

Pystern Blot: an open-source software for reproducible Western blot figure preparation, management, and integrity documentation

Etienne Boulter

`etienne.boulter@inserm.fr`

Inserm <https://orcid.org/0000-0001-6597-9086>

Chloé C. Féral

Inserm <https://orcid.org/0000-0001-9756-2082>

Method Article

Keywords: Western blot, Image integrity, Open-source software

Posted Date: October 14th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11306385/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Abstract

The transition from photographic film to digital acquisition transformed the Western blot from a self-labeling, archivable physical artifact into a digital data file, gaining dynamic range and quantitative capability at the cost of its self-documenting structure. Photographic film was self-documenting: researchers annotated antibody identity, lane contents, and exposure time directly on the film. Digital acquisition replaced this permanent, physical and recoverable record with image files stripped of biological context, losing archival interpretability. Here we present Pystern Blot, an open-source desktop application that addresses both the data-management and the figure-preparation problems. It restores these properties by treating Western blot images as informatic objects with a documented source identity and processing history, while persistently associating biological context with each image, and by automating the provenance documentation that would otherwise be reconstructed retrospectively at submission. It provides a 16-bit image-processing and semi-automated panel-assembly workflow in which alignment, molecular-weight annotation and reusable lane labels are properties of the project rather than layout work repeated by hand at each assembly and revision. It records each operation in a timestamped audit log whose entries are hash-chained from the point of logging onward, and stores figures and images in structured projects for long-term record keeping. Pystern Blot generates machine-readable JSON and human-readable HTML integrity reports, and it can export both the untouched imported source file and an annotated source-context TIFF for image-integrity review. It supports chemiluminescent and selected multichannel near-infrared fluorescence workflows, and exchanges projects through a portable .pbarchive format whose contents are hash-checked on import. Altogether, Pystern Blot restores key self-documenting properties of physical film to digital Western blot workflows and makes traceability a routine consequence of figure preparation, while reducing the time spent assembling and revising the figure itself.

Introduction

Western blotting has remained one of the most widely performed protein detection assays in cell and molecular biology for over four decades; it is named in the titles, abstracts or keywords of more than 400,000 PubMed-listed publications (Moritz, 2020; Burnette, 1981; Towbin et al., 1979). For most of that history, detection was performed by exposing the membrane to photographic or X-ray film — a physical analogue record that, for all its quantitative limitations, had one important practical virtue: it was self-labeling. A researcher could write directly on the film with a marker pen — antibody identity, experiment number, lane contents, exposure time — creating an annotation permanently and unambiguously associated with the image. Films could be sorted, filed in labeled folders and retrieved years later.

The progressive adoption of digital imaging systems over the past two decades has transformed the blot from a photographic artifact into a digital data file. Modern instruments routinely digitize signal at 12- to 16-bit depth, encoding up to 65,536 discrete intensity levels per pixel — far exceeding film — and in principle enabling more rigorous analysis of band intensities and signal linearity. Even high-dynamic-range systems, which resolve several orders of magnitude in signal intensity through deep-well sensors

or multi-exposure fusion, typically encode their output as 16-bit unsigned integers, making this the *de facto* standard for native Western blot acquisition.

Digital acquisition removed the very property that made the film archives work: its physicality. While film exposures were tangible, self-annotating objects, digital files are untethered from their experiments. Annotation, once as simple as writing on film, has become more important and more difficult: instrument metadata fields are inconsistently populated, frequently stripped during export, and not standardized across platforms. Digital files can also fail fatally: silent bit-level corruption can render them inaccessible years later, unlike film, which degrades interpretably. Moreover, digital archives depend on fragile infrastructure (storage media, filesystems, and backup regimes) often beyond researchers' control, making entire experimental records vulnerable to single failures. The cumulative effect is a steady erosion of interpretability, loss of scientific data and institutional memory. This attrition is measurable across research data generally, not only for blots: for 516 published biological studies, the odds that the underlying data were still extant fell by roughly 17% for every year elapsed since publication (Vines et al., 2014). Policy can counter it: in a comparison of journal archiving policies, mandates requiring both data deposition and a data availability statement raised the odds of finding the data online almost 1000-fold relative to no policy, whereas merely recommending archiving had little effect (Vines et al., 2013).

Beyond data management, digital acquisition made image manipulation easier than photographic film, which had inherent physical constraints. A landmark visual screen of 20,621 published papers containing Western blots found that 3.8% contained at least one figure with inappropriate image duplication, with at least half of those exhibiting features suggestive of deliberate manipulation (Bik et al., 2016). Automated screening has since reported suspected manipulations at a similar rate, in 5.7–8.6% of the papers screened, with a striking concentration on gel electrophoresis images (83%) (Bucci, 2018). The response has been detection after publication; a complementary approach is to preserve the record needed to assess a figure before submission.

Concerns about blot image presentation raised two decades ago (Rossner and Yamada, 2004) — inappropriate contrast adjustment, bands added or removed, lanes spliced together and duplicated panels — remain relevant today, and narrowly cropped blots lacking molecular-weight markers remain the norm in published figures (Kroon et al., 2022). Journal guidelines derived from that work also require that nonlinear adjustments, such as gamma changes, be disclosed in the figure legend. In response, journals have converged on formal image-integrity requirements, though their stringency varies. For example, Nature Portfolio requires unprocessed original images of gels and blots to be submitted with the final accepted version and published alongside the paper, as well as documentation of all image acquisition and processing software in the Methods. EMBO Press screens all image-based data before acceptance and may decline publication where source data are systematically absent, citing institutional data retention limits (often limited to 10 years or unspecified). Cell Press requires that original, unprocessed data be available on request of editors or reviewers.

In practice, however, the standard figure preparation workflow — exporting from instrument software, adjusting in Fiji/ImageJ (Schindelin et al., 2012) or a general-purpose image editor, and manually exporting — often alters images without documenting the operations applied. Yet this is exactly the information those policies demand — the original image, the operations applied to it, and the relationship between the two. Researchers must then reconstruct this processing history from memory at submission, an unverifiable process that shifts the entire compliance burden from tools to individual researchers. That burden scales with the number of figures submitted rather than with the number of blots analyzed: a laboratory need not run Western blots at high throughput for it to apply, because every blot panel sent to such a journal carries the same documentation requirement.

We argue that the root cause of these structural deficiencies is a category error: the Western blot image is still treated, conceptually and operationally, as a photograph rather than as a data file whose identity, processing history, and relationship to upstream experimental data should be tracked with the same rigor as any other scientific dataset. This reframing has immediate practical consequences: an informatic object has a defined identity (a content hash of the imported file), a processing history (an audit log of operations applied), and associated metadata (instrument settings, acquisition parameters, biological annotation, timestamp).

Existing tools address parts of this workflow, including image processing, figure assembly, quantitative gel analysis and repository-scale image management, but these functions remain divided across systems. In practice, assembling a single multi-condition blot figure means cropping in one application, transferring the result to another, positioning and aligning each panel, and typing marker and condition labels for every panel in turn, work that consumes research time without adding scientific content to the figure. Much of it is then repeated whenever a crop, an exposure or a lane assignment changes.

The processing record is difficult to reconstruct at submission because no single tool holds an account of what was done. Among the tools considered here, we did not identify one that combines blot-specific panel preparation with source retention and a portable, structured processing record intended for external review. Pystern Blot was developed to address this specific gap by coupling panel assembly, source retention and machine-readable provenance within a single workflow, reducing the time and labor spent on figure layout while producing the processing record as a by-product of that work.

Results

Positioning relative to existing software

Existing software occupies distinct points along the path from image processing to figure preparation and data management (Table 1). Pystern Blot was written to close a specific gap in that landscape: no reviewed tool couples blot-specific figure assembly to a portable, machine-readable record of what was done to the source images. Fiji/ImageJ (Schindelin et al., 2012) provides extensive manipulation capabilities, while the ImageJ Gels plugin remains a widely used open-source route to blot densitometry;

neither common ImageJ workflows nor the Gels plugin automatically provide a structured record of the analysis steps. Fiji's macro recorder can capture commands, but only when used and as a command history rather than a content-linked lineage. QuickFigures (Mazo, 2021), designed around fluorescence microscopy figures, streamlines panel layout, alignment and export from ImageJ; tools for gel and Western blot figures were added after publication. FigureJ (Mutterer and Zinck, 2013) is a closer antecedent to the present approach: it treats panels as views into datasets, preserves access to source data and exports a textual processing description. That description remains a free-text note rather than a schema-versioned record with content hashes, source re-verification or explicit disclosure flags, and its panel model is oriented towards microscopy rather than lanes, molecular-weight ladders and multiple exposures.

