ICHNOS Tool

User and Technical Documentation

with the ichnos_vail application

Document type:	User and technical documentation
Software:	ichnos-tool 0.1.0
Date:	October 2026
Team:	iGEM 2026 Patras-Med, University of Patras
Project:	ICHNOS
Repository:	gitlab.igem.org/2026/software/patras-med/ichnos-tool
Platform:	macOS or Linux (tested on macOS, Apple Silicon)
Python:	3.10 or later
Interface:	ichnos_vail (Streamlit, in the browser) or command line
License:	Apache License 2.0

iGEM 2026 Patras-Med

ICHNOS · Foundational Advance

Contents

About this guide 2

1 Project Overview 3
- 1.1 Introduction . . . 3
- 1.2 Background . . . 3
- 1.3 Objectives . . . 3
- 1.4 What the tool will and will not tell you . . . 4

2 System Architecture 5
- 2.1 Components . . . 5
- 2.2 Workflow . . . 6
- 2.3 Imaging configuration . . . 6

3 Installation Guide 7
- 3.1 Prerequisites . . . 7
- 3.2 Download and install . . . 7
- 3.3 Optional extras . . . 7
- 3.4 Check the installation . . . 8
- 3.5 Start ichnos_vail . . . 8
- 3.6 Repository layout . . . 8

4 Configuration 10
- 4.1 The image manifest . . . 10
- 4.2 Naming rules . . . 11
- 4.3 What a good experiment contains . . . 11
- 4.4 Analysis settings . . . 12
- 4.5 Application settings . . . 12

5 User Guide: the ichnos_vail application 13
- 5.1 Application interface . . . 13
- 5.2 Start page . . . 14
- 5.3 Step 1 · Check images . . . 14
- 5.4 Step 2 · Images → cells . . . 16
- 5.5 Step 3 · Analyse an experiment . . . 17
- 5.6 Step 4 · Decode a new culture . . . 20
- 5.7 Jobs . . . 22
- 5.8 Worked example: try it with the team's data . . . 23

6 Command-line use 25
- 6.1 The four steps . . . 25
- 6.2 Useful options . . . 25
- 6.3 Other scripts . . . 26
- 6.4 Example: simulate the model . . . 26

7 Technical Details **27**
7.1 Segmentation and measurement . . . 27
7.2 Analysis of an experiment . . . 27
7.3 Decoders . . . 28
7.4 How a verdict is chosen . . . 28
7.5 Reproducibility . . . 28
7.6 Libraries used . . . 29

8 Troubleshooting **30**

9 Known Limitations **32**
9.1 How the tool behaved on the October 2026 reference . . . 32
9.2 Measurement and design limits . . . 32
9.3 Software scope . . . 33
9.4 What would remove these limits . . . 33

10 Support, Contact and Attribution **34**

About this guide

The ICHNOS tool turns fluorescence microscope images of yeast carrying the ICHNOS stress reporter into two answers — **how much stress** a culture received and **how long ago** — and says, for every answer, whether it can be trusted. This guide shows anyone how to install it and use it, mainly through its browser application, **ichnos_vail**.

How to read this guide. If you only want to use the tool, read chapters 3 (install), 4 (prepare your manifest) and 5 (the app). Chapters 6–7 are for people who want the command line or the method behind each result. Chapter 9 lists what the results can and cannot support — read it before reporting a number.

Chapter 1

Project Overview

1.1 Introduction

The ICHNOS tool is the software of the iGEM 2026 team Patras-Med. It takes microscope images of yeast that carry the ICHNOS reporter and answers two questions about a culture: which stress dose it received, and how much time has passed since the stress. Every answer comes with a **verdict** that says whether it can be trusted.

The tool does three things:

1. **Images → cells.** It finds every cell on the bright-field image with Cellpose, measures its green and red fluorescence, and corrects for exposure time.
2. **Model vs data.** It simulates the ICHNOS model (oxidative and ER variants) and compares the predictions with the measured responses.
3. **Decoding.** It estimates the dose and the time since stress of a new culture, each with a verdict.

You can run all of this from **ichnos_vail**, a local application that opens in your web browser, or from the command line. The two give identical results: the app only runs the same scripts and shows you their outputs.

1.2 Background

ICHNOS is a biosensor in *S. cerevisiae*. A stress-responsive sensor (Yap1 for oxidative stress with H_2O_2, the UPR for ER stress with DTT) drives a TIP–TetR negative feedback loop, which makes the output change more linearly with dose. The output is a **tandem fluorescent timer**: sfGFP matures in minutes and mCherry in hours. The sensor says **how much** stress there was; the red/green ratio of the timer says **when** it started.

Reading these two quantities from images is not trivial. Cells must be found reliably, the two channels must be measured on the same cells, autofluorescence and bleed-through must be removed, and days with different transformations must not be mixed. The tool packages all of this into one reproducible route.

1.3 Objectives

- Segment yeast cells on bright-field images and measure green and red per cell.
- Normalise intensities for exposure time and remove the autofluorescence of non-transfected cells.

- Test every dose against the same day's unstressed culture, with confidence intervals and detection limits.
- Compare the measured response with the ICHNOS model.
- Decode the dose and time since stress of a new culture, and attach to each answer the decoder's own score on reference days it never saw.
- Make every result reproducible: each run records its settings, input checksum and code version, and never overwrites an earlier result.
- Offer all of this to non-programmers through a browser application.

1.4 What the tool will and will not tell you

The tool will	The tool will not
Find cells on bright-field and measure each cell's green and red	Give a dose from the ICHNOS model alone: the model is uncalibrated
Remove autofluorescence and test each dose against the same day's 0 µM	Pool different days into one control
Give a dose or a time only when the reference experiment supports one	Report a number when the verdict says the answer cannot be trusted
Show, with every answer, how well the decoder did on held-out days	Turn passing software tests into a claim of experimental accuracy

With the team's October 2026 data as reference, no new image reaches the verdict "dose estimate". Those data contain no dose signal that carries over to a new day, and the tool reports exactly that (chapter 9).

Chapter 2

System Architecture

2.1 Components

The tool is two Python packages, a set of command-line scripts, and the ichnos_vail application on top. ichnos_vail computes nothing itself: it builds the scripts' command lines, runs them, and shows what they wrote.

Component	What it does
ichnos_vail (`app/ichnos_vail.py`, `ichnos_vail/`)	Streamlit app in the browser. Checks inputs, shows the exact command, runs it as a background job, displays results
Background jobs (`ichnos_vail/jobs.py`, `chain.py`)	Run each step in a separate process, so a long run continues if the browser closes. Status and log are kept on disk
Manifest check (`ichnos_vail/manifest_check.py`)	Finds missing columns, missing image files and naming problems before a run starts
Image pipeline (`ichnos_image`)	Bright-field alignment, Cellpose segmentation, per-cell green/red, QC, exposure normalisation
Real-data analysis (`ichnos_image`)	Time courses, induction test against same-day 0 µM, model comparison, leave-one-day-out decoder test
Empirical decoders (`ichnos_image.field_decoder`)	Dose decoder (paired or absolute) and time-window decoder, each with a verdict and a track record
ICHNOS model (`ichnos`)	SBML models for ox (H_2O_2) and er (DTT), simulation, stress protocols, model-based decoders
Scripts (`scripts/`)	One command-line entry point per step; every app button runs one of them

2.2 Workflow

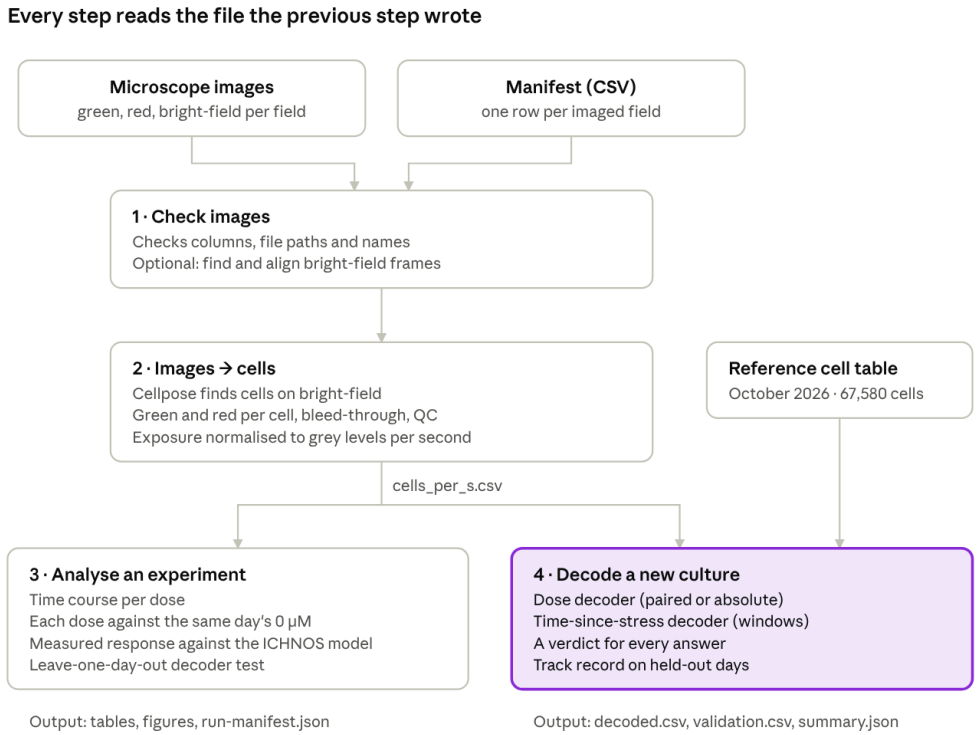

Figure 2.1: Workflow of the ICHNOS tool. Every step reads the file the previous step wrote; decoding also uses a reference cell table.

