Yin & Yang: Demonstrating complementary provenance from noWorkflow & YesWorkflow

João F. Pimentel, Saumen Dey, Timothy McPhillips,
Khalid Belhajjame, David Koop, Leonardo Murta,
Vanessa Braganholo, **Bertram Ludäscher**

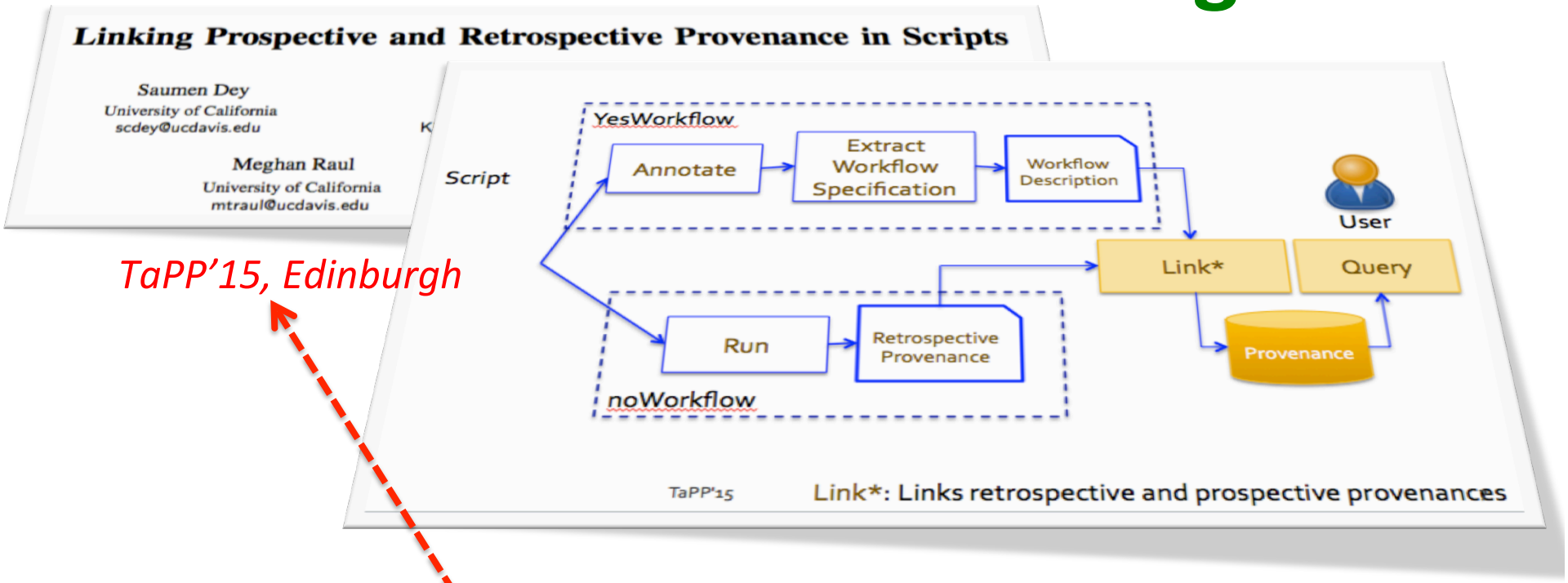
Instituto de Computação

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

DAUPHINE
UNIVERSITÉ PARIS

UNIVERSITY OF MASSACHUSETTS DARTMOUTH 1895

Provenance of the Yin & Yang Demo



TaPP'15, Edinburgh

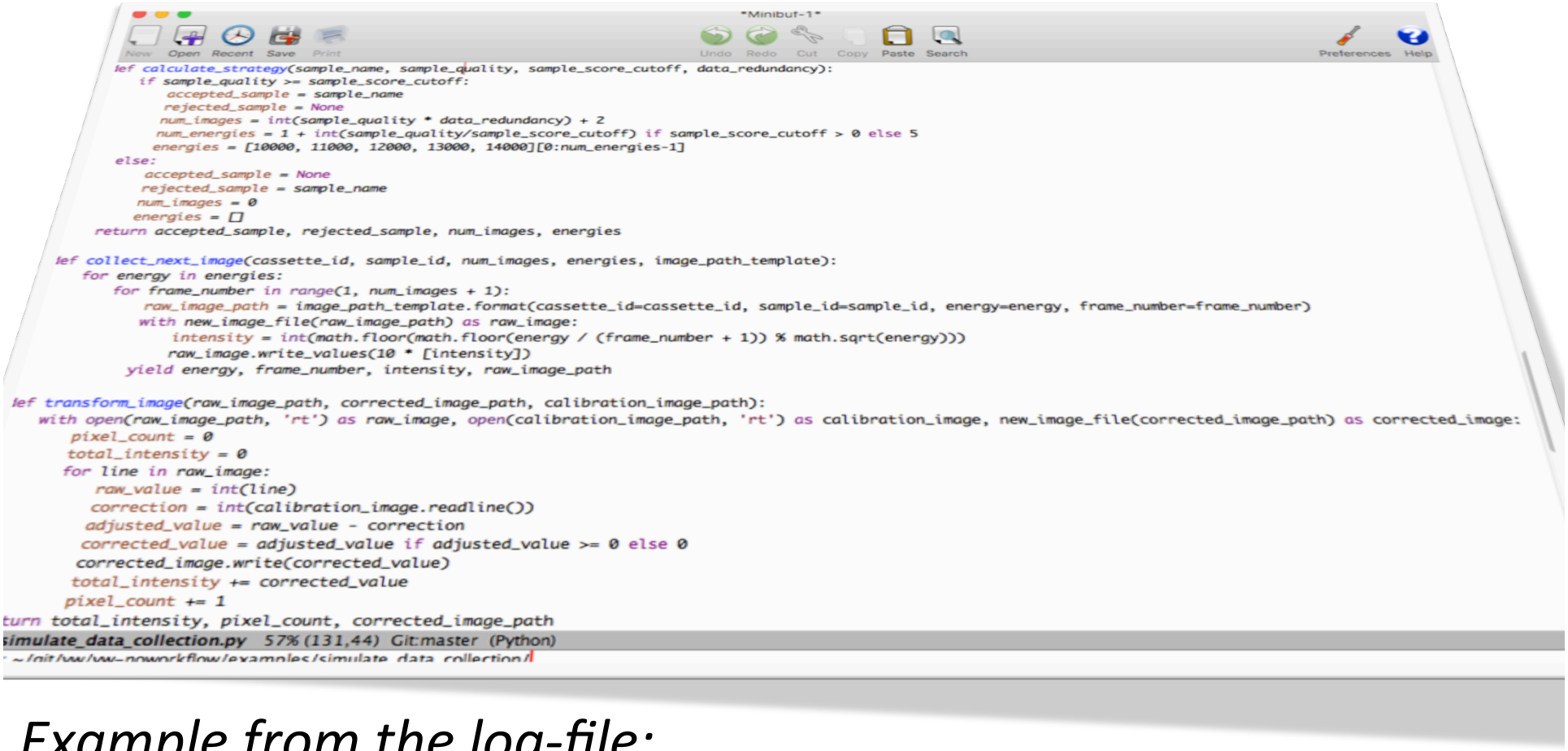
SCHLOSS DAGSTUHL
Leibniz-Zentrum für Informatik

