unlike the [y, x, band] arrays returned by at_tile(), so nrow() gives the width.

Value

An integer vector of length 2, c(width, height).

See Also

at_n_levels(), at_is_pyramidal()

Other images: at_bands(), at_close_image(), at_example_image(), at_hsi_meta(), at_image_capabilities(), at_is_pyramidal(), at_is_spectral(), at_meta(), at_n_bands(), at_n_levels(), at_pixel_size(), at_read_image(), at_tile(), at_wavelengths(), hyperannotatR-classes

Examples

```
# Image handles come from at_read_image(); see the images vignette.
```

---

at_example_image *Read a bundled example image*

Description

Read a bundled example image

Usage

```
at_example_image(
  which = c("tissue", "multiplex", "cube"),
  call = rlang::caller_env()
)
```

Arguments

which	One of "tissue" (RGB raster), "multiplex" (4-channel pyramidal TIFF), or "cube" (40-band ENVI spectral cube).
call	The calling environment, for error reporting.

Details

The tissue example is a PNG and needs **magick**; the multiplex example needs **tiff**; the cube needs nothing beyond hyperannotatR.

Value

An annot_image.

See Also

Other example data: at_example_path(), at_example_project(), at_example_session()

Other images: at_bands(), at_close_image(), at_dims(), at_hsi_meta(), at_image_capabilities(), at_is_pyramidal(), at_is_spectral(), at_meta(), at_n_bands(), at_n_levels(), at_pixel_size(), at_read_image(), at_tile(), at_wavelengths(), hyperannotatR-classes

Examples

```
img <- at_example_image("cube")
at_dims(img)
```

---

at_example_path *Path to a bundled example file*

Description

Path to a bundled example file

Usage

```
at_example_path(
  which = c("tissue", "multiplex", "cube"),
  call = rlang::caller_env()
)
```

Arguments

which	One of "tissue", "multiplex", or "cube".
call	The calling environment, for error reporting.

Value

A single string file path to the example asset.

See Also

Other example data: at_example_image(), at_example_project(), at_example_session()

Examples

```
at_example_path("tissue")
```

at_example_project *A populated example project*

Description

A populated example project

Usage

```
at_example_project(call = rlang::caller_env())
```

Arguments

call	The calling environment, for error reporting.

Value

An annot_project over the tissue example with one layer of three labelled regions.

See Also

Other example data: at_example_image(), at_example_path(), at_example_session()

Other projects: at_add_layer(), at_add_roi(), at_layers(), at_project(), at_remove_layer(), at_remove_roi(), at_rois(), at_summary(), at_validate()

Examples

```
proj <- at_example_project()
at_rois(proj)
```

at_example_session *An example batch session*

Description

Copies the tissue example into a temporary directory n times so the queue has real files to iterate.

Usage

```
at_example_session(n = 5L, call = rlang::caller_env())
```

Arguments

n	Number of images in the queue. Default 5.
call	The calling environment, for error reporting.

Value

An annot_session.

See Also

Other example data: `at_example_image()`, `at_example_path()`, `at_example_project()`

Other sessions: `at_current()`, `at_goto()`, `at_manifest()`, `at_next()`, `at_prev()`, `at_resume()`, `at_session()`, `at_session_status()`, `at_set_status()`

Examples

```
sess <- at_example_session(3)
at_session_status(sess)
```

---

at_export_all *Export a whole session*

Description

Export every selected image's annotations in the requested formats, isolating per-image failures so one bad image never aborts the run.

Usage

```
at_export_all(
  session,
  dir,
  formats = c("mask_tiff", "geojson", "qupath", "rds", "csv"),
  scope = c("complete", "all", "current", "flagged"),
  mask_type = c("labelled", "binary", "multiclass"),
  level = 0L,
  overwrite = FALSE,
  progress = TRUE,
  call = rlang::caller_env()
)
```

Arguments

session	An annot_session.
dir	Output directory (created if needed).
formats	Any of "mask_tiff", "geojson", "qupath", "rds", "csv".
scope	"complete" (default), "all", "current", or "flagged".
mask_type	Mask type for "mask_tiff".
level	Integer pyramid level. Default 0.
overwrite	Logical; overwrite existing files. Default FALSE.
progress	Logical; show a progress bar. Default TRUE.
call	The calling environment, for error reporting.

Details

Derived filenames use safe portable components; display names stay in the receipt and annotation metadata. Case-folded collisions are given deterministic suffixes, including cross-format and sidecar collisions. The complete plan (including manifests) rejects descendant symlinks before writing outputs. Existing mask/legend bundles are skipped as a whole unless `overwrite=TRUE`. Files are staged and publication failures roll back prior files. Callers must serialize concurrent exports and avoid changing the destination filesystem during a run: preflight and multiple renames are not a race-free filesystem transaction. Receipt files are replaced each run, independently of the item overwrite policy.

Value

An invisible export-receipt tibble::tibble with columns `entry_id`, `image`, `format`, `path`, `bytes`, `n_rois`, `status`, and `message`. Also written to `_export_manifest.csv` alongside the exports. Additive columns `sidecar_path` and `sidecar_bytes` describe mask legends. Byte counts are doubles. Status is `"ok"` for a complete published bundle, `"error"` for a failed item, or `"skipped"` when an existing bundle is protected. Every image-format request has a row, even if loading the image fails. Receipt publication or annotation-summary failures warn and are also retained in `receipt_error` / `summary_error` attributes; the item receipt is returned.

See Also

`at_write_masks()`, `at_manifest()`

Other batch: `at_batch_apply()`, `at_batch_check_geometry()`, `at_summary_table()`

Examples

```
sess <- at_example_session(2)
dir <- file.path(tempdir(), "hyperannotatR-export")
at_export_all(sess, dir, formats = "geojson", scope = "all", progress = FALSE)
```

---

at_export_qupflowr *Export annotations as a qupflowR handoff directory*

Description

Write a new, self-contained handoff directory for a partner package such as qupflowR. It contains only neutral, relative files: `image.json`, `annotations_qupath.geojson` (QuPath 0.4+ dialect), optionally `annotations.geojson`, one integer mask per layer under `masks/` with a `.legend.json` sidecar, optional `tables/rois.csv`, an optional hyperannotatR-internal `objects/project.rds`, `hsi_manifest.json` for spectral images, `manifest.json` (`at_interop_manifest()`) and `integrity.json`, the SHA-256 inventory of every other file. The handoff digest is the SHA-256 of the canonical `integrity.json`. Absolute paths are never used as identity.

Usage

```
at_export_qupflowr(
  x,
  destination,
  formats = c("qupath_geojson", "mask_tiff", "manifest"),
  overwrite = FALSE,
  level = 0L,
  mask_type = c("labelled", "multiclass", "binary"),
  scope = c("all", "current", "complete"),
  call = rlang::caller_env()
)
```

Arguments

x	An annot_project or annot_session (materialised entries are exported under entries/<entry_id>/, where entry_id is the session's stable entry id, or entry-NNNN by queue position for sessions without stable ids).
destination	Path of the handoff directory to create.
formats	Any of "qupath_geojson", "geojson", "mask_tiff", "mask_npy", "csv", "rds", "manifest". manifest.json and integrity.json are always written.
overwrite	Logical; replace an existing handoff directory.
level	Integer pyramid level for coordinates and masks. Default 0.
mask_type	Mask type for mask files: "labelled" (default; one value per ROI, joinable by roi_id through the legend), "multiclass" or "binary".
scope	For sessions: "all" materialised entries (default), "current" or "complete".
call	The calling environment, for error reporting.

Details

Files are written into a hidden sibling staging directory and moved into place only when complete, so a failed export never leaves a directory that looks valid. An existing destination is replaced only with overwrite = TRUE, and only when it is itself a handoff directory; if the new directory cannot be moved into place the previous one is restored. The optional objects/project.rds holds no pixels, read identity or local path (the image source is reduced to its file name).

Value

An at_handoff_receipt (invisibly): a list with destination, handoff_digest, files (a tibble of path, size_bytes, sha256, format, role, entry_id), manifest, skipped entries and overwritten.

See Also

at_import_qupflowr(), at_interop_manifest()

Other interop: at_annotation_revision(), at_commit_qupflowr(), at_import_qupflowr(), at_interop_capabilities(), at_interop_manifest(), at_stage_qupflowr()

Examples

```
dest <- file.path(tempdir(), "handoff-example")
rc <- at_export_qupflowr(at_example_project(), dest, overwrite = TRUE)
rc$handoff_digest
rc$files[, c("path", "format")]
```

`at_extract` *Extract summary statistics under ROIs*

Description

Compute per-band summary statistics of image values inside each ROI. Reads one tile covering each ROI's clipped bounding box; an image-sized bounding box can allocate an image-sized tile. Polygon centre membership and point containing cells follow the pixel-coverage contract in `at_mask()`. For example, point `(5, 5)` selects matrix row 6, column 6.

Usage

```
at_extract(
  project,
  img = NULL,
  stat = "mean",
  layer = NULL,
  label = NULL,
  level = 0L,
  bands = NULL,
  call = rlang::caller_env(),
  nonfinite = c("propagate", "omit", "error")
)
```

Arguments

`project`	An annot_project.
`img`	An annot_image; defaults to the project's image. Supplying an alternate image asserts spatial registration on the same level-0 pixel grid. Its level-0 extent must match; differing pyramid dimensions are handled using the alternate image's actual level dimensions.
`stat`	One or more of `"mean"`, `"median"`, `"sd"`, `"min"`, `"max"`, `"sum"`, `"n"`, or `"all"` (every statistic).
`layer`, `label`	Optional filters.
`level`	Integer pyramid level. Default `0`.
`bands`	Optional band indices; all bands by default.
`call`	The calling environment, for error reporting.

`nonfinite` Policy for NA, NaN and infinite samples: `"propagate"` (default) returns NA statistics for any affected ROI/band; `"omit"` uses finite samples only; `"error"` aborts on selected nonfinite samples. `n` always counts selected pixels. A wholly invalid selection has NA statistics under omit (including sum); singleton sample SD is NA.

Value

A long tibble::tibble with columns `roi_id`, `layer`, `label`, band (integer), `band_name`, `wavelength` (`NA` when not spectral), `unit` (declared wavelength unit, NA if unknown), `stat`, `value`, `n_px` (selected support), `n_valid` (finite samples), and `n_invalid`. Empty selections produce zero rows. Band order follows requested indices; wavelength values and units are preserved, without implicit conversion. The `analysis` attribute records policy, level, source, bands and limitations. Values come directly from tiles, independently of display contrast. The raster backend itself uses display conversion and cannot preserve original high-bit-depth samples; raw TIFF/OME/ENVI readers have separate contracts.

See Also

`at_extract_spectrum()`, `at_extract_pixels()`

Other extraction: `at_extract_pixels()`, `at_extract_spectrum()`

Examples

```
proj <- at_example_project()
at_extract(proj, stat = "mean")
```

---

at_extract_pixels *Extract per-pixel values under ROIs*

Description

Extract per-pixel values under ROIs

Usage

```
at_extract_pixels(
  project,
  img = NULL,
  layer = NULL,
  level = 0L,
  bands = NULL,
  max_px = 1e+06,
  call = rlang::caller_env()
)
```

Arguments

project	An annot_project.
img	An annot_image; defaults to the project's image. Supplying an alternate image asserts spatial registration on the same level-0 pixel grid. Its level-0 extent must match; differing pyramid dimensions are handled using the alternate image's actual level dimensions.
layer	Optional layer filter.
level	Integer pyramid level. Default 0.
bands	Optional band indices; all bands by default.
max_px	Maximum number of pixels to extract before aborting. Default 1e6.
call	The calling environment, for error reporting.

Value

A long tibble::tibble with columns roi_id, layer, label, x, y, band, and value. A 0-row tibble with these columns when nothing matches.

See Also

Other extraction: at_extract(), at_extract_spectrum()

---

at_extract_spectrum *Extract mean/median spectra for a spectral cube*

Description

Extract mean/median spectra for a spectral cube

Usage

```
at_extract_spectrum(
  project,
  img = NULL,
  stat = "mean",
  layer = NULL,
  label = NULL,
  level = 0L,
  call = rlang::caller_env(),
  nonfinite = c("propagate", "omit", "error")
)
```

Arguments

`project`	An annot_project.
`img`	An annot_image; defaults to the project's image. Supplying an alternate image asserts spatial registration on the same level-0 pixel grid. Its level-0 extent must match; differing pyramid dimensions are handled using the alternate image's actual level dimensions.
`stat`	`"mean"` (default) or `"median"`.
`layer, label`	Optional filters.
`level`	Integer pyramid level. Default `0`.
`call`	The calling environment, for error reporting.
`nonfinite`	Policy for NA, NaN and infinite samples: `"propagate"` (default) returns NA statistics for any affected ROI/band; `"omit"` uses finite samples only; `"error"` aborts on selected nonfinite samples. `n` always counts selected pixels. A wholly invalid selection has NA statistics under omit (including sum); singleton sample SD is NA.

Value

The long tibble of `at_extract()`, restricted to spectral bands and sorted by ROI, declared unit, wavelength, then band index. Values are not converted between units; equal wavelengths retain band order.

See Also

Other extraction: `at_extract()`, `at_extract_pixels()`

Examples

```
cube <- at_example_image("cube")
proj <- at_project(cube, at_layer_add(at_layer("roi"),
                                      at_roi_circle(16, 16, 5, label = "roi")))
at_extract_spectrum(proj)
```

`at_fix_geometry` *Repair the geometry of a project*

Description

Apply `sf::st_make_valid()` to invalid ROIs, drop duplicate consecutive vertices, re-orient holes that wind like their exterior ring, and clip out-of-bounds geometry to the image bounds (an intersection, so a polygon stays a polygon), reporting what changed.

Usage

```
at_fix_geometry(project, verbose = TRUE, call = rlang::caller_env())
```

Arguments

project	An annot_project.
verbose	Logical; whether to report repairs. Default TRUE.
call	The calling environment, for error reporting.

Value

The annot_project with repaired geometry. Repairs are reported via cli::cli_inform() when verbose is TRUE. Issues that cannot be repaired (zero area, degenerate, empty, NA coordinates, or a clip that would leave nothing inside the image) are left in place and reported.

See Also

at_check_geometry()

Other geometry: at_check_containment(), at_check_geometry(), at_clamp(), at_flip_y(), at_roi_buffer(), at_roi_contains(), at_roi_distance(), at_roi_overlaps(), at_roi_rescale(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_rois_overlap(), at_snap(), at_transform()

---

at_flip_y *Flip an ROI's y-axis*

Description

Convert between image orientation (top-left origin, y down) and GIS orientation (bottom-left origin, y up). Self-inverse.

Usage

```
at_flip_y(roi, height, call = rlang::caller_env())
```

Arguments

roi	An annot_roi.
height	Numeric image height in pixels at the ROI's reference level.
call	The calling environment, for error reporting.

Value

An annot_roi with the y-axis flipped.

See Also

Other geometry: at_check_containment(), at_check_geometry(), at_clamp(), at_fix_geometry(), at_roi_buffer(), at_roi_contains(), at_roi_distance(), at_roi_overlaps(), at_roi_rescale(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_rois_overlap(), at_snap(), at_transform()

Examples

```
at_flip_y(at_roi_rect(0, 0, 10, 5, label = "a"), height = 100)
```

`at_goto` *Jump to a specific image*

Description

Jump to a specific image

Usage

```
at_goto(session, i, call = rlang::caller_env())
```

Arguments

`session`	An annot_session.
`i`	Integer image index (1-based) within the queue.
`call`	The calling environment, for error reporting.

Value

A new annot_session with the cursor at `i`.

See Also

Other sessions: `at_current()`, `at_example_session()`, `at_manifest()`, `at_next()`, `at_prev()`, `at_resume()`, `at_session()`, `at_session_status()`, `at_set_status()`

`at_hsi_meta` *Hyperspectral and acquisition metadata of an image*

Description

Return the backend-agnostic metadata record hyperannotatR keeps for an image: acquisition identity, storage layout, value semantics, geometry convention, plane selection, and calibration/transform digests. Fields a backend cannot establish are `NA` (never guessed), and `wavelength_status` in `at_bands()` flags missing or duplicated wavelengths instead of interpolating them.

Usage

```
at_hsi_meta(x, call = rlang::caller_env())
```

Arguments

x	An annot_image.
call	The calling environment, for error reporting.

Value

A named list with the elements vendor, instrument, acquisition_id, format, backend, image_kind ("spectral", "multichannel", "rgb" or "grayscale"), width, height, n_bands, n_levels, dtype, value_unit (one of "raw", "reflectance", "radiance", "intensity", "absorbance", "unknown"), scale_factor, offset, nodata, valid_range, interleave, byte_order, header_bytes, axis_order, pixel_size, pixel_unit, origin, y_direction, series_id, plane (list of zero-based c, z, t), wavelength_unit, calibration, calibration_digest, transform_digest, profile, window_read and pan_sharpened.

See Also

at_bands(), at_read_stats()

Other images: at_bands(), at_close_image(), at_dims(), at_example_image(), at_image_capabilities(), at_is_pyramidal(), at_is_spectral(), at_meta(), at_n_bands(), at_n_levels(), at_pixel_size(), at_read_image(), at_tile(), at_wavelengths(), hyperannotatR-classes

Other hsi: at_band_operations(), at_band_view(), at_convert_values(), at_cubert_export_envi(), at_read_stats(), at_tivita_profile()

Examples

```
at_hsi_meta(at_example_image("cube"))[c("image_kind", "value_unit", "interleave")]
```

---

at_image_capabilities *Inspect the effective reading capabilities of an image*

Description

Report how an image's pixels are read: whether opening it reads metadata only, whether tiles are read as windows from the file, what the samples mean, and where they come from.

Usage

```
at_image_capabilities(x)
```

Arguments

x	An annot_image or annot_project.

Value

A list with `metadata_only` and `window_read` (logical; `NA` when a third-party backend does not declare them), `samples` (`"raw-scalar-v1"` for stored values, `"display-converted-0-255"` for display conversions, `"unqualified"` when undeclared) and `io` (`"binary"`, `"native"`, `"eager"`, `"memory"` or `"unknown"`), plus an optional `lifecycle`. An image whose handle holds decoded pixels reports memory access even for a backend that normally reads windows.

See Also

Other images: `at_bands()`, `at_close_image()`, `at_dims()`, `at_example_image()`, `at_hsi_meta()`, `at_is_pyramidal()`, `at_is_spectral()`, `at_meta()`, `at_n_bands()`, `at_n_levels()`, `at_pixel_size()`, `at_read_image()`, `at_tile()`, `at_wavelengths()`, `hyperannotatR-classes`

Other backends: `at_backend_detect()`, `at_backend_get()`, `at_backend_list()`, `at_backend_register()`, `at_close_image()`, `at_cubert_export_envi()`, `at_read_image()`, `at_tile()`, `at_tivita_profile()`

Examples

```
at_image_capabilities(at_example_image("cube"))$window_read
```

---

at_import_qupflowr *Import a qupflowR handoff or neutral interchange file*

Description

Read only the neutral contract, validate it, and return a namespaced hyperannotatR result together with a conversion report. For a handoff directory the SHA-256 inventory (missing, extra, duplicate, resized, altered or symlinked files), the manifest schema and major version, ROI id uniqueness and the id join between manifest and GeoJSON, layers, planes, mask dimensions, integer values and legend completeness are all checked. Every file the manifest references must be a clean relative path listed in the verified inventory; unlisted operating-system metadata files (`.DS_Store`, `._*`, `Thumbs.db`, `desktop.ini`) are ignored with a warning. Any failure aborts with a classified `at_error`: `at_validation_error` for integrity, schema and join failures, `at_protocol_error` for an unsupported major version, `at_limit_error` for oversized inventories and `at_io_error` for missing files. Nothing is written, no analysis is started, and foreign objects are never returned as if they were hyperannotatR objects: imported ROIs carry `source = "imported"` and partner fields under `attributes$qupflowr`. Features of a single GeoJSON file that carry no id get a minted one and are flagged (`attributes$qupflowr$id_minted`), so `at_stage_qupflowr()` creates them instead of joining them to an existing ROI; two features with the same id are refused (`DUPLICATE_ID`). Handoff coordinates are read at the export level recorded in the manifest. An RDS file is refused, because deserialising it is not a neutral read.