Each step writes its own output folder, and the next step reads a file from it. Analysis (step 3) and decoding (step 4) both start from the same per-cell table; decoding also needs a reference experiment, which is the team's October 2026 data by default.

2.3 Imaging configuration

The reference data were taken on the team's microscope. New images should use the same setup, because the baseline, thresholds and dose classes all come from the reference.

Parameter	Value (October 2026 data)
Microscope	Olympus CKX41
Camera	QImaging MicroPublisher 3.3 RTV behind a 0.5X adapter
Objective	40X (4X, 10X, 40X and 60X are known to the tool)
Pixel size	0.1725 µm per pixel
Bit depth	8-bit frames, saturation at 255
Channels per field	green, red and bright-field of the same position
Exposure	green and red of one field share one exposure
Bleed-through (green → red)	0.05, indicative: no single-colour controls were imaged

Chapter 3

Installation Guide

You install the tool once, in its own Python environment. The first segmentation also downloads a 1.15 GB Cellpose model.

3.1 Prerequisites

- macOS or Linux (the tool is tested on macOS with Apple Silicon; the GitLab tests run on Linux).
- Python 3.10 or later. On macOS, run Python as `python3`.
- Git, to download the repository.
- A few GB of free disk space for the environment, the Cellpose model and outputs.
- A GPU is recommended. Cellpose takes about 2 minutes per field on an Apple GPU and is several times slower on a CPU.
- A web browser (any recent Chrome, Firefox or Safari) for ichnos_vail.

3.2 Download and install

Run these commands in a terminal, one block at a time.

```
# 1. Download the tool
git clone https://gitlab.igem.org/2026/software/patras-med/ichnos-tool.git
cd ichnos-tool

# 2. Create and activate a virtual environment
python3 -m venv .venv
source .venv/bin/activate

# 3. Install the tool with image analysis, the model and the app
pip install -e ".[image,model,ichnos_vail]"

# 4. Install Cellpose (needed for bright-field segmentation)
pip install cellpose
```

Each time you open a new terminal, enter the folder and activate the environment again with `source .venv/bin/activate`.

3.3 Optional extras

Extra	Install with	Contains
`image`	`pip install -e ".[image]"`	scikit-image, Pillow, tifffile, matplotlib
`model`	`pip install -e ".[model]"`	python-libsbml, tellurium, PyYAML
`cellpose`	`pip install -e ".[cellpose]"`	Cellpose and PyTorch
`ichnos_vail`	`pip install -e ".[ichnos_vail]"`	Streamlit 1.50 or later
`dev`	`pip install -e ".[dev]"`	pytest, to run the tests

The base install (NumPy, SciPy, pandas) is enough to decode a cell table that already exists.

3.4 Check the installation

```
pip install -e ".[dev]"
python3 -m pytest -q
```

All tests should pass. The Cellpose tests need the model download, so the first run is slower. Passing tests check that the software behaves as designed; they do not show that dose or time estimates are accurate on real data.

3.5 Start ichnos_vail

```
streamlit run app/ichnos_vail.py
```

The app opens in your browser at `http://localhost:8501`. To stop it, press Ctrl+C in the terminal.

For long runs on macOS, stop the computer from sleeping:

```
caffeinate -i streamlit run app/ichnos_vail.py
```

The app is served to this computer only. It runs scripts with paths typed into it, so `.streamlit/config.toml` keeps it on `localhost`. Do not change that address to share it on a network.

3.6 Repository layout

Folder or file	Contents
`app/ichnos_vail.py`	The ichnos_vail application
`ichnos_vail/`	App helpers: commands, background jobs, manifest check, results display
`ichnos_image/`	Image pipeline, analysis of real data, empirical decoders
`ichnos/`	ICHNOS model (SBML, parameters), simulation, model-based decoders

Folder or file	Contents
`scripts/`	Command-line entry points for every step
`data/reference/`	The October 2026 reference cell table (`cells_all_per_s.csv.gz`)
`data/segmentation_reference/`	Hand-marked cells for scoring segmentation
`examples/`	Example configurations (`analysis_202610.json`, `workflow.json`)
`docs/`	Detailed documentation: user guide, data contract, limitations, run log
`tests/`	Automated tests
`outputs/`	Created on first use: every run's results and the app's jobs

Chapter 4

Configuration

The only file you must write yourself is the **image manifest**: a CSV with one row per imaged field. Everything else has sensible defaults that you change in the app.

4.1 The image manifest

Create it in a spreadsheet and save it as CSV. Image paths are relative to the manifest's own folder. A blank cell means "not recorded", never zero.

```
session_id,sample_id,condition_id,timepoint,acquisition_order,green_path,red_path,
↪ bright_field_path,exposure_ms_green,exposure_ms_red,nd_filter_green,nd_filter_
↪ red,objective,burner_hours,lamp_warmup_minutes,measurement_time_hours
20261101_h2o2_tr,ctrl_2h_R1,h2o2_0uM_2h,0,0,img/g001.tif,img/r001.tif,img/bf001.ti
↪ f,400,400,,,40X,,,2.0
20261101_h2o2_tr,d300_2h_R1,h2o2_300uM_2h,0,1,img/g002.tif,img/r002.tif,img/bf002.
↪ tif,400,400,,,40X,,,2.1
20261103_h2o2_nt,nt_2h_R1,h2o2_0uM_2h,0,0,img/g101.tif,img/r101.tif,img/bf101.tif,
↪ 800,800,,,40X,,,2.0
```

Column	Required	Meaning
`session_id`	yes	One imaging day of one stressor: `<yyyymmdd>_<stressor>_<tr|nt>`
`sample_id`	for analysis	One field (one image); unique within a condition
`condition_id`	for analysis	`<stressor>_<dose>uM_<label>` for a known dose; any other name for an unknown dose
`timepoint`, `acquisition_order`	yes	Acquisition index and order. Not elapsed time
`green_path`, `red_path`	yes	Green and red fluorescence images of the field
`bright_field_path`	yes (may be empty)	Bright-field image of the same position
`exposure_ms_green`, `exposure_ms_red`	yes	Exposure in ms; must be positive
`nd_filter_green`, `nd_filter_red`	yes (may be empty)	Neutral-density filter, if any
`objective`	yes	`4X`, `10X`, `40X` or `60X`
`burner_hours`, `lamp_warmup_minutes`	yes (may be empty)	Lamp state, if recorded

Column	Required	Meaning
`measurement_time_hours`	for analysis	Hours from adding the stress to imaging; blank if unknown

Optional columns used when present: `sampling_time_hours`, `specimen_id`, `biological_replicate_id`, `gain_setting`, `expect_cells` (false for medium-only frames) and `log_flags`. The full list is in `docs/data_contract.md`.

4.2 Naming rules

The tool reads the stressor, the culture type and the dose from the names, so they must follow these patterns.

Name	Pattern	Example
Session	`<yyyymmdd>_<stressor>_<tr|nt>` (`tr` transfected, `nt` non-transfected)	`20261007_h2o2_tr`
Known dose	`<stressor>_<dose>uM_<label>`	`h2o2_300uM_2h`
Same-day control	`<stressor>_0uM_<label>`	`h2o2_0uM_ctrl`
Unknown dose (to decode)	any other name	`mystery_A`

Stressors are `h2o2` (model variant ox) and `dtt` (model variant er). Doses are in µM. Fields R1, R2, R3 of one sample are fields of the same culture, not independent replicates.

