WORKING WITH AI · IMAGE ANALYSIS · BUILD-IT-YOURSELF

Colocalization Analyzer — the prompts, in order

Homework. Ten sends, R0 through R8. Paste each block below into Claude (in Cowork) in order, let it build, look at what it made, then move to the next. You don't need any coding or image-analysis background — Claude writes the code; your job is to describe your science clearly and check that each step makes sense.

What you'll build: the same colocalization analyzer we made in the workshop, feature for feature — a drag-and-drop loader with auto-thresholding, per-channel segmentation and object measurements, the full family of colocalization measures with a significance test and a heatmap, quality control, scale bars, a plain-English summary, exports, and a self-test against known answers. When you finish, it should do everything the reference tool does.

Four habits that make this work

- **Describe before you ask.** The very first prompt is nothing but description. The richer your description, the better everything that follows.
- **One step at a time.** Every block ends with "stop and wait." That one line keeps Claude from racing ahead and building five things at once (it will, otherwise).
- **Verify as you go.** After each step, glance at what it made before continuing. A short check note follows each prompt to tell you what to look for.
- **Rung 5 is two sends.** Ask Claude how to test the result first, let it answer, then send the second block to implement it.

What to have ready: three files, provided with this session. `Session3_Practice_DRG_Nav16_Nav12.tif` is the image (DRG neurons labeled for Nav1.6 and Nav1.2). `..._metadata.txt` carries the pixel size and channel names — load it together with the image. `..._GroundTruth.csv` lists every planted cluster: **don't open it until Rung 8**. Everyone builds on the same image, so we can compare results. Once it works, point it at your own data.

> **No shortcuts.** Build every rung yourself from these prompts — start from a blank page, with no starting file to load. Constructing each piece by hand is the whole point: it's how you come to understand what every number means and why you can trust the finished tool. (Fernando's version exists; you'll compare against it at the end, not build from it.)

RUNG 0 Describe your images (do this first — no building yet)

All description. This is where you hand Claude your biology so everything downstream is grounded in it.

```
I'm going to have you help me build an image-analysis tool step by step,
for a colocalization study. First, here is the full context in four
dimensions - please don't build anything yet.

SAMPLE: dorsal root ganglion (DRG) sensory neurons. Two proteins are
  labeled in two separate fluorescence channels: Channel 1 = the sodium
  channel Nav1.6; Channel 2 = the sodium channel Nav1.2. I want to know
  whether these two channels sit in the same places or in different places.
IMAGING: Leica confocal, 40x / 1.30 NA oil objective, pixel size 0.227
  micrometers/pixel, field ~232 micrometers across. Two channels. I will
  provide a two-channel image (a maximum-intensity projection) + pixel size.
STRUCTURE: the field contains about a dozen large round neuron cell bodies
  (roughly 22-52 micrometers across) on a dark background; most of the frame
  is background. Both sodium channels sit in bright CLUSTERS on and just
  inside the plasma membrane, so the clusters form a rim around each cell,
  with a few scattered inside. Nav1.6 clusters are larger (about 3
  micrometers across) than Nav1.2 clusters (about 2 micrometers). The
  question is whether Nav1.6 and Nav1.2 clusters occupy the same locations
  or separate ones.
QUESTION: measure colocalization between the two channels rigorously - not
  just one correlation number, but a test of whether any overlap is more
  than chance, and a map of where overlap occurs.
```

```
Please read this back to me in your own words and flag anything ambiguous
or missing before we start building.

After completing these instructions, stop and wait for the next rung
before doing anything.
```

✓ **Before you continue:** Claude should restate your setup — two-channel confocal, the two sodium-channel isoforms, the colocalization question. If it misread anything, correct it now.

RUNG 1 Build the loader — and meet the naive trap

Build the whole loader yourself, from a blank page — no template. It ends by showing the simplest colocalization number, which looks convincing but is misleading; that's the lesson. This is the biggest rung; take it in one go so the page is solid.

```
Now let's start building - from a blank page, no template. Make ONE
self-contained HTML page. Inline every library you need (including anything
for reading TIFFs) so it works offline with no internet downloads.

LOADING
1. A drag-and-drop zone (that also opens a file picker on click) accepting
   several files at once: a two-channel image, a metadata .txt, and
   optionally a brightfield/DIC image.
2. Read the image whether it is a PNG/JPG; an RGB TIFF (Red = Channel 1,
   Green = Channel 2); a two-page grayscale TIFF (page 1 = Ch 1, page 2 =
   Ch 2); or a multi-page z-stack (max-project it to one plane per channel).
3. Read the metadata .txt for the pixel size (micrometers/pixel) and the two
   channel names, and fill them in automatically.
4. REQUIRE the pixel size: if none is provided, show a clear warning and do
   NOT display or analyze until it is supplied (I can also type it in a
   field).

CONTROLS & DISPLAY
5. Editable Channel 1 / Channel 2 name fields that relabel everything.
6. A 'swap channels' button, and a 'load demo image' button so I can try it
   with no file.
7. Show each channel as its own grayscale image on separate tabs.

FIRST ANALYSIS
8. Compute the Pearson correlation between the two channels over the WHOLE
   image, and show it with a one-sentence plain explanation.

Show me the two channels and the Pearson number first.

After completing these instructions, stop and wait for the next rung
before doing anything.
```

✓ **Before you continue:** Drag in the image + metadata together; the channel names and pixel size (0.2271 µm/px) fill in, and both channels display. The whole-image Pearson should come back around 0.67 — write it down, it looks like strong colocalization. Try loading with NO metadata: it should refuse and warn.