At repository scale, OMERO (Allan et al., 2012) provides server-based management of biological image data built on the OME data model (Goldberg et al., 2005), and has been extended with provenance-aware metadata capture (Russell and Sorger, 2017). OMERO.figure composes panels from stored images and preserves links to their sources (The Open Microscopy Environment, 2026), but the workflow is designed for microscopy and records a saved figure layout rather than the per-source operation record considered here. IOCBIO Gel (Kütt et al., 2023) takes a different, blot-specific route: it retains the original image and analysis operations so that quantitative measurements can be re-derived and corrected, but it does not provide panel assembly or a verification-oriented artifact for editors or reviewers. GelGenie (Aquilina et al., 2025) applies deep-learning segmentation to band identification and quantification in gel images; it is likewise analysis-oriented and is not directed at panel assembly or at a verification-oriented artifact for editors or reviewers. GelBox (Gulbulak et al., 2024) combines image adjustment, background correction and band fitting, and stores the raw data, metadata, image adjustments and band-level analyses together in a single file to improve traceability; it is likewise directed at band quantification rather than at panel assembly or a verification-oriented artifact. Instrument-bundled proprietary applications, including Bio-Rad Image Lab, LI-COR Image Studio and Empiria Studio, and Cytiva ImageQuant, offer hardware integration and may guide users towards native 16-bit workflows. Their processing records and exchange options vary by product and version, and the evidence considered here did not establish portable, structured documentation that travels independently of the vendor.

What ultimately separates these tools is who the processing record is written for. Where a record exists at all, it is written for the person doing the work, for the analyst re-deriving a measurement, or for the system storing the images (Table 1). Pystern Blot instead produces an artifact intended to travel with the project to a third party — an editor, reviewer, reader or collaborator — while retaining source identity and project state. Among the tools reviewed here, we did not identify another that combines blot-specific figure assembly, content hashing, live source re-verification, schema-versioned provenance and explicit disclosure flags in a portable project workflow.

Overview, workflow and key features

The primary output of a Pystern Blot session is a publication- and presentation-ready annotated panel figure from which data management and integrity documentation are produced automatically as a structural by-product. The application is free and open source under GPLv3 and is available at <https://github.com/TheBrickalista/PysternBlot>. At its core, Pystern Blot treats Western blot images as informatic objects with three persistent properties: a content-addressed identity (SHA-256 hash (National Institute of Standards and Technology, 2015) of the imported file), a processing history (timestamped, hash-chained operation log), and biological context (antibody, lane, and exposure metadata). This design ensures that source identity and processing steps remain intrinsically linked to the final figure. Pystern Blot is implemented as a cross-platform desktop application (Python ≥ 3.10 , PySide6/Qt 6) and runs on macOS, Windows, and Linux.

The interface is organized into seven tabs: Home (start screen and quick actions), Library (projects in the workspace), Figure (panel composition and all publication and integrity exports), Original Image (per-blot adjustment, cropping, molecular-weight calibration and annotated exports), Legend (lane and group labels), Preferences (ladder presets and saved entries reused across projects) and About (license, credits and citation). A per-tab user guide is provided in the online documentation.

A typical workflow comprises six steps: (1) import a 16-bit TIFF from the instrument or an 8-bit legacy image, with JPEG rejected at import; (2) enter antibody name and detection modality metadata; (3) apply levels, rotation, and crop adjustments interactively, aided by a live pixel-intensity histogram with draggable gate lines that reflect the levels window in real time; (4) calibrate molecular-weight markers by clicking on band positions; (5) annotate the figure by defining lane labels and group headings (the panel legend); and (6) export the annotated panel figure in the desired format, alongside an integrity report and, when needed, either the untouched source file or an annotated source-context TIFF. The metadata entered at steps 2 and 5 — antibody identity, detection modality, lane identification and per-channel annotation for near-infrared (NIR) blots — constitutes the digital equivalent of writing on the film: biological context persistently and unambiguously associated with the image within the project. These steps are carried out in a single application against one project model, so no intermediate cropped copy is exported and reassembled elsewhere.

Several elements of panel layout that are otherwise performed by hand are instead encoded as project properties, eliminating repetitive manual work. For example, the shared crop template makes vertical alignment across blots a geometric consequence of panel design rather than a manual adjustment; the molecular-weight annotation is drawn from the ladder calibration entered by clicking band positions; and lane and group labels are positioned against the crop box by the legend renderer, ensuring consistent placement.

Annotation reuse further reduces effort: antibody names, lane and group labels, and ladder presets are saved at application level and offered as selectable entries in any subsequent project. Because experimental conditions typically recur across the figures of a single study, and frequently across successive studies, this replaces repeated typing with selection from a list, a change that directly

reduces the time and hands-on layout work required to reach a finished panel, none of which contributes scientific content to the figure.

Automatic traceability logging and integrity reporting

Pystern Blot automatically generates a structured audit trail for every action that changes a project in a way that affects the final figure, including imports, level adjustments, crops and exports. Each log entry records when, what and how changes were made, with hash-chaining to ensure internal consistency. No decision by the user to log anything is required. Transient interface state is not recorded: the log describes the project as it is saved. In current versions, new entries are linked by `prev_hash` and `entry_hash` fields, so the log is hash-chained from the first chained entry onward. Verification tests the internal consistency of retained log entries by detecting modifications that invalidate links between them. Because the chain is neither externally signed nor independently anchored, it cannot by itself establish that the retained history is complete and does not prevent deliberate rewriting followed by recomputation of downstream hashes; nor does it authenticate acquisition history prior to import.

On demand, Pystern Blot generates a dual-format integrity report (JSON plus a human-readable HTML file; Fig. 2) to facilitate journal review and long-term reproducibility. The JSON file includes: i) source verification: SHA-256 hashes of original files with live re-verification against stored assets, including bit depth, pixel dimensions, crop coordinates, rotation, and levels parameters; ii) processing history: full operation log (or a summary in the base report) with hash-chained entries for internal consistency; iii) disclosure flags: explicit warnings for non-default gamma adjustments — per blot and per NIR channel — or full-scale clipping (assessed via 3×3 erosion to distinguish isolated vs. coherent saturated regions) and iv) context preservation: annotated source-context TIFFs showing crop region overlaid on the full membrane.

The report is written as a JSON file conforming to the schema `pysternblot.integrity_report.v2`, or, in the detailed variant that additionally embeds the full chronological operation log, `pysternblot.detailed_integrity_report.v2`. It contains the Pystern Blot version, system platform, project metadata, the SHA-256 of the current project.json, the status of operation-log chain verification, and for each blot: source image SHA-256 with live re-verification against the stored file, bit depth, sample format (integer or floating-point) and pixel dimensions, crop coordinates, rotation and levels parameters, and ladder calibration. Any non-default gamma adjustment is explicitly flagged in the report — per blot and per NIR channel — and surfaced as a live amber badge in the interface, with a note that gamma adjustment must be disclosed under journal image-integrity guidelines. Digital full-scale clipping in the stored source image is assessed separately. On import, and before any display transform is applied, the number of pixels at full scale for the source bit depth is counted, together with the number that survive a 3×3 erosion of the full-scale mask. The erosion distinguishes isolated or one-pixel-wide full-scale features from spatially coherent full-scale regions: a pixel surviving it has eight full-scale neighbors and therefore lies within a region with spatial extent, whereas isolated pixels and one-pixel-wide structures

do not survive. Isolated full-scale pixels are reported as a note; spatially coherent full-scale regions raise a warning, both for the whole image and, computed separately at report time, for the crop region that reaches the panel. Because full-scale clipping represents unrecoverable intensity information, the report records its presence in the source and, separately, whether it occurs within the region selected for the final panel. The assessment is performed on the imported source before display transforms and therefore does not assess clipping introduced later by levels, gamma, or other display settings. Assets imported by earlier versions carry no assessment and are reported as not assessed rather than as unclipped. Floating-point sources, such as LI-COR Image Studio exports, have no fixed detector ceiling, so clipping cannot be assessed from the file; they are reported as not assessable, a state distinct from both unclipped and not assessed. Images with full-scale clipping are flagged rather than rejected, since an exposure containing full-scale clipping may legitimately be retained as the discarded member of an exposure series — itself a decision the operation log records. The HTML report presents this information in tabular form accessible to journal editors and reviewers without specialist software. Together with byte-for-byte source export and annotated context export, this output can facilitate journal image-integrity review, but it does not by itself establish that an imported file was instrument-original before entry into the workflow.

Crop documentation and exposure management

Lane selection and cropping are among the most problematic aspects of Western blot figure preparation and raise two distinct integrity concerns. First, cropping can conceal membrane context, making it impossible for reviewers to assess whether relevant bands were omitted. Second, without molecular-weight marker annotations, a cropped blot offers no independent basis for confirming that the displayed band corresponds to the protein of interest (Kroon et al., 2022).

Pystern Blot addresses both concerns through complementary mechanisms. To preserve the context, it generates the source-context TIFF: every blot can be exported as a full source image rendered with the current display settings and the selected crop region overlaid as a dashed rectangle (Fig. 2c), providing reviewers with complete membrane visual context. This file is not itself an unprocessed original; rather, it is an annotated derivative that clarifies the relationship between the published panel and the stored source asset. For absolute verification, the software can also separately re-export the imported source file byte for byte, with SHA-256 verification on export. To ensure interpretability, molecular-weight markers are calibrated by the user, establishing a calibrated size scale recorded in the integrity report and displayed in the final panel. This allows independent assessment of whether the annotated band position is consistent with the expected molecular weight of the target protein.

Pystern Blot further addresses cropping transparency through a shared crop template model: all blots in a panel share a single crop size (Panel.crop_template.w, h), while each retains its own position (Blot.crop.x, y). Resizing the interactive crop rectangle updates the shared template and propagates

immediately to all blots. This enforces perfect vertical alignment and eliminates manual adjustments. All crop operations are logged at pixel precision.