4.3 What a good experiment contains

You need	Why
The same reporter, objective, camera and settings for reference and new images	Baseline, thresholds and dose classes come from the reference
Green, red and bright-field frames of the same position	Cells are found on bright-field and measured in green and red
An unstressed (0 µM) culture on **every** day	All comparisons are made within a day; the paired decoder needs it
A non-transfected culture of each stressor, at least once	Autofluorescence baseline and the reporter-positive threshold
The time from adding the stress to each image	Time axis of every analysis
Several doses on at least 3 independent days (reference)	Validation holds each day out
Bleed-through controls (green-only cells imaged in red)	Otherwise an indicative 0.05 is assumed

4.4 Analysis settings

Step 3 writes a small JSON configuration for you. From the command line you copy `examples/analysis_202610.json` and edit it.

Setting	Default	Meaning
`cells`	reference table	The per-second cell table to analyse
`steps`	all four	`timecourse`, `induction`, `model_compare`, `decoder_lodo`
`n_boot`	1000	Bootstrap resamples for intervals
`seed`	0	Random seed; recorded so reruns are identical
`positive_quantile`	0.9	A cell is reporter-positive above this quantile of non-transfected cells
`clearance_per_h`	none, 1, 3	Model stress scenarios: constant, or first-order clearance per hour
`decoders`	absolute, paired, model	Decoders scored in the leave-one-day-out test
`min_cells`	20	Fields with fewer usable cells are skipped

4.5 Application settings

Setting	Where	Default
Server address	`.streamlit/config.toml`	`localhost` only; keep it that way
Jobs folder	environment variable `ICHNOS_VAIL_JOBS_DIR`	`outputs/ichnos_vail_jobs/` in the repository
Output folders	each page's "Output folder" field	`outputs/<step>_<date>-<time>`; must not exist yet

Chapter 5

User Guide: the ichnos_vail application

ichnos_vail takes you from images to answers in four steps, one page each, in the order of the sidebar. You never need to type a command: each page fills in the command for you, shows it, and runs it when you press the button.

5.1 Application interface

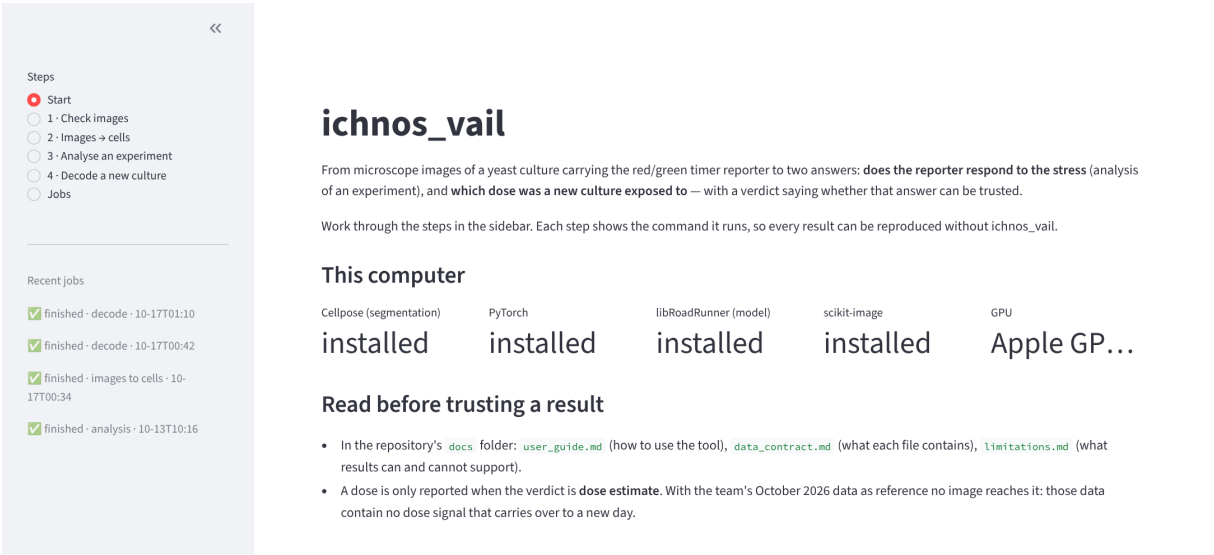

Figure 5.1: Start page. The sidebar lists the steps and the most recent jobs.

The window has two parts.

- **Sidebar (left).** The six pages: Start, 1 · Check images, 2 · Images → cells, 3 · Analyse an experiment, 4 · Decode a new culture, and Jobs. Under them, the last five jobs with their state.
- **Main area (right).** The page you selected. Every page that runs something has the same parts: input fields, an **Advanced** section that can stay closed, the command it will run ("This runs exactly the following"), a start button, and a live job panel.

Three things hold on every page:

- **Paths** can be absolute or relative to the repository folder. A path you typed is remembered when you switch pages.

- **Output folders** must be new. The app suggests one with the date and time, such as `outputs/cells_20261017-1631`. It never overwrites an earlier result.
- **Jobs keep running** if you close the browser tab. Find them again under **Jobs**.

5.2 Start page

The Start page checks what is installed on this computer: Cellpose, PyTorch, libRoadRunner (for the model) and scikit-image, and whether a GPU was found (CUDA or Apple GPU). A yellow warning appears if Cellpose is missing, with the command to install it. A blue note appears if no GPU is found: segmentation will then run on the CPU, several times slower.

It also reminds you of the one rule that matters most: report a dose only when its verdict is **dose estimate**.

5.3 Step 1 · Check images

This page finds problems in your manifest in seconds, before a run that can take hours.

1. Type the path of your manifest in **Manifest (CSV)**, for example `examples/demo/manifest.csv`.
2. Press **Check**.
3. Read the four numbers: fields, sessions, errors and warnings.
4. Read the table of problems. **Errors** (red) must be fixed before step 2. **Warnings** (orange) let the pipeline run but limit a later analysis.
5. Open **First rows** to see the manifest as the tool read it.

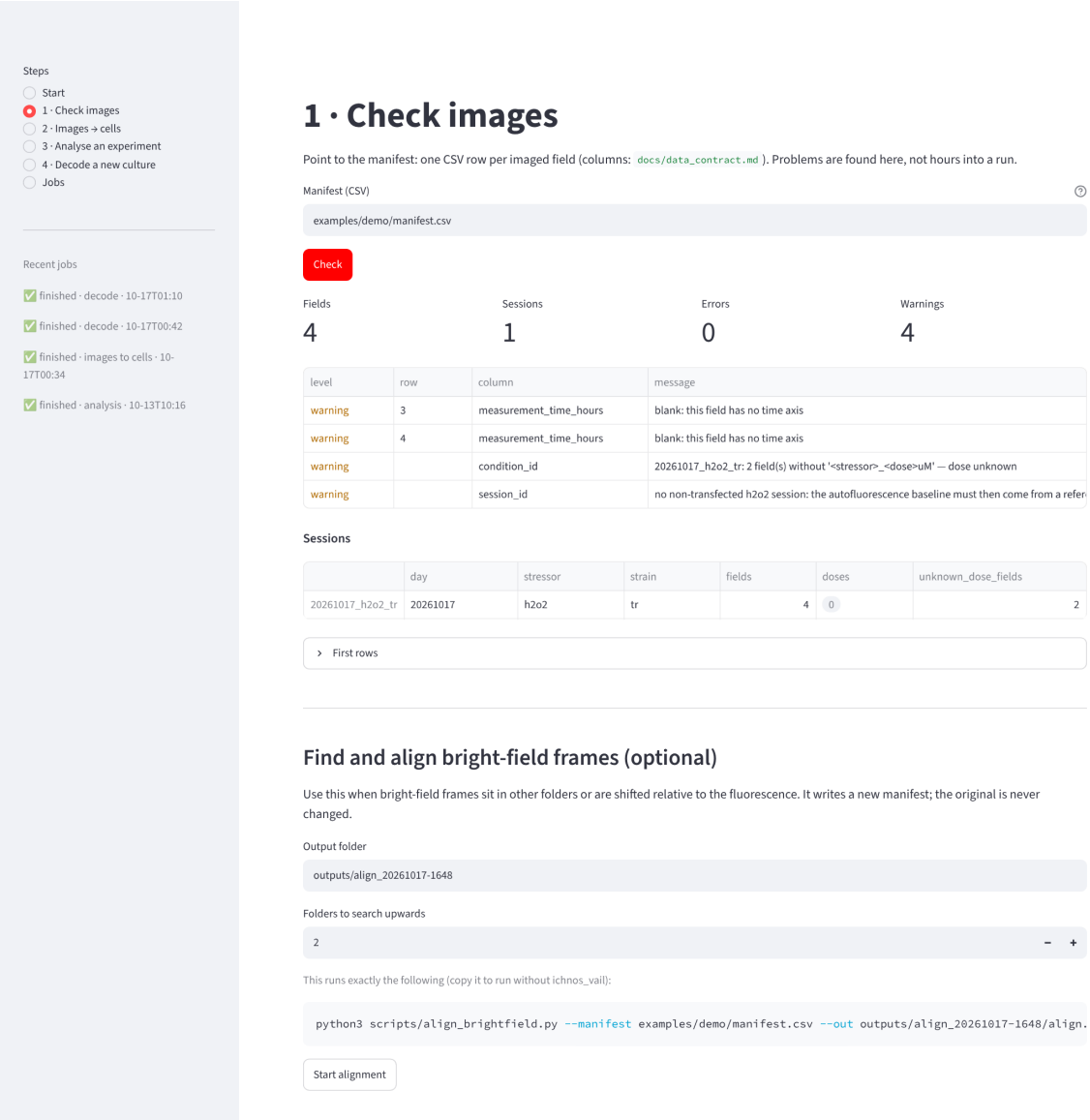

Figure 5.2: Check images. This manifest has no errors and four warnings: two fields without a measurement time, two fields of unknown dose (the culture to decode), and no non-transfected session on that day.