Dagstuhl'16
Reproducibility Seminar

was_derived_from__via

Provenance-Week'16 Demo!

Using Provenance from Script Runs



```
def calculate_strategy(sample_name, sample_quality, sample_score_cutoff, data_redundancy):
    if sample_quality >= sample_score_cutoff:
        accepted_sample = sample_name
        rejected_sample = None
        num_images = int(sample_quality * data_redundancy) + 2
        num_energies = 1 + int(sample_quality/sample_score_cutoff) if sample_score_cutoff > 0 else 5
        energies = [10000, 11000, 12000, 13000, 14000][0:num_energies-1]
    else:
        accepted_sample = None
        rejected_sample = sample_name
        num_images = 0
        energies = []
    return accepted_sample, rejected_sample, num_images, energies

def collect_next_image(cassette_id, sample_id, num_images, energies, image_path_template):
    for energy in energies:
        for frame_number in range(1, num_images + 1):
            raw_image_path = image_path_template.format(cassette_id=cassette_id, sample_id=sample_id, energy=energy, frame_number=frame_number)
            with new_image_file(raw_image_path) as raw_image:
                intensity = int(math.floor(math.floor(energy / (frame_number + 1)) % math.sqrt(energy)))
                raw_image.write_values(10 * [intensity])
            yield energy, frame_number, intensity, raw_image_path

def transform_image(raw_image_path, corrected_image_path, calibration_image_path):
    with open(raw_image_path, 'rt') as raw_image, open(calibration_image_path, 'rt') as calibration_image, new_image_file(corrected_image_path) as corrected_image:
        pixel_count = 0
        total_intensity = 0
        for line in raw_image:
            raw_value = int(line)
            correction = int(calibration_image.readline())
            adjusted_value = raw_value - correction
            corrected_value = adjusted_value if adjusted_value >= 0 else 0
            corrected_image.write(corrected_value)
            total_intensity += corrected_value
            pixel_count += 1
    return total_intensity, pixel_count, corrected_image_path
```

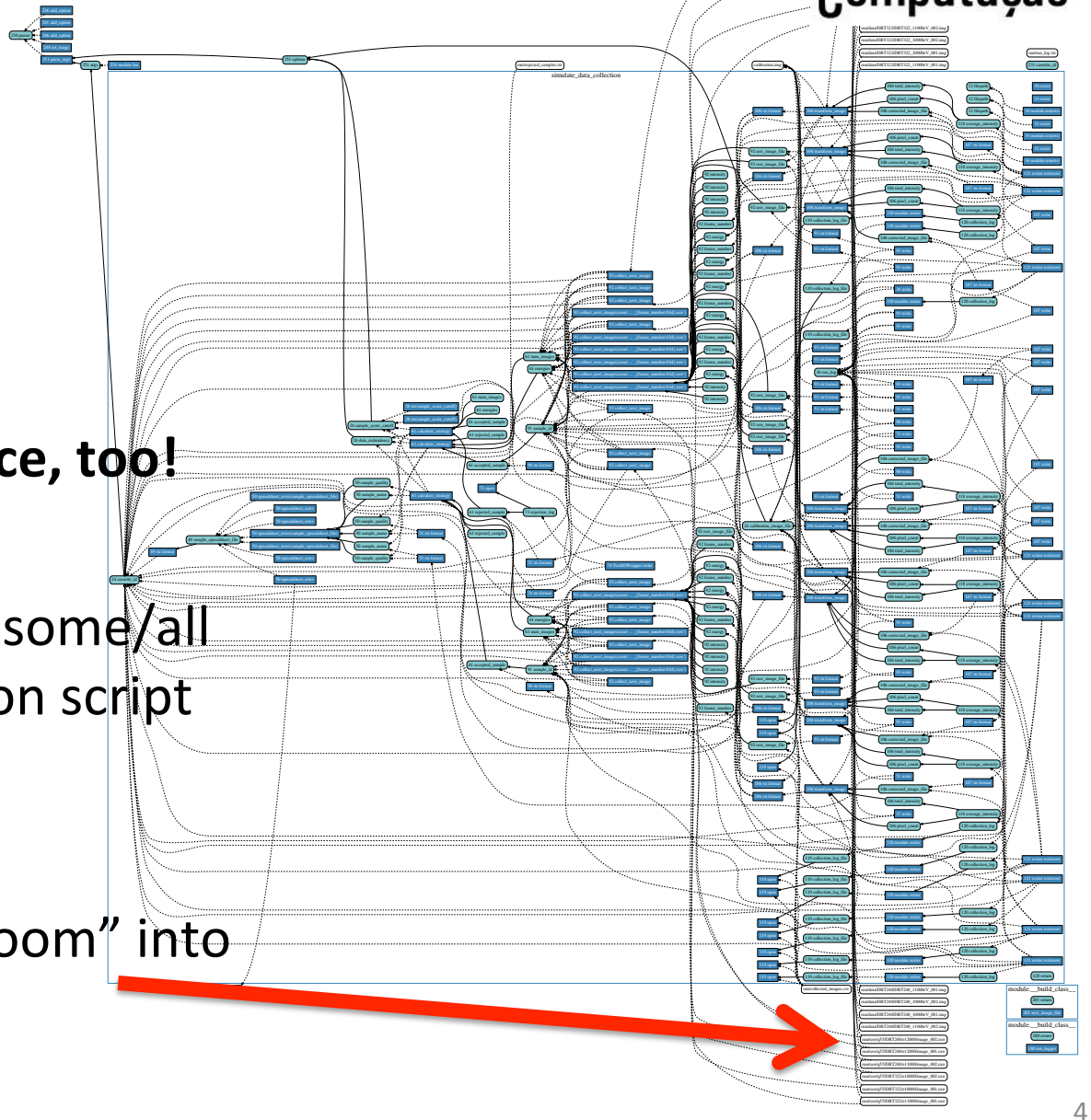
Example from the log-file:

`2016-06-07 20:32:36 Wrote run/data/DRT240/`**`DRT240_11000eV_002.img`**

But how was that image derived?? *(“Provenance for Self!”)*

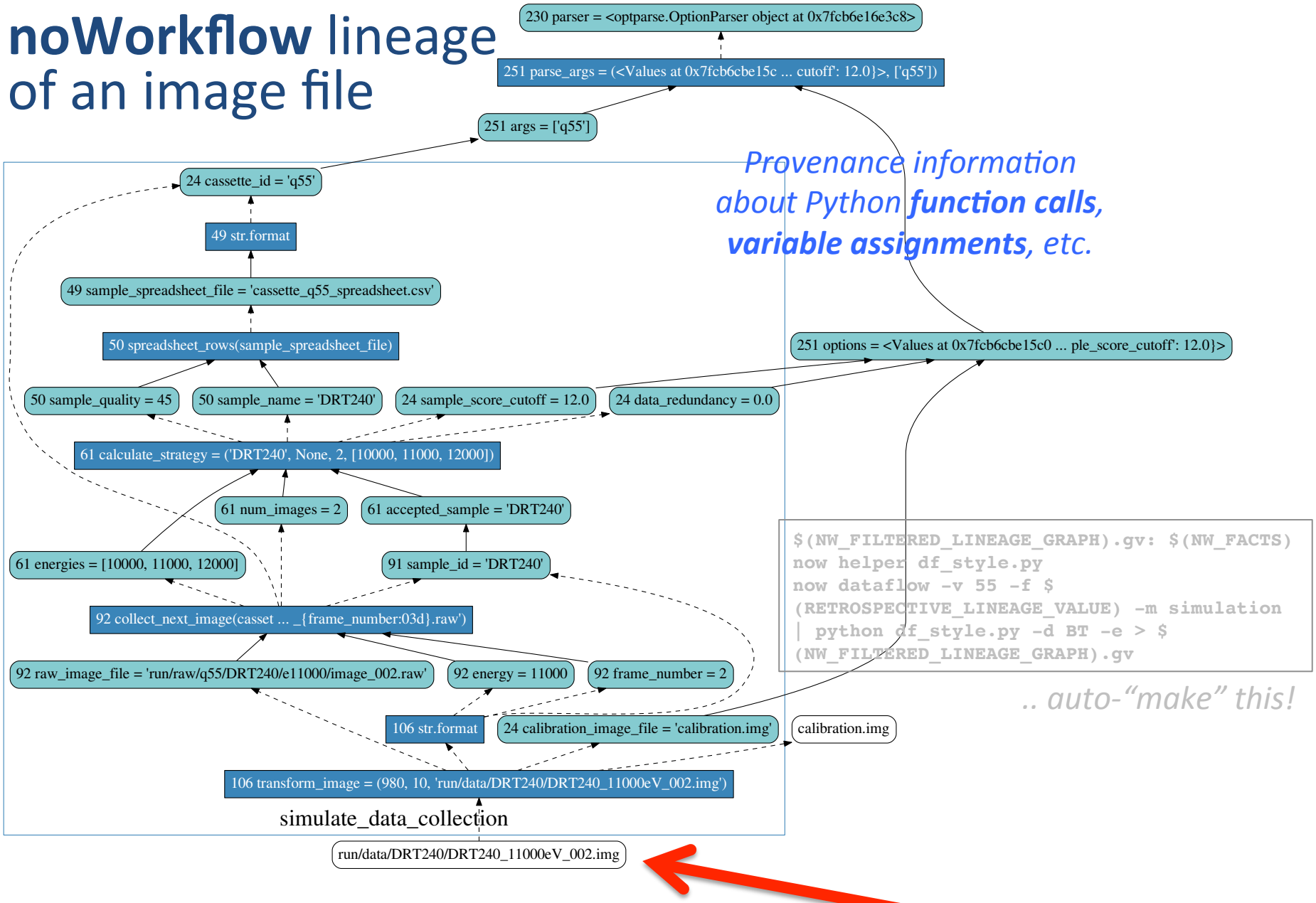
noWorkflow: not **o**nly **Workflow!**

Instituto de Computação



- **Scripts have provenance, too!**
- **Transparently** capture some/all provenance from Python script **runs**.
- Use filter **queries** to “zoom” into relevant parts ..