Usage

```
at_import_qupflowr(
  file,
  format = c("auto", "handoff", "qupath_geojson", "geojson", "mask", "npy"),
  expected_revision = NULL,
```

```
  call = rlang::caller_env()
)
```

Arguments

file	A handoff directory (or its manifest.json/integrity.json), a QuPath GeoJSON, a GeoJSON, a mask TIFF/PNG with legend sidecar, or a .npy mask.
format	"auto" (default), "handoff", "qupath_geojson", "geojson", "mask" or "npy".
expected_revision	Optional "sha256:..." annotation revision the caller expects the handoff to carry; a different revision aborts with an at_conflict_error (REVISION_CONFLICT).
call	The calling environment, for error reporting.

Value

An at_import_report: a list with status, format, source_name, handoff_digest, manifest, checks (tibble of check, status, detail), objects (layers: named list of annot_layer; masks: named list of annot_mask; images: image records), conversion (tibble with one row per ROI: roi_id, layer, label, source_object_id, object_type, geometry_fidelity, geometry_match ("exact" bytes, "equivalent" within 1e-9 after level normalisation, "changed", or NA without a manifest record), notes) and annotation_revision of the imported layers.

See Also

at_export_qupflowr(), at_stage_qupflowr()

Other interop: at_annotation_revision(), at_commit_qupflowr(), at_export_qupflowr(), at_interop_capabilities(), at_interop_manifest(), at_stage_qupflowr()

Examples

```
dest <- file.path(tempdir(), "handoff-import-example")
at_export_qupflowr(at_example_project(), dest, overwrite = TRUE)
rep <- at_import_qupflowr(dest)
rep$checks
```

---

at_inspect_saved *Inspect a saved checkpoint without modifying it*

Description

Validate a saved project or session file and report the state of its image sources, without repairing anything or reading pixels.

Usage

```
at_inspect_saved(path)
```

Arguments

path	A saved project or session RDS file.

Value

A list with valid (logical), class (the saved object's class), issues (character; why the file is invalid or ambiguous) and sources (one at_inspect_source() report per stored project). Invalid or ambiguous structures are reported without guessing repairs.

See Also

at_inspect_source(), at_load_session()

Other io: at_inspect_source(), at_load_project(), at_load_session(), at_read_geojson(), at_read_qupath(), at_read_rois_csv(), at_relink_source(), at_roi_from_geojson(), at_save_project(), at_save_session(), at_write_geojson(), at_write_qupath(), at_write_rois_csv()

---

at_inspect_source *Inspect an image source without changing annotations or reading pixels*

Description

Report whether the file an image (or project) was read from is still available and unchanged, so a saved project can be reopened or relinked deliberately.

Usage

```
at_inspect_source(x)
```

Arguments

x	An annot_image or annot_project.

Value

A list with status, path, backend, options (the recorded reader options) and options_inferred (TRUE for a legacy image whose options were not recorded). status is "available", "missing", "changed", "in_memory" (pixels held in memory, no source file) or "unresolved". Size and modification time detect many source replacements, but are not a content checksum. Composite sources also validate every declared companion input, such as an ENVI header and payload.

See Also

at_relink_source(), at_inspect_saved()

Other io: at_inspect_saved(), at_load_project(), at_load_session(), at_read_geojson(), at_read_qupath(), at_read_rois_csv(), at_relink_source(), at_roi_from_geojson(), at_save_project(), at_save_session(), at_write_geojson(), at_write_qupath(), at_write_rois_csv()

`at_interop_capabilities` *Partner interoperability capabilities*

Description

Report what this hyperannotatR installation can exchange with a partner package such as qupflowR, and how far each capability has been qualified. Profiles follow the partner contract: I0 file interchange, I1 public R API, I2 the local control service (`annotatr-control-v1`) and I3 the embeddable app object from `at_app()`. A profile is `"supported"` only with implementation, runtime dependencies and qualification evidence; implemented profiles that have not passed their tests against a real partner client are `"planned"`; profiles whose optional runtime is missing are `"unavailable"`. Every entry carries a `reason` when it is not supported.

Usage

```
at_interop_capabilities(
  target = c("package", "hsi", "app", "control"),
  call = rlang::caller_env()
)
```

Arguments

`target`	One or more of `"package"`, `"hsi"`, `"app"`, `"control"`.
`call`	The calling environment, for error reporting.

Value

A list of class `at_capabilities` with `schema`, `schema_version`, `annotatr_version`, `r_version`, `source_revision`, `consumer`, `profiles` (I0-I3), `limits`, and the requested sections `package`, `hsi`, `app`, `control`, plus `digest`: the SHA-256 of the canonical JSON of the report without `digest` itself.

See Also

`at_interop_manifest()`, `at_control_capabilities()`

Other interop: `at_annotation_revision()`, `at_commit_qupflowr()`, `at_export_qupflowr()`, `at_import_qupflowr()`, `at_interop_manifest()`, `at_stage_qupflowr()`

Examples

```
caps <- at_interop_capabilities()
caps$profiles$I0$status
caps$digest
```

`at_interop_manifest` *Build a partner interop manifest*

Description

Describe hyperannotatR objects for a partner such as qupflowR without exposing local paths: images are identified by content hash (`image_id`) and file basename; annotations by string ROI ids and an `at_annotation_revision()`; masks by type, integer dtype, background, dimensions, level and legend. The coordinate convention, band/wavelength table, HSI value semantics, plane, calibration and transform digests are recorded explicitly. QuPath hierarchy and native ROI shapes, which hyperannotatR does not model, are reported as `"unknown"` or `"approximated"`, never invented.

Usage

```
at_interop_manifest(
  x,
  destination = NULL,
  hash_sources = TRUE,
  call = rlang::caller_env()
)
```

Arguments

`x`	An annot_project, annot_session, annot_image, annot_layer, annot_roi or `annot_mask`.
`destination`	Optional path: an existing directory (the manifest is written as `manifest.json` inside it) or a `.json` file path. An existing file is never overwritten.
`hash_sources`	Logical; hash image source files (needed for a content-derived `image_id`). Default `TRUE`.
`call`	The calling environment, for error reporting.
	In a session manifest every ROI and layer record carries the `entry_id` of its queue entry, so entries that share an image stay separate. An ROI whose `review_status` attribute is not one of `"unreviewed"`, `"reviewed"`, `"rejected"` or `"staged"` is refused (`REVIEW_STATUS_INVALID`).

Value

The manifest (a list of class `at_interop_manifest`); invisibly when `destination` is given.

See Also

`at_export_qupflowr()`, `at_interop_capabilities()`

Other interop: `at_annotation_revision()`, `at_commit_qupflowr()`, `at_export_qupflowr()`, `at_import_qupflowr()`, `at_interop_capabilities()`, `at_stage_qupflowr()`

Examples

```
m <- at_interop_manifest(at_example_project())
m$annotations$annotation_revision
m$images[[1]]$image_id
```

`at_is_pyramidal` *Is the image pyramidal?*

Description

Is the image pyramidal?

Usage

```
at_is_pyramidal(x, call = rlang::caller_env())
```

Arguments

`x`	An annot_image.
`call`	The calling environment, for error reporting.

Value

A logical scalar. `TRUE` when more than one pyramid level exists.

See Also

Other images: `at_bands()`, `at_close_image()`, `at_dims()`, `at_example_image()`, `at_hsi_meta()`, `at_image_capabilities()`, `at_is_spectral()`, `at_meta()`, `at_n_bands()`, `at_n_levels()`, `at_pixel_size()`, `at_read_image()`, `at_tile()`, `at_wavelengths()`, `hyperannotatR-classes`

`at_is_spectral` *Is the image a spectral cube?*

Description

Is the image a spectral cube?

Usage

```
at_is_spectral(x, call = rlang::caller_env())
```

Arguments

x	An annot_image.
call	The calling environment, for error reporting.

Value

A logical scalar. TRUE when the image carries per-band wavelengths.

See Also

Other images: at_bands(), at_close_image(), at_dims(), at_example_image(), at_hsi_meta(), at_image_capabilities(), at_is_pyramidal(), at_meta(), at_n_bands(), at_n_levels(), at_pixel_size(), at_read_image(), at_tile(), at_wavelengths(), hyperannotatR-classes

---

at_layer *Create a layer*

Description

Create a layer

Usage

```
at_layer(
  name,
  labels = character(),
  style = at_style(),
  ...,
  call = rlang::caller_env()
)
```

Arguments

name	Single string layer name (unique within a project).
labels	Character vector; the controlled label vocabulary for the layer.
style	An annot_style, from at_style().
...	Additional named metadata stored on the layer. Set role = "calibration_reference" to default to two-click lines in the app. For each label, at least three valid lines are required for a mean reference length in level-0 pixels; five are recommended. The app also reports their standard deviation. Reference layers are not copied to subsequent images. No physical pixel size is assigned automatically.
call	The calling environment, for error reporting.

Value

An annot_layer with no ROIs and a resolved colour mapping.

See Also

at_style(), at_layer_add()

Other layers: at_layer_add(), at_layer_n(), at_layer_remove(), at_layer_rois(), at_style()

Examples

```
lyr <- at_layer("tumour", labels = c("tumour", "stroma"))
at_layer_n(lyr)
```

---

at_layer_add *Add an ROI to a layer*

Description

Add an ROI to a layer

Usage

```
at_layer_add(layer, roi, call = rlang::caller_env())
```

Arguments

layer	An annot_layer.
roi	An annot_roi.
call	The calling environment, for error reporting.

Value

A new annot_layer with the ROI appended. The label vocabulary and colour mapping are extended if the ROI introduces a new label (following the colour specification given to at_layer()). A missing, empty or colliding ROI identifier is re-minted to one unused in the layer. The input is unchanged.

See Also

Other layers: at_layer(), at_layer_n(), at_layer_remove(), at_layer_rois(), at_style()

Examples

```
lyr <- at_layer("tumour", labels = "tumour")
lyr <- at_layer_add(lyr, at_roi_rect(0, 0, 10, 10, label = "tumour"))
at_layer_n(lyr)
```

at_layer_n *Number of ROIs in a layer*

Description

Number of ROIs in a layer

Usage

```
at_layer_n(layer, call = rlang::caller_env())
```

Arguments

layer	An annot_layer.
call	The calling environment, for error reporting.

Value

An integer scalar.

See Also

Other layers: at_layer(), at_layer_add(), at_layer_remove(), at_layer_rois(), at_style()

at_layer_remove *Remove an ROI from a layer*

Description

Remove an ROI from a layer

Usage

```
at_layer_remove(layer, id, call = rlang::caller_env())
```

Arguments

layer	An annot_layer.
id	Single string ROI identifier.
call	The calling environment, for error reporting.

Value

A new annot_layer without the identified ROI. Warns if the ROI is not present and returns the layer unchanged.

See Also

Other layers: at_layer(), at_layer_add(), at_layer_n(), at_layer_rois(), at_style()

`at_layer_rois` *Layer ROI table*

Description

Layer ROI table

Usage

```
at_layer_rois(layer, image = NULL, call = rlang::caller_env())
```

Arguments

`layer`	An annot_layer.
`image`	annot_image defining actual pyramid dimensions. Required when any ROI is stored at a nonzero level; no scale is inferred without it.
`call`	The calling environment, for error reporting.

Value

An ROI table (see at_rois() for the column contract). A 0-row table with the same columns when the layer has no ROIs.

See Also

Other layers: at_layer(), at_layer_add(), at_layer_n(), at_layer_remove(), at_style()

`at_layers` *Layer summary table*

Description

Layer summary table

Usage

```
at_layers(project, call = rlang::caller_env())
```

Arguments

`project`	An annot_project.
`call`	The calling environment, for error reporting.

Value

A tibble::tibble with one row per layer and the columns `name` (character), `n_rois` (integer), `labels` (list of character), `visible` (logical), `locked` (logical), and `z` (integer). A 0-row tibble with these columns when the project has no layers.

See Also

Other projects: `at_add_layer()`, `at_add_roi()`, `at_example_project()`, `at_project()`, `at_remove_layer()`, `at_remove_roi()`, `at_rois()`, `at_summary()`, `at_validate()`

`at_load_project` *Load a project*

Description

Load a project

Usage

```
at_load_project(path, call = rlang::caller_env())
```

Arguments

`path`	Path to a saved project `.rds`.
`call`	The calling environment, for error reporting.

Value

The annot_project. Its image is reopened from the saved source descriptor; when the source is missing, changed or ambiguous the annotations still load, pixel access fails with a pointer to `at_relink_source()`, and `at_inspect_source()` reports why. Warns if the file was written by a newer hyperannotatR.

See Also

Other io: `at_inspect_saved()`, `at_inspect_source()`, `at_load_session()`, `at_read_geojson()`, `at_read_qupath()`, `at_read_rois_csv()`, `at_relink_source()`, `at_roi_from_geojson()`, `at_save_project()`, `at_save_session()`, `at_write_geojson()`, `at_write_qupath()`, `at_write_rois_csv()`

at_load_session *Load a session*

Description

Load a session

Usage

```
at_load_session(path, call = rlang::caller_env())
```

Arguments

path	Path to a saved session .rds.
call	The calling environment, for error reporting.

Value

The annot_session. A file written before sessions recorded queue entry identities gets one per queue row. Stored projects are reopened lazily by at_current(). Warns if written by a newer hyperannotatR.

See Also

at_save_session(), at_resume()

Other io: at_inspect_saved(), at_inspect_source(), at_load_project(), at_read_geojson(), at_read_qupath(), at_read_rois_csv(), at_relink_source(), at_roi_from_geojson(), at_save_project(), at_save_session(), at_write_geojson(), at_write_qupath(), at_write_rois_csv()

at_manifest *Session progress manifest with per-label counts*

Description

Session progress manifest with per-label counts

Usage

```
at_manifest(session, call = rlang::caller_env())
```

Arguments

session	An annot_session.
call	The calling environment, for error reporting.

Value

A tibble::tibble with `idx`, `entry_id`, `export_stem`, `name`, `path`, `status`, `n_layers`, `n_rois`, and one integer column per label in the session vocabulary giving its ROI count per image. `n_rois` counts the ROIs of materialised projects (otherwise the manifest's stored count). A label that clashes with one of the fixed column names gets a `label_` prefix.

See Also

Other sessions: `at_current()`, `at_example_session()`, `at_goto()`, `at_next()`, `at_prev()`, `at_resume()`, `at_session()`, `at_session_status()`, `at_set_status()`

`at_mask` *Rasterise ROIs to a mask ("Schablone")*

Description

Produce a binary, labelled, or multi-class integer mask from annotations. This is the package's core output artifact.

Usage

```
at_mask(
  x,
  type = c("binary", "labelled", "multiclass"),
  layer = NULL,
  label = NULL,
  level = 0L,
  background = 0L,
  touches = FALSE,
  dims = NULL,
  values = NULL,
  overlap = c("last", "first", "max", "min", "error", "bitor"),
  engine = c("stars", "terra"),
  call = rlang::caller_env(),
  image = NULL
)
```

Arguments

`x`	An annot_roi, annot_layer, or annot_project.
`type`	`"binary"` (a logical matrix), `"labelled"` (one integer id per ROI), or `"multiclass"` (one id per label class).
`layer`	Optional layer name(s) to restrict to (projects only). Unknown names are an error.
`label`	Optional label(s) to restrict to.
`level`	Integer pyramid level at which to rasterise. Default `0`.

background	Integer background value. Default 0. Foreground codes must differ from background. Binary masks reject 1 (foreground), and normalize other accepted backgrounds to the stored logical FALSE/0.
touches	Logical; see the coverage contract. Default FALSE.
dims	Optional c(width, height); taken from the image (projects) or the geometry bounding box otherwise, including containing cells for points. Dimensions must be positive and finite; fractional extents round up.
values	Optional named integer vector mapping labels to explicit class codes (e.g. c(specular = 1L, blood = 2L, shadow = 4L)), used only for type = "multiclass". When NULL (default) codes follow first-seen label order: 1, 2, 3, ..., or the single bits 1, 2, 4, ... with overlap = "bitor". Explicit bitfield codes must be distinct powers of two.
overlap	How overlapping ROIs resolve: "last" (later z-order wins, the default), "first", "max", "min", "error" (abort on any overlap), or "bitor" (bitwise-OR the overlapping class bits into a bitfield mask; type = "multiclass" or "binary" only, with background = 0; at most 31 classes).
engine	"stars" or "terra" (if installed). Centre-based polygons and points use shared membership; other coverage uses the selected engine.
call	The calling environment, for error reporting.
image	Optional annot_image for standalone ROIs or layers. Required when changing coordinate levels; dimensions alone do not define a scale. Projects always use their own image. Without dims, the image supplies the full mask extent at level.

Details

Pixel-coverage contract. Pixel (i, j) covers the half-open square [j-1, j) x [i-1, i) with (0, 0) at the top-left corner of the top-left pixel; its centre is (j - 0.5, i - 0.5). With touches = FALSE (default) a pixel is included when its centre lies inside the polygon. Boundary ties use the symbolic sample (x + epsilon, y + epsilon^2) as epsilon tends to zero from above, without a finite coordinate shift. Equivalently, scanline edges are active on [minY, maxY) and crossing pairs fill [left, right). Hole rings exclude pixels by parity regardless of winding; MULTIPOLYGON components combine by union. Thin polygons containing no selected centres have empty masks. This rule is shared by both engines and all overlap paths.

POINT and MULTIPOINT geometries select their half-open containing cells: (5, 5) selects row 6, column 6. Extraction uses the same membership and bounds. With touches = TRUE, polygon coverage uses the selected engine's all-touched rasterisation; lines and other geometry types also retain the selected engine's rasterisation behavior. Engine equality is guaranteed for centre-based polygons and points. ROI coordinates use the image convention (top-left origin, y down).

Full-grid allocation is checked against getOption("hyperannotatR.max_read_bytes", 1024^3) before rasterisation, using a conservative estimate of 32 bytes per pixel for arrays and working copies. The default allows 33,554,432 pixels. This is not a bound on total process memory or geometry/native-library overhead. Trusted larger masks require raising that option; over-budget requests fail with READ_BUDGET_EXCEEDED, and unsupported array sizes with MASK_TOO_LARGE.

Value

An annot_mask: a matrix (logical for "binary", otherwise integer) with attributes legend (a tibble with columns value, label, layer, roi_id, n_px, colour), level, dims, mask_type, source_project, mask_metadata and created. An all-background mask of the correct dimensions is returned when there are no ROIs. The mask_metadata attribute has schema version 1 and records encoding (binary, instance, categorical, or bitfield), background, overlap policy, grid (dimensions, level, origin, stride), source descriptor and entry provenance, and the complete label codebook. Supplied values retain mapped-but-absent classes with zero counts. Bitfields require unique positive single-bit codes and zero background; category 3 is exclusive unless bitfield encoding is explicitly declared.

See Also

at_write_mask(), at_read_mask(), at_mask_stats()

Other masks: at_mask_agreement(), at_mask_boundary(), at_mask_derive(), at_mask_legend(), at_mask_preview(), at_mask_stack(), at_mask_stats(), at_read_mask(), at_read_npy(), at_write_mask(), at_write_masks(), at_write_npy()

Examples

```
proj <- at_example_project()
m <- at_mask(proj, type = "multiclass")
table(as.integer(m))
```

---

at_mask_agreement *Agreement between two masks*

Description

Per-class Dice and Jaccard (IoU) plus an overall accuracy and Cohen's kappa, comparing a reference mask a against a second mask b (e.g. a hyperannotatR mask against a .npy ground truth, or two annotators). Versioned metadata, grids, source provenance, encodings and semantic codebooks are validated. Plain matrices are explicit-code legacy categorical inputs and require finite representable integer values; missing values are rejected.