A complementary legend export zone provides an annotated intermediate view compatible with presentation and laboratory record-keeping: a region containing the full crop box, panel legend, and calibrated molecular-weight markers. This export always carries a complete size scale (with channel-specific bands omitted for NIR blots) and maintains lane caption alignment regardless of zone size. Zone geometry and marker settings are stored per blot, with all changes recorded in the operation log (`legend_zone_changed`, `legend_zone_markers_changed`, `legend_zone_side_changed`). The result sits between the cropped panel and the full source-context TIFF: an annotated view of the region shown in the figure, with its immediate membrane context retained.

For exposure management, Pystern Blot treats each exposure of a membrane as a separate Blot entry. The `Blot.included_in_final` flag allows users to exclude blots from the final figure while retaining full editability and visibility in the blot selector, replacing the risky practice of deleting alternative exposures. To improve usability, each blot may be given a cosmetic display name without breaking the link to the original file. This label is purely presentational: the original filename is preserved unchanged in the integrity report and tooltip, and all renaming is recorded as an operation in the log.

16-bit image processing pipeline

Pystern Blot's pipeline preserves the native bit depth of integer sources throughout processing to avoid unnecessary quantization, a critical requirement for accurate Western blot analysis. Input handling is strict and transparent: all sources enter through a single loader, `load_image_as_uint16()`; native 16-bit grayscale TIFF files are read without conversion; floating-point TIFF exports (LI-COR Image Studio) are mapped into the 16-bit display pipeline by a single recorded scale factor (`Methods`), leaving the stored original untouched; legacy 8-bit sources (grayscale, palette, RGB, RGBA, and LA modes) are accepted but flagged with warnings at import, in the integrity report, and at export; JPEG files are hard-rejected at all import entry points with an explicit scientific explanation because lossy compression alters pixel values and typical scanner JPEG export may also apply undocumented display transforms.

All subsequent display operations — levels mapping, gamma correction, inversion, rotation, horizontal and vertical flipping, and cropping — never silently reduce bit depth. The pipeline does not implicitly down-convert native 16-bit inputs to 8-bit, and operations requiring numerical transformation use higher-precision intermediates before conversion back to `uint16`, while accepted 8-bit sources retain their original 0-255 value range without upscaling.

Rotation is implemented with bicubic resampling in Pillow's `I` mode and clipped back to `uint16`, so it should be regarded as a documented display transform, not a quantification step. The cropped panel image is cached to disk as a 16-bit preview TIFF, which the Figure tab renders, whereas the Original Image tab recomputes its full-membrane view in memory. Both are derived from the stored source,

which is written once at import and only read thereafter, including by the integrity module for re-hashing and bit-depth verification.

The quantitative significance of retaining native bit depth lies in avoiding unnecessary quantization during processing. A 16-bit image can encode 65,536 discrete gray levels (vs. 256 for an 8-bit image) preserving fine intensity differences through intermediate operations. While individual operations (levels, gamma, rotation) can alter pixel values, all changes are recorded explicitly in the Pystern Blot operation log. Export is likewise explicit, logged and optimized for its use case. Presentation-quality PNG and PDF exports render the composed panel at screen-appropriate bit depth suitable for slide decks and reports; production-quality SVG export preserves all annotation text as fully editable vector objects for downstream assembly in Illustrator, Affinity Designer, or equivalent applications — ensuring a clean and documented handoff.

NIR multichannel fluorescence support

Western blot membranes probed with NIR-conjugated secondary antibodies produce one grayscale TIFF file per spectral channel. Pystern Blot supports the Cytiva Typhoon workflow through the BlotChannel model and NirImportDialog GUI (Fig. 4). LI-COR Odyssey exports import through the same multichannel path, both the floating-point TIFFs exported by Image Studio from an Odyssey CLx and the 16-bit integer TIFFs of older Odyssey systems; channel (700 or 800), instrument model, serial number, software version, acquisition time and pixel size are read from the TIFF tags, and the channel is stored as the instrument's channel name rather than as a wavelength. Cytiva Typhoon TIFF metadata is parsed via Tag 270: each Typhoon TIFF embeds scan metadata in TIFF Tag 270, which Pystern Blot parses to extract wavelength (nm), filter name, scan number, and pixel size. Where a sibling .inf sidecar file accompanies the Typhoon scan, it is parsed on import and its acquisition fields — scale type (linear or square-root), scan mode, PMT and laser settings, and applied corrections — are stored per asset and surfaced in the integrity report, with the scale type shown prominently because it governs the quantitative interpretation of pixel values. The import dialog displays this metadata for user verification and allows import of one or two channel files. In the Figure tab, NIR blots render as stacked per-channel grayscale rows with per-band channel visibility controlled by wavelength assignment in ladder presets, enabling channel-specific marker display. Because the Odyssey CLx excites its 700 and 800 channels at 685 and 785 nm, LI-COR channels follow the corresponding assignments; other channels display all assigned bands.

Portable archival format

For project transfer and long-term archival, Pystern Blot implements a .pbarchive format: a ZIP file containing a manifest, all project JSON files, and all referenced source image assets. In format version 2, the manifest records per-project SHA-256 digests, and import performs a two-pass validation: assets are SHA-256-verified against their path component and project JSON bytes are SHA-256-verified against the manifest before any file is written to the receiving workspace. Imported projects receive an

imported_from_archive operation log entry, while legacy format version 1 archives remain importable without project-level verification. This mechanism enables self-consistent, hash-checked transfer of a project and its source data between workstations or institutions, though it is not a cryptographically signed archive format.

Discussion

Conceptual Framework

Pystern Blot is organized around the view that the Western blot image is an informatic object with documented source identity and processing history, and that routine documentation should be produced by the tool rather than reconstructed retrospectively by the researcher. Its design reflects four principles: fidelity by default (bit depth is not silently reduced during processing); transparency by design (operations are logged automatically); traceability as a primary output (integrity reports are a core deliverable of each session); and accessibility by design (a point-and-click GUI requires no scripting, while the GPLv3 source enables inspection and extension). Its distinguishing contribution is that routine figure preparation can simultaneously produce a structured processing record aligned with common journal image-integrity requests.

Practical Impact

The practical significance during figure preparation follows from the same design. Pystern Blot removes the handoff between cropping and assembly and converts much of the remaining layout work into project state: alignment follows from the shared crop template, marker annotation from the ladder calibration, and label placement from the legend geometry, while saved entries and ladder presets make annotation typed once and available in every later figure. Revision benefits even more: a change of crop, exposure or lane grouping re-renders the panel with its annotation intact rather than requiring the downstream assembly to be redone. The practical consequence is a reduction in the time spent preparing figures, and in the share of a project devoted to presentation rather than to the experiment or the conceptualization.

The practical significance at manuscript submission is most apparent by comparison. Under a standard ImageJ-based workflow, a researcher submitting to a journal with image-integrity requirements must locate source image files, reconstruct the specific adjustments applied, prepare a written description of those adjustments, and provide source material on request. Under a Pystern Blot workflow, much of that information is already packaged: the integrity report provides structured processing documentation; the annotated source-context TIFF documents the relationship between the published panel and the full source membrane; and the untouched source file can be re-exported byte for byte with SHA-256 verification on export. The hash-chained operation log provides an internal consistency check on the retained project history, and the .pbarchive format adds manifest-based consistency checks for shared projects. A further distinction from proprietary instrument software is portability: the integrity record is emitted as plain JSON and HTML that travel with the project rather than remaining confined to a vendor

ecosystem. By narrowing these gaps, Pystern Blot addresses the documentation side of a problem whose scale is measurable in the published literature (Bik et al., 2016).

These features are designed to reduce the amount of retrospective reconstruction required for image-integrity documentation. They do not, however, guarantee compliance with every journal policy, authenticate the provenance of an image before import, or replace repository-scale infrastructure such as OMERO (Allan et al., 2012). The processing record similarly ends at export. Pystern Blot logs each export it generates, but subsequent edits made in external figure-composition software such as Illustrator or Affinity Designer fall outside its operation log, and exported figure files are not themselves hashed in the current version.

The film analogy is worth making precise, because it clarifies which problem Pystern Blot is solving. A well-labeled film archive retained both image and context in the same physical object. Many digital laboratories retain folders of files in which that contextual link has been weakened or lost. Pystern Blot is designed to restore that interpretability in digital-native form: the SHA-256 asset store provides filename-independent identity for the imported file; per-blot annotation records the biological context that was once written on the film; the `included_in_final` flag and operation log document exposure-selection decisions that were previously unrecorded; and the `.pbarchive` format packages project state and source assets for transfer. The content hash also makes later corruption of a stored source file detectable when integrity reports are generated or archives are imported. Pystern Blot is not a backup system and does not protect against loss. Nor does it independently authenticate acquisition history before import: if a file has already been altered before entering the workflow, Pystern Blot will document that altered file consistently from the point of import onward. Within that trust boundary, however, it restores much of the archival interpretability that annotated film once provided.