The check looks for	Level
A required column missing	error
An image file that does not exist	error
An exposure that is not a positive number	error
A measurement time below 0	error
An unknown objective	error
Two rows with the same session, condition and sample	error
A blank measurement time	warning
A session name that does not follow the pattern	warning
Fields with no dose in their name	warning
A transfected day without a 0 µM culture	warning
No non-transfected session for a stressor	warning

The check looks for	Level

When there are no errors, the manifest path is passed to step 2 automatically.

Find and align bright-field frames (optional). Use this when the bright-field frames are in other folders, or shifted relative to the fluorescence. Choose an output folder and how many folders upward to search (default 2), then press **Start alignment**. It writes a new manifest, `images_aligned.csv`, with the bright-field path and shift of each field. Your original manifest is never changed, and step 2 uses the aligned one.

5.4 Step 2 · Images → cells

This page turns every field into rows of a cell table: it finds cells on bright-field, measures green and red in each, applies QC, and converts intensities to grey levels per second. It is the long step: about 2 minutes per field with Cellpose on a GPU.

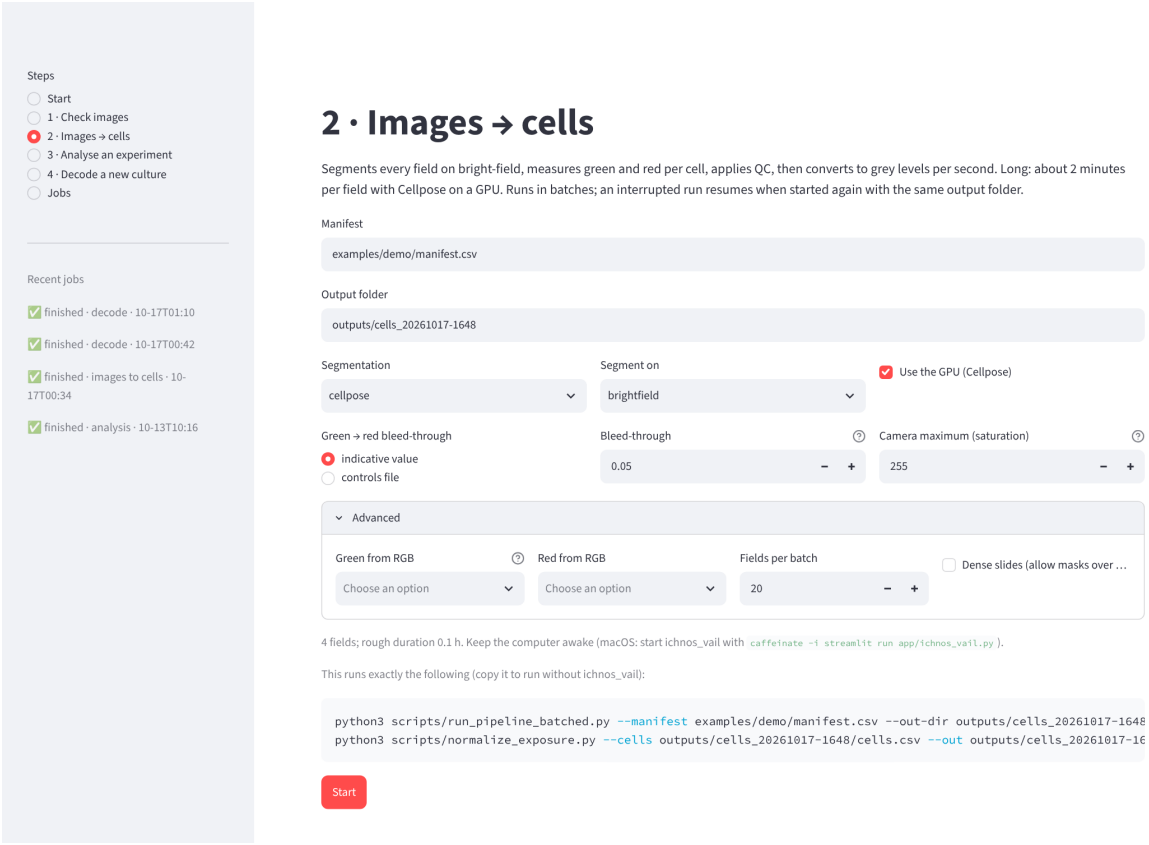

Figure 5.3: Images → cells, with the Advanced section open. The page estimates the run time and shows the two commands it will run.

Field	Default	When to change it
Manifest	from step 1	Use another manifest
Output folder	`outputs/cells_<date>-<time>`	Keep it; use the same folder to resume an interrupted run
Segmentation	`cellpose`	Other methods (`brightfield_rim`, `transmitted`, `sparse`, `otsu`) are for testing

Field	Default	When to change it
Segment on	`brightfield`	Only if you have no bright-field (`sum`, `green`, `red`)
Use the GPU	on	Off if no GPU was found on the Start page
Green → red bleed-through	indicative value 0.05	Choose **controls file** if you imaged green-only control cells
Camera maximum (saturation)	255	4095 for 12-bit, 65535 for 16-bit images
Green / Red from RGB (Advanced)	empty	Only for colour (RGB) files: which channel holds green and red
Fields per batch (Advanced)	20	Smaller batches lose less work if interrupted
Dense slides (Advanced)	off	Turn on if fields are refused for covering over 50 % of the frame

Press **Start**. The job panel shows the step (1 of 2 for segmentation, 2 of 2 for normalisation), the minutes elapsed, a progress bar of batches, and the end of the log. When the job finishes, the page shows the path of the cell table, `cells_per_s.csv`, and fills it into steps 3 and 4.

If the computer sleeps or restarts, the job is marked **interrupted**. Start the same run with the same output folder: finished batches are kept and the run continues.

5.5 Step 3 · Analyse an experiment

This page answers the first question: **does the reporter respond to the stress, and how large an effect could this experiment have detected?** It runs four analyses in a few minutes.