noWorkflow lineage of an image file



*Provenance information about Python **function calls**, **variable assignments**, etc.*

```
$(NW_FILTERED_LINEAGE_GRAPH).gv: $(NW_FACTS)
now helper df_style.py
now dataflow -v 55 -f $
(RETROSPECTIVE_LINEAGE_VALUE) -m simulation
| python df_style.py -d BT -e > $
(NW_FILTERED_LINEAGE_GRAPH).gv
```

.. auto-"make" this!

$ **now** dataflow -f "run/data/DRT240/**DRT240_11000eV_002.img**"

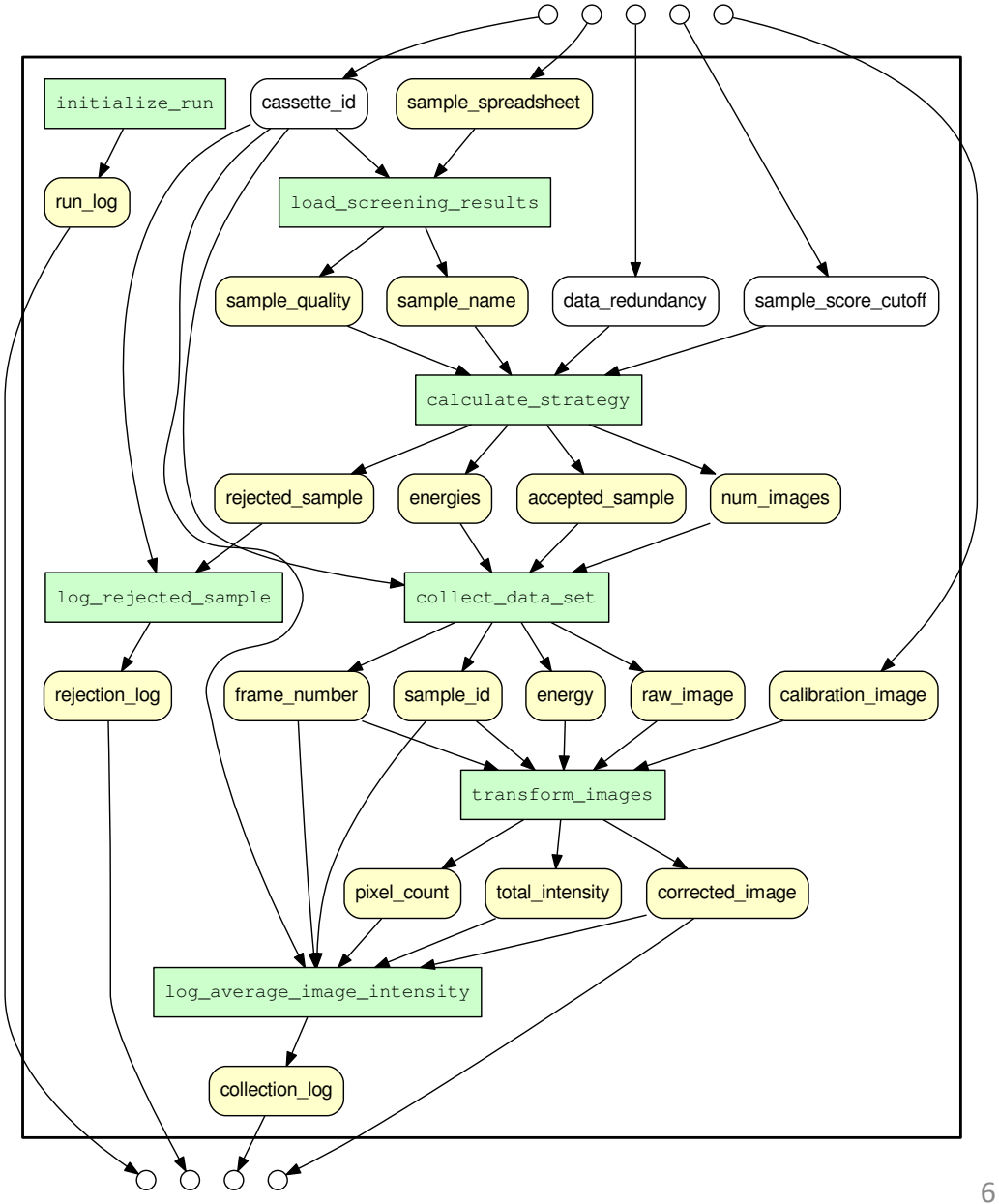
YesWorkflow: Yes, scripts are Workflows, too!

