



FOUNDATIONS FOR AI IN THE AUTOMATED LAB

FUTURE LABS AUTOMATION & TECHNOLOGY WEST
NOVEMBER 2025

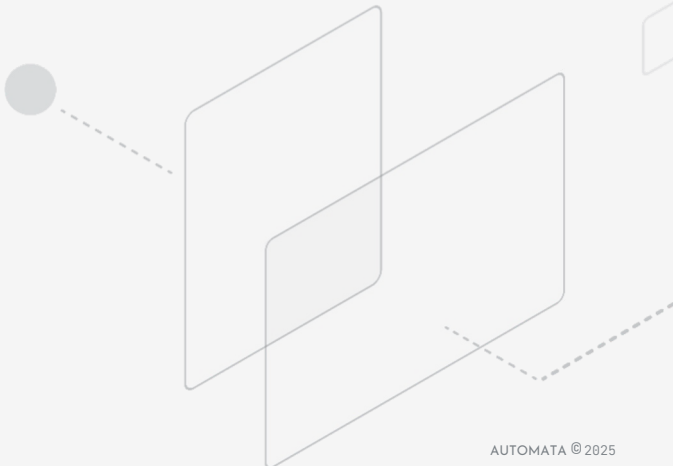
JESSE MAYER



01 AI IN LAB AUTOMATION TODAY

02 LIMITATIONS AND SOLUTIONS

03 DISCUSSION/Q&A



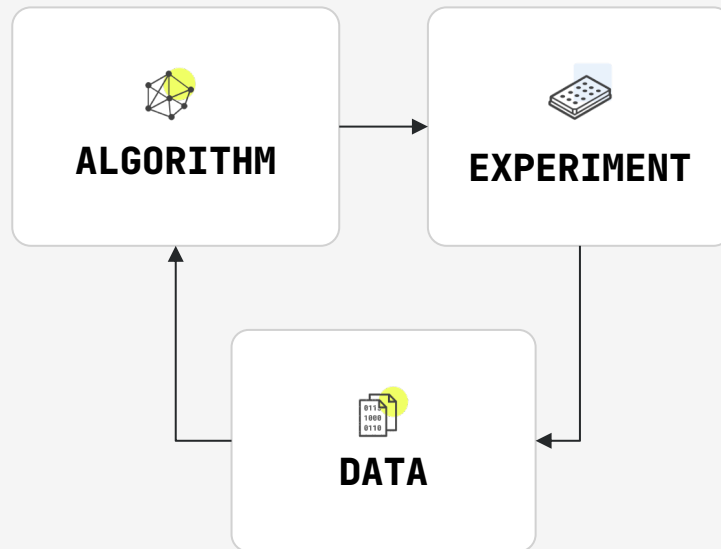


AI HYPE IS OFF THE CHARTS.

AI HYPE IS OFF THE CHARTS

"GET MORE OF THE AI"

“GO GET MORE CHATGPT LICENSES”