FAIR Alignment

In a broader sense, Pystern Blot aligns with several FAIR data principles (Wilkinson et al., 2016) for Western blot-related project records: content-addressed identity improves findability within a workspace; the JSON-based `.pbarchive` and integrity reports improve accessibility without proprietary dependencies; and the operation log plus integrity report improve reusability by preserving provenance. Interoperability remains the least developed dimension. The `.pbarchive` schema is documented and human-readable, but it is a project-specific schema rather than a community standard, and export to shared standards such as OME-TIFF (Goldberg et al., 2005; Linkert et al., 2010) or direct BioImage Archive (Hartley et al., 2022) submission is not supported. Pystern Blot data are therefore inspectable and portable on their own terms, but not yet readily integrated into repository-scale community infrastructure.

The software itself addresses several FAIR4RS elements (Barker et al., 2022): it is findable through a Zenodo DOI (10.5281/zenodo.20185279) and PyPI; accessible as source code and prebuilt binaries under the GPL-3.0-only license; reusable through versioned releases, a machine-readable CITATION.cff file, public documentation, and continuous-integration workflows; and partly interoperable at the artifact level through standard TIFF input and plain JSON outputs. These properties improve software

accessibility and reuse, although they should not be conflated with full interoperability of the project data model.

Limitations and Scope

Nevertheless, Pystern Blot has limitations, and it is worth stating explicitly what it does not perform. It is not an image-analysis or densitometry tool: it performs no quantitative normalization, band detection, or intensity measurement. It does not modify source files after import, but it also does not certify that an imported file is instrument-original. The annotated context TIFF and legend-zone export are provenance aids, not unprocessed source files. The audit log supports internal consistency verification of retained entries rather than external authentication, and the archive format is hash-checked rather than externally signed. Instrument coverage is likewise bounded: Pystern Blot parses acquisition metadata from Cytiva Typhoon and LI-COR Odyssey files only, and its LI-COR support has been tested on Odyssey CLx exports from Image Studio 3.1.4 and on 16-bit exports from one older Odyssey system; other Odyssey models and software versions are untested. For floating-point exports, full-scale clipping cannot be assessed from the file. For NIR blots, levels and protein-label changes are logged with their values but without identifying the channel, crop entries in the log record the blot-level rather than the per-channel crop, and the per-blot crop, levels and protein-label fields of the integrity report do not yet reflect per-channel settings. Finally, the current evidence base is primarily software testing and inspection of exported artifacts rather than benchmark studies of user performance or journal-review outcomes. The workflow claims made here are correspondingly mechanistic: they describe manual operations that the implementation consolidates or removes, and they should not be read as measured time savings, which would require a controlled comparison against conventional multi-application workflows that we have not performed.

None of these boundaries negates the problem the tool was built to solve. Pystern Blot in its current form provides a practical and unusually explicit workflow for blot-specific figure preparation, source retention, and provenance documentation. A Western blot panel prepared with it can arrive at submission accompanied by a structured processing record, but the strength of that record remains bounded by the provenance of the imported file and by the absence of external cryptographic signing.

Methods

Software architecture

Pystern Blot is implemented in Python ≥ 3.10 with PySide6 $\geq 6.11.2$ (Qt 6 bindings), Pydantic ≥ 2.6 (data model validation and serialization), NumPy ≥ 1.24 (Harris et al., 2020), Pillow $\geq 12.3.0$, tifffile $\geq 2023.1.23$ (floating-point and tiled TIFF reading), and certifi $\geq 2024.7.4$ (CA certificates for HTTPS requests). The application is organized into eight core library modules: `models.py` (Pydantic v2 data model hierarchy: Project $\rightarrow$ Panel $\rightarrow$ Blot $\rightarrow$ BlotChannel); `storage.py` (workspace management, SHA-256 asset store, crop preview cache, NIR import, archive export/import); `image_utils.py` (bit-depth handling,

image transforms, and multichannel TIFF utilities); `render.py` (Qt `QGraphicsScene` builders for the Figure and Original Image views); `integrity.py` (SHA-256 traceability and JSON/HTML report generation); `logchain.py` (hash-chained operation-log construction and verification); `update_check.py` (opt-in comparison of the installed version against the GitHub Releases API, disabled by default and gated behind an explicit consent prompt); and `selfcheck.py` (windowless diagnostic of the version lookup, CA bundle and HTTPS request, invoked by `--selfcheck`). The user interface in `pysternblot/ui/` is assembled via Python mixins with no Qt multiple inheritance; operation logging is initiated in `project_io_mixin.py` and chained via `logchain.py`.

16-bit image pipeline

All import paths read source files through `load_tiff_source()`, which opens them with Pillow and, only for files Pillow cannot open (such as tiled, multi-resolution floating-point TIFFs), falls back to `tifffile`, reading exclusively the full-resolution page; unsupported modes and sample formats raise a `ValueError`. `load_image_as_uint16()` returns native 16-bit sources unchanged. Floating-point sources are mapped to `uint16` by `bridge_float_to_uint16()` with a single scale factor (65535 divided by the finite source maximum), with no offset and no percentile clipping, so intensities remain proportional to the source; NaN and infinite values are mapped to zero and counted, negative values are clipped to zero and counted, and the method, source minimum and maximum, scale factor and counts are recorded per asset as `float_display_scale`. This mapping serves the display pipeline only: saturation assessment and provenance use the true floating-point values, and for floating-point sources the initial display levels are set from the 0.1st and 99.9th percentiles of the mapped image, remaining adjustable and non-destructive. 8-bit sources (L, P, RGB, RGBA, and LA modes) are accepted; values remain in the 0-255 range with no upscaling, and `levels_white` is automatically set to 255. `is_jpeg()` performs magic-byte detection to hard-reject JPEG files at all import entry points with a scientific explanation message. `get_bit_depth()` reads source bit depth from file metadata without decoding pixel data for integrity-report population and amber-flag logic. `apply_levels_uint16()` performs black-point, white-point, and gamma correction as `float32` operations clipped to `uint16`. `rotate_uint16()` converts to Pillow I (`int32`) mode, rotates with bicubic resampling, and clips back to `uint16`. `crop_uint16()` extracts the selected region. The processing order is fixed: levels are applied first, then rotation and any flip, then cropping, so that the histogram gates shown in the interface always refer to the full membrane. Horizontal and vertical flips use `numpy.fliplr()` and `numpy.flipud()`. `save_uint16_tiff()` writes output using `Image.frombuffer("I;16", ...)`, avoiding the mode argument to `Image.fromarray()`, which was deprecated in Pillow 11.3.0 and remains deprecated where it would change the array data type.

Operation log

Provenance-relevant persisted project mutations routed through `project_io_mixin.py` call `self.log_operation(operation, *, target_type, target_id, field, old_value, new_value, note)`, which serializes

values via `_plain_log_value()` and appends an `OperationLogEntry` to `project.operation_log`. New entries are added through `append_log_entry()` in `logchain.py`, which sets `prev_hash` to the preceding entry hash (or to a genesis hash for the first chained entry), computes `entry_hash` from a canonical JSON payload, and thereby hash-chains the log from the first chained entry onward. The log is persisted as part of `project.json` by `workspace.save_project()`, and `verify_log_chain()` reports whether the stored chain is ok, broken, partial, or not_chained.

Integrity report generation

`build_integrity_report()` in `integrity.py` constructs a dict containing project metadata, panel layout, operation-log chain status, and per-blot traceability records. For each blot, `_asset_info()` opens the stored original file, verifies its SHA-256 hash against the stored path, and records bit depth and sample format, dimensions, image mode, associated source-path metadata and, where present, the floating-point display scale and LI-COR acquisition metadata. `build_detailed_integrity_report()` adds the serialized operation log and updates the schema identifier to `pysternblot.detailed_integrity_report.v2`; the base schema is `pysternblot.integrity_report.v2`. `write_integrity_json()` serializes the report to JSON (Bray, 2017); `write_integrity_html()` generates a self-contained HTML file using only Python standard library and Pillow.

Cytiva Typhoon NIR import

`parse_typhoon_tag270()` in `storage.py` parses TIFF Tag 270 (ImageDescription) key=value metadata produced by Cytiva Typhoon / Amersham TYPHOON scanners. Keys extracted include serial number, datetime, laser wavelength (nm), filter name, scan number (used to derive `channel_index` and `channel_total`), pixel size (μm), PMT HV (V), and the scanner control software string.

`import_nir_blot_typhoon()` imports 1 or 2 Typhoon TIFF files, populates `BlotChannel` entries with instrument metadata, and appends `nir_channel_imported` log entries. When a sibling `.inf` sidecar is present alongside the imported TIFF, it is parsed and its acquisition fields — scale type, scan mode, PMT and laser settings, and applied corrections — are stored on the corresponding asset and included in the integrity report. `parse_licor_metadata()` reads the channel, instrument model, serial number, software version, acquisition date and time, and pixel size from the TIFF tags of LI-COR Odyssey exports; the channel is stored as `channel_label` and the wavelength field is left unset. For marker display only, `_marker_channel_key()` in `render.py` matches LI-COR 700 and 800 channels to the 685 and 785 nm band assignments of ladder presets; nothing is written to the project.

Legend export zone

`export_legend_zone_png()` in `export_mixin.py` renders the legend export zone.