- Use YW annotations `@begin...@end, @in, @out` to **reveal hidden conceptual workflow**
  (**prospective** provenance)

- Script isn't changed:
  - annotations via **comments** (=> **language independent**)

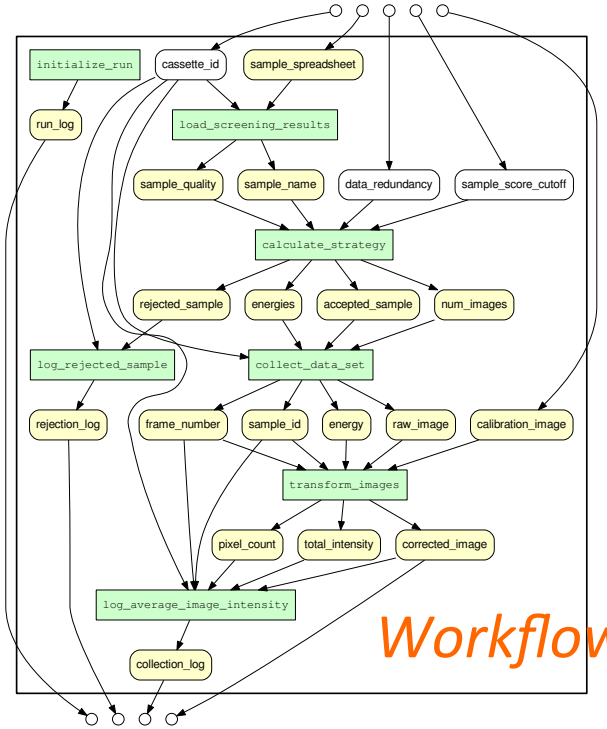
- For understanding and sharing the "big picture"

- **Query** and visualize!