RUNG 2 Find the objects (segmentation) — with auto-thresholding

Turn the picture into countable objects. Each control is a real decision about what counts as a punctum — and the tool proposes a sensible starting threshold for you.

```
Now find the individual puncta in each channel - segmentation. Add these
controls, and explain each in plain words:
```

```
1. A brightness THRESHOLD per channel (group touching bright pixels into
   blobs).
2. An 'AUTO THRESHOLD' checkbox, ON by default, that sets each channel's
   threshold automatically from the image background - use Otsu's method
   with a background-noise floor. When it's off, I set the thresholds myself.
3. A NOISE FLOOR: discard blobs smaller than a minimum number of touching
   pixels (default 3).
4. A MINIMUM SIZE in square micrometers, using the pixel size (start
   around 0.5 square micrometers - real clusters here are 2-3 micrometers
   across, so anything much smaller is debris).
Then split blobs that are really two touching objects (watershed).

Add a separate black-and-white 'Binary' tab for EACH channel, and show the
object count at each step (above threshold -> after removing noise -> final
objects after splitting).
Show these first so I can check the objects look right.

After completing these instructions, stop and wait for the next rung
before doing anything.
```

✓ **Before you continue:** On load, thresholds are set for you; each channel has its own Binary tab. Toggle Auto threshold off and drag a threshold by hand — watch objects appear and disappear. That's you learning the knob.

RUNG 3 Measure the objects

Basic morphometry, per channel — how many, how big, how far apart.

```
Now measure the objects you found. For each channel report: the number of
objects, the mean object area (square micrometers) and mean diameter
(micrometers), the mean nearest-neighbour distance (how far apart same-type
objects sit, in micrometers), and a clustered / random / evenly-spaced score
(Clark-Evans: below 1 clustered, about 1 random, above 1 evenly spaced).
Also plot the distribution of Channel 1 object sizes. Put it all in a clearly
labeled results panel.

After completing these instructions, stop and wait for the next rung
before doing anything.
```

✓ **Before you continue:** Per-channel counts, mean sizes, and spacing appear. Concrete check: Nav1.6 clusters should average about 3 µm across and Nav1.2 about 2 µm. If your numbers are far off, your pixel size is probably wrong — fix that before going on.

RUNG 4 Measure colocalization properly — four ways

The heart of it. Notice how restricting Pearson to real signal changes the answer.

```
Now measure colocalization properly - four measures, each explained briefly:

1. Recompute Pearson, but ONLY over pixels that belong to detected signal
   in either channel (not the whole image). Show it next to the whole-image
   Pearson from Rung 1, and explain why the whole-image value was inflated
   by the shared dark background.
2. Li's ICQ (intensity correlation quotient) as a second signal-based number
   (roughly -0.5 to +0.5; positive means the two tend to be bright together).
3. Manders' coefficients M1 and M2: the fraction of Channel 1 signal that
   overlaps Channel 2, and the reverse (they can differ).
4. Object-based: for each Channel 1 object, is there a Channel 2 object
   within a distance I can set (default 0.6 micrometers)? Report the
```

```
   percentage of Channel 1 objects that have a Channel 2 partner. Draw an
   overlay: Channel 1 objects circled GREEN if they have a nearby Channel 2
   object, RED if not; mark Channel 2 objects with a cross. Put a small
   legend on the image explaining the colors.

Show the overlay and the numbers.

After completing these instructions, stop and wait for the next rung
before doing anything.
```

✓ **Before you continue:** The signal-only Pearson should collapse to roughly zero (about −0.05) from the 0.67 you wrote down at Rung 1. Same image, opposite conclusion — that drop is the whole point. Then read the green/red overlay against its legend.

RUNG 5 · BEAT A Ask Claude how to test it (send this alone, let it answer)

You're asking for the method, not the code. This is the habit of a critical user.

```
Before I trust these colocalization numbers, I want to know whether any
overlap I see is real or just what you'd expect by chance (two channels
that are both busy will overlap somewhat even if unrelated). What is the
best way to test whether the colocalization is more than chance, and how
would I verify it? Propose the method before writing any code.

After completing these instructions, stop and wait for the next rung
before doing anything.
```

✓ **Before you continue:** Claude should propose a randomization test — shuffle the object positions many times and see how often chance produces as much overlap as you actually see. Read its reasoning before the next send.