`_compute_export_geometry()` reconciles the stored `LegendZone` rectangle with the blot crop rectangle

and the shared `Panel.crop_template`, expanding the exported region so that it always contains the full crop box, and returns the offset of that box within the region. The panel legend is drawn into a `QGraphicsScene` by `draw_legend_into_scene()`, the same renderer used by the Figure tab, positioned on the crop-box offset rather than on the drawn zone. `_draw_legend_zone_markers()` draws molecular-weight ticks and labels from the blot's existing overlay-ladder assignments and calibration, ignoring the `show_in_final` and highlight-only curation applied to the final panel; for NIR blots, bands assigned to a different channel are omitted. The scene is rasterized at 2× scale and written as PNG. `LegendZone` is an optional field on `Blot`, so `project.json` files written by earlier versions load unchanged.

Software comparison

Features summarized in Table 1 were assessed from the publications cited for each tool and from the publicly available documentation for the versions accessible at the time of writing, and for Pystern Blot from the implementation described here. A capability was scored positively only where it was explicitly documented in those materials. "Not identified" indicates that the reviewed materials did not establish the capability and is not evidence of its absence. Capabilities of proprietary instrument software vary by product and version and are therefore reported only as vendor-dependent, without product-level conclusions. The tools were not installed and exercised systematically; the comparison is a documentation survey, not a benchmark.

Full-scale clipping assessment

`compute_saturation_stats()` records the maximum pixel value, the count and fraction of pixels at full scale for the source bit depth, and the count surviving a 3×3 binary erosion of the full-scale mask, implemented as eight shifted logical ANDs in NumPy with border pixels treated as below full scale. The computation is short-circuited when no pixel is at full scale. Whole-image statistics are stored on the asset as an immutable property of the imported source image; full-scale clipping within the current crop region is recomputed at report time, since only that region reaches the panel and the crop may change. The reporting threshold is a single module-level constant: any pixel surviving erosion raises a warning, on the basis that such a pixel already has eight full-scale neighbors and therefore belongs to a spatially coherent full-scale region rather than being an isolated or one-pixel-wide feature. For floating-point sources, which have no detector ceiling, no full-scale count is computed: the asset records the assessment as not assessable together with the true source maximum, and fields without meaning in the absence of a ceiling are left empty.

Unexposed image-import paths

The image utilities can additionally detect multipage TIFFs and split a single interleaved RGB/RGBA TIFF into separate channels. These low-level paths are covered by the test suite but are not exposed in the

import dialog, and are therefore not part of the supported user-facing workflow.

Archive format

The .pbarchive format is a standard ZIP file (zipfile.ZIP_DEFLATED) containing: pbarchive/manifest.json (format identifier, version, creation timestamp, project IDs, asset SHA-256 list, and, in format version 2, per-project SHA-256 digests); pbarchive/projects/<id>/project.json; and pbarchive/assets/<sha256>/original.<ext>. `import_archive()` performs a two-pass import: it first validates archive member paths, verifies each asset against the SHA-256 encoded in its path, verifies each project JSON against the manifest hash when present, and cross-checks manifest inventory against archive contents before any file is written; only then are validated assets and projects written to the workspace, with an `imported_from_archive` log entry appended on save. Version 1 archives remain readable but do not carry project-level hash verification.

Installation and availability

Pystern Blot is available from source at <https://github.com/TheBrickalista/PysternBlot> under GPL-3.0-only. Each release is also published to the Python Package Index as the `pysternblot` package, and standalone application bundles for macOS (.app) and Windows (.exe) are attached to the corresponding GitHub release; the macOS bundle (macOS 15 or later, Apple Silicon) is signed with an Apple Developer ID certificate and notarized. Versioned releases are archived on Zenodo (concept DOI: 10.5281/zenodo.20185279; v1.2.3: 10.5281/zenodo.23102409). User and developer documentation, including installation, a quickstart tutorial, a per-tab user guide, and conceptual background, is hosted at <https://pysternblot.readthedocs.io>. Installation requires Python ≥ 3.10 . The released package is installed with `pip install pysternblot`; to install from source:

```
git clone https://github.com/TheBrickalista/PysternBlot.git
```

```
cd PysternBlot && pip install -e .
```

```
python -m pysternblot
```

Validation

The behavior on which the provenance and integrity claims rest was validated with the automated test suite described below (650 tests passing for v1.2.3), using inputs constructed to exercise each failure mode together with real instrument files distributed with the test suite: a chemiluminescence TIFF and a two-channel Cytiva Typhoon NIR acquisition. Operation-log verification was tested by modifying a logged value, deleting an entry, swapping two adjacent entries and altering a stored entry hash: in each case `verify_log_chain()` reported the chain as broken, and for modified, deleted and re-hashed entries it also

located the first broken entry. Unaltered logs verified as ok, logs written before chaining was introduced as `not_chained`, and such legacy logs extended with new entries as `partial`. Archive import was tested with archives in which the project JSON, or a single operation-log entry within it, had been edited after export; both were rejected on manifest-hash mismatch and no project was written to the receiving workspace. Archives with path-traversal, backslash or absolute member names, or with a non-hexadecimal asset directory, were rejected, assets listed in the manifest but absent from the archive were flagged, and archives exceeding the per-member size, aggregate size, member count or compression-ratio limits were rejected. Byte-for-byte source export was verified to reproduce the stored asset exactly, independently of display settings and with one verified file per channel for NIR blots, and a corrupted copy was deleted and reported as an error. Full-scale clipping detection was tested on synthetic patterns: an isolated full-scale pixel and a one-pixel-wide line produced no coherent region, a full-scale 20×20 block eroded to its 18×18 interior, and 8-bit sources were assessed against 255 rather than 65,535. The floating-point display bridge was tested for proportional scaling without offset, for NaN, infinite and negative values, and for sources whose maximum is not positive. These tests establish the internal behavior of the software; they are not a user study and do not measure time savings.

Continuous integration and testing

Pull requests to the main development and release branches automatically execute the test suite via GitHub Actions. In v1.2.3, the pytest suite contains 650 passing tests covering the image-processing pipeline, 8-bit import and JPEG rejection, crop-handle geometry, data-model behavior, storage and archive integrity, source export, multichannel TIFF handling, legend-zone geometry and rendering, full-scale clipping reporting, LI-COR import and marker-channel routing, update checking, version-string consistency, and ladder rendering. Dependencies are resolved from a fully hash-pinned uv lockfile, and pull requests additionally run pip-audit against that locked dependency tree and CodeQL static analysis; third-party GitHub Actions are pinned by commit SHA. Tagged releases trigger automated PyPI publication through OpenID Connect trusted publishing, with no long-lived credentials stored in the repository, and generation of standalone macOS (.app) and Windows (.exe) application bundles via PyInstaller, attached directly to the GitHub release. The macOS bundle is signed with an Apple Developer ID certificate under the hardened runtime, notarized with notarytool and stapled, and the archived bundle is subsequently unpacked and re-verified in CI to confirm that its structure survived packaging. Both bundles are then run in CI with a `--selfcheck` option that reports the installed version and the presence of the CA certificate bundle and performs a live verified HTTPS request; a bundle that fails this check fails the release.

Use of AI-assisted tools

Software development used Claude Code (Anthropic), an AI coding assistant, for implementation, refactoring and test writing. Every change was reviewed as a diff by E.B. and committed manually, and

changes to the release branches are merged through pull requests subject to the continuous-integration checks described above. Large language models were also used in preparing this manuscript. Claude (Anthropic) assisted with drafting, editing and revising the text and with checking its consistency against the source code. The schematic diagrams in Figs. 1, 3 and 4 were designed by the authors with Claude's assistance and checked against the source code; the application screenshots and software outputs they contain, all panels of Fig. 2 (panel a reproducing the first lines of a genuine report, typeset with the abbreviations stated in its legend), and all blot images are genuine outputs of the software and instruments, and no AI tool was used to generate or modify any blot image. The authors reviewed and edited all AI-assisted content, verified it against the cited sources and the software. AI tools do not meet authorship criteria and are not listed as authors.

Declarations

Acknowledgements

The authors would like to acknowledge Philippe Lenormand for testing the application and for feedback. The authors acknowledge the IRCAN CytoMed and PICMI facilities, which are supported by grants from the Conseil Départemental 06, the FEDER, the Ministère de l'Enseignement Supérieur, the Région Provence Alpes-Côte d'Azur, the Canceropôle Sud, the Fondation ARC and INSERM.

Funding

This study was supported by grants from the French National Cancer Institute, INCa-PLBio (R24132AA) and ANR (9RPV25014MRS), by the French government, through the UCA JEDI Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-15-IDEX-01 and a grant Emergence from Canceropôle Sud to E.B. funded by the French National Institute of Cancer, Région Sud, and the Canceropôle Sud.

Competing interests

The authors declare no competing interests.

Ethics approval

Not applicable. This study describes software and reports no research on human participants or animals.

Author contributions

E.B.: conceptualization, software design, development and validation, writing — original draft. C.C.F.: supervision, funding, writing — review and editing.

Data availability

Worked example data — a tutorial .pbarchive together with representative ECL and NIR source TIFFs — are distributed with the software in the examples/ directory of the repository and in the same archived Zenodo record as the software (DOI: 10.5281/zenodo.20185279). These artifacts reproduce the project structure, operation log, and integrity reports described in this study. Any additional source files are available from the corresponding author on request. The Pystern Blot software is available at <https://github.com/TheBrickalista/PysternBlot> (DOI: 10.5281/zenodo.20185279), with documentation at <https://pysternblot.readthedocs.io>.