Usage

```
at_mask_agreement(
  a,
  b,
  background = 0L,
  labels = NULL,
  call = rlang::caller_env(),
  alignment = c("check", "assert"),
  code_alignment = c("strict", "by-label", "by-id")
)
```

Arguments

a	Reference annot_mask or integer matrix.
b	Comparison annot_mask or integer matrix, same dimensions as a.
background	Integer value treated as unlabeled/background and excluded from the per-class rows (still counted for overall accuracy/kappa). Defaults to declared metadata background, or 0 for plain matrices. Explicit values must agree with metadata; both inputs must share background.
labels	Optional named vector or legend tibble (value, label) overriding display labels only. It cannot certify semantic codebook alignment.
call	The calling environment, for error reporting.
alignment	"check" (default) verifies shared grid/source metadata; "assert" records the caller's assertion of spatial alignment. Dimensions must always agree. Identical inputs permit a self-grid comparison without source descriptors; legacy pairs retain explicit-code/grid assumptions.
code_alignment	"strict" (default) requires compatible code mappings; "by-label" explicitly maps categorical/binary/bitfield codes by unique semantic labels; "by-id" maps instance codes by stable unique roi_id. Remapping requires matching identity sets in both codebooks. Mixed plain matrices and annotated inputs require alignment="assert" and categorical encoding; plain codes then assert the annotated codebook.

Details

Categorical/binary classes use equality; bitfields use class memberships, with overall accuracy measuring exact membership-set agreement and kappa NA. Instance maps use stable unique ROI IDs, never display class names. No spatial instance matching is inferred. Absent classes retain rows with NA Dice/IoU; background-only accuracy is 1 and kappa is NA (zero Cohen denominator). Mean Dice/IoU exclude absent classes and are NA if none are defined.

Value

A tibble::tibble with one row per class: value, label, n_true, n_pred, tp, fp, fn, dice, iou. The overall attribute is a list of accuracy, kappa, mean_dice, mean_iou, and n_px, encoding, alignment, code_alignment, background and nonfinite (always "error"). A comparison attribute retains validated input metadata and legends for interpretation.

See Also

at_mask(), at_read_npy()

Other masks: at_mask(), at_mask_boundary(), at_mask_derive(), at_mask_legend(), at_mask_preview(), at_mask_stack(), at_mask_stats(), at_read_mask(), at_read_npy(), at_write_mask(), at_write_masks(), at_write_npy()

Examples

```
m <- at_mask(at_example_project(), "multiclass")
ag <- at_mask_agreement(m, m)
```

```
attr(ag, "overall")$kappa
```

`at_mask_boundary` *Object boundaries of a mask*

Description

Object boundaries of a mask

Usage

```
at_mask_boundary(mask, width = 1L, call = rlang::caller_env())
```

Arguments

`mask`	An `annot_mask`.
`width`	Integer boundary thickness in pixels. Default `1`.
`call`	The calling environment, for error reporting.

Value

A logical matrix marking boundary pixels (where a 4-neighbour has a different value).

See Also

Other masks: `at_mask()`, `at_mask_agreement()`, `at_mask_derive()`, `at_mask_legend()`, `at_mask_preview()`, `at_mask_stack()`, `at_mask_stats()`, `at_read_mask()`, `at_read_npy()`, `at_write_mask()`, `at_write_masks()`, `at_write_npy()`

`at_mask_derive` *Derive a training mask from layer masks*

Description

Keep foreground state codes inside the requested anatomy class and outside artefacts, using each mask's declared background and class membership. An anatomy bitfield includes composite samples containing the requested bit.

Usage

```
at_mask_derive(
  state,
  anatomy,
  artefact = NULL,
  keep_label = "wound",
  background = NULL,
  call = rlang::caller_env(),
  alignment = c("legacy", "assert")
)
```

Arguments

state	An annot_mask of state (class) codes.
anatomy	An annot_mask of anatomy codes, sharing state's dimensions and level.
artefact	Optional annot_mask (bitfield); pixels with any artefact bit set are dropped. NULL (default) applies no artefact exclusion.
keep_label	The anatomy label whose region is kept (e.g. "wound").
background	Output background value. NULL (default) preserves the state mask's background. Must not collide with any state class code.
call	The calling environment, for error reporting.
alignment	"legacy" (default) validates declared source descriptors and grids; masks with only legacy metadata use a recorded legacy assumption. Modern masks with missing or conflicting provenance require "assert", explicitly asserting registration without resampling. Dimensions must always agree. Entry IDs and runtime cache generations do not establish source compatibility. Combining the same object records self-grid.

Value

An annot_mask retaining state encoding and the full state codebook, with counts recomputed, zero counts for absent classes, and alignment provenance recorded in mask_metadata$derivation.

See Also

at_mask(), at_mask_stack()

Other masks: at_mask(), at_mask_agreement(), at_mask_boundary(), at_mask_legend(), at_mask_preview(), at_mask_stack(), at_mask_stats(), at_read_mask(), at_read_npy(), at_write_mask(), at_write_masks(), at_write_npy()

Examples

```
proj <- at_example_project()
# A single-layer project derives trivially; the four-layer HSI workflow passes
# separate anatomy/state/artefact masks. See the masks vignette.
m <- at_mask(proj, "multiclass")
at_mask_derive(m, m, keep_label = at_mask_legend(m)$label[1])
```

at_mask_legend *Legend of a mask*

Description

Legend of a mask

Usage

```
at_mask_legend(mask, call = rlang::caller_env())
```

Arguments

mask	An annot_mask.
call	The calling environment, for error reporting.

Value

The mask's legend tibble::tibble (value, label, layer, roi_id, n_px, colour).

See Also

Other masks: at_mask(), at_mask_agreement(), at_mask_boundary(), at_mask_derive(), at_mask_preview(), at_mask_stack(), at_mask_stats(), at_read_mask(), at_read_npy(), at_write_mask(), at_write_masks(), at_write_npy()

Examples

```
at_mask_legend(at_mask(at_example_project(), "labelled"))
```

at_mask_preview *Downsampled preview of a mask*

Description

Nearest-neighbour downsample only (interpolating a label mask would invent labels that do not exist).

Usage

```
at_mask_preview(mask, max_dim = 1024L, call = rlang::caller_env())
```

Arguments

mask	An annot_mask.
max_dim	Integer maximum dimension of the preview. Default 1024.
call	The calling environment, for error reporting.

Value

A downsampled annot_mask with membership-aware counts recomputed. Metadata records the preview dimensions and sampling stride, its parent grid, and the original source provenance. No full-resolution counts are copied onto the sampled matrix.

See Also

Other masks: at_mask(), at_mask_agreement(), at_mask_boundary(), at_mask_derive(), at_mask_legend(), at_mask_stack(), at_mask_stats(), at_read_mask(), at_read_npy(), at_write_mask(), at_write_masks(), at_write_npy()

---

at_mask_stack *Stack per-layer masks into a 3D array*

Description

Stack per-layer masks into a 3D array

Usage

```
at_mask_stack(project, layers = NULL, level = 0L, call = rlang::caller_env())
```

Arguments

project	An annot_project.
layers	Optional layer name(s) to include; all by default. Unknown names are an error.
level	Integer pyramid level. Default 0.
call	The calling environment, for error reporting.

Value

A 3D integer array [y, x, layer], one multi-class plane per layer; the third dimension is named by layer. Codes are local to each layer; the legends attribute is a named list of the per-layer legends (see at_mask_legend()) that maps them back to labels.

See Also

Other masks: at_mask(), at_mask_agreement(), at_mask_boundary(), at_mask_derive(), at_mask_legend(), at_mask_preview(), at_mask_stats(), at_read_mask(), at_read_npy(), at_write_mask(), at_write_masks(), at_write_npy()

at_mask_stats *Per-value statistics of a mask*

Description

Per-value statistics of a mask

Usage

```
at_mask_stats(mask, pixel_size = NULL, call = rlang::caller_env())
```

Arguments

mask	An annot_mask.
pixel_size	Optional positive physical pixel size c(x, y), or one value for square pixels, in the mask's own pixel grid. When NULL, area_physical is NA.
call	The calling environment, for error reporting.

Value

A tibble::tibble with one row per mask value: value, label, n_px, area_px, area_physical, frac_total, bounding box (bbox_xmin, bbox_ymin, bbox_xmax, bbox_ymax), and centroid (centroid_x, centroid_y). Every bitfield membership contributes to its class count, including composite samples; categorical counts are exclusive. Mapped-but-absent classes have zero counts. A 0-row tibble is returned only when the codebook is empty.

See Also

Other masks: at_mask(), at_mask_agreement(), at_mask_boundary(), at_mask_derive(), at_mask_legend(), at_mask_preview(), at_mask_stack(), at_read_mask(), at_read_npy(), at_write_mask(), at_write_masks(), at_write_npy()

Examples

```
at_mask_stats(at_mask(at_example_project(), "labelled"))
```

at_meta *Image metadata*

Description

Image metadata

Usage

```
at_meta(x, key = NULL, call = rlang::caller_env())
```

Arguments

x	An annot_image.
key	Optional single string. When supplied, the corresponding metadata element is returned; otherwise the whole metadata list is returned.
call	The calling environment, for error reporting.

Value

When key is NULL, a list of all metadata (including source, backend, dtype, pixel_unit, and any backend-specific entries). When key is supplied, the single element (or NULL if absent).

See Also

Other images: at_bands(), at_close_image(), at_dims(), at_example_image(), at_hsi_meta(), at_image_capabilities(), at_is_pyramidal(), at_is_spectral(), at_n_bands(), at_n_levels(), at_pixel_size(), at_read_image(), at_tile(), at_wavelengths(), hyperannotatR-classes

at_n_bands *Number of bands (channels)*

Description

Number of bands (channels)

Usage

```
at_n_bands(x, call = rlang::caller_env())
```

Arguments

x	An annot_image.
call	The calling environment, for error reporting.

Value

An integer scalar.

See Also

Other images: at_bands(), at_close_image(), at_dims(), at_example_image(), at_hsi_meta(), at_image_capabilities(), at_is_pyramidal(), at_is_spectral(), at_meta(), at_n_levels(), at_pixel_size(), at_read_image(), at_tile(), at_wavelengths(), hyperannotatR-classes

---

at_n_levels *Number of pyramid levels*

Description

Number of pyramid levels

Usage

```
at_n_levels(x, call = rlang::caller_env())
```

Arguments

x	An annot_image.
call	The calling environment, for error reporting.

Value

An integer scalar; 1 when the image has no pyramid.

See Also

Other images: at_bands(), at_close_image(), at_dims(), at_example_image(), at_hsi_meta(), at_image_capabilities(), at_is_pyramidal(), at_is_spectral(), at_meta(), at_n_bands(), at_pixel_size(), at_read_image(), at_tile(), at_wavelengths(), hyperannotatR-classes

at_next *Advance to the next image*

Description

Advance to the next image

Usage

```
at_next(session, call = rlang::caller_env())
```

Arguments

session	An annot_session.
call	The calling environment, for error reporting.

Value

A new annot_session with the cursor advanced (clamped at the end).

See Also

Other sessions: at_current(), at_example_session(), at_goto(), at_manifest(), at_prev(), at_resume(), at_session(), at_session_status(), at_set_status()

at_pixel_size *Physical pixel size*

Description

Physical pixel size

Usage

```
at_pixel_size(x, call = rlang::caller_env())
```

Arguments

x	An annot_image.
call	The calling environment, for error reporting.

Value

A numeric vector of length 2, the physical size of one level-0 pixel as c(x, y), in the units given by at_meta(x, "pixel_unit").

See Also

Other images: at_bands(), at_close_image(), at_dims(), at_example_image(), at_hsi_meta(), at_image_capabilities(), at_is_pyramidal(), at_is_spectral(), at_meta(), at_n_bands(), at_n_levels(), at_read_image(), at_tile(), at_wavelengths(), hyperannotatR-classes

---

`at_plot` *Plot generic for hyperannotatR objects*

Description

Plot generic for hyperannotatR objects

Usage

```
at_plot(x, ...)
```

Arguments

`x`	An annot_image, annot_project, annot_layer, or `annot_mask`.
`...`	Passed to the specific plot function. For layers, passed to `at_layer_rois()`: supply `image = img` for any nonzero stored ROI level. Layer plot geometry uses level-0 coordinates. The same image argument is supported by `ggplot2::autoplot(layer, image = img)`.

Value

A ggplot2::ggplot object.

See Also

Other plots: at_plot_image(), at_plot_mask(), at_plot_overlay(), at_plot_project(), at_plot_spectrum(), at_plot_summary()

Examples

```
at_plot(at_example_project())
```

`at_plot_image` *Plot an image*

Description

Plot an image

Usage

```
at_plot_image(
  img,
  level = NULL,
  bands = NULL,
  rgb = NULL,
  max_dim = 1024L,
  call = rlang::caller_env()
)
```

Arguments

`img`	An annot_image.
`level`	Pyramid level to sample; chosen automatically when `NULL`. Plot axes always use level-0 image coordinates, scaled independently by actual width and height ratios.
`bands`	Optional band index for single-band display.
`rgb`	Optional length-3 vector of band indices, or wavelengths in nm for a spectral cube, for a false-colour composite. Each channel of an `rgb` composite is contrast-stretched on its own; the default natural-colour display of an image with three or more bands shows an 8-bit image with its stored values and stretches deeper data in its first three bands jointly, preserving their colour balance.
`max_dim`	Maximum display dimension (auto-downsampled). Default `1024`. Each displayed cell covers its full source block; final partial blocks end at the image boundary. When `level` is `NULL`, the finest pyramid level that fits within `max_dim` is sampled.
`call`	The calling environment, for error reporting.

Value

A ggplot2::ggplot object. Non-finite samples are drawn transparent.

See Also

Other plots: `at_plot()`, `at_plot_mask()`, `at_plot_overlay()`, `at_plot_project()`, `at_plot_spectrum()`, `at_plot_summary()`

Examples

```
at_plot_image(at_example_image("tissue"))
```

`at_plot_mask` *Plot a mask*

Description

Plot a mask

Usage

```
at_plot_mask(mask, legend = TRUE, call = rlang::caller_env())
```

Arguments

`mask`	An `annot_mask`.
`legend`	Logical; show the fill legend. Default `TRUE`.
`call`	The calling environment, for error reporting.

Value

A ggplot2::ggplot object.

See Also

Other plots: `at_plot()`, `at_plot_image()`, `at_plot_overlay()`, `at_plot_project()`, `at_plot_spectrum()`, `at_plot_summary()`

Examples

```
at_plot_mask(at_mask(at_example_project(), "labelled"))
```

at_plot_overlay *Plot ROIs overlaid on the image*

Description

Plot ROIs overlaid on the image

Usage

```
at_plot_overlay(project, img = NULL, layer = NULL, call = rlang::caller_env())
```

Arguments

project	An annot_project.
img	An annot_image; the project's image by default.
layer	Optional layer filter.
call	The calling environment, for error reporting.

Value

A ggplot2::ggplot object with the image and ROI outlines (points as markers), both in level-0 image coordinates.

See Also

Other plots: at_plot(), at_plot_image(), at_plot_mask(), at_plot_project(), at_plot_spectrum(), at_plot_summary()

Examples

```
at_plot_overlay(at_example_project())
```

at_plot_project *Plot a project's ROIs*

Description

Plot a project's ROIs

Usage

```
at_plot_project(
  project,
  layer = NULL,
  label = NULL,
  show_labels = TRUE,
  alpha = NULL,
  call = rlang::caller_env()
)
```

Arguments

project	An annot_project.
layer, label	Optional filters.
show_labels	Logical; draw label text at centroids. Default TRUE.
alpha	Fill opacity; a sensible default when NULL.
call	The calling environment, for error reporting.

Value

A ggplot2::ggplot object in level-0 image coordinates with the y axis pointing down, as in the image. Empty projects give a valid empty plot.

See Also

Other plots: at_plot(), at_plot_image(), at_plot_mask(), at_plot_overlay(), at_plot_spectrum(), at_plot_summary()

Examples

```
at_plot_project(at_example_project())
```

---

at_plot_spectrum *Plot spectra*

Description

Plot spectra

Usage

```
at_plot_spectrum(
  spectra,
  colour_by = c("label", "roi_id", "layer"),
  ribbon = TRUE,
  call = rlang::caller_env()
)
```

Arguments

spectra	The long tibble from at_extract_spectrum() or at_extract(). Its declared unit is used on the axis; absent units remain unknown. Mixed units reject: explicitly convert coordinates before combining them.
colour_by	One of "label", "roi_id", or "layer".
ribbon	Logical; draw a min-max ribbon per colour group (and statistic) wherever the group holds more than one ROI. Default TRUE.
call	The calling environment, for error reporting.

Value

A ggplot2::ggplot object with one line per ROI and statistic.

See Also

Other plots: at_plot(), at_plot_image(), at_plot_mask(), at_plot_overlay(), at_plot_project(), at_plot_summary()

---

at_plot_summary *Plot a project summary*

Description

Plot a project summary

Usage

```
at_plot_summary(project, call = rlang::caller_env())
```

Arguments

project	An annot_project.
call	The calling environment, for error reporting.

Value

A ggplot2::ggplot object of per-label ROI area distributions.

See Also

Other plots: at_plot(), at_plot_image(), at_plot_mask(), at_plot_overlay(), at_plot_project(), at_plot_spectrum()

at_prev *Step back to the previous image*

Description

Step back to the previous image

Usage

```
at_prev(session, call = rlang::caller_env())
```

Arguments

session	An annot_session.
call	The calling environment, for error reporting.

Value

A new annot_session with the cursor decremented (clamped at 1).

See Also

Other sessions: at_current(), at_example_session(), at_goto(), at_manifest(), at_next(), at_resume(), at_session(), at_session_status(), at_set_status()

at_project *Create an annotation project*

Description

Create an annotation project

Usage

```
at_project(image, layers = list(), ..., call = rlang::caller_env())
```

Arguments

image	An annot_image the annotations refer to.
layers	An annot_layer, a list of them, or an empty list (default).
...	Additional named project metadata (e.g. name, description).
call	The calling environment, for error reporting.

Value

An annot_project bundling the image, layers, metadata, and provenance (creation time, hyperannotatR version, R version, author, and an edit log).

See Also

at_add_layer(), at_add_roi(), at_rois(), at_mask()

Other projects: at_add_layer(), at_add_roi(), at_example_project(), at_layers(), at_remove_layer(), at_remove_roi(), at_rois(), at_summary(), at_validate()

Examples

```
# An annot_image comes from at_read_image() or at_example_image():
# proj <- at_project(at_example_image("tissue"),
#                    at_layer("tissue", labels = "tissue"))
```

---

at_read_geojson *Read ROIs from a GeoJSON file*

Description

Read ROIs from a GeoJSON file

Usage

```
at_read_geojson(
  path,
  layer_name = NULL,
  level = 0L,
  flip_y = FALSE,
  height = NULL,
  call = rlang::caller_env(),
  coordinate_level = NULL,
  legacy_levels = c("error", "recorded")
)
```

Arguments

path	Path to a GeoJSON file.
layer_name	Layer name for the imported ROIs; taken from the features' layer property when NULL.
level	Integer level to record when a feature has no level property.
flip_y	Logical; invert the y-axis on read. Default FALSE.
height	Image height in pixels, required when flip_y = TRUE.
call	The calling environment, for error reporting.

coordinate_level
: Optional integer overriding every recorded `level` without scaling coordinates. Use only when the actual coordinate level is known, for example `0L` for legacy hyperannotatR exports that wrote level-0 coordinates beside a nonzero source level. `NULL` (default) uses the schema and `legacy_levels` policy. Coordinates are never automatically repaired.

legacy_levels
: How to interpret files without the coordinate schema marker: `"error"` (default) rejects ambiguous nonzero recorded levels; `"recorded"` explicitly trusts those levels. Level-0 files remain unambiguous.

Details

A FeatureCollection, a single Feature or a bare array of features is accepted. Features without a geometry (`"geometry": null`) carry no region and are skipped with a warning; any other document is refused with an `at_validation_error` of code `INVALID_GEOJSON`.

Value

A named list of annot_layer objects (one per distinct `layer` property), or a single annot_layer when `layer_name` is supplied.

See Also

Other io: `at_inspect_saved()`, `at_inspect_source()`, `at_load_project()`, `at_load_session()`, `at_read_qupath()`, `at_read_rois_csv()`, `at_relink_source()`, `at_roi_from_geojson()`, `at_save_project()`, `at_save_session()`, `at_write_geojson()`, `at_write_qupath()`, `at_write_rois_csv()`

---

at_read_image *Read an image*

Description

Read an image into a lightweight annot_image handle, auto-detecting the backend from the file unless one is named.