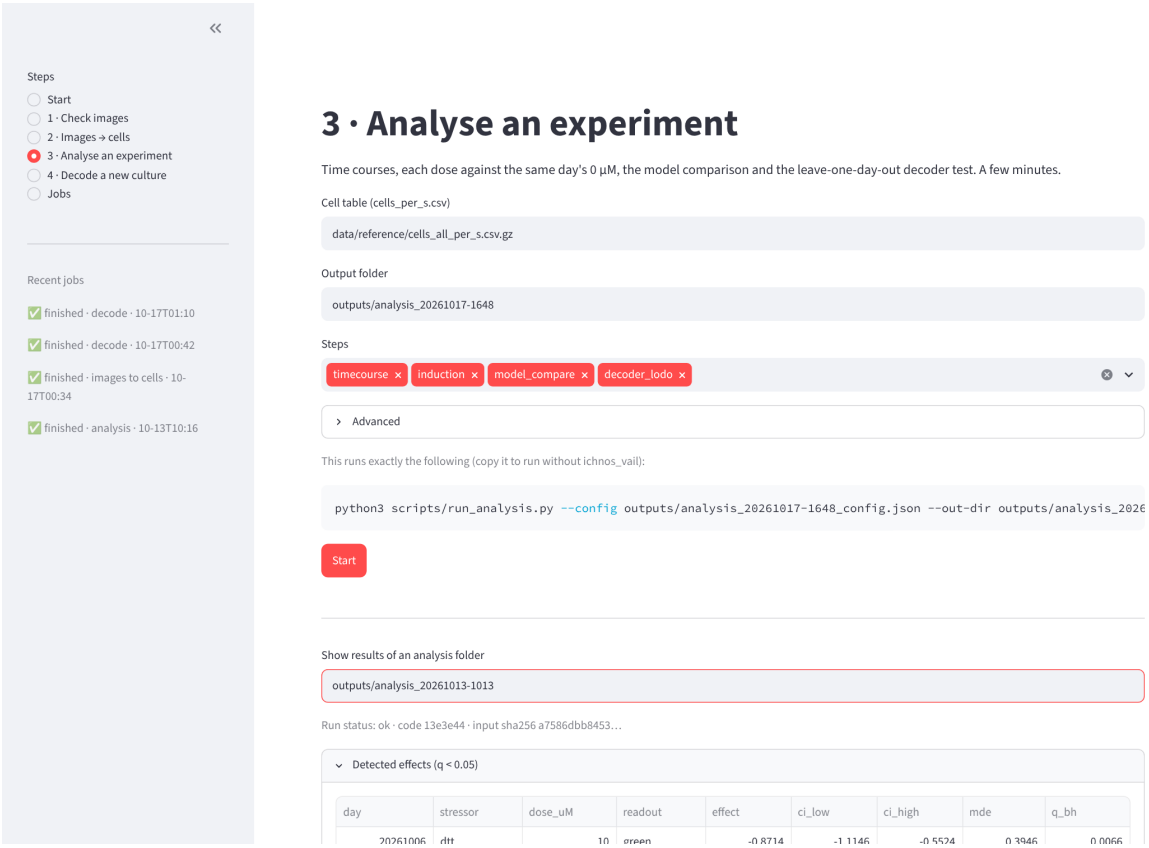

Figure 5.4: Analyse an experiment. The cell table defaults to the October 2026 reference.

1. **Cell table.** The `cells_per_s.csv` from step 2. It defaults to the team's reference table, so you can try the page right away.
2. **Output folder.** A new folder.
3. **Steps.** Keep all four, or remove those you do not need.
4. **Advanced** (optional): resamples, seed, reporter-positive quantile, smallest field, model clearance scenarios, decoders. See section 4.4.
5. Press **Start**, wait for the job to finish, and the results appear below.

To look at an older analysis, type its folder in **Show results of an analysis folder**.

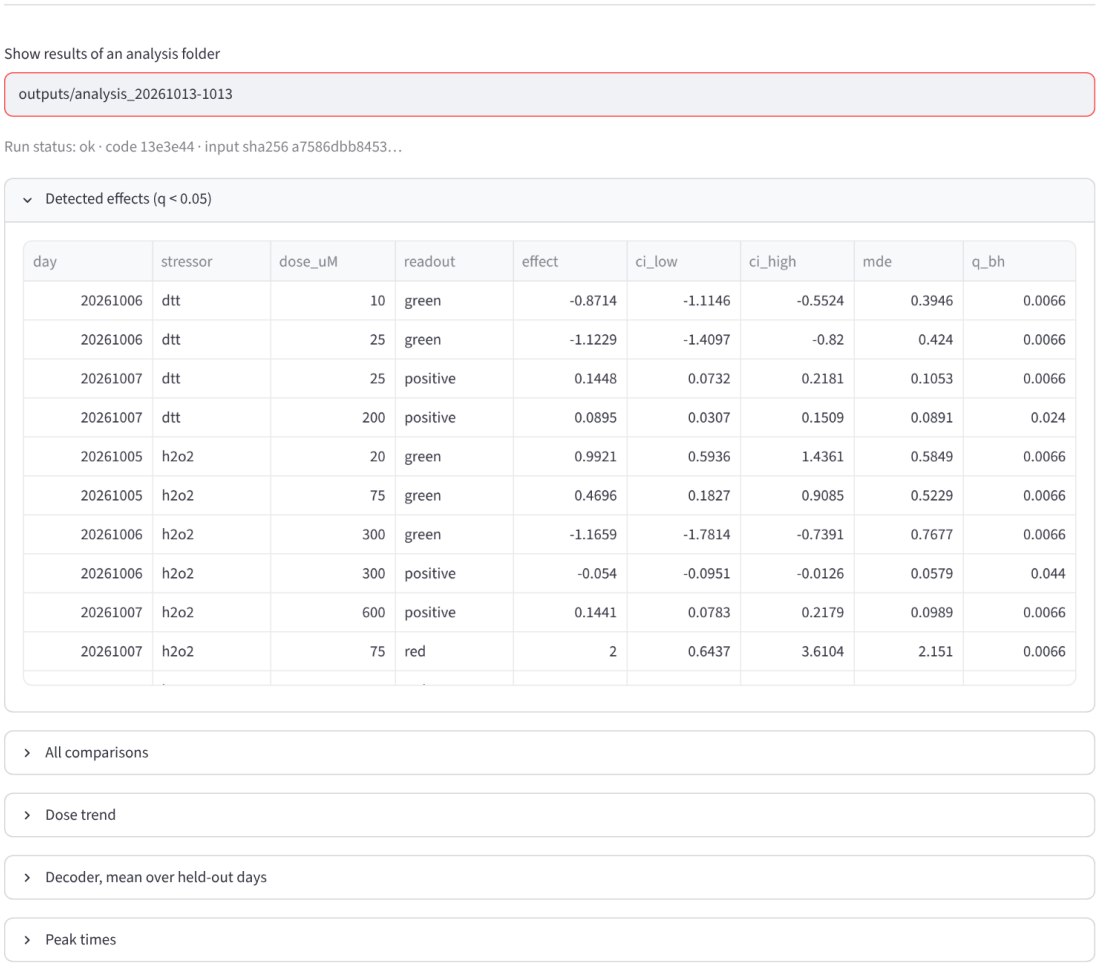

Figure 5.5: Results. The run status, code version and input checksum come first, then the tables.

Result	What it tells you
Run status line	Whether the run was complete, the code version and the input checksum
Detected effects (q < 0.05)	Doses that differ from the same day's 0 µM after correction for multiple tests. `effect` with its 95 % interval; `mde` is the smallest effect this experiment could have detected
All comparisons	Every day × dose × readout, detected or not
Dose trend	Whether the effect grows with dose (slope per 10-fold dose)
Decoder, mean over held-out days	Accuracy against chance and information gain in bits, for each decoder
Peak times	When each curve peaks; a peak at the first or last sample is only a bound
Time course figures	Green, red and red/green over time, one panel per dose, one colour per day
Model against data	Measured fold over 0 µM against the model, for constant stress and two clearance rates
Single-field decoder	The leave-one-day-out posterior: could a single image's dose be recovered on a day the decoder never saw?

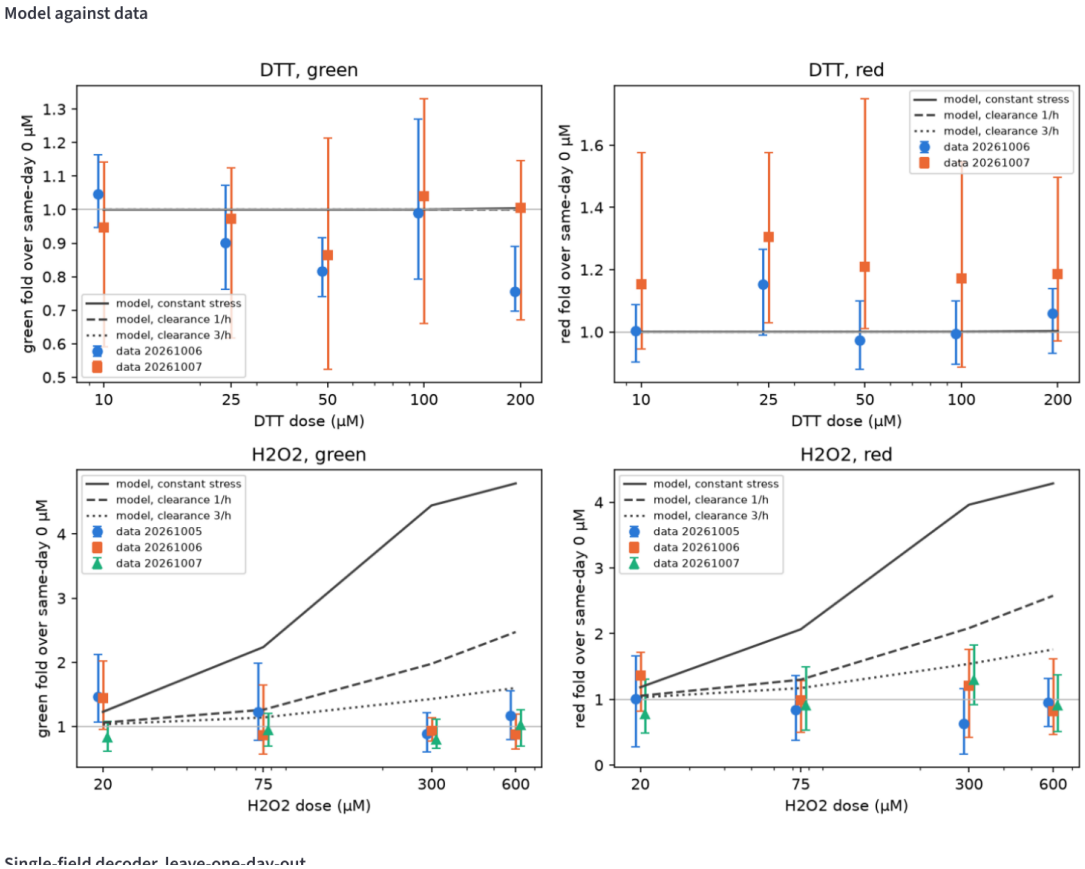

Figure 5.6: Model against data for the October 2026 reference. Points are measured folds with 95 % intervals; lines are the model under three stress scenarios.