Code availability

Pystern Blot source code is freely available at <https://github.com/TheBrickalista/PysternBlot> under the GNU General Public License v3.0 (GPL-3.0-only). Each tagged release is archived on Zenodo; the concept DOI 10.5281/zenodo.20185279 always resolves to the most recent release, and version-specific DOIs are available from that record. The version described here is v1.2.3 (DOI: 10.5281/zenodo.23102409).

References

1. Allan, C., J.-M. Burel, J. Moore, C. Blackburn, M. Linkert, S. Loynton, D. MacDonald, W.J. Moore, C. Neves, A. Patterson, M. Porter, A. Tarkowska, B. Loranger, J. Avondo, I. Lagerstedt, L. Lianas, S. Leo, K. Hands, R.T. Hay, A. Patwardhan, C. Best, G.J. Kleywegt, G. Zanetti, and J.R. Swedlow. 2012. OMERO: flexible, model-driven data management for experimental biology. *Nat. Methods*. 9:245–253. doi:10.1038/nmeth.1896.
2. Aquilina, M., N.J.W. Wu, K. Kwan, F. Bušić, J. Dodd, L. Nicolás-Sáenz, A. O’Callaghan, P. Bankhead, and K.E. Dunn. 2025. GelGenie: an AI-powered framework for gel electrophoresis image analysis. *Nat. Commun*. 16:4087. doi:10.1038/s41467-025-59189-0.
3. Barker, M., N.P. Chue Hong, D.S. Katz, A.-L. Lamprecht, C. Martinez-Ortiz, F. Psomopoulos, J. Harrow, L.J. Castro, M. Gruenpeter, P.A. Martinez, and T. Honeyman. 2022. Introducing the FAIR Principles for research software. *Sci. Data*. 9:622. doi:10.1038/s41597-022-01710-x.
4. Bik, E.M., A. Casadevall, and F.C. Fang. 2016. The prevalence of inappropriate image duplication in biomedical research publications. *mBio*. 7:e00809-16. doi:10.1128/mbio.00809-16.
5. Bray, T. 2017. The JavaScript Object Notation (JSON) Data Interchange Format. Internet Engineering Task Force. RFC 8259. doi:10.17487/RFC8259.
6. Bucci, E.M. 2018. Automatic detection of image manipulations in the biomedical literature. *Cell Death Dis*. 9:400. doi:10.1038/s41419-018-0430-3.
7. Burnette, W.N. 1981. “Western Blotting”: Electrophoretic transfer of proteins from sodium dodecyl sulfate-polyacrylamide gels to unmodified nitrocellulose and radiographic detection with antibody and radioiodinated protein A. *Anal. Biochem*. 112:195–203. doi:10.1016/0003-2697(81)90281-5.

8. Goldberg, I.G., C. Allan, J.-M. Burel, D. Creager, A. Falconi, H. Hochheiser, J. Johnston, J. Mellen, P.K. Sorger, and J.R. Swedlow. 2005. The Open Microscopy Environment (OME) Data Model and XML file: open tools for informatics and quantitative analysis in biological imaging. *Genome Biol.* 6:R47. doi:10.1186/gb-2005-6-5-r47.
9. Gulbulak, U., A.G. Wellette-Hunsucker, T. Kampourakis, and K.S. Campbell. 2024. GelBox: open-source software to improve rigor and reproducibility when analyzing gels and immunoblots. *Am. J. Physiol. Heart Circ. Physiol.* 327:H715–H721. doi:10.1152/ajpheart.00144.2024.
10. Harris, C.R., K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T.E. Oliphant. 2020. Array programming with NumPy. *Nature.* 585:357–362. doi:10.1038/s41586-020-2649-2.
11. Hartley, M., G.J. Kleywegt, A. Patwardhan, U. Sarkans, J.R. Swedlow, and A. Brazma. 2022. The BioImage Archive – Building a home for life-sciences microscopy data. *J. Mol. Biol.* 434:167505. doi:10.1016/j.jmb.2022.167505.
12. Kroon, C., L. Breuer, L. Jones, J. An, A. Akan, E.A. Mohamed Ali, F. Busch, M. Fislage, B. Ghosh, M. Hellrigel-Holderbaum, V. Kazezian, A. Koppold, C.A. Moreira Restrepo, N. Riedel, L. Scherschinski, F.R. Urrutia Gonzalez, and T.L. Weissgerber. 2022. Blind spots on western blots: Assessment of common problems in western blot figures and methods reporting with recommendations to improve them. *PLoS Biol.* 20:e3001783. doi:10.1371/journal.pbio.3001783.
13. Kütt, J., G. Margus, L. Kask, T. Rätsepso, K. Soodla, R. Bernasconi, R. Birkedal, P. Järv, M. Laasmaa, and M. Vendelin. 2023. Simple analysis of gel images with IOCBIO Gel. *BMC Biol.* 21:225. doi:10.1186/s12915-023-01734-8.
14. Linkert, M., C.T. Rueden, C. Allan, J.-M. Burel, W. Moore, A. Patterson, B. Loranger, J. Moore, C. Neves, D. MacDonald, A. Tarkowska, C. Sticco, E. Hill, M. Rossner, K.W. Eliceiri, and J.R. Swedlow. 2010. Metadata matters: access to image data in the real world. *J. Cell Biol.* 189:777–782. doi:10.1083/jcb.201004104.
15. Mazo, G. 2021. QuickFigures: A toolkit and ImageJ PlugIn to quickly transform microscope images into scientific figures. *PLoS One.* 16:e0240280. doi:10.1371/journal.pone.0240280.
16. Moore, J., C. Allan, S. Besson, J.-M. Burel, E. Diel, D. Gault, K. Kozlowski, D. Lindner, M. Linkert, T. Manz, W. Moore, C. Pape, C. Tischer, and J.R. Swedlow. 2021. OME-NGFF: a next-generation file format for expanding bioimaging data-access strategies. *Nat. Methods.* 18:1496–1498. doi:10.1038/s41592-021-01326-w.
17. Moritz, C.P. 2020. 40 years Western blotting: A scientific birthday toast. *J. Proteomics.* 212:103575. doi:10.1016/j.jprot.2019.103575.
18. Mutterer, J., and E. Zinck. 2013. Quick-and-clean article figures with FigureJ. *J. Microsc.* 252:89–91. doi:10.1111/jmi.12069.

19. National Institute of Standards and Technology. 2015. Secure Hash Standard (SHS). National Institute of Standards and Technology, Gaithersburg, MD.
20. Rossner, M., and K.M. Yamada. 2004. What's in a picture? The temptation of image manipulation. *J. Cell Biol.* 166:11–15. doi:10.1083/jcb.200406019.
21. Russell, D.P.W., and P.K. Sorger. 2017. Maintaining the provenance of microscopy metadata using OMERO.forms software. *bioRxiv.* 109199. doi:10.1101/109199.
22. Schindelin, J., I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D.J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, and A. Cardona. 2012. Fiji: an open-source platform for biological-image analysis. *Nat. Methods.* 9:676–682. doi:10.1038/nmeth.2019.
23. The Open Microscopy Environment. 2026. OMERO.figure. <https://www.openmicroscopy.org/omero/figure/> (accessed October 8, 2026).
24. Towbin, H., T. Staehelin, and J. Gordon. 1979. Electrophoretic transfer of proteins from polyacrylamide gels to nitrocellulose sheets: procedure and some applications. *Proc. Natl. Acad. Sci. USA.* 76:4350–4354. doi:10.1073/pnas.76.9.4350.
25. Vines, T.H., A.Y.K. Albert, R.L. Andrew, F. Débarre, D.G. Bock, M.T. Franklin, K.J. Gilbert, J.-S. Moore, S. Renaut, and D.J. Rennison. 2014. The availability of research data declines rapidly with article age. *Curr. Biol.* 24:94–97. doi:10.1016/j.cub.2013.11.014.
26. Vines, T.H., R.L. Andrew, D.G. Bock, M.T. Franklin, K.J. Gilbert, N.C. Kane, J.-S. Moore, B.T. Moyers, S. Renaut, D.J. Rennison, T. Veen, and S. Yeaman. 2013. Mandated data archiving greatly improves access to research data. *The FASEB Journal.* 27:1304–1308. doi:10.1096/fj.12-218164.
27. Wilkinson, M.D., M. Dumontier, Ij.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne, J. Bouwman, A.J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C.T. Evelo, R. Finkers, A. Gonzalez-Beltran, A.J.G. Gray, P. Groth, C. Goble, J.S. Grethe, J. Heringa, P.A.C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S.J. Lusher, M.E. Martone, A. Mons, A.L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M.A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Sci. Data.* 3:160018. doi:10.1038/sdata.2016.18.

Table

Table 1 is available in the Supplementary Files section.