Usage

```
at_read_image(path, backend = NULL, ..., call = rlang::caller_env())
```

Arguments

path
: Single string path to an image file.

backend
: Optional backend name; auto-detected when `NULL`.

...
: Passed to the backend's reader. They must be plain serializable values (numbers, strings, logicals and lists of them): they are recorded in the image's source descriptor so a saved project can reopen the file the same way.

call
: The calling environment, for error reporting.

Value

An annot_image whose `source` is the absolute path of the file, with a `source_descriptor` (path, backend, options, file signature) and a per-read cache identity.

See Also

`at_backend_list()`, `at_tile()`

Other images: `at_bands()`, `at_close_image()`, `at_dims()`, `at_example_image()`, `at_hsi_meta()`, `at_image_capabilities()`, `at_is_pyramidal()`, `at_is_spectral()`, `at_meta()`, `at_n_bands()`, `at_n_levels()`, `at_pixel_size()`, `at_tile()`, `at_wavelengths()`, `hyperannotatR-classes`

Other backends: `at_backend_detect()`, `at_backend_get()`, `at_backend_list()`, `at_backend_register()`, `at_close_image()`, `at_cubert_export_envi()`, `at_image_capabilities()`, `at_tile()`, `at_tivita_profile()`

Examples

```
img <- at_read_image(at_example_path("tissue"))
at_dims(img)
```

---

at_read_mask *Read a mask into editable ROIs*

Description

Polygonise an existing mask file (e.g. Cellpose, StarDist, or QuPath output, or a prior hyperannotatR export) into an editable annot_layer. Explicit metadata and legend/level arguments take precedence over embedded RDS metadata, then the `<path>.legend.json` sidecar. Bitfields polygonise each class membership. Legacy masks default to exclusive values and zero background, recorded as `legacy-default`; incomplete legacy bitfield legends require an explicit encoding override. Malformed/future metadata and inconsistent dimensions fail rather than silently discarding labels. Sampled previews or translated grids cannot be polygonised as editable ROIs; import the original-resolution mask with unit stride and zero origin.

Usage

```
at_read_mask(
  path,
  level = NULL,
  connectivity = c(8L, 4L),
  simplify = 0,
  min_area = 0,
  legend = NULL,
  call = rlang::caller_env(),
  metadata = NULL
)
```

Arguments

path	Path to a mask file (TIFF, PNG, RDS, or NPY). NPY sidecar storage orientation is recovered automatically for polygonisation. PNG input must be unsigned 8/16-bit grayscale without palette, colour, alpha channels or tRNS transparency; original class codes are preserved. TIFF requires the raw tiff reader; PNG requires a magick build with exact grayscale capability.
level	Integer pyramid level to record on the ROIs. NULL (default) uses the RDS mask or legend sidecar level, falling back to 0 for an external mask without level metadata. An explicit level overrides metadata.
connectivity	Single finite whole number 8 (default) or 4, including 8L/4L. Eight-connectivity joins corner-touching pixels into one ROI; four-connectivity requires shared edges. Explicit vectors reject.
simplify	Non-negative simplification tolerance for the polygons.
min_area	Minimum polygon area in pixels to keep. Default 0.
legend	Optional legend tibble (value, label) overriding the embedded / sidecar / pixel-value labelling.
call	The calling environment, for error reporting.
metadata	Optional named list overriding semantic mask_metadata fields, such as encoding, background, or source. Nested grid fields merge with recovered metadata. Explicit level and legend take precedence over this list. Overrides do not resample pixels. Stored versioned metadata must be structurally valid before overrides are applied.

Value

An annot_layer with one ROI per connected region. Recovered semantic metadata is retained in layer$meta$mask_metadata, including empty layers.

See Also

at_write_mask(), at_mask()

Other masks: at_mask(), at_mask_agreement(), at_mask_boundary(), at_mask_derive(), at_mask_legend(), at_mask_preview(), at_mask_stack(), at_mask_stats(), at_read_npy(), at_write_mask(), at_write_masks(), at_write_npy()

Examples

```
m <- at_mask(at_example_project(), "labelled")
f <- tempfile(fileext = ".tif")
at_write_mask(m, f)
at_read_mask(f)
```

at_read_npy *Read a NumPy* .npy *integer mask*

Description

Read a NumPy integer array (e.g. a ground-truth annotation mask) into a lossless annot_mask – values are preserved exactly, including composite bitfield codes. Use this (not at_read_mask(), which polygonises) when the raw integer matrix is what you need, for example to derive a training mask or compute agreement against another mask.

Usage

```
at_read_npy(
  path,
  level = NULL,
  transpose = FALSE,
  legend = NULL,
  call = rlang::caller_env(),
  metadata = NULL
)
```

Arguments

path	Path to a .npy file.
level	Integer pyramid level to record. NULL (default) recovers the sidecar level, falling back to 0 without metadata. Explicit values override the sidecar declaration without resampling pixels.
transpose	Logical; transpose the array on read, for files stored in the native cube order (width, height) = [x, y] (e.g. TIVITA/HyperGui masks) so the result lands in hyperannotatR's [y, x] convention. Default FALSE assumes the standard image convention (height, width).
legend	Optional legend tibble (value, label) overriding the sidecar / pixel-value labelling.
call	The calling environment, for error reporting.
metadata	Optional named list overriding semantic mask_metadata fields, such as encoding, background, or source. Nested grid fields merge; explicit level and legend take precedence. No pixels are resampled. Versioned declarations must be structurally valid before overrides apply.

Details

The header is validated strictly (NPY format versions 1-3, 2-D shapes, scalar integer dtypes of 1, 2, 4 or 8 bytes in either byte order) and the payload length is checked before reading; values that do not fit a 32-bit integer mask (including INT_MIN, R's missing-value code) are refused, never truncated. Reads are bounded by option hyperannotatR.max_read_bytes and headers by hyperannotatR.max_npy_header_bytes.

Value

An integer annot_mask preserving encoding, background, codebook, grid and source metadata. Without encoding metadata, imports use exclusive instance values and zero background with status legacy-default; signed representable codes are preserved. A declared storage orientation must match transpose, even for square arrays. Malformed sidecars are rejected.

See Also

at_write_npy(), at_read_mask()

Other masks: at_mask(), at_mask_agreement(), at_mask_boundary(), at_mask_derive(), at_mask_legend(), at_mask_preview(), at_mask_stack(), at_mask_stats(), at_read_mask(), at_write_mask(), at_write_masks(), at_write_npy()

Examples

```
m <- at_mask(at_example_project(), "labelled")
f <- tempfile(fileext = ".npy")
at_write_npy(m, f)
at_read_npy(f)
```

---

at_read_qupath *Read ROIs from a QuPath GeoJSON file*

Description

Reads QuPath 0.4+ exports (a native feature array, a FeatureCollection or a single Feature) and the legacy hyperannotatR 0.1 dialect. classification.name (or the joined names of a derived class, e.g. "Tumor: Positive") becomes the ROI label, the classification colour (a current color RGB array or a legacy colorRGB integer) the layer style colour, and features without a classification are labelled "unclassified". Everything else QuPath carries is kept, never invented, under each ROI's attributes$qupath: the object id and type, name, measurements (with explicit value_state), the image plane (zero-based c, z, t; c = NA for all channels), metadata, a nucleus geometry, and roi_native = "ellipse" with geometry_fidelity = "approximated" for polygonised ellipses. A hyperannotatR id stored in metadata.annotatr_roi_id is restored as the ROI id; ordinary external ids are kept unchanged.

Usage

```
at_read_qupath(
  path,
  layer_name = "qupath",
  level = 0L,
  call = rlang::caller_env(),
  coordinate_level = NULL,
  legacy_levels = c("error", "recorded")
)
```

Arguments

path	Path to a QuPath GeoJSON file.
layer_name	Layer name for the imported ROIs. Default "qupath".
level	Integer fallback level for features without level metadata. Default 0 (standard full-resolution QuPath coordinates).
call	The calling environment, for error reporting.
coordinate_level	Optional integer overriding every recorded level without scaling coordinates. Use only when the actual coordinate level is known, for example 0L for legacy hyperannotatR exports that wrote level-0 coordinates beside a nonzero source level. NULL (default) uses the schema and legacy_levels policy. Coordinates are never automatically repaired.
legacy_levels	How to interpret files without the coordinate schema marker: "error" (default) rejects ambiguous nonzero recorded levels; "recorded" explicitly trusts those levels. Level-0 files remain unambiguous.

Details

Input is validated strictly and keys are matched exactly: malformed structure, contradictory duplicate fields (identical duplicates, which QuPath 0.7.0 emits, are accepted), conflicting colours or object types for a feature, conflicting colours for one label, and repeated ROI ids are refused with an at_validation_error of code INVALID_GEOJSON.

Value

An annot_layer.

See Also

Other io: at_inspect_saved(), at_inspect_source(), at_load_project(), at_load_session(), at_read_geojson(), at_read_rois_csv(), at_relink_source(), at_roi_from_geojson(), at_save_project(), at_save_session(), at_write_geojson(), at_write_qupath(), at_write_rois_csv()

---

at_read_rois_csv *Read ROIs from a CSV with WKT geometry*

Description

Read ROIs from a CSV with WKT geometry

Usage

```
at_read_rois_csv(
  path,
  layer_name = "csv",
  call = rlang::caller_env(),
  coordinate_level = NULL,
  legacy_levels = c("error", "recorded")
)
```

Arguments

path	Path to a CSV with at least label and geometry (WKT) columns. Other columns (roi_id, level, author, source, created, modified) are used when present. Every column is read as text, so labels and ids such as "T", "007" or "NA" are kept verbatim. level is the pyramid level of the WKT coordinates.
layer_name	Layer name for the imported ROIs. Default "csv".
call	The calling environment, for error reporting.
coordinate_level	Optional integer overriding every recorded level without scaling coordinates. Use only when the actual coordinate level is known, for example 0L for legacy hyperannotatR exports that wrote level-0 coordinates beside a nonzero source level. NULL (default) uses the schema and legacy_levels policy. Coordinates are never automatically repaired.
legacy_levels	How to interpret files without the coordinate_schema marker: "error" (default) rejects ambiguous nonzero recorded levels; "recorded" explicitly trusts those levels. Level-0 files remain unambiguous.

Value

An annot_layer.

See Also

Other io: at_inspect_saved(), at_inspect_source(), at_load_project(), at_load_session(), at_read_geojson(), at_read_qupath(), at_relink_source(), at_roi_from_geojson(), at_save_project(), at_save_session(), at_write_geojson(), at_write_qupath(), at_write_rois_csv()

---

at_read_stats *Read accounting for an image*

Description

Report how many tiles were requested from an image's backend, how many payload bytes were read from its file by windowed readers, and how many bytes were returned as decoded arrays. Cache hits are not counted as reads.

Usage

```
at_read_stats(img, reset = FALSE, call = rlang::caller_env())
```

Arguments

img	An annot_image.
reset	Logical; reset the counters for this image after reporting.
call	The calling environment, for error reporting.

Value

A list with tiles (integer), file_bytes, decoded_bytes and max_tile_bytes (doubles), plus window_read (logical) and the current max_tile_bytes_limit from option hyperannotatR.max_tile_bytes.

See Also

Other hsi: at_band_operations(), at_band_view(), at_convert_values(), at_cubert_export_envi(), at_hsi_meta(), at_tivita_profile()

Examples

```
cube <- at_example_image("cube")
invisible(at_tile(cube, xrange = c(1, 4), yrange = c(1, 4), bands = 1:2))
at_read_stats(cube)$tiles
```

---

at_relink_source *Explicitly relink or revalidate an image source*

Description

Point a saved image (or project) at its source file again, for example after the file moved, or accept a source that changed since it was saved. The replacement is read with the selected reader and must have the same geometry; annotations are never changed.

Usage

```
at_relink_source(x, path, backend = NULL, options = NULL)
```

Arguments

x	An annot_image or annot_project. Annotations and geometry are preserved.
path	Replacement source path (may be the same path to revalidate).
backend	Reader backend; defaults to the saved backend.
options	Reader options as a list; defaults to the saved options.

Value

The updated image or project. Dimensions, pyramid levels, band count and wavelengths must agree. Relinking explicitly accepts the new source and refreshes image metadata from the selected reader.

See Also

`at_inspect_source()`

Other io: `at_inspect_saved()`, `at_inspect_source()`, `at_load_project()`, `at_load_session()`, `at_read_geojson()`, `at_read_qupath()`, `at_read_rois_csv()`, `at_roi_from_geojson()`, `at_save_project()`, `at_save_session()`, `at_write_geojson()`, `at_write_qupath()`, `at_write_rois_csv()`

---

at_remove_layer *Remove a layer from a project*

Description

Remove a layer from a project

Usage

```
at_remove_layer(project, name, call = rlang::caller_env())
```

Arguments

`project`	An annot_project.
`name`	Single string layer name.
`call`	The calling environment, for error reporting.

Value

A new annot_project without the named layer. The input is unchanged.

See Also

Other projects: `at_add_layer()`, `at_add_roi()`, `at_example_project()`, `at_layers()`, `at_project()`, `at_remove_roi()`, `at_rois()`, `at_summary()`, `at_validate()`

at_remove_roi *Remove an ROI from a project*

Description

Remove an ROI from a project

Usage

```
at_remove_roi(project, id, call = rlang::caller_env())
```

Arguments

project	An annot_project.
id	Single string ROI identifier.
call	The calling environment, for error reporting.

Value

A new annot_project without the identified ROI. Warns and returns the project unchanged if the identifier is not found.

See Also

Other projects: at_add_layer(), at_add_roi(), at_example_project(), at_layers(), at_project(), at_remove_layer(), at_rois(), at_summary(), at_validate()

at_resume *Resume a saved session*

Description

A more discoverable alias for at_load_session().

Usage

```
at_resume(path, call = rlang::caller_env())
```

Arguments

path	Path to a saved session .rds file.
call	The calling environment, for error reporting.

Value

The restored annot_session.

See Also

at_save_session(), at_load_session()

Other sessions: at_current(), at_example_session(), at_goto(), at_manifest(), at_next(), at_prev(), at_session(), at_session_status(), at_set_status()

---

at_roi_area *ROI area*

Description

ROI area

Usage

```
at_roi_area(
  roi,
  level = 0L,
  units = c("px", "physical"),
  call = rlang::caller_env(),
  image = NULL
)
```

Arguments

roi	An annot_roi.
level	Integer pyramid level at which to express the area. Default 0.
units	Either "px" (pixels squared, the default) or "physical". Physical area uses image calibration at level zero, in squared image$pixel_unit, and is invariant to reporting level. Unknown or pixel-only image calibration rejects. Without an image, legacy ROI pixel_size=c(x,y) is supported at stored level zero, in squared calibration units. Nonzero ROI levels require explicit pixel_size_level and pixel_unit; differing calibration and stored levels require an image for the geometric transform. Pixel sizes must be finite and positive. Geometric area is not raster support count.
call	The calling environment, for error reporting.
image	Optional annot_image defining actual pyramid dimensions. Required when level differs from the ROI's stored level.

Value

A numeric scalar area, or NA_real_ for non-areal geometry (points, lines).

See Also

Other rois: at_roi_bbox(), at_roi_centroid(), at_roi_circle(), at_roi_ellipse(), at_roi_freehand(), at_roi_from_geojson(), at_roi_from_sf(), at_roi_point(), at_roi_polygon(), at_roi_rect()

Examples

```
at_roi_area(at_roi_rect(0, 0, 10, 5, label = "a"))
```

`at_roi_bbox` *ROI bounding box*

Description

ROI bounding box

Usage

```
at_roi_bbox(roi, call = rlang::caller_env())
```

Arguments

`roi`	An annot_roi.
`call`	The calling environment, for error reporting.

Value

A numeric vector `c(xmin, ymin, xmax, ymax)` in the ROI's stored coordinates.

See Also

Other rois: `at_roi_area()`, `at_roi_centroid()`, `at_roi_circle()`, `at_roi_ellipse()`, `at_roi_freehand()`, `at_roi_from_geojson()`, `at_roi_from_sf()`, `at_roi_point()`, `at_roi_polygon()`, `at_roi_rect()`

Examples

```
at_roi_bbox(at_roi_circle(50, 50, 10, label = "a"))
```

`at_roi_buffer` *Buffer an ROI*

Description

Buffer an ROI

Usage

```
at_roi_buffer(roi, dist, call = rlang::caller_env())
```

Arguments

roi	An annot_roi.
dist	Numeric buffer distance in pixels (may be negative to erode).
call	The calling environment, for error reporting.

Value

An annot_roi with the buffered geometry.

See Also

Other geometry: at_check_containment(), at_check_geometry(), at_clamp(), at_fix_geometry(), at_flip_y(), at_roi_contains(), at_roi_distance(), at_roi_overlaps(), at_roi_rescale(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_rois_overlap(), at_snap(), at_transform()

Examples

```
at_roi_buffer(at_roi_point(10, 10, label = "a"), dist = 5)
```

---

at_roi_centroid *ROI centroid*

Description

ROI centroid

Usage

```
at_roi_centroid(roi, call = rlang::caller_env())
```

Arguments

roi	An annot_roi.
call	The calling environment, for error reporting.

Value

A numeric vector `c(x, y)` in the ROI's stored coordinates (at its own `level`).

See Also

Other rois: at_roi_area(), at_roi_bbox(), at_roi_circle(), at_roi_ellipse(), at_roi_freehand(), at_roi_from_geojson(), at_roi_from_sf(), at_roi_point(), at_roi_polygon(), at_roi_rect()

Examples

```
at_roi_centroid(at_roi_rect(0, 0, 10, 10, label = "a"))
```

`at_roi_circle` *Create a circular ROI*

Description

The circle is approximated by a regular polygon of `n_seg` segments.

Usage

```
at_roi_circle(
  x,
  y,
  r,
  label,
  level = 0L,
  n_seg = 64L,
  ...,
  call = rlang::caller_env()
)
```

Arguments

`x`, `y`	Numeric centre coordinates in image pixels.
`r`	Positive numeric radius in pixels.
`label`	Single string class label.
`level`	Integer pyramid level the coordinates refer to. Default `0`.
`n_seg`	Integer number of polygon segments approximating the circle.
`...`	Additional named attributes stored on the ROI. The reserved names `author`, `source`, and `id` set those fields; all others become user attributes.
`call`	The calling environment, for error reporting.

Value

An annot_roi with `POLYGON` geometry and `attributes$shape = "circle"`, so exports can flag the polygon as an approximation.

See Also

Other rois: `at_roi_area()`, `at_roi_bbox()`, `at_roi_centroid()`, `at_roi_ellipse()`, `at_roi_freehand()`, `at_roi_from_geojson()`, `at_roi_from_sf()`, `at_roi_point()`, `at_roi_polygon()`, `at_roi_rect()`

Examples

```
r <- at_roi_circle(50, 50, r = 10, label = "nucleus")
```

at_roi_contains *Test whether points fall inside an ROI*

Description

Test whether points fall inside an ROI

Usage

```
at_roi_contains(roi, x, y, call = rlang::caller_env())
```

Arguments

roi	An annot_roi.
x, y	Numeric coordinate vectors of equal length in the ROI's stored coordinate level.
call	The calling environment, for error reporting.

Value

A logical vector of length `length(x)`.

See Also

Other geometry: at_check_containment(), at_check_geometry(), at_clamp(), at_fix_geometry(), at_flip_y(), at_roi_buffer(), at_roi_distance(), at_roi_overlaps(), at_roi_rescale(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_rois_overlap(), at_snap(), at_transform()

Examples

```
at_roi_contains(at_roi_rect(0, 0, 10, 10, label = "a"), c(5, 20), c(5, 20))
```

at_roi_distance *Distance between two ROIs*

Description

Distance between two ROIs

Usage

```
at_roi_distance(roi1, roi2, call = rlang::caller_env(), image = NULL)
```

Arguments

roi1, roi2	annot_roi objects.
call	The calling environment, for error reporting.
image	annot_image defining actual pyramid dimensions; required for any nonzero ROI level because measurements are reported at level 0.