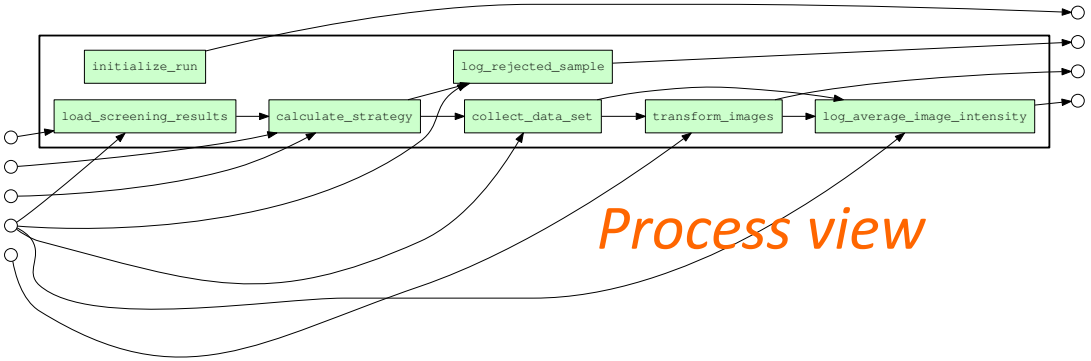


ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Alternate YW Views



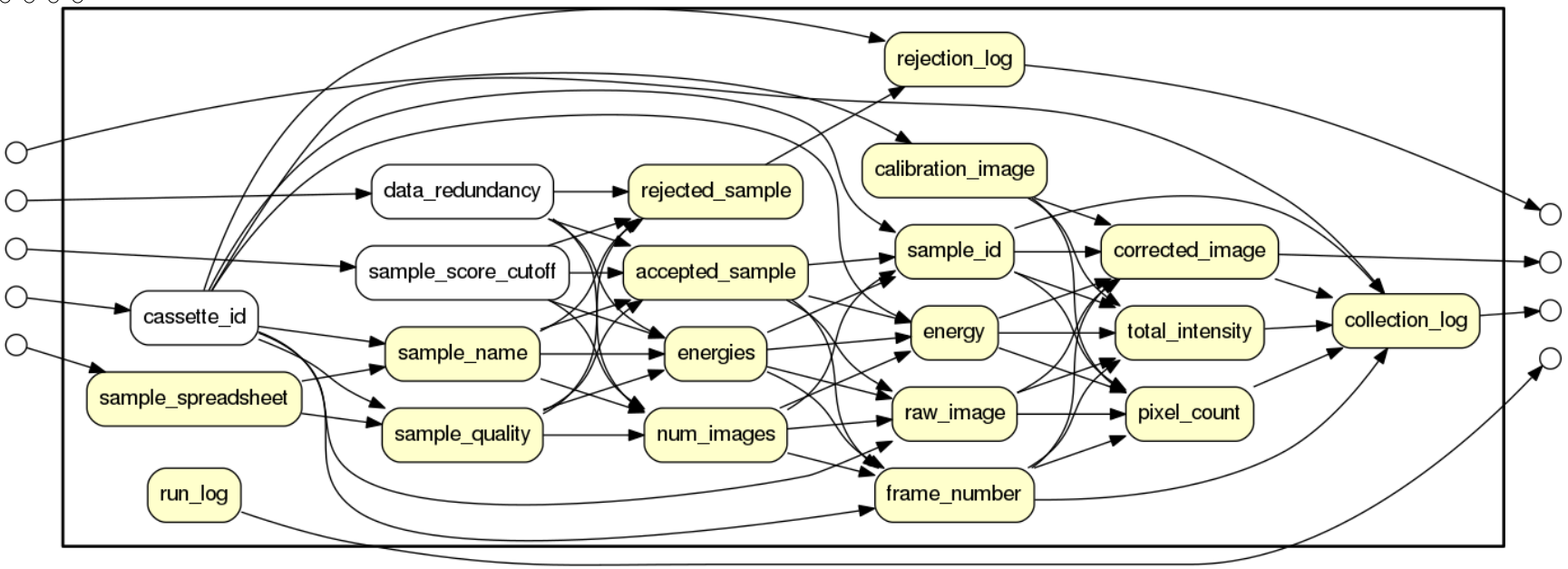
Workflow view



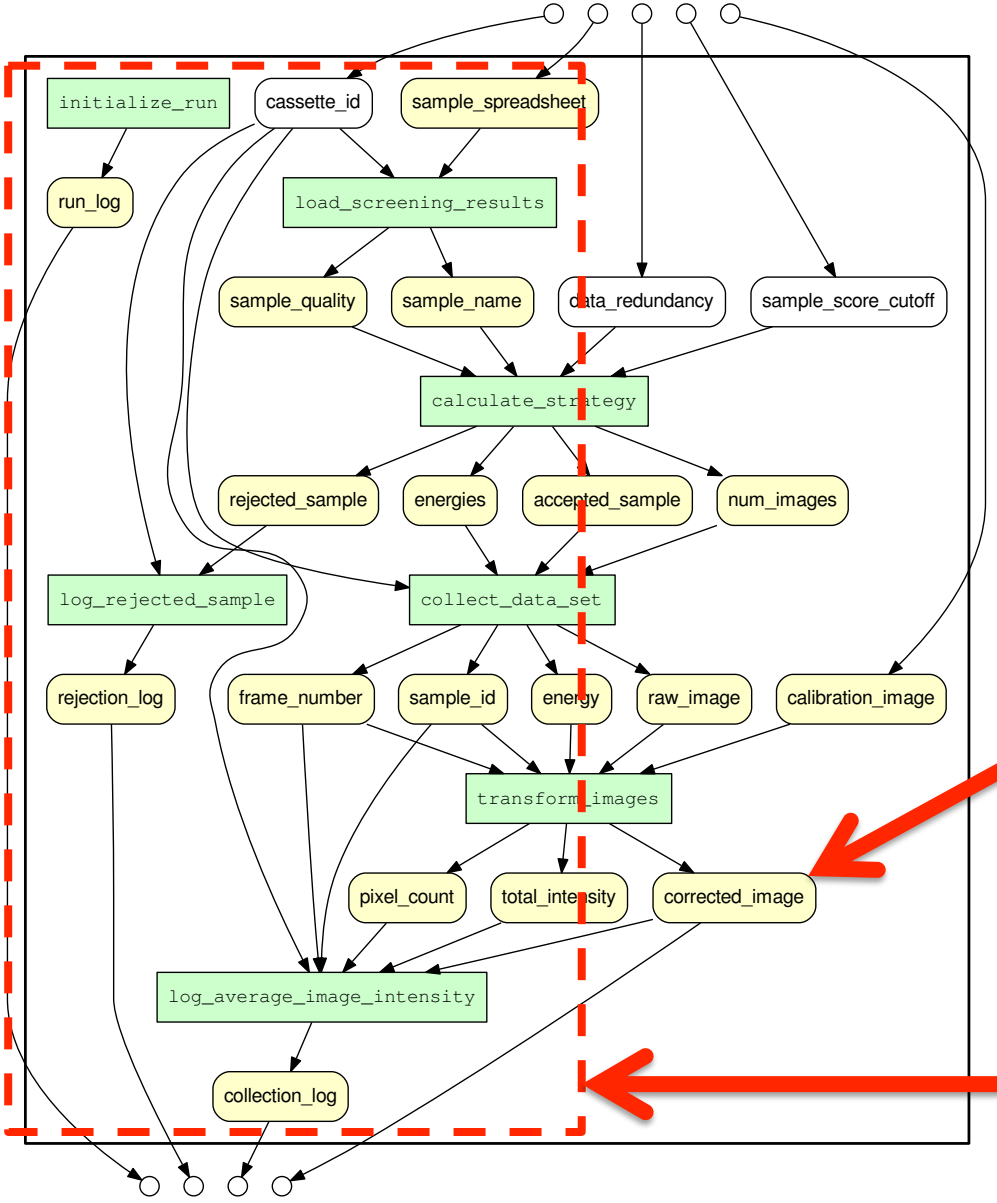
Process view

Data view

simulate_data_collection



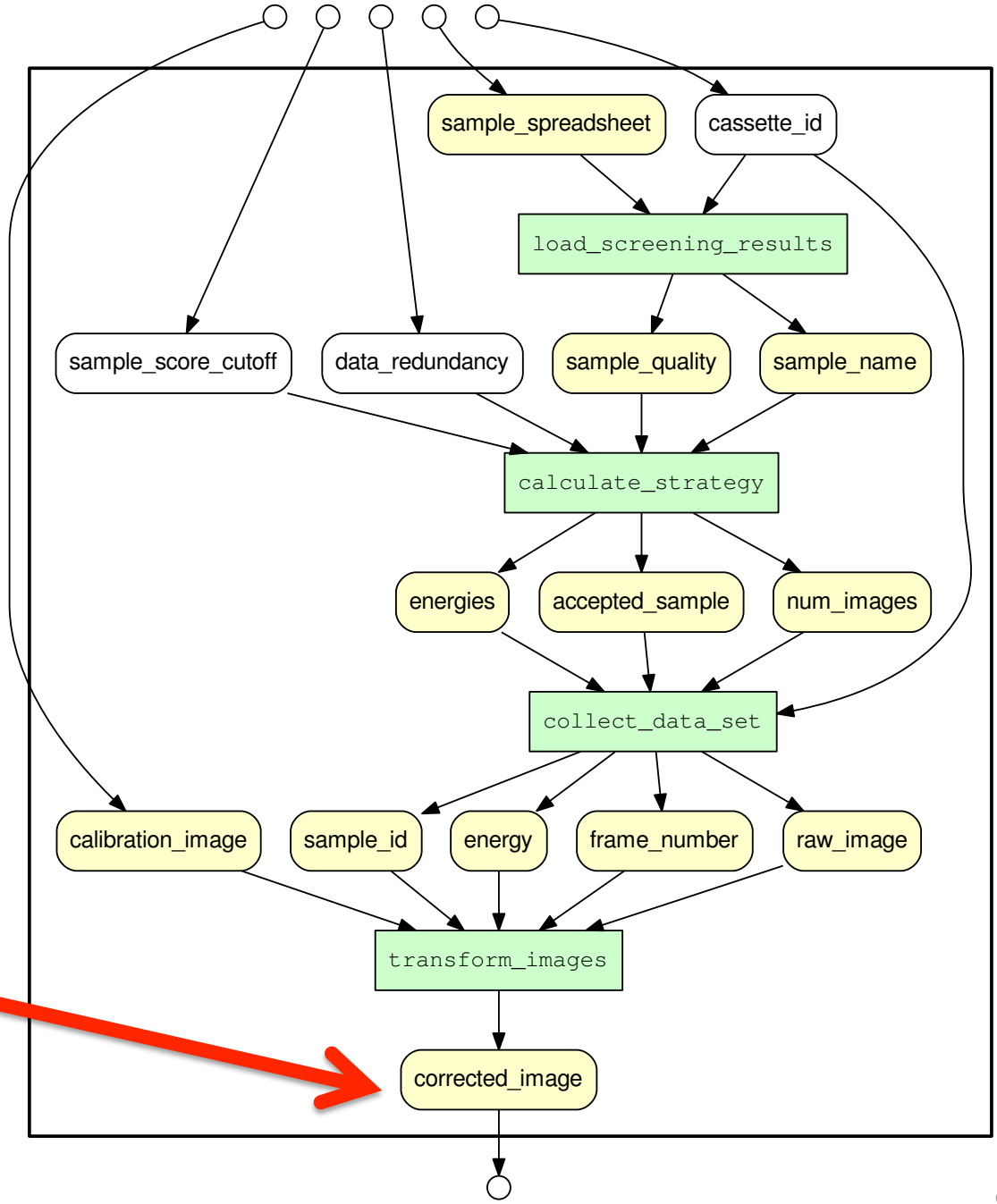
What is the **lineage** of "`corrected_image`"?