Figures

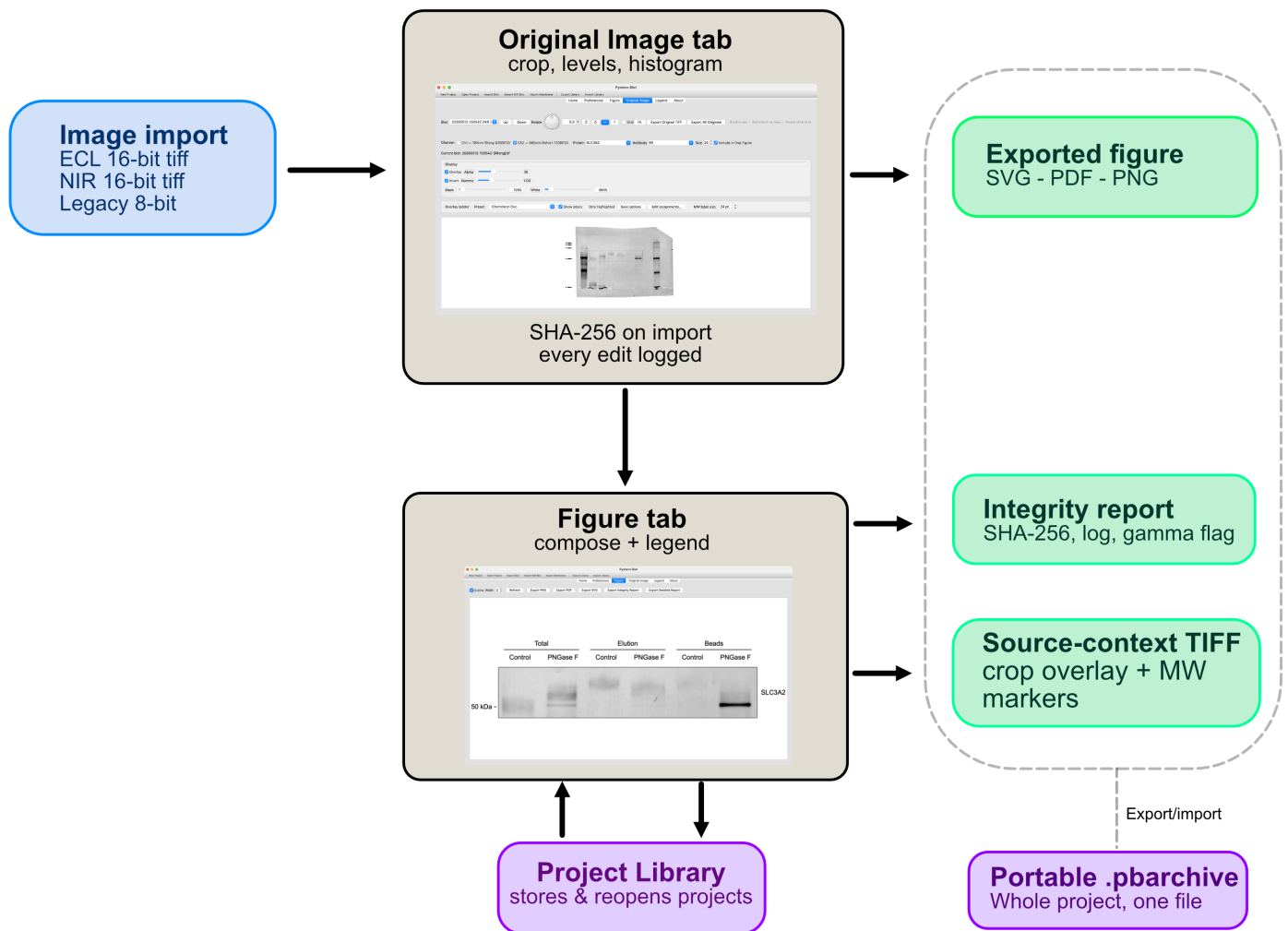


Figure 1

Pystern Blot workflow and provenance outputs. Schematic of a complete figure-preparation session, with two representative application screenshots embedded. A 16-bit TIFF is imported from the instrument (enhanced chemiluminescence [ECL] or near-infrared [NIR] fluorescence) – legacy 8-bit sources are accepted but flagged – and a SHA-256 fingerprint is recorded on import. In the Original Image tab the source membrane is cropped and adjusted with interactive levels and a live pixel-intensity histogram, and every edit is written to the operation log. The adjusted blots are composed into a multi-condition panel in the Figure tab, where lane and group annotations are defined. Pystern Blot generates three provenance outputs (dashed group, right): from the Original Image tab, an annotated source-context TIFF showing the full source membrane with the selected crop region and molecular-weight markers overlaid; and, from the composed figure in the Figure tab, the exported figure (SVG, PDF or PNG) and a dual-format integrity report carrying the SHA-256 hashes, operation-log chain status, and explicit flags for any non-default gamma adjustment. The untouched imported source file can also be re-exported byte for byte. Underlying the session, the project library persists and reopens projects, and serializes an entire project – original images, operation log and integrity metadata – into a single portable .pbarchive file for exchange with collaborators, reviewers or editors. Arrows denote data flow; the project library and .pbarchive exchange projects bidirectionally (export/import).

a Detailed integrity report (JSON, first lines)

```
pysternblot.detailed_integrity_report.v2

{
  "schema": "pysternblot.detailed_integrity_report.v2",
  "created_utc": "2026-10-08T13:38:41+00:00",
  "pysternblot_version": "1.2.3",
  "operation_log_chain": {
    "status": "ok",
    "first_broken_index": null,
    "chained_from_index": 0,
    "n_entries": 36,
    "n_chained": 36,
    "message": "Operation log chain verified ..."
  },
  "system": {
    "python": "3.12.10",
    "platform": "macOS-26.7.1-arm64-arm-64bit"
  },
  "project": {
    "id": "proj_a81bb8b29f",
    "name": "SLC3A2 PNGase His Purification",
    "created_utc": "2026-10-08T13:21:14Z",
    "modified_utc": "2026-10-08T13:21:14Z",
    "app_version": "1.2.3",
    "license": "GPL-3.0-only",
    "project_json_path": ".../project.json",
    "project_json_sha256": "611d5355...c5bd7e"
  },
  "panel": {
    "layout_order": [
      "blot_01"
    ],
    "style": {
      "font_family": "Arial",
      "font_size_pt": 10.0,
      "top_header_height_px": 80,
      "ladder_col_width_px": 60,
      "protein_col_width_px": 120,

```

b HTML integrity report

Pystern Blot integrity report

Project: SLC3A2 PNGase His Purification
Project ID: proj_a81bb8b29f
Created UTC: 2026-10-08T13:38:41+00:00
Pystern Blot version: 1.2.3
Schema: pysternblot.detailed_integrity_report.v2

Blot provenance

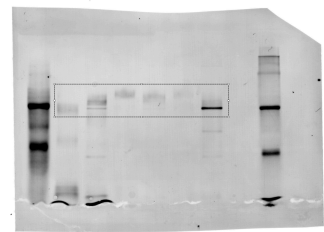
Blot	Protein	Source SHA256	Bit depth	Bit depth warning	Gamma warning	Saturation	Acquisition	Source size	Crop	Rotation	Levels	Overlay
blot_01	Protein	13cd7fa44143b08a02b47725cfdbdbd1513784dc78c3573bda712b8ac23b0e784	16			Whole image: None Crop region: None		2000 x 1500	x=50.0, y=50.0, w=300.0, h=200.0	0.0	[0-8535, v=1.0]	False

Chronological operation log

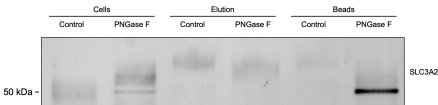
Operation log chain verified (36 entries).

Time UTC	Operation	Target type	Target ID	Field	Old value	New value	Note
2026-10-08T13:22:55+00:00	nir_channel_imported	blot			null	null	Typchoon: IRong 825BP30, 785nm, channel 1/2
2026-10-08T13:22:55+00:00	nir_channel_imported	blot			null	null	Typchoon: IRshort

c Source-context TIFF export



d Final figure

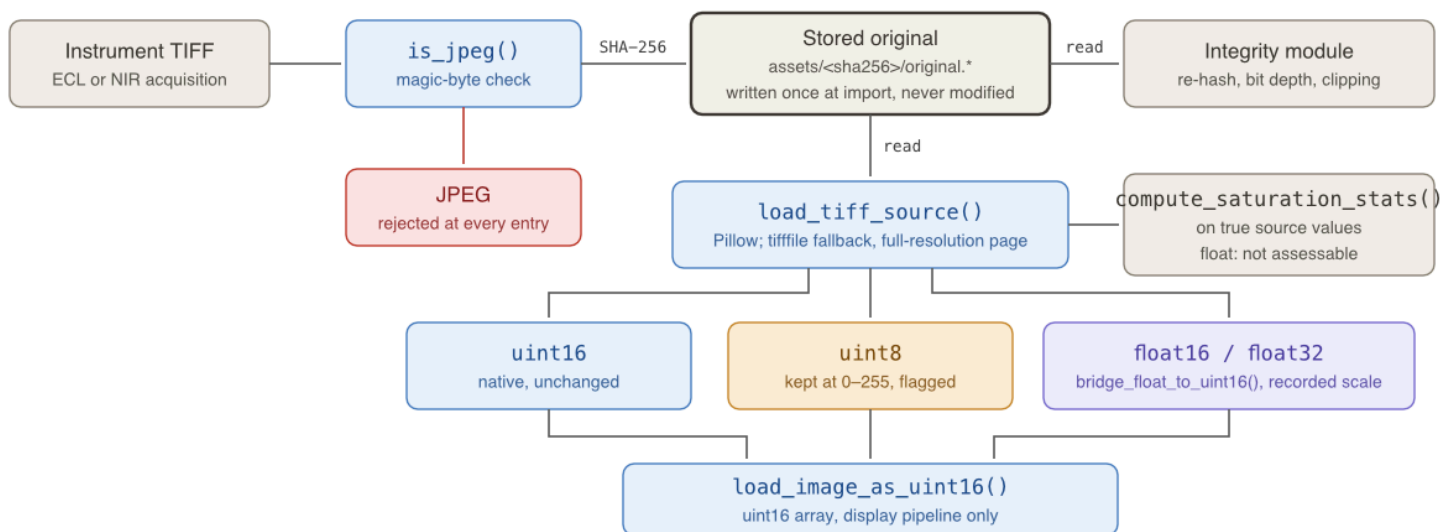


All four panels are real outputs from a single Pystern Blot v1.2.3 session (project: SLC3A2 PNGase His Purification; schema: pysternblot.detailed_integrity_report.v2).