Value

A numeric scalar distance in level-0 pixels.

See Also

Other geometry: at_check_containment(), at_check_geometry(), at_clamp(), at_fix_geometry(), at_flip_y(), at_roi_buffer(), at_roi_contains(), at_roi_overlaps(), at_roi_rescale(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_rois_overlap(), at_snap(), at_transform()

---

at_roi_ellipse *Create an elliptical ROI*

Description

Create an elliptical ROI

Usage

```
at_roi_ellipse(
  x,
  y,
  rx,
  ry,
  rotation = 0,
  label,
  level = 0L,
  n_seg = 64L,
  ...,
  call = rlang::caller_env()
)
```

Arguments

x, y	Numeric centre coordinates in image pixels.
rx, ry	Positive numeric semi-axis lengths in pixels.
rotation	Rotation of the major axis in degrees, clockwise in image orientation. Default 0.
label	Single string class label.

level	Integer pyramid level the coordinates refer to. Default 0.
n_seg	Integer number of polygon segments approximating the ellipse.
...	Additional named attributes stored on the ROI. The reserved names author, source, and id set those fields; all others become user attributes.
call	The calling environment, for error reporting.

Value

An annot_roi with POLYGON geometry and attributes$shape = "ellipse", so exports can flag the polygon as an approximation.

See Also

Other rois: at_roi_area(), at_roi_bbox(), at_roi_centroid(), at_roi_circle(), at_roi_freehand(), at_roi_from_geojson(), at_roi_from_sf(), at_roi_point(), at_roi_polygon(), at_roi_rect()

Examples

```
r <- at_roi_ellipse(50, 50, rx = 20, ry = 10, rotation = 30, label = "cell")
```

---

at_roi_freehand *Create a freehand ROI*

Description

A freehand trace is stored as a closed polygon, optionally simplified.

Usage

```
at_roi_freehand(
  coords,
  label,
  level = 0L,
  simplify = 0,
  ...,
  call = rlang::caller_env()
)
```

Arguments

coords	Two-column numeric matrix of vertices in drawing order.
label	Single string class label.
level	Integer pyramid level the coordinates refer to. Default 0.
simplify	Non-negative Douglas-Peucker tolerance (dTolerance) applied to the closed polygon. 0 (default) keeps every vertex.
...	Additional named attributes stored on the ROI. The reserved names author, source, and id set those fields; all others become user attributes.
call	The calling environment, for error reporting.

Value

An annot_roi with POLYGON geometry.

See Also

Other rois: at_roi_area(), at_roi_bbox(), at_roi_centroid(), at_roi_circle(), at_roi_ellipse(), at_roi_from_geojson(), at_roi_from_sf(), at_roi_point(), at_roi_polygon(), at_roi_rect()

Examples

```
pts <- cbind(c(0, 5, 10, 5), c(0, 2, 0, -2))
r <- at_roi_freehand(pts, label = "trace")
```

---

at_roi_from_geojson *Convert a GeoJSON feature to an ROI*

Description

Turn a single parsed GeoJSON feature (a list with geometry and optional properties) into an annot_roi. Useful for consuming interactive drawing output.

Usage

```
at_roi_from_geojson(
  feature,
  label = NULL,
  level = 0L,
  source = "manual",
  call = rlang::caller_env()
)
```

Arguments

feature	A parsed GeoJSON feature: a list with a geometry element (type and coordinates) and optional properties.
label	Single string class label; taken from properties$label when NULL.
level	Integer pyramid level. Default 0.
source	Provenance source string. Default "manual".
call	The calling environment, for error reporting.

Value

An annot_roi.

See Also

Other io: at_inspect_saved(), at_inspect_source(), at_load_project(), at_load_session(), at_read_geojson(), at_read_qupath(), at_read_rois_csv(), at_relink_source(), at_save_project(), at_save_session(), at_write_geojson(), at_write_qupath(), at_write_rois_csv()

Other rois: at_roi_area(), at_roi_bbox(), at_roi_centroid(), at_roi_circle(), at_roi_ellipse(), at_roi_freehand(), at_roi_from_sf(), at_roi_point(), at_roi_polygon(), at_roi_rect()

Examples

```
feat <- list(geometry = list(type = "Polygon",
  coordinates = list(list(c(0, 0), c(4, 0), c(4, 4), c(0, 4), c(0, 0)))))
at_roi_from_geojson(feat, label = "region")
```

---

at_roi_from_sf *Create an ROI from an existing sf geometry*

Description

Create an ROI from an existing sf geometry

Usage

```
at_roi_from_sf(geometry, label, level = 0L, ..., call = rlang::caller_env())
```

Arguments

geometry	An sf, sfc, or sfg geometry in image pixel coordinates.
label	Single string class label.
level	Integer pyramid level the coordinates refer to. Default 0.
...	Additional named attributes stored on the ROI. The reserved names author, source, and id set those fields; all others become user attributes.
call	The calling environment, for error reporting.

Value

An annot_roi wrapping the supplied geometry.

See Also

Other rois: at_roi_area(), at_roi_bbox(), at_roi_centroid(), at_roi_circle(), at_roi_ellipse(), at_roi_freehand(), at_roi_from_geojson(), at_roi_point(), at_roi_polygon(), at_roi_rect()

Examples

```
g <- sf::st_point(c(3, 4))
r <- at_roi_from_sf(g, label = "imported", source = "imported")
```

at_roi_overlaps *Test whether two ROIs intersect*

Description

Test whether two ROIs intersect

Usage

```
at_roi_overlaps(roi1, roi2, call = rlang::caller_env(), image = NULL)
```

Arguments

roi1, roi2	annot_roi objects.
call	The calling environment, for error reporting.
image	Optional annot_image; required for mixed coordinate levels.

Value

A logical scalar: TRUE when the ROIs share any point, including ROIs that only touch along an edge or at a corner.

See Also

Other geometry: at_check_containment(), at_check_geometry(), at_clamp(), at_fix_geometry(), at_flip_y(), at_roi_buffer(), at_roi_contains(), at_roi_distance(), at_roi_rescale(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_rois_overlap(), at_snap(), at_transform()

at_roi_point *Create a point ROI*

Description

Create a point ROI

Usage

```
at_roi_point(x, y, label, level = 0L, ..., call = rlang::caller_env())
```

Arguments

x, y	Numeric coordinates of the point, in image pixels.
label	Single string class label.
level	Integer pyramid level the coordinates refer to. Default 0.
...	Additional named attributes stored on the ROI. The reserved names author, source, and id set those fields; all others become user attributes.
call	The calling environment, for error reporting.

Value

An annot_roi with POINT geometry.

See Also

at_roi_rect(), at_roi_polygon()

Other rois: at_roi_area(), at_roi_bbox(), at_roi_centroid(), at_roi_circle(), at_roi_ellipse(), at_roi_freehand(), at_roi_from_geojson(), at_roi_from_sf(), at_roi_polygon(), at_roi_rect()

Examples

```
r <- at_roi_point(10, 20, label = "cell")
at_roi_centroid(r)
```

---

at_roi_polygon *Create a polygon ROI*

Description

Create a polygon ROI

Usage

```
at_roi_polygon(
  coords,
  label,
  level = 0L,
  hole = NULL,
  ...,
  call = rlang::caller_env()
)
```

Arguments

coords	Either a two-column numeric matrix of exterior-ring vertices, or a list of such matrices where the first is the exterior ring and any others are holes. Rings are closed automatically.
label	Single string class label.
level	Integer pyramid level the coordinates refer to. Default 0.
hole	Optional list of two-column matrices giving additional holes, combined with any holes supplied via coords.
...	Additional named attributes stored on the ROI. The reserved names author, source, and id set those fields; all others become user attributes.
call	The calling environment, for error reporting.

Value

An annot_roi with POLYGON geometry.

See Also

Other rois: at_roi_area(), at_roi_bbox(), at_roi_centroid(), at_roi_circle(), at_roi_ellipse(), at_roi_freehand(), at_roi_from_geojson(), at_roi_from_sf(), at_roi_point(), at_roi_rect()

Examples

```
sq <- matrix(c(0, 0, 10, 0, 10, 10, 0, 10), ncol = 2, byrow = TRUE)
r <- at_roi_polygon(sq, label = "region")
```

---

at_roi_rect *Create a rectangular ROI*

Description

Create a rectangular ROI

Usage

```
at_roi_rect(
  xmin,
  ymin,
  xmax,
  ymax,
  label,
  level = 0L,
  ...,
  call = rlang::caller_env()
)
```

Arguments

xmin, ymin, xmax, ymax	Numeric bounds in image pixels. xmax/ymax must exceed xmin/ymin.
label	Single string class label.
level	Integer pyramid level the coordinates refer to. Default 0.
...	Additional named attributes stored on the ROI. The reserved names author, source, and id set those fields; all others become user attributes.
call	The calling environment, for error reporting.

Value

An annot_roi with POLYGON geometry (an axis-aligned rectangle).

See Also

Other rois: at_roi_area(), at_roi_bbox(), at_roi_centroid(), at_roi_circle(), at_roi_ellipse(), at_roi_freehand(), at_roi_from_geojson(), at_roi_from_sf(), at_roi_point(), at_roi_polygon()

Examples

```
r <- at_roi_rect(0, 0, 10, 5, label = "tissue")
at_roi_bbox(r)
```

---

at_roi_rescale *Rescale an ROI between pyramid levels*

Description

Transform an ROI's coordinates from one pyramid level to another, assuming a power-of-two pyramid. For an exact transform against a real image pyramid (which need not be power-of-two), use `at_transform()` with the annot_image.

Usage

```
at_roi_rescale(
  roi,
  from_level = roi$level,
  to_level,
  call = rlang::caller_env()
)
```

Arguments

`roi`	An annot_roi.
`from_level`	Integer source pyramid level. Defaults to the ROI's own `level`; any other value is an error, because the coordinates are stored at that level.
`to_level`	Integer target pyramid level.
`call`	The calling environment, for error reporting.

Value

An annot_roi with coordinates at `to_level` and its `level` field set to `to_level`. Power-of-two scaling is exact in floating point, so the transform is exactly invertible.

See Also

`at_transform()`

Other geometry: at_check_containment(), at_check_geometry(), at_clamp(), at_fix_geometry(), at_flip_y(), at_roi_buffer(), at_roi_contains(), at_roi_distance(), at_roi_overlaps(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_rois_overlap(), at_snap(), at_transform()

Examples

```
r <- at_roi_rect(0, 0, 100, 100, label = "a")
r2 <- at_roi_rescale(r, from_level = 0, to_level = 2)
identical(
  sf::st_coordinates(sf::st_geometry(r)),
  sf::st_coordinates(sf::st_geometry(at_roi_rescale(r2, 2, 0)))
)
```

`at_roi_ring` *Ring (annulus) along an ROI margin*

Description

Build a band straddling an ROI's boundary – the natural shape for a "penumbra" region along an injury or wound margin. The ring extends `outer` units outside the boundary and `inner` units inside it, formed as the set difference of the outward and inward buffers.

Usage

```
at_roi_ring(
  roi,
  outer,
  inner = 0,
  label = roi$label,
  call = rlang::caller_env()
)
```

Arguments

`roi`	An annot_roi.
`outer`	Non-negative outward width (dilation) of the ring.
`inner`	Non-negative inward width (erosion) of the ring. Default `0`, a purely external band.
`label`	Label for the derived ring ROI. Default the source ROI's label.
`call`	The calling environment, for error reporting.

Value

A derived annot_roi (an annulus) with `source = "derived"`.

See Also

`at_roi_buffer()`, `at_roi_difference()`

Other geometry: `at_check_containment()`, `at_check_geometry()`, `at_clamp()`, `at_fix_geometry()`, `at_flip_y()`, `at_roi_buffer()`, `at_roi_contains()`, `at_roi_distance()`, `at_roi_overlaps()`, `at_roi_rescale()`, `at_roi_setops`, `at_roi_simplify()`, `at_rois_overlap()`, `at_snap()`, `at_transform()`

Examples

```
inj <- at_roi_rect(10, 10, 20, 20, label = "injury")
at_roi_ring(inj, outer = 3, label = "penumbra")
```

`at_roi_setops` *Set operations on ROIs*

Description

Combine ROIs geometrically. Each accepts several annot_roi via `...`, or a single annot_layer. Multi-part results are returned as one ROI with `MULTIPOLYGON` geometry (the return type is always a single annot_roi). When every input is a polygon the result is a `POLYGON` or `MULTIPOLYGON`; parts of lower dimension (edges or corners that ROIs only touch along) are dropped, and an empty result is an empty polygon, with a warning.

Usage

```
at_roi_union(..., call = rlang::caller_env(), image = NULL)

at_roi_intersect(..., call = rlang::caller_env(), image = NULL)

at_roi_difference(..., call = rlang::caller_env(), image = NULL)

at_roi_symdiff(..., call = rlang::caller_env(), image = NULL)
```

Arguments

`...`	Two or more annot_roi, or a single annot_layer.
`call`	The calling environment, for error reporting.
`image`	Optional annot_image defining actual pyramid dimensions. Required for mixed levels. With it, results use level 0; without it, all operands must share a level and the result retains that level.

Details

Bare ROIs at different pyramid levels are an error with code `MIXED_LEVELS`; pass `image` to combine them at level 0. The result takes the label and author of the first ROI, but not its lock, review status, QuPath object identity or shape attributes.

Value

A single annot_roi with `source = "derived"`.

See Also

Other geometry: `at_check_containment()`, `at_check_geometry()`, `at_clamp()`, `at_fix_geometry()`, `at_flip_y()`, `at_roi_buffer()`, `at_roi_contains()`, `at_roi_distance()`, `at_roi_overlaps()`, `at_roi_rescale()`, `at_roi_ring()`, `at_roi_simplify()`, `at_rois_overlap()`, `at_snap()`, `at_transform()`

Examples

```
a <- at_roi_rect(0, 0, 6, 6, label = "x")
b <- at_roi_rect(4, 4, 10, 10, label = "x")
at_roi_area(at_roi_intersect(a, b))
```

`at_roi_simplify` *Simplify an ROI*

Description

Simplify an ROI

Usage

```
at_roi_simplify(roi, tolerance, call = rlang::caller_env())
```

Arguments

`roi`	An annot_roi.
`tolerance`	Non-negative Douglas-Peucker tolerance (`dTolerance`).
`call`	The calling environment, for error reporting.

Value

An annot_roi with simplified geometry.

See Also

Other geometry: `at_check_containment()`, `at_check_geometry()`, `at_clamp()`, `at_fix_geometry()`, `at_flip_y()`, `at_roi_buffer()`, `at_roi_contains()`, `at_roi_distance()`, `at_roi_overlaps()`, `at_roi_rescale()`, `at_roi_ring()`, `at_roi_setops`, `at_rois_overlap()`, `at_snap()`, `at_transform()`

Examples

```
at_roi_simplify(at_roi_circle(50, 50, 10, n_seg = 128, label = "a"), 1)
```

`at_rois` *Query the ROIs of a project*

Description

Return a tidy table of ROIs, optionally filtered by layer and/or label.

Usage

```
at_rois(project, layer = NULL, label = NULL, call = rlang::caller_env())
```

Arguments

`project`	An annot_project.
`layer`	Optional single layer name to filter to.
`label`	Optional character vector of labels to filter to.
`call`	The calling environment, for error reporting.

Details

Coordinates in the `geometry` column, together with `area_px` and the centroid, are expressed at pyramid level 0. The `level` column is 0; `source_level` records the stored level at which each ROI was defined.

Older tables used `level` for the stored reference level despite returning level-0 geometry. Consumers needing that provenance must now use `source_level`. The stored ROI geometry, level, and identifier are unchanged.

Value

A tibble::tibble with one row per ROI and, in this exact order, the columns:

`roi_id` Unique identifier (character).

`layer` Layer name (character).

`label` Class label (character).

`geom_type` Geometry type, e.g. `"POLYGON"` (character).

`level` Coordinate pyramid level, always 0 (integer).

`source_level` Stored ROI reference level (integer provenance).

`area_px` Area in level-0 pixels (double); `NA` for points/lines.

`centroid_x`,`centroid_y` Centroid at level 0 (double).

`n_vertices` Vertex count (integer).

`created`,`modified` Timestamps (POSIXct).

`author` Author (character).

`source` Provenance source (character).

`geometry` The `sf` geometry column (`sfc`), at level 0.

Returns a 0-row tibble with these columns if no ROIs match.

See Also

at_layers(), at_mask()

Other projects: at_add_layer(), at_add_roi(), at_example_project(), at_layers(), at_project(), at_remove_layer(), at_remove_roi(), at_summary(), at_validate()

Examples

```
# proj <- at_example_project()
# at_rois(proj)
```

---

at_rois_overlap *Pairwise ROI overlaps within a layer*

Description

Pairwise ROI overlaps within a layer

Usage

```
at_rois_overlap(layer, call = rlang::caller_env(), image = NULL)
```

Arguments

layer	An annot_layer.
call	The calling environment, for error reporting.
image	annot_image defining actual pyramid dimensions; required for any nonzero ROI level because measurements are reported at level 0.

Value

A tibble::tibble with columns id_a, id_b (character), overlap_px (double, level-0 pixels), and jaccard (double). A 0-row tibble with these columns when no ROIs overlap with positive area.

See Also

Other geometry: at_check_containment(), at_check_geometry(), at_clamp(), at_fix_geometry(), at_flip_y(), at_roi_buffer(), at_roi_contains(), at_roi_distance(), at_roi_overlaps(), at_roi_rescale(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_snap(), at_transform()

`at_save_project` *Save a project*

Description

The project is written atomically: to a temporary file in the target directory that then replaces `path`, so a failed write keeps the previous file. An image read with `at_read_image()` is saved with a versioned source descriptor (path, reader, options and file signature) instead of its pixels, and is reopened from it on load. A hand-built in-memory image keeps its pixels.

Usage

```
at_save_project(project, path, overwrite = FALSE, call = rlang::caller_env())
```

Arguments

`project`	An annot_project.
`path`	Output `.rds` path.
`overwrite`	Logical; overwrite an existing file. Default `FALSE`.
`call`	The calling environment, for error reporting.

Value

The path, invisibly.

See Also

`at_load_project()`, `at_inspect_saved()`

Other io: `at_inspect_saved()`, `at_inspect_source()`, `at_load_project()`, `at_load_session()`, `at_read_geojson()`, `at_read_qupath()`, `at_read_rois_csv()`, `at_relink_source()`, `at_roi_from_geojson()`, `at_save_session()`, `at_write_geojson()`, `at_write_qupath()`, `at_write_rois_csv()`

Examples

```
p <- tempfile(fileext = ".rds")
at_save_project(at_example_project(), p)
```

`at_save_session` *Save a session*

Description

The session is written atomically, with each stored project's image saved as a source descriptor rather than pixels (see `at_save_project()`).

Usage

```
at_save_session(
  session,
  path = NULL,
  overwrite = FALSE,
  call = rlang::caller_env()
)
```

Arguments

`session`	An annot_session.
`path`	Optional output path; defaults to `_session.rds` in `out_dir`.
`overwrite`	Logical; overwrite an existing file, including the default `_session.rds`. Default `FALSE`.
`call`	The calling environment, for error reporting.

Value

The session as saved (validated, with persisted projects), invisibly.

See Also

`at_load_session()`, `at_resume()`

Other io: `at_inspect_saved()`, `at_inspect_source()`, `at_load_project()`, `at_load_session()`, `at_read_geojson()`, `at_read_qupath()`, `at_read_rois_csv()`, `at_relink_source()`, `at_roi_from_geojson()`, `at_save_project()`, `at_write_geojson()`, `at_write_qupath()`, `at_write_rois_csv()`

`at_session` *Create a batch annotation session*

Description

Create a batch annotation session

Usage

```
at_session(
  paths,
  labels = character(),
  layers = NULL,
  out_dir = .session_out_dir(),
  autosave = TRUE,
  ...,
  call = rlang::caller_env()
)
```

Arguments

`paths`	Character vector of image file paths forming the queue.
`labels`	Character vector of the global label vocabulary. Duplicates are dropped.
`layers`	An annot_layer, a list of them, or `NULL`. Applied as a template to every image visited.
`out_dir`	Directory for autosave and exports. Defaults to a new, unique directory under `tempdir()`, which R deletes when the process ends: pass a persistent directory to keep autosaved work. Separate sessions never share a default directory. A relative path is resolved against the working directory when the session is created.
`autosave`	Logical; whether to autosave on annotation commits. Default `TRUE`.
`...`	Additional named session metadata.
`call`	The calling environment, for error reporting.