From here on "upwards":
What led (leads) to this?

*.. and what is **irrelevant** and should be **pruned?**?*

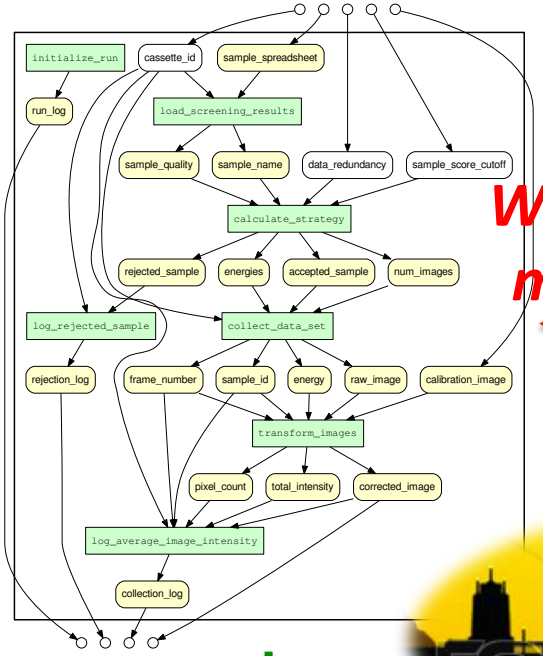
Subgraph resulting from **lineage query** on YW workflow model

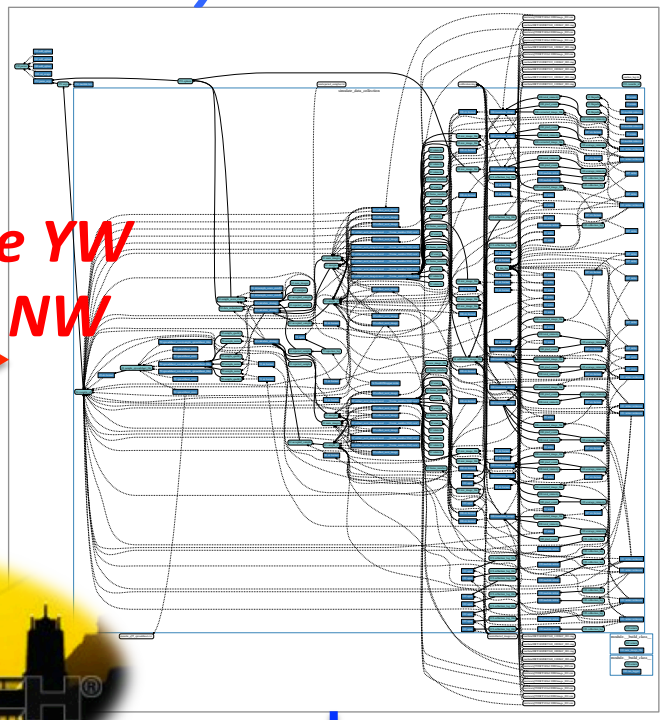


What is the lineage of `corrected_image`***?***

YesWorkflow: *Conceptual* workflow model

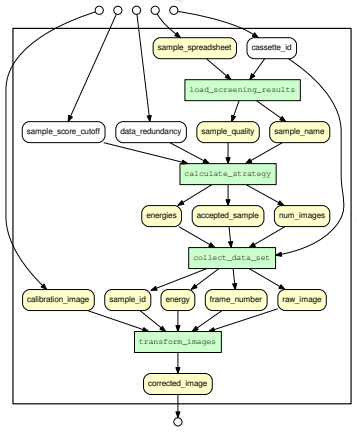
noWorkflow: *Python* trace model





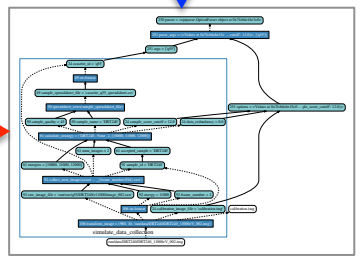
Would like to use YW model to query NW data!

lineage query



But how do we bridge this gap???

lineage query



We're off to see the Wizard of Prov …

- Enrich YW conceptual view with NW Python provenance!
- Get the best of both worlds!
- ***How hard can it be to bridge YW and NW …***
  (cf. TaPP'15 prototype)

We're off to see the Wizard,
The wonderful Wizard of Prov!
--
We hear he is a wiz of a wiz
If ever a wiz there was.
--
If ever, oh ever, a wiz there was,
The Wizard of Prov is one because,
Because, because, because, because, because,
Because of the wonderful things he does.