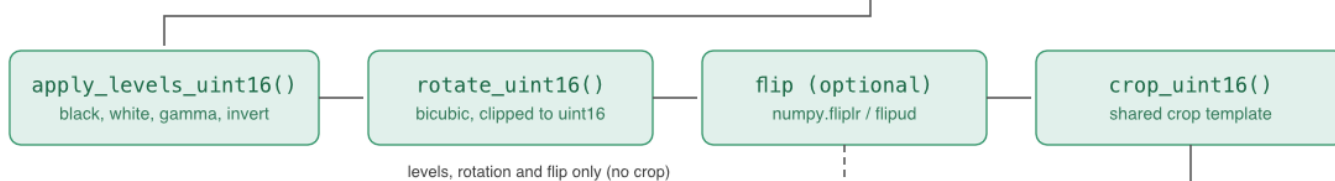
Figure 2

Integrity reporting and exported outputs. All four panels are real outputs from a single Pystern Blot v1.2.3 session (project: SLC3A2 PNGase His Purification; schema: pysternblot.detailed_integrity_report.v2) on a two-channel NIR blot. (a) First lines of the detailed integrity report (JSON): the schema identifier, creation time and software version; the operation-log chain verification (status ok, 36 of 36 entries chained); the system description; the project record, including the SHA-256 hash of the project file; and the beginning of the panel layout and style settings. The full report continues with the per-blot provenance records and the complete hash-chained operation log. For legibility, SHA-256 values are shortened to their first and last characters, file paths to their file names, and long strings are truncated (...). (b) HTML integrity report rendered in a browser, showing the project header, the blot provenance table (source SHA-256, bit depth, bit-depth, gamma and saturation warning columns, acquisition, source size, crop, rotation, levels, overlay), the operation-log chain status and the chronological operation log. For this NIR blot, the per-blot crop, levels and protein-label fields show default values, because these fields do not yet reflect per-channel settings (see Limitations). (c) Annotated source-context TIFF export of the 685 nm channel: the full source membrane rendered with the current display settings, with the selected crop region indicated by an overlaid dashed rectangle and the calibrated molecular-weight markers annotated at right. (d) Final figure exported from the Figure tab for the same blot, with condition and lane labels, the calibrated 50 kDa marker and the protein label; one of the two NIR channels was excluded from the final figure, as recorded in the operation log.

a Import (format enforced at entry)



b Processing (uint16 arrays, float32 intermediates, fixed order)



c Cache and render



Boxes with code font denote functions or files; arrows denote data flow. The stored original is only read after import.

Figure 3

16-bit source image processing pipeline. Schematic of the source image pipeline from the imported instrument file to the two rendering paths, annotated with the data type carried at each stage. (a) Import: every import entry point rejects JPEG files by magic-byte detection (`is_jpeg`). The imported file is stored once in a content-addressed asset store (`assets/<sha256>/original.*`) and only read thereafter, by the display pipeline and by the integrity module (re-hashing, bit depth and full-scale clipping). `load_tiff_source` reads the stored file with Pillow, falling back to `tiff` for tiled floating-point TIFFs (full-resolution page only); full-scale clipping is assessed on the true source values and is reported as not assessable for floating-point sources. `load_image_as_uint16` returns native 16-bit sources unchanged, keeps legacy 8-bit sources in the 0–255 range and flags them, and maps floating-point sources to `uint16` for display by a single recorded scale factor (`bridge_float_to_uint16`). (b) Processing on `uint16` arrays with `float32` intermediates, in fixed order: `apply_levels_uint16` → `rotate_uint16` → optional flip → `crop_uint16`, using the shared crop template. (c) Cache and render: the cropped result is written to a 16-bit `preview_crop` TIFF that the Figure tab renders, whereas the Original Image tab recomputes the full-membrane view in memory from levels, rotation and flip, without cropping. Boxes in code font denote functions or files; arrows denote data flow.

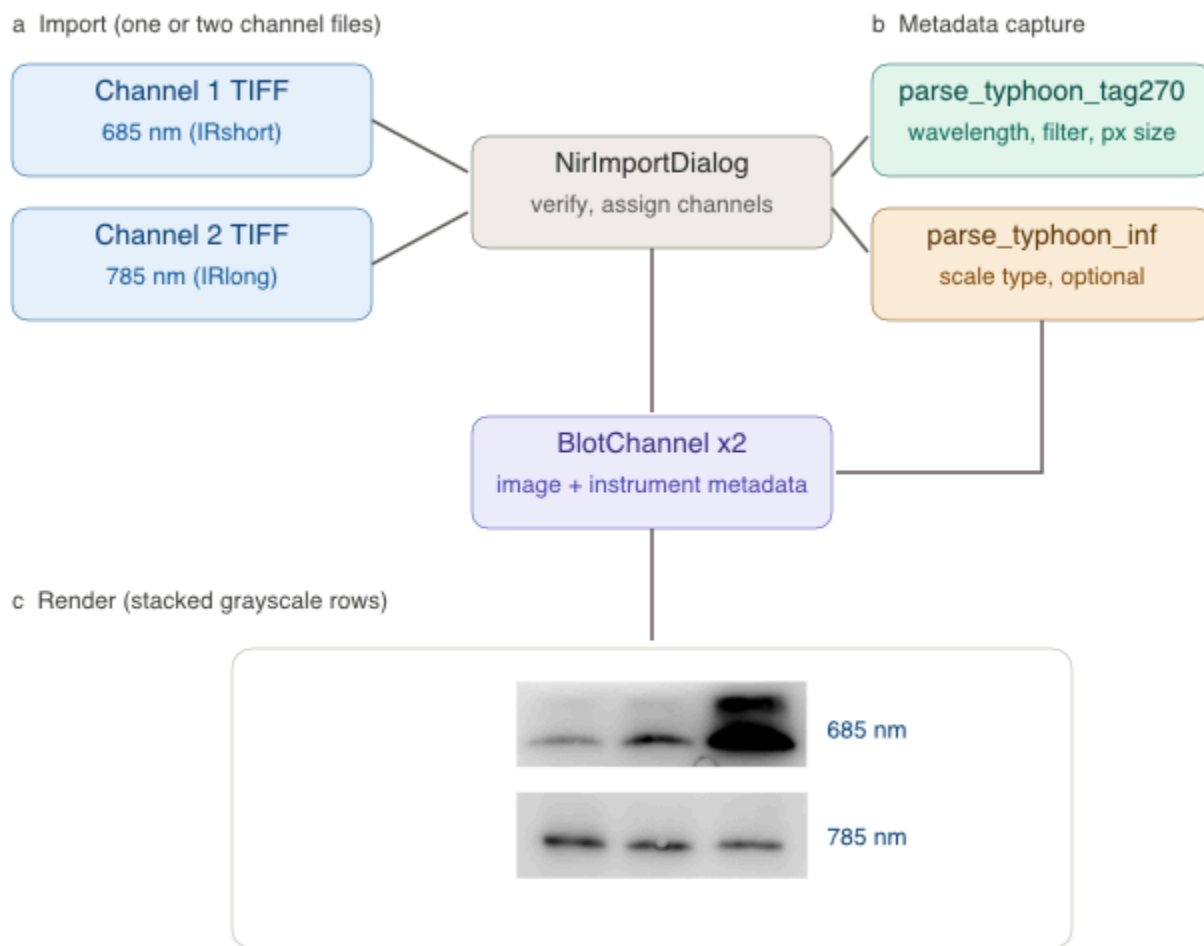


Figure 4

NIR multichannel fluorescence workflow. Schematic of the near-infrared (NIR) multichannel import and rendering path. (a) Import: one or two single-channel 16-bit TIFFs (here 685 nm, IRshort, and 785 nm, IRLong) are passed to NirImportDialog, which verifies the files and assigns them to channels, populating one BlotChannel object per channel with the image and its instrument metadata. (b) Metadata capture: parse_typhoon_tag270 extracts wavelength, filter name, scan number and pixel size from TIFF Tag 270, while parse_typhoon_inf reads the optional sibling .inf sidecar for acquisition fields including the scale type, which governs the quantitative interpretation of pixel values. Both are attached to the corresponding BlotChannel. (c) Render: the NIR blot is drawn as stacked per-channel grayscale rows, each labeled with its laser excitation wavelength. Boxes denote input files and code objects; connecting lines denote data flow.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Table1.png