Value

An annot_session with a manifest over `paths`, all statuses `"pending"`, the cursor at the first image, and projects not yet materialised.

See Also

`at_next()`, `at_current()`, `at_resume()`

Other sessions: `at_current()`, `at_example_session()`, `at_goto()`, `at_manifest()`, `at_next()`, `at_prev()`, `at_resume()`, `at_session_status()`, `at_set_status()`

Examples

```
# Build a session over the bundled example image copied several times:
# sess <- at_example_session(3)
```

`at_session_status` *The session manifest*

Description

The session manifest

Usage

```
at_session_status(session, call = rlang::caller_env())
```

Arguments

`session`	An annot_session.
`call`	The calling environment, for error reporting.

Value

The manifest tibble::tibble with columns `idx`, `entry_id`, `export_stem`, `path`, `name`, `status`, `project_path`, `n_rois`, and `modified`.

See Also

Other sessions: `at_current()`, `at_example_session()`, `at_goto()`, `at_manifest()`, `at_next()`, `at_prev()`, `at_resume()`, `at_session()`, `at_set_status()`

`at_set_status` *Set the status of a queued image*

Description

Set the status of a queued image

Usage

```
at_set_status(session, i, status, call = rlang::caller_env())
```

Arguments

`session`	An annot_session.
`i`	Integer image index (1-based) within the queue.
`status`	One of `"pending"`, `"in_progress"`, `"complete"`, `"flagged"`, `"skipped"`.
`call`	The calling environment, for error reporting.

Value

A new annot_session with the status updated.

See Also

Other sessions: at_current(), at_example_session(), at_goto(), at_manifest(), at_next(), at_prev(), at_resume(), at_session(), at_session_status()

`at_snap` *Snap ROI vertices to a grid*

Description

Snap ROI vertices to a grid

Usage

```
at_snap(roi, grid = 1, call = rlang::caller_env())
```

Arguments

`roi`	An annot_roi.
`grid`	Positive numeric grid spacing in pixels. Default 1.
`call`	The calling environment, for error reporting.

Value

An annot_roi with vertices rounded to the nearest grid multiple. A geometry that snapping makes invalid (e.g. a collapsed ring) is repaired with a warning.

See Also

Other geometry: at_check_containment(), at_check_geometry(), at_clamp(), at_fix_geometry(), at_flip_y(), at_roi_buffer(), at_roi_contains(), at_roi_distance(), at_roi_overlaps(), at_roi_rescale(), at_roi_ring(), at_roi_setops, at_roi_simplify(), at_rois_overlap(), at_transform()

Examples

```
at_snap(at_roi_point(3.4, 5.6, label = "a"), grid = 1)
```

`at_stage_qupflowr` *Stage partner changes against a hyperannotatR project*

Description

Compute, without applying, the changes a partner patch would make to an hyperannotatR project: ROIs are joined by id and classified as `create`, `update`, `unchanged`, `delete` (only with `delete_missing = TRUE`) or `conflict`. Reviewed or locked ROIs, and ROIs in locked layers, are never changed automatically, and nothing is created in or moved into a locked layer; such changes become conflicts. Incoming ROIs whose id was minted on import (features without an id) are always created, with an id unique in the project. Incoming ROIs at another pyramid level are compared, and updates stored, at the level of the existing ROI. A patch that repeats an ROI id aborts (`DUPLICATE_ID`). When `expected_revision` is given and the project has moved on, staging aborts with an `at_conflict_error` (`REVISION_CONFLICT`).

Usage

```
at_stage_qupflowr(
  x,
  patch,
  expected_revision = NULL,
  delete_missing = FALSE,
  call = rlang::caller_env()
)
```

Arguments

`x`	An annot_project, or the path of a saved project `.rds` (then the commit re-reads and re-checks that file).
`patch`	An import report from `at_import_qupflowr()`, a handoff directory, or one or more annot_layer objects.
`expected_revision`	Optional revision of `x` the partner based its patch on (see `at_annotation_revision()`).
`delete_missing`	Logical; treat ROIs absent from the patch's layers as deletions. Default `FALSE`.
`call`	The calling environment, for error reporting.

Value

An `at_staged_patch`: a list with `patch_id`, `state = "staged"`, `created`, `base_revision`, `proposed_revision`, `operations` (tibble of `op`, `roi_id`, `layer`, `label`, `reason`), `summary` (counts by `op`), `base`, `proposed` (the would-be project), `target_path` and `patch_digest`.

See Also

`at_commit_qupflowr()`

Other interop: `at_annotation_revision()`, `at_commit_qupflowr()`, `at_export_qupflowr()`, `at_import_qupflowr()`, `at_interop_capabilities()`, `at_interop_manifest()`

Examples

```
proj <- at_example_project()
partner <- at_layer_add(at_layer("regions"), at_roi_rect(10, 10, 40, 40, label = "tumour",
                                                         id = "partner-1"))
st <- at_stage_qupflowr(proj, partner, expected_revision = at_annotation_revision(proj))
st$summary
```

`at_style` *Layer style*

Description

Construct a style specification for an annot_layer.

Usage

```
at_style(
  colour = NULL,
  fill_alpha = 0.3,
  stroke_width = 2,
  visible = TRUE,
  locked = FALSE,
  z = 1L,
  call = rlang::caller_env()
)
```

Arguments

`colour`	Either `NULL` (default; the Okabe-Ito colourblind-safe palette is cycled over the layer's labels), a single colour recycled across labels, or a named character vector mapping labels to colours. The style is stored with the resolved mapping so colours are stable across sessions; the original specification is kept too, so labels added later follow it.
`fill_alpha`	Fill opacity in `[0, 1]`. Default `0.3`.
`stroke_width`	Stroke width in pixels. Default `2`.
`visible`	Logical; whether the layer is drawn. Default `TRUE`.
`locked`	Logical; whether the layer is protected from edits. Default `FALSE`.
`z`	Integer draw order (higher is drawn later, on top). Affects mask overlap resolution. Default `1`.
`call`	The calling environment, for error reporting.

Value

A named list of class `annot_style`.

See Also

Other layers: at_layer(), at_layer_add(), at_layer_n(), at_layer_remove(), at_layer_rois()

Examples

```
at_style(fill_alpha = 0.5, z = 2)
```

`at_summary` *Summarise a project*

Description

Summarise a project

Usage

```
at_summary(project, call = rlang::caller_env())
```

Arguments

`project`	An annot_project.
`call`	The calling environment, for error reporting.

Value

An `annot_summary` object: a list with `n_layers`, `n_rois`, per-label ROI counts, and total area per label.

See Also

Other projects: at_add_layer(), at_add_roi(), at_example_project(), at_layers(), at_project(), at_remove_layer(), at_remove_roi(), at_rois(), at_validate()

Examples

```
at_summary(at_example_project())
```

at_summary_table *Session summary statistics*

Description

Session summary statistics

Usage

```
at_summary_table(
  session,
  by = c("image", "label", "layer"),
  call = rlang::caller_env()
)
```

Arguments

session	An annot_session.
by	Grouping: "image", "label", or "layer".
call	The calling environment, for error reporting.

Value

A tibble::tibble with the grouping key plus n_rois, total_area_px, mean_area, and sd_area. Image summaries also include entry_id, so repeated display names remain separate. A 0-row tibble when there are no ROIs.

See Also

Other batch: at_batch_apply(), at_batch_check_geometry(), at_export_all()

at_tile *Read a tile from an image*

Description

The single, format-independent tile accessor. Returns a numeric array with dimensions [y, x, band] — always three-dimensional, even for a single band.

Usage

```
at_tile(
  img,
  level = 0L,
  xrange = NULL,
  yrange = NULL,
  bands = NULL,
  call = rlang::caller_env()
)
```

Arguments

img	An annot_image.
level	Integer pyramid level. Default 0.
xrange, yrange	Optional integer c(min, max) pixel ranges (1-based, inclusive) at level. NULL (default) means the full extent.
bands	Optional integer band indices. NULL (default) means all bands.
call	The calling environment, for error reporting.

Value

A numeric array with dimensions [y, x, band]. An image whose pixel handle was not saved is re-opened from its source descriptor; a missing or ambiguous source is an error with code IMAGE_UNAVAILABLE and a changed source IMAGE_CHANGED (see at_inspect_source() and at_relink_source()).

See Also

Other images: at_bands(), at_close_image(), at_dims(), at_example_image(), at_hsi_meta(), at_image_capabilities(), at_is_pyramidal(), at_is_spectral(), at_meta(), at_n_bands(), at_n_levels(), at_pixel_size(), at_read_image(), at_wavelengths(), hyperannotatR-classes

Other backends: at_backend_detect(), at_backend_get(), at_backend_list(), at_backend_register(), at_close_image(), at_cubert_export_envi(), at_image_capabilities(), at_read_image(), at_tivita_profile()

Examples

```
img <- at_example_image("tissue")
dim(at_tile(img, xrange = c(1, 16), yrange = c(1, 16)))
```

at_tile_source *Build a tile-source descriptor for the canvas*

Description

Build a tile-source descriptor for the canvas

Usage

```
at_tile_source(
  img,
  level_max = NULL,
  tile_size = 512L,
  embed = TRUE,
  view = NULL,
  call = rlang::caller_env()
)
```

Arguments

img	An annot_image.
level_max	Optional maximum level to expose; the image maximum by default.
tile_size	Tile size in pixels. Default 512.
embed	Logical; embed a downsampled base image as a data URI (for the self-contained canvas). Default TRUE.
view	Optional band view: a list with operation (see at_band_operations()), bands and params. NULL shows natural-colour RGB (nearest bands to 640/550/460 nm for spectral cubes). Display only. A view that does not apply to img (for example bands it does not have) falls back to the default view and is reported in viewError.
call	The calling environment, for error reporting.

Value

A list describing the image for the widget: width, height (level-0 pixels), nBands, and the pyramid description tileSize, minLevel and maxLevel (informational: the built-in canvas displays a single embedded image rather than streaming tiles). When embed is TRUE it also has dataUri (a PNG data URI); when no image could be encoded, dataUri is absent and displayError says why, and the canvas shows that message instead of an image. viewError says why view was not applied, when it was not.

See Also

Other shiny: atCanvasOutput(), at_annotate(), at_app(), at_canvas(), at_canvas_fit(), at_canvas_proxy(), at_canvas_set_annotations(), at_canvas_set_band(), at_canvas_set_overlay(), at_canvas_set_selection(), at_canvas_set_tool(), renderAtCanvas()

`at_tivita_profile` *Declare a TIVITA SpecCube layout profile*

Description

A profile states everything needed to decode a bare `*_SpecCube.dat` file; hyperannotatR never infers these from a file name. The default arguments describe the documented 640 x 480 x 100-band layout (500-995 nm), which is a profile like any other and can be replaced for a different device.

Usage

```
at_tivita_profile(
  name = .TIVITA_DEFAULT_PROFILE,
  width = 640L,
  height = 480L,
  bands = 100L,
  header_values = 3L,
  value_bytes = 4L,
  endian = c("big", "little"),
  axis_order = "x,y,band",
  wavelengths = seq(500, 995, by = 5),
  wavelength_unit = "nm",
  value_unit = "reflectance",
  call = rlang::caller_env()
)
```

Arguments

`name`	Profile identifier recorded in the image metadata.
`width`, `height`, `bands`	Cube dimensions (columns x, rows y, bands).
`header_values`	Number of leading values before the payload.
`value_bytes`	Bytes per value (`4` for float32, `8` for float64).
`endian`	`"big"` or `"little"`.
`axis_order`	Storage order from slowest to fastest varying axis; only `"x,y,band"` (the numpy C-order TIVITA layout) is supported.
`wavelengths`	Numeric band-centre wavelengths (length `bands`) or `NULL` when unknown.
`wavelength_unit`	Wavelength unit, default `"nm"`.
`value_unit`	What the values mean: one of `"raw"`, `"reflectance"`, `"radiance"`, `"intensity"`, `"absorbance"`, `"unknown"`.
`call`	The calling environment, for error reporting.

Value

A list of class `at_tivita_profile`.

See Also

`at_read_image()`

Other hsi: `at_band_operations()`, `at_band_view()`, `at_convert_values()`, `at_cubert_export_envi()`, `at_hsi_meta()`, `at_read_stats()`

Other backends: `at_backend_detect()`, `at_backend_get()`, `at_backend_list()`, `at_backend_register()`, `at_close_image()`, `at_cubert_export_envi()`, `at_image_capabilities()`, `at_read_image()`, `at_tile()`

Examples

```
at_tivita_profile()$bands
small <- at_tivita_profile("bench_4x3x2", width = 4, height = 3, bands = 2,
                           wavelengths = c(600, 700))
```

---

at_training_check *Check a training dataset*

Description

Run independent checks on an exported training dataset: the integrity inventory (schema, safe paths, symbolic links, sizes and SHA-256 hashes, missing and unlisted files), the manifest schema, split leakage (every group and image in exactly one split, every tile in its image's split), mask dimensions and orientation, integer label range against the class legend, the recorded overlap counts, and a mask -> ROI -> mask round trip on a sample of tiles. When the source `x` is supplied, each sampled tile is also re-rasterised from the annotations and compared exactly. Tile files outside the verified inventory are never opened. A check that cannot run on a malformed manifest is reported as `"fail"` rather than raising an error; only an unsupported major version aborts (`PROTOCOL_MISMATCH`).

Usage

```
at_training_check(
  dataset,
  x = NULL,
  max_roundtrip_tiles = 20L,
  call = rlang::caller_env()
)
```

Arguments

`dataset`	Path of a dataset written by `at_training_export()`.
`x`	Optional source annot_session or annot_project.
`max_roundtrip_tiles`	Maximum number of tiles sampled for round trips.
`call`	The calling environment, for error reporting.

Value

A tibble of class `at_training_check` with `check`, `status` (`"ok"`, `"warn"`, `"fail"`) and `detail`.

See Also

Other training: `at_training_export()`, `at_training_import()`

`at_training_export` *Export a DNN training dataset*

Description

Tile annotated images into a portable training dataset with integer masks, deterministic grouped splits and a complete provenance manifest. Splits are assigned per group (by default per image, or by `subject_id`/`sample_id` from `groups`), so tiles of one group never occur in two splits. Masks are written as integer TIFF (or NPY) with background `0`; class codes are shared across the dataset and instance ids are only produced for `mask_type = "instance"`. Tiles that do not fit the image are excluded as partial; empty and ambiguous (overlapping labels) tiles are counted and handled by policy. Image bytes are excluded unless `include_images = TRUE`.

Usage

```
at_training_export(
  x,
  destination,
  split = NULL,
  tile_size = 256L,
  overlap = 0L,
  level = 0L,
  mask_type = c("labelled", "instance", "binary"),
  bands = NULL,
  include_images = FALSE,
  seed = 1L,
  group_by = c("image_id", "sample_id", "subject_id"),
  groups = NULL,
  fractions = c(train = 0.7, validation = 0.15, test = 0.15),
  label_map = NULL,
  empty_tiles = c("keep", "exclude"),
  mask_format = c("tiff", "npy"),
  normalization = NULL,
  overwrite = FALSE,
  call = rlang::caller_env()
)
```

Arguments

x	An annot_session (materialised entries) or annot_project.
destination	New dataset directory.
split	Optional explicit assignment: a named list (train, validation, test) of group ids. With group_by = "image_id" entry ids ("entry-0001") or entry names are accepted too and mapped to their image's group. NULL assigns groups by fractions and seed.
tile_size	Tile edge length in pixels at level.
overlap	Tile overlap in pixels (0 <= overlap < tile_size).
level	Pyramid level.
mask_type	"labelled", "instance" or "binary".
bands	Band indices recorded (and written with include_images); every entry must have them. With include_images = TRUE all entries must share the same band table (BAND_MISMATCH otherwise).
include_images	Logical; also write float32 NPY image tiles.
seed	Split seed (recorded).
group_by	Group column used for splitting: "image_id" (default), "sample_id" or "subject_id".
groups	Optional data frame with entry_id (the session's entry id, or the queue-position id "entry-NNNN"; else name) plus subject_id and/or sample_id columns (character).
fractions	Named split fractions. Split names (here and in split) are used as directory names and must consist of letters, digits, ., _ or -.
label_map	Optional named integer class codes (label = code, codes >= 1); defaults to first-seen label order.
empty_tiles	"keep" (default) or "exclude" tiles without foreground.
mask_format	"tiff" (default) or "npy".
normalization	Optional list describing value normalisation applied downstream (recorded, not applied).
overwrite	Logical; replace an existing dataset directory.
call	The calling environment, for error reporting.

Details

mask_type uses training terminology: "labelled" is a semantic class-code mask (hyperannotatR "multiclass"), "instance" has one id per ROI (hyperannotatR "labelled"), "binary" marks foreground. The mapping is recorded in the manifest.

Value

An at_training_export list: destination, dataset_digest, manifest, files and checks (from at_training_check()).

See Also

at_training_check(), at_training_import()

Other training: at_training_check(), at_training_import()

Examples

```
sess <- at_example_session(2)
for (i in 1:2) sess$projects[[i]] <- at_example_project()
ds <- at_training_export(sess, file.path(tempdir(), "train-example"), tile_size = 256,
                         overwrite = TRUE)
ds$manifest$counts$tiles
# Both example entries show the same picture, so they form one image
# group and land in one split: tiles of one image never leak across splits.
ds$manifest$counts$images_by_split
```

---

at_training_import *Import model predictions as staged annotations*

Description

Convert an integer prediction mask into ROIs and stage them against a project; nothing is committed. The prediction must match the image dimensions at `level`, contain only integers, and every non-background value must be in `label_map`; otherwise the import aborts. Optional per-pixel confidence below `min_confidence` is set to background first. Model and prediction digests and the confidence policy are recorded on every ROI, which starts as `review_status = "unreviewed"`. Reviewed or locked annotations are never overwritten: with `replace = TRUE` they become staged conflicts.

Usage

```
at_training_import(
  x,
  predictions,
  label_map,
  level = 0L,
  source = "dnn_prediction",
 model = list(name = NA_character_, version = NA_character_, sha256 = NA_character_),
  confidence = NULL,
  min_confidence = NULL,
  layer = "predictions",
  replace = FALSE,
  stage = TRUE,
  call = rlang::caller_env()
)
```

Arguments

x	An annot_project.
predictions	A mask file (TIFF/PNG/NPY), an annot_mask, or an integer matrix [y, x].
label_map	Named integer vector label = code (codes >= 1; 0 is background).
level	Pyramid level of the prediction grid.
source	ROI source string. Default "dnn_prediction".
model	List with name, version and sha256 of the model.
confidence	Optional numeric matrix of per-pixel confidence.
min_confidence	Optional threshold applied to confidence (which is then required).
layer	Target layer name. Default "predictions".
replace	Logical; stage removal of existing unreviewed ROIs of layer not present in the prediction.
stage	Logical; return an at_staged_patch (default) or, with FALSE, the prediction layer only.
call	The calling environment, for error reporting.

Value

An at_staged_patch (or an annot_layer when stage = FALSE) with attribute prediction_digest.

See Also

at_commit_qupflowr()

Other training: at_training_check(), at_training_export()

---

at_transform *Transform an ROI between pyramid levels using real image dimensions*

Description

Unlike at_roi_rescale() (which assumes a power-of-two pyramid), this uses the actual level dimensions of an annot_image, so it is exact for pyramids with any downsampling ratio.

Usage

```
at_transform(
  roi,
  from_level = roi$level,
  to_level,
  img,
  call = rlang::caller_env()
)
```

Arguments

`roi`	An annot_roi.
`from_level`	Integer source pyramid level; defaults to, and must equal, the ROI's `level`, because that is the level its coordinates are stored at.
`to_level`	Integer target pyramid level.
`img`	The annot_image whose pyramid defines the level dimensions.
`call`	The calling environment, for error reporting.