THE RIGHT AI FOR THE JOB

01

REACTIVE ML

Call results that can not
be defined objectively

02

PROACTIVE ML

Identify issues and
present findings beyond
the hypothesis

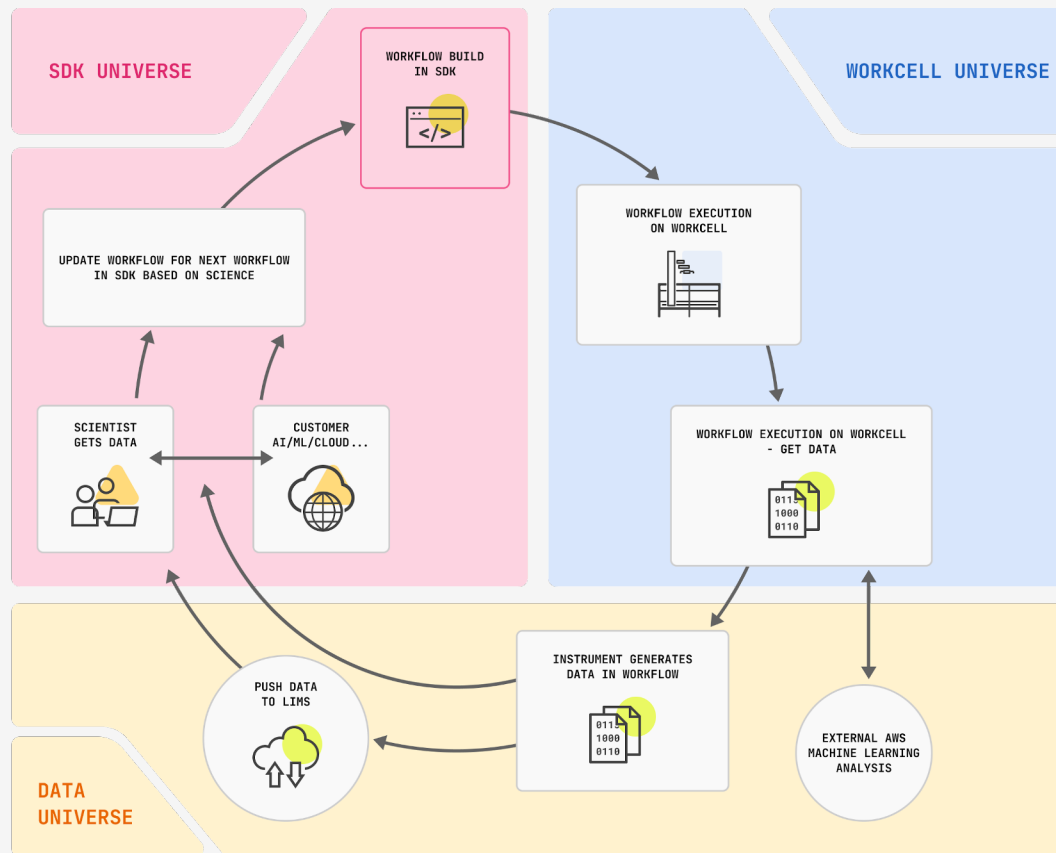
03

GENERATIVE AI (LLMs)

Build SOPs, summarize
data, close the loop

REACTIVE ML

- Train based on the characteristics of compounds or results that have proven successful in the past
- Can be used for hit-picking, process optimization, or experimental iteration
- Comes in the form of DoE and analytical studio software



PROACTIVE AI

- Train based on all data that has been produced on the system, both scientific and non-scientific
- Identify patterns that associate with any outcome
- Present these to the scientist/operator before they ask

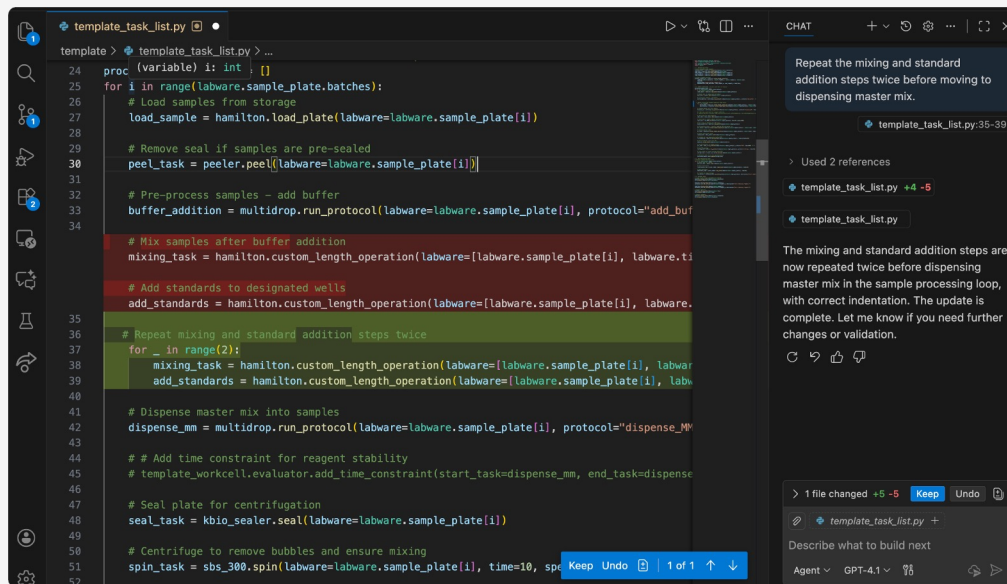


"Std_ELISA" usually errors when the plate in position C2 is blue. Do you want to continue?"