RUNG 5 · BEAT B Now implement what it proposed

Turn the agreed method into code.

```
Good. Now implement that: repeatedly randomize the positions of the objects
within the cell area, measure the colocalization each time to build a
'by chance' distribution, and compare it to the real observed value. Report
the observed value, the chance level, and a p-value (how surprising the real
value is). State plainly whether the colocalization is above chance or not.

After completing these instructions, stop and wait for the next rung
before doing anything.
```

✓ **Before you continue:** You get: observed overlap, chance level, a p-value, and a plain verdict. On this image expect a decisive answer — roughly 70% of Nav1.6 clusters have a Nav1.2 neighbour within 0.6 µm, against only a few percent by chance ($p < 0.001$). Note that the raw 70% meant nothing until you had the chance level to compare it against. On other data 'not above chance' is an equally real result, not a failure.

RUNG 6 Show WHERE they overlap — a colocalization map

A picture of colocalization, pixel by pixel.

```
Now show me WHERE the channels overlap, as a heatmap on its OWN tab. For
each pixel in the detected signal, compute how much the two channels rise
and fall together (Li's Intensity Correlation Analysis - the product of each
channel's difference from its own average). Display it as a map: WARM colors
where the two channels are bright together (colocalized), COOL colors where
one is bright and the other dark (mutually exclusive), dark where neutral.
Compute it only over the detected signal, not the background. Add a color
calibration bar on the map so I can read it.
```

```
After completing these instructions, stop and wait for the next rung
before doing anything.
```

✓ **Before you continue:** A heatmap appears. Mostly warm = strong colocalization; a mix of warm and cool = weak. It should agree with the numbers from Rung 4.

RUNG 7 Finishing layer — QC, scale bars, summary, and exports

The polish that makes the tool honest, readable, and shareable by anyone.

```
Finally, add the finishing layer so someone without an imaging background
can use this and share the results:

1. Quality checks using the pixel size: is the image finely enough sampled
   (Nyquist)? what is the signal-to-noise ratio? are any pixels saturated?
   is there sign of bleed-through between the two channels?
2. A distance SCALE BAR on every image tab, auto-sized to a round number
   (e.g. 2, 5, 10, 20 micrometers) from the pixel size.
3. An 'Analysis summary' in plain English that reads off the numbers and
   tells me: what the image shows, what it CANNOT tell me (resolution
   limits), how the image could be improved, and what to analyze next.
4. Export buttons: save the current view as a PNG, save all tabs as PNGs,
   save a full HTML report of the whole analysis, and save the per-object
   measurements as a CSV.

After completing these instructions, stop and wait for the next rung
before doing anything.
```

✓ **Before you continue:** A QC panel, scale bars on every tab, a written summary, and working export buttons (PNG / report / CSV). Read the summary out loud — if it makes sense to you, the tool is doing its job.

RUNG 8 Prove the tool works — test on known answers, and report

The whole tool is built. Before trusting it on the real image, check it on data where you already know the truth.

```
Now prove the whole tool works, on data where we already know the answer.
Generate a synthetic two-channel image with KNOWN planted objects: a set
number of Channel 1 objects, a set fraction of them placed on a Channel 2
object (colocalized) and the rest placed independently, plus some Channel
2-only objects and scattered debris. Keep a record of the planted truth.

Run the complete pipeline on this synthetic image and compare its output to
the planted truth. Report a validation table: objects planted vs detected
in each channel; colocalized objects planted vs correctly called; precision,
recall, and F1 for the colocalization calls; and matched / missed /
false-positive counts. Then generate a downloadable validation report (HTML)
I can archive with the analysis.

After completing these instructions, stop and wait for the next rung
before doing anything.
```

✓ **Before you continue:** You get a validation table (precision, recall, F1; matched / missed / false-positive) and a report file. It won't be perfect — and the gaps are the lesson: they show you exactly where the tool is conservative or over-eager. This is the bar for every tool: recover known answers before you trust unknown ones.

Now open the ground truth. The practice image ships with `..._GroundTruth.csv` — every cluster that was planted in it, with position, size and whether it was given a partner. Check your detections against it. One thing will look wrong: it lists 430 Nav1.6 and 1,150 Nav1.2 clusters, but a correct analysis finds only about 235 and 360 objects. Your tool is not

broken — clusters that touch get merged into one object. That gap between ‘what is there’ and ‘what a method counts’ is worth sitting with: it is why watershed splitting exists, and why any count you publish belongs to the method as much as to the biology.

That’s the analyzer.

Compare what you built to the finished version Fernando will send — they should do the same things. The reason to build it yourself is to see exactly how each number is made, so you can trust it, question it, and adjust it. If a step goes sideways, tell Claude what you expected versus what you saw and ask it to fix that one thing — don’t start over.

Two rules of thumb as you go: if an overlay marks something a biologist wouldn’t, the tool (not your eye) is wrong; and one image is never a validation — once it works, run it the same way on several images before you believe a result.