5.6 Step 4 · Decode a new culture

This page answers the second question: **which dose was my culture exposed to, and how long ago?** Both decoders are trained on a reference experiment. Their scores on reference days they never saw are shown next to every answer.

Before you start, check two things in your manifest:

- The same-day unstressed culture is named `<stressor>_0uM_<label>`. The unknown samples have any other name.
- `measurement_time_hours` is blank if you do not know when the stress came.

Field	Default	Meaning
Reference cell table	October 2026 reference	The experiment the decoders learn from
New culture's cell table	from step 2	The `cells_per_s.csv` of the culture to decode
Stressor	first in the reference	`h2o2` or `dtt`
Leave reference days out	none	Leave out the day you are decoding, if it is also in the reference

Field	Default	Meaning
Decoder	paired	**paired** compares each image with the same-day 0 µM; **absolute** uses the image alone and is sensitive to day-to-day differences
Smallest field (cells)	20	Fields with fewer usable cells get “too few cells”
Time windows	`0,2,4,6,8`	Window edges in hours: 0–2, 2–4, 4–6, 6–8 h
Output folder	`outputs/decoded_<date>-<time>`	A new folder

Press **Decode**. It takes a minute or two. The result appears below; an older one can be opened with **Show a decoded folder**.

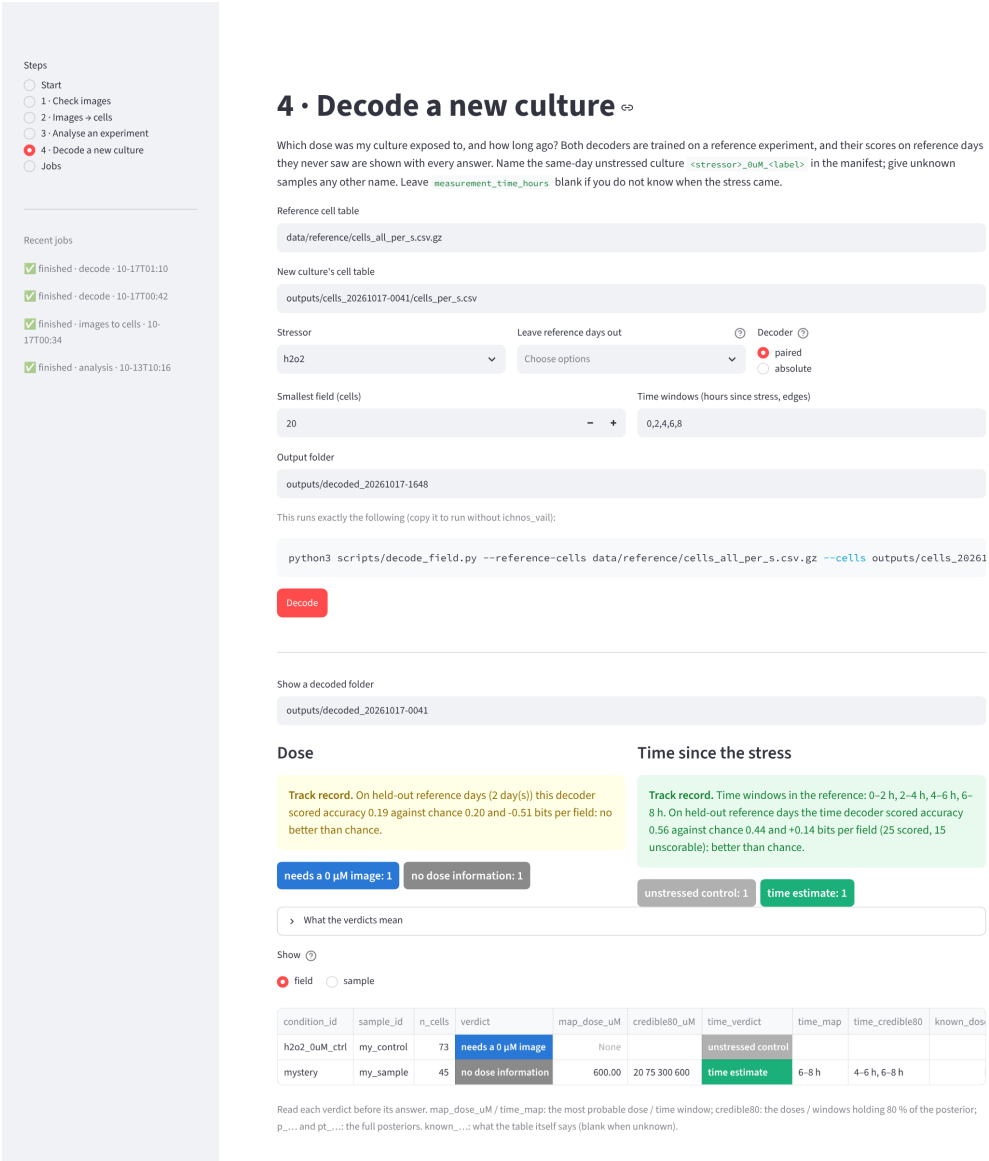

Figure 5.7: Decode a new culture. Each answer has a colour-coded verdict. Here the dose decoder has no track record better than chance, so the dose is not reported; the time decoder beat chance on held-out days and gives a time window.

How to read the result, in this order:

1. **Track record (the coloured box).** How the decoder scored on reference days it never saw. Yellow means no better than chance; green means better than chance.
2. **Verdict counts.** How many fields got each verdict.
3. **The table.** Switch between **field** (one image) and **sample** (the fields of one condition averaged). Read each verdict before its number.

Dose verdict	Meaning	What to do
`dose estimate`	Informative answer from a decoder that beat chance on held-out days	Report `map_dose_uM` with its 80 % credible set `credible80_uM`
`not validated`	The answer points somewhere, but this decoder did no better than chance on held-out days	Do not report a dose
`no dose information`	The image cannot tell the doses apart	Do not report a dose
`unlike reference`	Farther from every reference dose than reference images ever are	Check reporter, settings and stressor; extend the reference
`needs a 0 µM image`	Paired decoder without a same-day control	Image a 0 µM culture on the same day, or use the absolute decoder
`too few cells`	Fewer usable cells than the minimum	Image more cells or a denser field

Time verdict	Meaning	What to do
`time estimate`	Informative answer from a time decoder that beat chance on held-out days	Report `time_map` with `time_credible80`
`not validated`, `no time information`, `unlike reference`	As for the dose	Do not report a time
`too few reporter cells`	Fewer than 5 reporter-positive cells	Image more cells
`unstressed control`	A 0 µM culture has no time since stress	—
`time not available`	The reference cannot train a time decoder	Extend the reference

Column	Meaning
`map_dose_uM`, `time_map`	The most probable dose and time window
`credible80_uM`, `time_credible80`	The doses or windows that hold 80 % of the probability
`p_<dose>uM`, `pt_<window>`	The full probability over doses and windows
`known_dose_uM`, `known_time_h`	What the table itself says; blank when unknown

5.7 Jobs

Every run is a job. The Jobs page lists them all, newest first, with name, state and step. Choose one under **Open** to see its exact commands, its status and its log.