Diamonds are forever
Bridges aren't …

… new bridge-building can be stressful

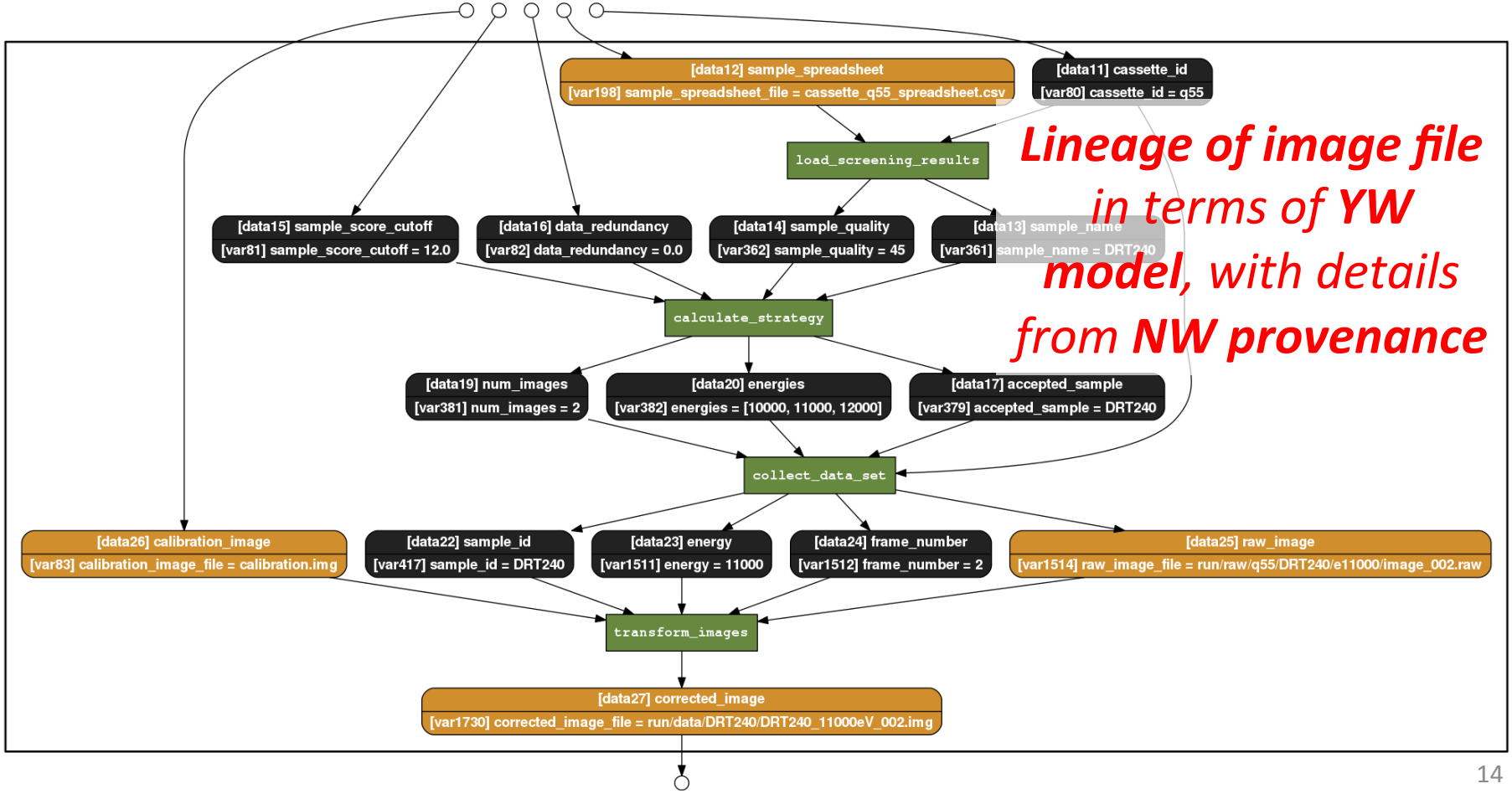
… even if just painting over.

Habemus Pons!

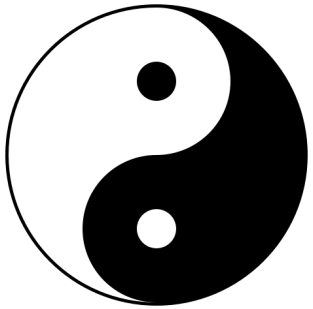
We've got the Bridge!

The bridge is the journey..

(The journey is the destination)



Lineage of image file *in terms of* ***YW model****, with details from* ***NW provenance***



~~Secret~~ Reproducible Sauce

- Combining provenance information from noWorkflow and YesWorkflow
- Using all the good stuff:
  - make, docker, Prolog, SQL, Graphviz
- Open source
  - github.com/yesworkflow-org/yw-noworkflow
  - github.com/gems-uff/yin-yang-demo
- Have a closer look at the demo!