Value

An annot_roi with coordinates at `to_level`. Invertible within floating-point precision (relative tolerance `1e-12`).

See Also

`at_roi_rescale()`

Other geometry: `at_check_containment()`, `at_check_geometry()`, `at_clamp()`, `at_fix_geometry()`, `at_flip_y()`, `at_roi_buffer()`, `at_roi_contains()`, `at_roi_distance()`, `at_roi_overlaps()`, `at_roi_rescale()`, `at_roi_ring()`, `at_roi_setops`, `at_roi_simplify()`, `at_rois_overlap()`, `at_snap()`

Examples

```
img <- at_example_image("multiplex")
r <- at_roi_rect(0, 0, 100, 100, label = "a")
at_transform(r, from_level = 0, to_level = 2, img = img)
```

`at_validate` *Validate a project's invariants*

Description

Check that layer names are unique, ROI identifiers are unique across the project, every ROI has one geometry at a pyramid level the image has, and all geometries are valid.

Usage

```
at_validate(project, call = rlang::caller_env())
```

Arguments

`project`	An annot_project.
`call`	The calling environment, for error reporting.

Value

The project, invisibly, if valid; otherwise an error describing the violated invariant.

See Also

`at_check_geometry()`

Other projects: `at_add_layer()`, `at_add_roi()`, `at_example_project()`, `at_layers()`, `at_project()`, `at_remove_layer()`, `at_remove_roi()`, `at_rois()`, `at_summary()`

Examples

```
proj <- at_validate(at_example_project())
```

---

at_wavelengths *Band centre wavelengths*

Description

Band centre wavelengths

Usage

```
at_wavelengths(x, call = rlang::caller_env())
```

Arguments

`x`	An annot_image.
`call`	The calling environment, for error reporting.

Value

A numeric vector of length `at_n_bands(x)` for a spectral image, or `numeric(0)` when the image is not spectral.

See Also

`at_is_spectral()`

Other images: `at_bands()`, `at_close_image()`, `at_dims()`, `at_example_image()`, `at_hsi_meta()`, `at_image_capabilities()`, `at_is_pyramidal()`, `at_is_spectral()`, `at_meta()`, `at_n_bands()`, `at_n_levels()`, `at_pixel_size()`, `at_read_image()`, `at_tile()`, `hyperannotatR-classes`

`at_write_geojson` *Write a project's ROIs to GeoJSON*

Description

Write a project's ROIs to GeoJSON

Usage

```
at_write_geojson(
  project,
  path,
  layer = NULL,
  level = 0L,
  flip_y = FALSE,
  overwrite = FALSE,
  call = rlang::caller_env()
)
```

Arguments

`project`	An annot_project.
`path`	Output file path.
`layer`	Optional layer filter.
`level`	Integer pyramid level for the coordinates and the exported `level` property. Default `0`. `source_level` separately records the stored ROI level, and `coordinate_schema` marks the file as recording the level of its coordinates, so `at_read_geojson()` restores ROIs at that level.
`flip_y`	Logical; invert the y-axis (for GIS-oriented consumers). Default `FALSE` (image orientation preserved). The consequence is documented in the interop vignette.
`overwrite`	Logical; overwrite an existing file. Default `FALSE`.
`call`	The calling environment, for error reporting.

Value

The output path, invisibly.

See Also

`at_read_geojson()`, `at_write_qupath()`

Other io: `at_inspect_saved()`, `at_inspect_source()`, `at_load_project()`, `at_load_session()`, `at_read_geojson()`, `at_read_qupath()`, `at_read_rois_csv()`, `at_relink_source()`, `at_roi_from_geojson()`, `at_save_project()`, `at_save_session()`, `at_write_qupath()`, `at_write_rois_csv()`

Examples

```
p <- tempfile(fileext = ".geojson")
at_write_geojson(at_example_project(), p)
```

`at_write_mask` *Write a mask to disk*

Description

Write a mask to disk

Usage

```
at_write_mask(
  mask,
  path,
  format = c("tiff", "png", "rds"),
  bits = c(16L, 8L),
  legend = TRUE,
  overwrite = FALSE,
  call = rlang::caller_env()
)
```

Arguments

`mask`	An `annot_mask`.
`path`	Output file path.
`format`	`"tiff"` (default), `"png"` (8-bit only), or `"rds"` (full-fidelity `annot_mask`).
`bits`	Single finite whole-number bit depth for TIFF: `16` (default) or `8`, including `16L`/`8L`. Explicit vectors are rejected.
`legend`	Logical; also write a `<path>.legend.json` sidecar describing the mask. Default `TRUE`.
`overwrite`	Logical; overwrite an existing file. Default `FALSE`.
`call`	The calling environment, for error reporting.

Details

TIFF exports preserve unsigned 8/16-bit codes; PNG exports are unsigned 8-bit only, regardless of `bits`. Samples must be finite integers in the selected unsigned range; invalid samples reject before writing. TIFF writing requires `tiff` or a capable `magick` build. PNG writing requires `magick` with exact grayscale support. Missing exact conversion capability produces an error rather than a quantized output. The caller's explicit path is respected. Mask and legend are staged before either is replaced; a preexisting sidecar also requires `overwrite=TRUE`. Publication failure rolls back prior files. Concurrent writers must be serialized.

Value

The output path, invisibly.

See Also

at_read_mask()

Other masks: at_mask(), at_mask_agreement(), at_mask_boundary(), at_mask_derive(), at_mask_legend(), at_mask_preview(), at_mask_stack(), at_mask_stats(), at_read_mask(), at_read_npy(), at_write_masks(), at_write_npy()

Examples

```
m <- at_mask(at_example_project(), "labelled")
f <- tempfile(fileext = ".tif")
at_write_mask(m, f)
```

---

at_write_masks *Batch-write the masks of a project*

Description

Batch-write the masks of a project

Usage

```
at_write_masks(
  project,
  dir,
  per = c("layer", "roi", "class", "project"),
  type = c("labelled", "binary", "multiclass"),
  level = 0L,
  overwrite = FALSE,
  call = rlang::caller_env()
)
```

Arguments

project	An annot_project.
dir	Output directory (created if needed).
per	One of "layer", "roi", "class", or "project"; controls how masks are split into files.
type	Mask type passed to at_mask().
level	Integer pyramid level. Default 0.
overwrite	Logical; overwrite existing files. Default FALSE.
call	The calling environment, for error reporting.

Details

Output plans reject symlinked descendants and aliases before writing. Callers must serialize concurrent writes and keep destination paths stable; preflight plus staged renames is not a race-free filesystem transaction.

Value

An invisible export-receipt tibble::tibble with columns `path`, `type`, `n_px`, `bytes`, `name`, `status`, `message`, `sidecar_path`, and `sidecar_bytes`. Counts and bytes use double precision. Display names are retained in `name`; filenames use safe components with unique suffixes. Each item is `"ok"`, `"error"`, or `"skipped"`; one failure does not abort later items. A TIFF and its legend form one staged output bundle.

See Also

Other masks: `at_mask()`, `at_mask_agreement()`, `at_mask_boundary()`, `at_mask_derive()`, `at_mask_legend()`, `at_mask_preview()`, `at_mask_stack()`, `at_mask_stats()`, `at_read_mask()`, `at_read_npy()`, `at_write_mask()`, `at_write_npy()`

---

at_write_npy *Write a mask to a NumPy* `.npy` *file*

Description

Write an `annot_mask` as a lossless integer NumPy array, for interchange with Python tools. A `<path>.legend.json` sidecar (value -> label, plus the mask metadata and storage orientation) is written alongside so the array stays self-describing. Unlike `at_write_mask()`, values are stored raw (no bit-depth normalisation), so bitfield artefact masks survive intact. The array and legend are staged as one output bundle; an existing sidecar also requires `overwrite = TRUE`. Values outside the chosen `dtype` are refused before anything is written. Concurrent writers must be serialised.

Usage

```
at_write_npy(
  mask,
  path,
  dtype = c("int32", "uint8", "int16"),
  transpose = FALSE,
  legend = TRUE,
  overwrite = FALSE,
  call = rlang::caller_env()
)
```

Arguments

mask	An annot_mask.
path	Output file path.
dtype	NumPy integer dtype: "int32" (default, wide enough for bitfield codes), "uint8", or "int16".
transpose	Logical; write the transpose, i.e. numpy shape (width, height) = [x, y], matching tools that store the native cube order (e.g. TIVITA/HyperGui). Default FALSE writes the standard image convention (height, width).
legend	Logical; also write a <path>.legend.json sidecar. Default TRUE.
overwrite	Logical; overwrite an existing file. Default FALSE.
call	The calling environment, for error reporting.

Value

The output path, invisibly.

See Also

at_read_npy(), at_write_mask()

Other masks: at_mask(), at_mask_agreement(), at_mask_boundary(), at_mask_derive(), at_mask_legend(), at_mask_preview(), at_mask_stack(), at_mask_stats(), at_read_mask(), at_read_npy(), at_write_mask(), at_write_masks()

Examples

```
m <- at_mask(at_example_project(), "labelled")
f <- tempfile(fileext = ".npy")
at_write_npy(m, f)
```

---

at_write_qupath *Write a project's ROIs as QuPath GeoJSON*

Description

The default dialect = "qupath" writes what QuPath 0.4 and later read natively (verified with QuPath 0.7.0): objectType, classification with an [r, g, b] colour, isLocked for locked ROIs, a deterministic UUID per ROI as the feature id (QuPath replaces non-UUID ids; existing UUIDs are kept), and hyperannotatR's own ROI id, layer, source and stored level in properties.metadata. dialect = "legacy" writes the hyperannotatR 0.1 form (object_type, signed colorRGB), which QuPath still reads with a deprecation warning. Publication uses a staged replacement.

Usage

```
at_write_qupath(
  project,
  path,
  layer = NULL,
  level = 0L,
  overwrite = FALSE,
  dialect = c("qupath", "legacy"),
  call = rlang::caller_env()
)
```

Arguments

project	An annot_project.
path	Output file path.
layer	Optional layer filter.
level	Integer pyramid level for the coordinates and the level property. Default 0, as QuPath expects; other levels are for consumers that explicitly support them.
overwrite	Logical; overwrite an existing file. Default FALSE.
dialect	"qupath" (default) or "legacy".
call	The calling environment, for error reporting.

Details

Every feature records the pyramid level of its written coordinates as the level property, the ROI's stored level as source_level, and the marker coordinate_schema = "annotatR-pixel-level-v1". QuPath keeps only metadata through its own import and export, so the QuPath dialect repeats the declaration there as annotatr_coordinate_schema and annotatr_coordinate_level; at_read_qupath() restores ROIs at that level.

QuPath keeps one colour per classification name, so a label that has different colours in different layers is written with the first colour found for it.

Value

The output path, invisibly.

See Also

at_read_qupath(), at_write_geojson()

Other io: at_inspect_saved(), at_inspect_source(), at_load_project(), at_load_session(), at_read_geojson(), at_read_qupath(), at_read_rois_csv(), at_relink_source(), at_roi_from_geojson(), at_save_project(), at_save_session(), at_write_geojson(), at_write_rois_csv()

Examples

```
p <- tempfile(fileext = ".geojson")
at_write_qupath(at_example_project(), p)
```

`at_write_rois_csv` *Write a project's ROIs to CSV*

Description

One row per ROI with a WKT `geometry` column. Coordinates and `level` are at level 0; `source_level` records the stored ROI level as provenance, and `coordinate_schema` marks the file as recording the level of its coordinates. WKT coordinates use 17 significant digits and a decimal point, independently of session display options, to preserve double-precision geometry.

Usage

```
at_write_rois_csv(project, path, overwrite = FALSE, call = rlang::caller_env())
```

Arguments

`project`	An annot_project.
`path`	Output `.csv` path.
`overwrite`	Logical; overwrite an existing file. Default `FALSE`.
`call`	The calling environment, for error reporting.

Value

The path, invisibly.

See Also

`at_read_rois_csv()`

Other io: `at_inspect_saved()`, `at_inspect_source()`, `at_load_project()`, `at_load_session()`, `at_read_geojson()`, `at_read_qupath()`, `at_read_rois_csv()`, `at_relink_source()`, `at_roi_from_geojson()`, `at_save_project()`, `at_save_session()`, `at_write_geojson()`, `at_write_qupath()`

Examples

```
p <- tempfile(fileext = ".csv")
at_write_rois_csv(at_example_project(), p)
```

atCanvasOutput *Shiny output for the annotation canvas*

Description

Shiny output for the annotation canvas

Usage

```
atCanvasOutput(outputId, width = "100%", height = "600px")
```

Arguments

outputId	Output id.
width, height	Output dimensions.

Value

A Shiny output element.

See Also

Other shiny: at_annotate(), at_app(), at_canvas(), at_canvas_fit(), at_canvas_proxy(), at_canvas_set_annotations(), at_canvas_set_band(), at_canvas_set_overlay(), at_canvas_set_selection(), at_canvas_set_tool(), at_tile_source(), renderAtCanvas()

hyperannotatR-classes *hyperannotatR S3 classes*

Description

hyperannotatR represents annotations with a small set of S3 classes. This page collects them so they can be cross-referenced from the documentation.

annot_image A backend-agnostic image handle holding metadata and the backend's pixel handle, read through at_tile(). Windowed backends (ENVI, TIVITA) read only the requested window; the raster and TIFF backends decode the whole image when it is read. Created by at_read_image() or at_example_image().

annot_roi A single region of interest: an sf geometry in image pixel coordinates at a declared pyramid level. Created by the at_roi_*() constructors.

annot_layer A named collection of ROIs sharing a label vocabulary and a visual style. Created by at_layer().

annot_project An image plus a named set of layers and provenance. Created by at_project().

annot_session A resumable batch-annotation session over a queue of images. Created by at_session().

`annot_mask` A rasterised mask: an integer matrix (binary, labelled, multiclass, or bitfield-encoded) with a legend. Created by `at_mask()`.

`annot_style` A layer style specification. Created by `at_style()`.

`annot_summary` A compact per-label project summary. Created by `at_summary()`.

See Also

`at_project()`, `at_layer()`

Other images: `at_bands()`, `at_close_image()`, `at_dims()`, `at_example_image()`, `at_hsi_meta()`, `at_image_capabilities()`, `at_is_pyramidal()`, `at_is_spectral()`, `at_meta()`, `at_n_bands()`, `at_n_levels()`, `at_pixel_size()`, `at_read_image()`, `at_tile()`, `at_wavelengths()`