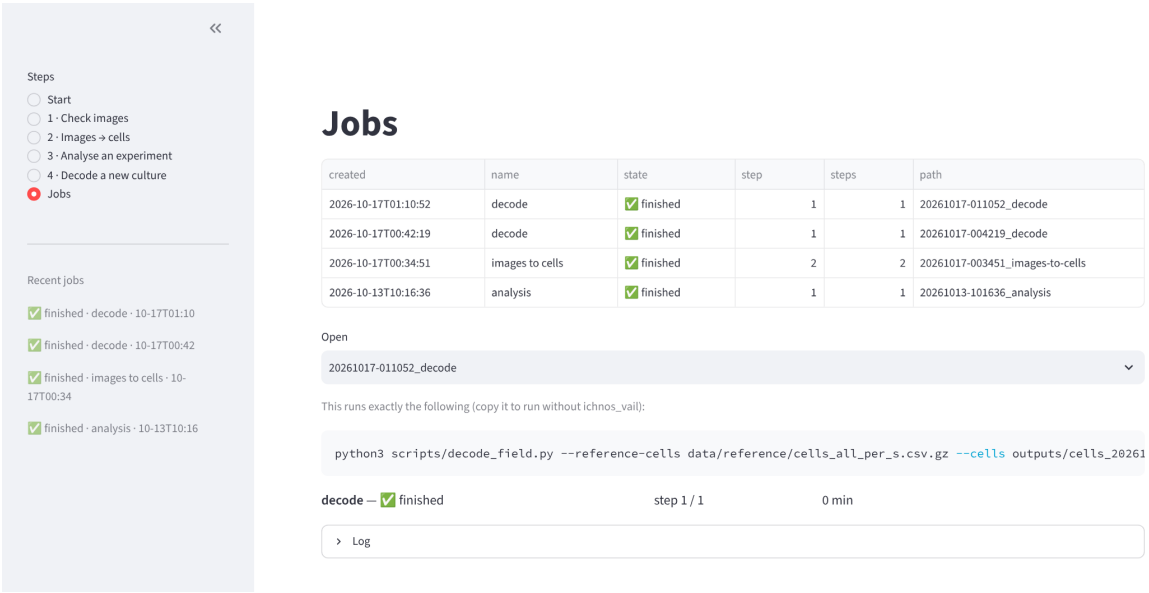

Figure 5.8: Jobs. Each job keeps its commands, status and log on disk.

State	Meaning
created	Written, not started yet
running	Running now; the **Stop** button ends it
stopping	Stop requested
finished	Done; results are in the output folder
failed	A command returned an error; the end of the log says why
stopped	You stopped it
interrupted	Its process is gone, for example after sleep or restart; segmentation can be resumed

Jobs are folders in `outputs/ichnos_vail_jobs/`, each with `job.json` (the commands), `status.json` and `log.txt`.

5.8 Worked example: try it with the team's data

You can try the whole decoding step without your own images. The 7 October H_2O_2 day plays the "new culture": its stressed samples are renamed `sample01` to `sample16` so their doses are hidden, and `key.csv` keeps the answers.

1. In a terminal, make the example:

```
python3 scripts/make_worked_example.py --cells
↪  data/reference/cells_all_per_s.csv.gz --session 20261007_h2o2_tr --out-dir
↪  outputs/example_user
```

2. In ichnos_vail, open **4 · Decode a new culture**.
3. Set the new culture's cell table to `outputs/example_user/cells_per_s.csv`, stressor `h2o2`, and leave out reference day `20261007_h2o2_tr`.

4. Press **Decode** and compare the answers with `outputs/example_user/key.csv`.

What you will see. On the two remaining H_2O_2 days the dose decoder scores at chance (accuracy 0.19 against 0.20). Of 48 fields, 23 are `unlike reference`, 15 `not validated` and 10 `no dose information`; none gets `dose estimate`. The most probable dose would have been right for only 15 % of the hidden fields, so declining to name a dose is the correct answer on these data.

Chapter 6

Command-line use

Every ichnos_vail button runs one of the scripts below, so the command line gives the same results. Use it for scripting, for a server without a browser, or to rerun a command copied from the app. Run each script with `--help` for all its options.

6.1 The four steps

```
# Step 1 (optional): find and align bright-field frames; writes a new manifest
python3 scripts/align_brightfield.py --manifest images.csv --out align/align.csv \
    --write-manifest align/images_aligned.csv --search-up 2

# Step 2: images to cells, in resumable batches, then per-second normalisation
python3 scripts/run_pipeline_batched.py --manifest align/images_aligned.csv
↪  --out-dir run1 --batch-size 20 -- \
    --segmentation-method cellpose --segmentation-source brightfield --cellpose-gpu
    ↪  \
    --bleed-default 0.05 --saturation-value 255
python3 scripts/normalize_exposure.py --cells run1/cells.csv --out
↪  run1/cells_per_s.csv

# Step 3: analyse an experiment (copy examples/analysis_202610.json and set "cells")
python3 scripts/run_analysis.py --config my_analysis.json --out-dir analysis1

# Step 4: decode a new culture against a reference
python3 scripts/decode_field.py --reference-cells
↪  data/reference/cells_all_per_s.csv.gz \
    --cells new_run/cells_per_s.csv --stressor h2o2 --out-dir new_run/decoded
```

6.2 Useful options

Script	Option	Effect
`run_pipeline_batched.py`	`--controls controls.csv`	Use imaged bleed-through controls instead of `--bleed-default`
	`--green-extraction G --red-extraction R`	Channel choice for colour (RGB) files
	`--saturation-value`	Camera maximum: 255 (8-bit), 4095 (12-bit), 65535 (16-bit)

Script	Option	Effect
	`--max-foreground-fraction 0.5`	Accept very dense fields
`decode_field.py`	`--decoder paired / absolute`	With or without the same-day 0 µM
	`--exclude-reference-session <id>`	Leave a reference day out; repeat for several
	`--time-windows 0,2,4,6,8`	Edges of the time windows in hours
	`--min-cells 20`	Smallest usable field
	`--positive-quantile 0.9`	Reporter-positive threshold

6.3 Other scripts

Script	Purpose
`make_worked_example.py`	Turn a reference day into a "new culture" with hidden doses (section 5.8)
`score_segmentation.py`	Score a segmentation method against hand-marked cells in `data/segmentation_reference/`
`check_exposure_linearity.py`	Test the camera background against exposure
`run_protocol.py`	Simulate the ICHNOS model for one stress protocol (dose, times, preincubation or equilibrium)
`run_workflow.py`	The model-based route from images to a dose × time posterior; needs calibration data (chapter 9)
`verify_end_to_end.py`	Synthetic end-to-end check of calibration, image processing and decoding

6.4 Example: simulate the model

```
python3 scripts/run_protocol.py --variant ox --profile default --dose 75
↪  --dose-units uM \
    --times-hours 0.5 1 2 3 --initialization finite-preincubation
    ↪  --preincubation-hours 2 \
    --out-dir outputs/ox_75uM_preincubation_2h
```

The output folder holds `results.csv` (all model outputs), `fluorescence.csv` (green, red and red/green over time, in model units of nM, not camera units), `metadata.json` (parameters, protocol, solver, versions) and `model.sbml`. Stress stays constant unless you add `--clearance-rate-per-hour`.

Chapter 7

Technical Details

This chapter explains how each result is computed, for readers who want to judge it rather than just use it. Every module named here is in the repository with its tests.

7.1 Segmentation and measurement

- **Masks from bright-field, not fluorescence.** Cellpose (default settings) runs on the bright-field frame, and the masks are shifted onto the fluorescence. Against 212 hand-marked cells on 7 crops it scored recall 0.91 and precision 0.90; fluorescence-based masks scored 0.44 and 0.31. A mask found this way does not depend on how bright the reporter is.
- **Per-cell values.** For each cell the pipeline measures mean green and red, subtracts the background, and removes green bleed-through from red. Saturated pixels are checked before extraction.
- **QC.** A usable cell has `qc_pass` true and does not touch the frame edge (`edge_flag` false). Fields where masks cover too much of the frame are refused unless allowed.
- **Exposure normalisation.** Intensities are divided by exposure time, giving grey levels per second (`corrected_mean_green_per_s`, `corrected_mean_red_per_s`). The red/green ratio needs no normalisation.

7.2 Analysis of an experiment

- **Autofluorescence baseline.** The median of usable non-transfected cells of each stressor is subtracted from every transfected cell.
- **Reporter-positive cells.** Green and red are both above the 90th percentile of non-transfected cells of that stressor.
- **Within-day comparisons.** Each dose is compared with the same day's 0 µM, never with a pooled control, because each day used a different transformation.
- **Statistics.** Bootstrap 95 % intervals, resampling cells within fields and fields within time points; the minimum detectable effect; Benjamini–Hochberg q-values across all comparisons. Seeds are recorded, so reruns are identical.
- **Model comparison.** The ICHNOS model is simulated for each dose under constant stress and under first-order clearance at 1 and 3 per hour, and its fold over 0 µM is set beside the measured fold of reporter-positive cells.

7.3 Decoders

The decoders that ichnos_vail offers are **empirical**: they learn from a reference experiment, not from the model.

Decoder	Features of a field	Needs
Paired (default)	Green, red and share of reporter-positive cells, minus the same day's 0 µM at the same time	A 0 µM culture imaged on the same day
Absolute	The same features as they are	One image; sensitive to anything that differs between days
Time	log(red/green) of the reporter-positive cells, plus their green level	At least 5 reporter-positive cells

The red/green ratio of the timer does not depend on how much reporter a cell carries, which is why the time decoder uses it. Each decoder places a Gaussian likelihood around each dose (or time window) with a uniform prior. Its spread combines the variation between days and within a day.