`renderAtCanvas` *Render an annotation canvas in Shiny*

Description

Render an annotation canvas in Shiny

Usage

```
renderAtCanvas(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`expr`	An expression producing an `at_canvas()`.
`env`	The environment to evaluate `expr` in.
`quoted`	Is `expr` already quoted?

Value

A Shiny render function.

See Also

Other shiny: `atCanvasOutput()`, `at_annotate()`, `at_app()`, `at_canvas()`, `at_canvas_fit()`, `at_canvas_proxy()`, `at_canvas_set_annotations()`, `at_canvas_set_band()`, `at_canvas_set_overlay()`, `at_canvas_set_selection()`, `at_canvas_set_tool()`, `at_tile_source()`

Index

∗ **backends**
at_backend_detect, 10
at_backend_get, 11
at_backend_list, 11
at_backend_register, 12
at_close_image, 27
at_cubert_export_envi, 38
at_image_capabilities, 52
at_read_image, 85
at_tile, 121
at_tivita_profile, 124
∗ **batch**
at_batch_apply, 16
at_batch_check_geometry, 17
at_export_all, 43
at_summary_table, 121
∗ **control**
at_control_capabilities, 29
at_control_close, 30
at_control_command, 30
at_control_connect, 31
at_control_events, 32
at_control_request_status, 33
at_control_start, 34
at_control_state, 36
at_control_stop, 36
∗ **example data**
at_example_image, 40
at_example_path, 41
at_example_project, 42
at_example_session, 42
∗ **extraction**
at_extract, 46
at_extract_pixels, 47
at_extract_spectrum, 48
∗ **geometry**
at_check_containment, 24
at_check_geometry, 25
at_clamp, 26
at_fix_geometry, 49
at_flip_y, 50
at_roi_buffer, 96
at_roi_contains, 99
at_roi_distance, 99
at_roi_overlaps, 104
at_roi_rescale, 107
at_roi_ring, 108
at_roi_setops, 109
at_roi_simplify, 110
at_rois_overlap, 112
at_snap, 117
at_transform, 129
∗ **hsi**
at_band_operations, 13
at_band_view, 14
at_convert_values, 37
at_cubert_export_envi, 38
at_hsi_meta, 51
at_read_stats, 91
at_tivita_profile, 124
∗ **images**
at_bands, 15
at_close_image, 27
at_dims, 39
at_example_image, 40
at_hsi_meta, 51
at_image_capabilities, 52
at_is_pyramidal, 58
at_is_spectral, 58
at_meta, 74
at_n_bands, 74
at_n_levels, 75
at_pixel_size, 76
at_read_image, 85
at_tile, 121
at_wavelengths, 131
hyperannotatR-classes, 139
∗ **interop**

at_annotation_revision, 8
at_commit_qupflowr, 27
at_export_qupflowr, 44
at_import_qupflowr, 53
at_interop_capabilities, 56
at_interop_manifest, 57
at_stage_qupflowr, 118
* **io**
at_inspect_saved, 54
at_inspect_source, 55
at_load_project, 63
at_load_session, 64
at_read_geojson, 84
at_read_qupath, 89
at_read_rois_csv, 90
at_relink_source, 92
at_roi_from_geojson, 102
at_save_project, 113
at_save_session, 114
at_write_geojson, 132
at_write_qupath, 136
at_write_rois_csv, 138
* **layers**
at_layer, 59
at_layer_add, 60
at_layer_n, 61
at_layer_remove, 61
at_layer_rois, 62
at_style, 119
* **masks**
at_mask, 65
at_mask_agreement, 67
at_mask_boundary, 69
at_mask_derive, 69
at_mask_legend, 71
at_mask_preview, 71
at_mask_stack, 72
at_mask_stats, 73
at_read_mask, 86
at_read_npy, 88
at_write_mask, 133
at_write_masks, 134
at_write_npy, 135
* **plots**
at_plot, 77
at_plot_image, 78
at_plot_mask, 79
at_plot_overlay, 80
at_plot_project, 80
at_plot_spectrum, 81
at_plot_summary, 82
* **projects**
at_add_layer, 5
at_add_roi, 6
at_example_project, 42
at_layers, 62
at_project, 83
at_remove_layer, 93
at_remove_roi, 94
at_rois, 111
at_summary, 120
at_validate, 130
* **rois**
at_roi_area, 95
at_roi_bbox, 96
at_roi_centroid, 97
at_roi_circle, 98
at_roi_ellipse, 100
at_roi_freehand, 101
at_roi_from_geojson, 102
at_roi_from_sf, 103
at_roi_point, 104
at_roi_polygon, 105
at_roi_rect, 106
* **sessions**
at_current, 39
at_example_session, 42
at_goto, 51
at_manifest, 64
at_next, 76
at_prev, 83
at_resume, 94
at_session, 115
at_session_status, 116
at_set_status, 116
* **shiny**
at_annotate, 6
at_app, 8
at_canvas, 17
at_canvas_fit, 19
at_canvas_proxy, 20
at_canvas_set_annotations, 20
at_canvas_set_band, 21
at_canvas_set_overlay, 22
at_canvas_set_selection, 23
at_canvas_set_tool, 23

at_tile_source, 123
atCanvasOutput, 139
renderAtCanvas, 140
∗ **training**
at_training_check, 125
at_training_export, 126
at_training_import, 128

annot_image, *7*, *9*, *12*, *14, 15*, *18*, *27*, *39, 40*, *46*, *48, 49*, *52*, *55*, *57–59*, *62*, *66*, *74–78*, *80*, *83*, *85, 86*, *92*, *95*, *100*, *104*, *107*, *109*, *112*, *122, 123*, *129–131*
annot_image *(*hyperannotatR-classes*)*, 139
annot_layer, *5*, *7, 8*, *54*, *57*, *59–62*, *65*, *77*, *83*, *85–87*, *90, 91*, *109*, *112*, *115*, *118, 119*, *129*
annot_layer *(*hyperannotatR-classes*)*, 139
annot_mask *(*hyperannotatR-classes*)*, 139
annot_project, *5–9*, *16*, *18*, *21*, *24, 25*, *27*, *39*, *42*, *45, 46*, *48–50*, *52*, *55*, *57*, *62, 63*, *65*, *72*, *77*, *80–82*, *84*, *92–94*, *111*, *113*, *118*, *120*, *125*, *127*, *129, 130*, *132*, *134*, *137, 138*
annot_project *(*hyperannotatR-classes*)*, 139
annot_roi, *6*, *26*, *50*, *57*, *60*, *65*, *95–110*, *117*, *130*
annot_roi *(*hyperannotatR-classes*)*, 139
annot_session, *7–9*, *16, 17*, *34*, *39*, *43*, *45*, *51*, *57*, *64*, *76*, *83*, *94*, *114–117*, *121*, *125*, *127*
annot_session *(*hyperannotatR-classes*)*, 139
annot_style *(*hyperannotatR-classes*)*, 139
annot_summary *(*hyperannotatR-classes*)*, 139
at_add_layer, 5
at_add_layer(), *6*, *42*, *63*, *84*, *93, 94*, *112*, *120*, *131*
at_add_roi, 6
at_add_roi(), *6*, *42*, *63*, *84*, *93, 94*, *112*, *120*, *131*
at_annotate, 6
at_annotate(), *8*, *10*, *19–24*, *34*, *123*, *139, 140*
at_annotation_revision, 8
at_annotation_revision(), *28*, *45*, *54*, *56, 57*, *118*
at_app, 8
at_app(), *6, 7*, *19–24*, *35*, *56*, *123*, *139, 140*
at_backend_detect, 10
at_backend_detect(), *11–13*, *27*, *38*, *53*, *86*, *122*, *125*
at_backend_get, 11
at_backend_get(), *10*, *12, 13*, *27*, *38*, *53*, *86*, *122*, *125*
at_backend_list, 11
at_backend_list(), *10, 11*, *13*, *27*, *38*, *53*, *86*, *122*, *125*
at_backend_register, 12
at_backend_register(), *10–12*, *27*, *38*, *53*, *86*, *122*, *125*
at_band_operations, 13
at_band_operations(), *14*, *38*, *52*, *92*, *123*, *125*
at_band_view, 14
at_band_view(), *13*, *21*, *38*, *52*, *92*, *125*
at_bands, 15
at_bands(), *27*, *40, 41*, *51–53*, *58, 59*, *74, 75*, *77*, *86*, *122*, *131*, *140*
at_batch_apply, 16
at_batch_apply(), *17*, *44*, *121*
at_batch_check_geometry, 17
at_batch_check_geometry(), *16*, *44*, *121*
at_canvas, 17
at_canvas(), *7*, *10*, *20–24*, *123*, *139, 140*
at_canvas_fit, 19
at_canvas_fit(), *7*, *10*, *19–24*, *123*, *139, 140*
at_canvas_proxy, 20
at_canvas_proxy(), *7*, *10*, *19–24*, *123*, *139*, *140*
at_canvas_set_annotations, 20
at_canvas_set_annotations(), *7*, *10*, *18–20*, *22–24*, *123*, *139, 140*
at_canvas_set_band, 21
at_canvas_set_band(), *7*, *10*, *19–24*, *123*, *139, 140*
at_canvas_set_overlay, 22
at_canvas_set_overlay(), *7*, *10*, *19–24*, *123*, *139, 140*
at_canvas_set_selection, 23
at_canvas_set_selection(), *7*, *10*, *19–22*, *24*, *123*, *139, 140*
at_canvas_set_tool, 23
at_canvas_set_tool(), *7*, *10*, *19–23*, *123*, *139, 140*

at_check_containment, 24
at_check_containment(), *25, 26*, *50*, *97*, *99, 100*, *104*, *107–110*, *112*, *117*, *130*
at_check_geometry, 25
at_check_geometry(), *17*, *24–26*, *50*, *97*, *99, 100*, *104*, *107–110*, *112*, *117*, *130*
at_clamp, 26
at_clamp(), *25*, *50*, *97*, *99, 100*, *104*, *107–110*, *112*, *117*, *130*
at_close_image, 27
at_close_image(), *10–13*, *15*, *38*, *40, 41*, *52, 53*, *58, 59*, *74, 75*, *77*, *86*, *122*, *125*, *131*, *140*
at_commit_qupflowr, 27
at_commit_qupflowr(), *8*, *45*, *54*, *56, 57*, *118*, *129*
at_control_capabilities, 29
at_control_capabilities(), *30–33*, *35–37*, *56*
at_control_close, 30
at_control_close(), *29*, *31–33*, *35–37*
at_control_command, 30
at_control_command(), *29, 30*, *32, 33*, *35–37*
at_control_connect, 31
at_control_connect(), *29–31*, *33*, *35–37*
at_control_events, 32
at_control_events(), *29–33*, *35–37*
at_control_request_status, 33
at_control_request_status(), *29–33*, *35–37*
at_control_start, 34
at_control_start(), *9, 10*, *29–33*, *36, 37*
at_control_state, 36
at_control_state(), *29–33*, *35*, *37*
at_control_stop, 36
at_control_stop(), *29–33*, *35, 36*
at_convert_values, 37
at_convert_values(), *13, 14*, *38*, *52*, *92*, *125*
at_cubert_export_envi, 38
at_cubert_export_envi(), *10–14*, *27*, *38*, *52, 53*, *86*, *92*, *122*, *125*
at_current, 39
at_current(), *43*, *51*, *64, 65*, *76*, *83*, *95*, *115–117*
at_dims, 39
at_dims(), *15*, *27*, *41*, *52, 53*, *58, 59*, *74, 75*, *77*, *86*, *122*, *131*, *140*
at_example_image, 40
at_example_image(), *15*, *27*, *40–43*, *52, 53*, *58, 59*, *74, 75*, *77*, *86*, *122*, *131*, *140*
at_example_path, 41
at_example_path(), *41–43*
at_example_project, 42
at_example_project(), *6*, *41*, *43*, *63*, *84*, *93, 94*, *112*, *120*, *131*
at_example_session, 42
at_example_session(), *39*, *41, 42*, *51*, *65*, *76*, *83*, *95*, *115–117*
at_export_all, 43
at_export_all(), *16, 17*, *121*
at_export_qupflowr, 44
at_export_qupflowr(), *8*, *28*, *54*, *56, 57*, *118*
at_extract, 46
at_extract(), *48, 49*, *82*
at_extract_pixels, 47
at_extract_pixels(), *47*, *49*
at_extract_spectrum, 48
at_extract_spectrum(), *47, 48*, *82*
at_fix_geometry, 49
at_fix_geometry(), *25, 26*, *50*, *97*, *99, 100*, *104*, *107–110*, *112*, *117*, *130*
at_flip_y, 50
at_flip_y(), *25, 26*, *50*, *97*, *99, 100*, *104*, *107–110*, *112*, *117*, *130*
at_goto, 51
at_goto(), *39*, *43*, *65*, *76*, *83*, *95*, *115–117*
at_hsi_meta, 51
at_hsi_meta(), *13–15*, *27*, *38*, *40, 41*, *53*, *58, 59*, *74, 75*, *77*, *86*, *92*, *122*, *125*, *131*, *140*
at_image_capabilities, 52
at_image_capabilities(), *10–13*, *15*, *27*, *38*, *40, 41*, *52*, *58, 59*, *74, 75*, *77*, *86*, *122*, *125*, *131*, *140*
at_import_qupflowr, 53
at_import_qupflowr(), *8*, *28*, *45*, *56, 57*, *118*
at_inspect_saved, 54
at_inspect_saved(), *55*, *63, 64*, *85*, *90, 91*, *93*, *103*, *113, 114*, *132*, *137, 138*
at_inspect_source, 55
at_inspect_source(), *39*, *55*, *63, 64*, *85*, *90, 91*, *93*, *103*, *113, 114*, *122*, *132*, *137, 138*
at_interop_capabilities, 56
at_interop_capabilities(), *8*, *28, 29*, *45*,

54, *57*, *118*
at_interop_manifest, 57
at_interop_manifest(), *8*, *28*, *44, 45*, *54*, *56*, *118*
at_is_pyramidal, 58
at_is_pyramidal(), *15*, *27*, *40, 41*, *52, 53*, *59*, *74, 75*, *77*, *86*, *122*, *131*, *140*
at_is_spectral, 58
at_is_spectral(), *15*, *27*, *40, 41*, *52, 53*, *58*, *74, 75*, *77*, *86*, *122*, *131*, *140*
at_layer, 59
at_layer(), *60–62*, *120*, *139, 140*
at_layer_add, 60
at_layer_add(), *60–62*, *120*
at_layer_n, 61
at_layer_n(), *60*, *62*, *120*
at_layer_remove, 61
at_layer_remove(), *60–62*, *120*
at_layer_rois, 62
at_layer_rois(), *60–62*, *77*, *120*
at_layers, 62
at_layers(), *6*, *42*, *84*, *93, 94*, *112*, *120*, *131*
at_load_project, 63
at_load_project(), *55*, *64*, *85*, *90, 91*, *93*, *103*, *113, 114*, *132*, *137, 138*
at_load_session, 64
at_load_session(), *55*, *63*, *85*, *90, 91*, *93–95*, *103*, *113, 114*, *132*, *137, 138*
at_manifest, 64
at_manifest(), *39*, *43, 44*, *51*, *76*, *83*, *95*, *115–117*
at_mask, 65
at_mask(), *46*, *68–73*, *87*, *89*, *134–136*
at_mask_agreement, 67
at_mask_agreement(), *67*, *69–73*, *87*, *89*, *134–136*
at_mask_boundary, 69
at_mask_boundary(), *67, 68*, *70–73*, *87*, *89*, *134–136*
at_mask_derive, 69
at_mask_derive(), *67–69*, *71–73*, *87*, *89*, *134–136*
at_mask_legend, 71
at_mask_legend(), *67–70*, *72, 73*, *87*, *89*, *134–136*
at_mask_preview, 71
at_mask_preview(), *67–73*, *87*, *89*, *134–136*
at_mask_stack, 72
at_mask_stack(), *67–73*, *87*, *89*, *134–136*
at_mask_stats, 73
at_mask_stats(), *67–72*, *87*, *89*, *134–136*
at_meta, 74
at_meta(), *15*, *27*, *40, 41*, *52, 53*, *58, 59*, *75*, *77*, *86*, *122*, *131*, *140*
at_n_bands, 74
at_n_bands(), *15*, *27*, *40, 41*, *52, 53*, *58, 59*, *74, 75*, *77*, *86*, *122*, *131*, *140*
at_n_levels, 75
at_n_levels(), *15*, *27*, *40, 41*, *52, 53*, *58, 59*, *74, 75*, *77*, *86*, *122*, *131*, *140*
at_next, 76
at_next(), *39*, *43*, *51*, *65*, *83*, *95*, *115–117*
at_pixel_size, 76
at_pixel_size(), *15*, *27*, *40, 41*, *52, 53*, *58*, *59*, *74, 75*, *86*, *122*, *131*, *140*
at_plot, 77
at_plot(), *78–82*
at_plot_image, 78
at_plot_image(), *77*, *79–82*
at_plot_mask, 79
at_plot_mask(), *77, 78*, *80–82*
at_plot_overlay, 80
at_plot_overlay(), *77–79*, *81, 82*
at_plot_project, 80
at_plot_project(), *77–80*, *82*
at_plot_spectrum, 81
at_plot_spectrum(), *77–82*
at_plot_summary, 82
at_plot_summary(), *77–82*
at_prev, 83
at_prev(), *39*, *43*, *51*, *65*, *76*, *95*, *115–117*
at_project, 83
at_project(), *6*, *42*, *63*, *93, 94*, *112*, *120*, *131*, *139, 140*
at_read_geojson, 84
at_read_geojson(), *55*, *63, 64*, *90, 91*, *93*, *103*, *113, 114*, *132*, *137, 138*
at_read_image, 85
at_read_image(), *10–13*, *15*, *27*, *38*, *40, 41*, *52, 53*, *58, 59*, *74, 75*, *77*, *113*, *122*, *125*, *131*, *140*
at_read_mask, 86
at_read_mask(), *67–73*, *88, 89*, *134–136*
at_read_npy, 88
at_read_npy(), *67–73*, *87*, *134–136*
at_read_qupath, 89

at_read_qupath(), *55*, *63, 64*, *85*, *91*, *93*, *103*, *113, 114*, *132*, *137, 138*
at_read_rois_csv, 90
at_read_rois_csv(), *55*, *63, 64*, *85*, *90*, *93*, *103*, *113, 114*, *132*, *137, 138*
at_read_stats, 91
at_read_stats(), *13, 14*, *38*, *52*, *125*
at_relink_source, 92
at_relink_source(), *55*, *63, 64*, *85*, *90, 91*, *103*, *113, 114*, *122*, *132*, *137, 138*
at_remove_layer, 93
at_remove_layer(), *6*, *42*, *63*, *84*, *94*, *112*, *120*, *131*
at_remove_roi, 94
at_remove_roi(), *6*, *42*, *63*, *84*, *93*, *112*, *120*, *131*
at_resume, 94
at_resume(), *39*, *43*, *51*, *64, 65*, *76*, *83*, *114–117*
at_roi_area, 95
at_roi_area(), *96–98*, *101–103*, *105–107*
at_roi_bbox, 96
at_roi_bbox(), *95*, *97, 98*, *101–103*, *105–107*
at_roi_buffer, 96
at_roi_buffer(), *25, 26*, *50*, *99, 100*, *104*, *107–110*, *112*, *117*, *130*
at_roi_centroid, 97
at_roi_centroid(), *95, 96*, *98*, *101–103*, *105–107*
at_roi_circle, 98
at_roi_circle(), *95–97*, *101–103*, *105–107*
at_roi_contains, 99
at_roi_contains(), *25, 26*, *50*, *97*, *100*, *104*, *107–110*, *112*, *117*, *130*
at_roi_difference *(*at_roi_setops*)*, 109
at_roi_difference(), *108*
at_roi_distance, 99
at_roi_distance(), *25, 26*, *50*, *97*, *99*, *104*, *107–110*, *112*, *117*, *130*
at_roi_ellipse, 100
at_roi_ellipse(), *95–98*, *102, 103*, *105–107*
at_roi_freehand, 101
at_roi_freehand(), *95–98*, *101*, *103*, *105–107*
at_roi_from_geojson, 102
at_roi_from_geojson(), *55*, *63, 64*, *85*, *90, 91*, *93*, *95–98*, *101–103*, *105–107*, *113, 114*, *132*, *137, 138*
at_roi_from_sf, 103
at_roi_from_sf(), *95–98*, *101–103*, *105–107*
at_roi_intersect *(*at_roi_setops*)*, 109
at_roi_overlaps, 104
at_roi_overlaps(), *25, 26*, *50*, *97*, *99, 100*, *107–110*, *112*, *117*, *130*
at_roi_point, 104
at_roi_point(), *95–98*, *101–103*, *106, 107*
at_roi_polygon, 105
at_roi_polygon(), *95–98*, *101–103*, *105*, *107*
at_roi_rect, 106
at_roi_rect(), *95–98*, *101–103*, *105, 106*
at_roi_rescale, 107
at_roi_rescale(), *25, 26*, *50*, *97*, *99, 100*, *104*, *108–110*, *112*, *117*, *129, 130*
at_roi_ring, 108
at_roi_ring(), *25, 26*, *50*, *97*, *99, 100*, *104*, *107*, *109, 110*, *112*, *117*, *130*
at_roi_setops, *25, 26*, *50*, *97*, *99, 100*, *104*, *107, 108*, 109, *110*, *112*, *117*, *130*
at_roi_simplify, 110
at_roi_simplify(), *25, 26*, *50*, *97*, *99, 100*, *104*, *107–109*, *112*, *117*, *130*
at_roi_symdiff *(*at_roi_setops*)*, 109
at_roi_union *(*at_roi_setops*)*, 109
at_rois, 111
at_rois(), *6*, *42*, *62, 63*, *84*, *93, 94*, *120*, *131*
at_rois_overlap, 112
at_rois_overlap(), *25, 26*, *50*, *97*, *99, 100*, *104*, *107–110*, *117*, *130*
at_save_project, 113
at_save_project(), *55*, *63, 64*, *85*, *90, 91*, *93*, *103*, *114*, *132*, *137, 138*
at_save_session, 114
at_save_session(), *55*, *63, 64*, *85*, *90, 91*, *93*, *95*, *103*, *113*, *132*, *137, 138*
at_session, 115
at_session(), *9*, *39*, *43*, *51*, *65*, *76*, *83*, *95*, *116, 117*, *139*
at_session_status, 116
at_session_status(), *39*, *43*, *51*, *65*, *76*, *83*, *95*, *115*, *117*
at_set_status, 116
at_set_status(), *39*, *43*, *51*, *65*, *76*, *83*, *95*,

115, *116*
at_snap, 117
at_snap(), *25, 26*, *50*, *97*, *99, 100*, *104*, *107–110*, *112*, *130*
at_stage_qupflowr, 118
at_stage_qupflowr(), *8*, *28*, *45*, *53, 54*, *56, 57*
at_style, 119
at_style(), *59–62*, *140*
at_summary, 120
at_summary(), *6*, *42*, *63*, *84*, *93, 94*, *112*, *131*, *140*
at_summary_table, 121
at_summary_table(), *16, 17*, *44*
at_tile, 121
at_tile(), *10–15*, *27*, *38*, *40, 41*, *52, 53*, *58, 59*, *74, 75*, *77*, *86*, *125*, *131*, *139, 140*
at_tile_source, 123
at_tile_source(), *7*, *10*, *18–24*, *139, 140*
at_tivita_profile, 124
at_tivita_profile(), *10–14*, *27*, *38*, *52, 53*, *86*, *92*, *122*
at_training_check, 125
at_training_check(), *127–129*
at_training_export, 126
at_training_export(), *125, 126*, *129*
at_training_import, 128
at_training_import(), *126*, *128*
at_transform, 129
at_transform(), *25, 26*, *50*, *97*, *99, 100*, *104*, *107–110*, *112*, *117*
at_validate, 130
at_validate(), *6*, *17*, *25*, *42*, *63*, *84*, *93, 94*, *112*, *120*
at_wavelengths, 131
at_wavelengths(), *15*, *27*, *40, 41*, *52, 53*, *58, 59*, *74, 75*, *77*, *86*, *122*, *140*
at_write_geojson, 132
at_write_geojson(), *55*, *63, 64*, *85*, *90, 91*, *93*, *103*, *113, 114*, *137, 138*
at_write_mask, 133
at_write_mask(), *67–73*, *87*, *89*, *135, 136*
at_write_masks, 134
at_write_masks(), *44*, *67–73*, *87*, *89*, *134*, *136*
at_write_npy, 135
at_write_npy(), *67–73*, *87*, *89*, *134, 135*
at_write_qupath, 136
at_write_qupath(), *55*, *63, 64*, *85*, *90, 91*, *93*, *103*, *113, 114*, *132*, *138*
at_write_rois_csv, 138
at_write_rois_csv(), *55*, *63, 64*, *85*, *90, 91*, *93*, *103*, *113, 114*, *132*, *137*
atCanvasOutput, 139
atCanvasOutput(), *7*, *10*, *19–24*, *123*, *140*

cli::cli_inform(), *50*

dim.annot_image *(*at_dims*)*, 39

ggplot2::ggplot, *77–82*

hyperannotatR-classes, 139

renderAtCanvas, 140
renderAtCanvas(), *7*, *10*, *19–24*, *123*, *139*

shiny::runApp(), *7, 8*

tempdir(), *115*
tibble::tibble, *11*, *13*, *15–17*, *24, 25*, *44*, *47, 48*, *63*, *65*, *68*, *71*, *73*, *111, 112*, *116*, *121*, *135*