Leave-one-day-out validation. Each reference day is held out in turn. Everything is fitted on the other days, and the held-out day is decoded. The scores are accuracy against chance (1/K for K doses) and information gain in bits: positive is better than guessing, negative is confidently wrong. This is the **track record** shown with every answer.

7.4 How a verdict is chosen

Each field goes through these checks in order, and the first that applies gives its dose verdict:

1. Fewer usable cells than the minimum (default 20) → `too few cells`.
2. Paired decoder and no same-day 0 µM → `needs a 0 µM image`.
3. Farther from every reference dose than the 99th percentile of reference fields (chi-squared distance) → `unlike reference`.
4. Posterior entropy at least 90 % of the prior's → `no dose information`.
5. Track record not above zero bits on held-out days → `not validated`.
6. Otherwise → `dose estimate`.

The time verdict follows the same order. Before those checks, a 0 µM culture gets `unstressed control`, and a reference that cannot train a time decoder gives `time not available`. A held-out reference field whose time window no other day covers is counted as unscorable, never as right or wrong.

7.5 Reproducibility

- Every script refuses to overwrite an existing output folder.
- Every run writes what it was run with: `run.json` or `run-manifest.json`, with the settings, the input sha256, the git commit and the sha256 of every output.
- ichnos_vail passes commands as argument lists, never through a shell, so a path with spaces is one argument and nothing typed is run as a command.
- Every run on the team's data, PASS or FAIL, is recorded in `docs/run_log.md`.

- Automated tests run on every push to GitLab, in three jobs (model, decoders, image and app).

7.6 Libraries used

Library	Purpose
NumPy, SciPy, pandas	Numerics, statistics, tables
scikit-image, tifffile, Pillow	Reading and processing images
Cellpose, PyTorch	Cell segmentation on bright-field
matplotlib	Figures
python-libsbml, tellurium, libRoadRunner	Loading and simulating the SBML model
PyYAML	Parameter profiles
Streamlit (1.50 or later)	The ichnos_vail application
pytest	Automated tests

Chapter 8

Troubleshooting

When something goes wrong, open the job's **Log** first: the job panel shows its last 40 lines, and the Jobs page shows the log of every job. The last lines usually name the cause.

Problem	Solution
`streamlit: command not found`	The environment is not active. Run `source .venv/bin/activate` in the repository folder, then start the app again
Start page shows **Cellpose: missing**	`pip install cellpose` in the active environment, then restart the app
Start page shows **GPU: none found**	Segmentation runs on the CPU and takes several times longer. Untick **Use the GPU** in step 2
`output folder already exists`	Every run writes a new folder. Change the output folder, or keep the suggested one with the current time
`cells.csv already exists: this run is complete`	Step 2 already finished in that folder. Choose a new folder, or use its `cells_per_s.csv`
`file not found` in Check images	Image paths are relative to the manifest's folder. Fix the path or move the manifest
`required column missing`	Add the column to the manifest. It may be empty, but it must exist
`unknown objective`	Use `4X`, `10X`, `40X` or `60X`
Job state **interrupted**	The computer slept or restarted. Start step 2 again with the same output folder; finished batches are kept. On macOS, start the app with `caffeinate -i`
Job state **failed**	Read the end of the log, fix the input, and start again in a new folder
Fields refused for foreground fraction	The slide is dense. Tick **Dense slides** under Advanced in step 2
`lacks [...]; run normalize_exposure.py first`	The cell table is not per second yet. Use `cells_per_s.csv`, not `cells.csv`
`session_id not of the form ...` or `condition_id does not start with ...`	Reference names break the naming rules (section 4.2)
`reference has no <stressor> sessions with non-transfected cells`	The reference needs a non-transfected session of that stressor for the baseline

Problem	Solution
`Cannot read session ids from the reference`	The reference table has no `session_id` column, or it is not a cell table
Every field is `needs a 0 µM image`	Name the same-day control `<stressor>_0uM_<label>`, or choose the absolute decoder
Many fields are `unlike reference`	The new images differ from the reference: check reporter, objective, exposure and stressor
The app is not reachable from another computer	By design: it is served to `localhost` only

Chapter 9

Known Limitations

The tool's limits come from its reference data, not from its design. Every step, decoder and safeguard runs as intended. On the October 2026 data it does what a trustworthy decoder must: it reports the effects the data contain, and it declines to name a dose or a time that the data cannot support. A reference experiment that meets section 9.4 is all it needs to return validated answers, with no change to the code. Full details are in `docs/limitations.md`.

9.1 How the tool behaved on the October 2026 reference

- **It found the effects the data contain.** The within-day test found 12 significant effects among 66 comparisons ($q < 0.05$). Among them is a dose-ordered H_2O_2 response on 7 October: +14 percentage points of reporter-positive cells at 600 µM, with higher red. Every comparison also reports its minimum detectable effect, so a null result says how large an effect it rules out.
- **It refused answers it could not support.** In the worked example (section 5.8) none of 48 fields received a dose estimate. The most probable dose would have been right for only 15 % of the hidden fields, so the abstention was the correct answer. Leave-one-day-out, no decoder beat chance on these data, and the verdicts say so.
- **It caught a model–data mismatch.** The uncalibrated model overpredicts the H_2O_2 response 1.7–5× at 75 µM and above, and predicts no response at the DTT doses used (200 µM or less). This is why the model-based decoder is not yet fully useful to users until the model is recalibrated.
- **The tandem timer remains untested in a high-expression regime, not refuted.** The underlying biological principle requires a robust, sudden wave of reporter synthesis to initiate the timing sequence. Given the subtle stress induction in our current setup, the cells did not produce a sharp enough transcriptional burst to alter the red/green ratio over time (slopes ranged merely from -0.08 to $+0.02$ h^{-1}). To guarantee data integrity when these specific dynamic conditions are not met, our computational decoder is designed to output "no time information" instead of a potentially inaccurate estimate. The algorithm functions exactly as intended, ensuring absolute accuracy while it awaits appropriately optimized and foremost a much bigger number of biological data to deliver precise temporal predictions.
- **Peak times are bounds.** On 7 October the 600 µM red signal was still rising at the last sample (7.6 h), so the tool reports a lower bound, not a peak.

9.2 Measurement and design limits

- **Few independent units**: 3 days for H_2O_2 and 2 for DTT, one culture per dose and day. Each day used a different transformation.

- **Segmentation** misses mainly small dark cells; 6 of 294 fields had no matching bright-field frame and were not analysed.
- **Bleed-through** of green into red is an indicative 0.05: no single-colour controls were imaged.
- **Red exposure linearity** is assumed: red was never imaged at several exposures.
- **No flat-field correction**: uneven illumination adds noise between fields.
- **Weak signals**: the median cell signal is a few grey levels on an 8-bit scale, so single-cell ratios are noisy.

9.3 Software scope

- The **model-based decoders** (`run_workflow.py`) are not offered in ichnos_vail. They need bleed-through controls, a session calibration and at least 3 independent cultures per condition, which these data do not have. On these data the model decoder was confidently wrong.
- Copper ($CuSO_4$) images can be processed, but there is no copper model or decoder.
- ichnos_vail runs on one computer only and is not a multi-user server.
- Passing tests verify software behaviour under controlled assumptions. They do not establish experimental accuracy.

9.4 What would remove these limits

- Single-colour controls, for bleed-through.
- One field imaged at several exposures, for red linearity, and flat-field frames.
- At least 3–4 independent days per stressor, each with its own 0 µM.
- DTT doses around the model's activation range (1 mM or more), or a recalibrated ER module.
- A measurement of how fast H_2O_2 disappears from the medium.
- Reference samples for a fluorescence-to-model calibration.

Chapter 10

Support, Contact and Attribution

Item	Details
Team	iGEM 2026 Patras-Med, University of Patras
Project	ICHNOS : a yeast biosensor that records how much stress a culture received and when
Software design, decisions and review	Marifylli Kathariou (dry lab)
AI assistance	The code was written with AI assistance; see the Attributions page of the team wiki
Repository	gitlab.igem.org/2026/software/patras-med/ichnos-tool
Questions and bug reports	An issue on the GitLab repository
Start the app	`streamlit run app/ichnos_vail.py`
Reference data	`data/reference/cells_all_per_s.csv.gz` (67,580 cells, October 2026)
Further documentation	`docs/user_guide.md`, `docs/architecture.md`, `docs/data_contract.md`, `docs/limitations.md`, `docs/run_log.md`
License	Apache License 2.0

The research repositories behind the tool (`Ichnos_PULSE`, `ICHNOS-ablation`, `ichnos-fisher`, `DryLabTool`) keep the supporting studies. Migrated code and model sources are recorded in `docs/PROVENANCE.md`.

End of Document

ICHNOS Tool v0.1.0 · iGEM 2026 Patras-Med · October 2026