Revvity

GENERATIVE AI

- Build workflows from SOPs
- Summarize data, based on common scientific principles or previous data
- Suggest future areas of investigation or the next iteration in an experiment



The screenshot displays a code editor with a Python script named `template_task_list.py`. The script defines a function `template_task_list` that takes a variable `i` as an integer. It performs a series of steps for each sample in a plate, including loading, peeling, buffer addition, mixing, standard addition, and dispensing. The script is annotated with comments in various colors. To the right of the code editor is a chat window titled "CHAT" with a message from the AI agent: "Repeat the mixing and standard addition steps twice before moving to dispensing master mix." Below the chat window, a status bar indicates "1 file changed +5 -5" and "Keep Undo".

```
template > template_task_list.py > ...
24 proc (variable) i: int | {}
25 for i in range(labware.sample_plate.batches):
26     # Load samples from storage
27     load_sample = hamilton.load_plate(labware=labware.sample_plate[i])
28
29     # Remove seal if samples are pre-sealed
30     peel_task = peeler.peel(labware=labware.sample_plate[i])
31
32     # Pre-process samples - add buffer
33     buffer_addition = multidrop.run_protocol(labware=labware.sample_plate[i], protocol="add_buf
34
35     # Mix samples after buffer addition
36     mixing_task = hamilton.custom_length_operation(labware=[labware.sample_plate[i], labware.ti
37
38     # Add standards to designated wells
39     add_standards = hamilton.custom_length_operation(labware=[labware.sample_plate[i], labware.
40
41     # Repeat mixing and standard addition steps twice
42     for _ in range(2):
43         mixing_task = hamilton.custom_length_operation(labware=[labware.sample_plate[i], labwar
44         add_standards = hamilton.custom_length_operation(labware=[labware.sample_plate[i], labw
45
46     # Dispense master mix into samples
47     dispense_mm = multidrop.run_protocol(labware=labware.sample_plate[i], protocol="dispense_MM
48
49     # Add time constraint for reagent stability
50     # template_workcell.evaluator.add_time_constraint(start_task=dispense_mm, end_task=dispense
51
52     # Seal plate for centrifugation
53     seal_task = kbio_sealer.seal(labware=labware.sample_plate[i])
54
55     # Centrifuge to remove bubbles and ensure mixing
56     spin_task = sbs_300.spin(labware=labware.sample_plate[i], time=10, speed=1000)
```

Repeat the mixing and standard addition steps twice before moving to dispensing master mix.

template_task_list.py:35-39

Used 2 references

- template_task_list.py +4 -5
- template_task_list.py

The mixing and standard addition steps are now repeated twice before dispensing master mix in the sample processing loop, with correct indentation. The update is complete. Let me know if you need further changes or validation.

> 1 file changed +5 -5 Keep Undo

template_task_list.py +

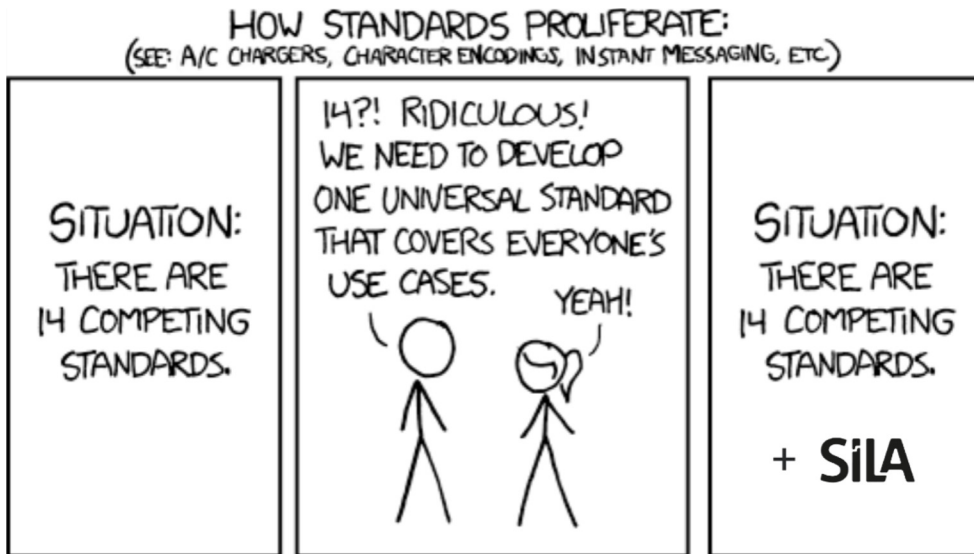
Describe what to build next

Agent GPT-4.1

LIMITATIONS

DRIVERS/APIs

- Our primary connection to the device producing the data
- Often closed-source, with both instrument and scheduler vendors to blame
- Open standards, while ideal, have consistently failed in this space



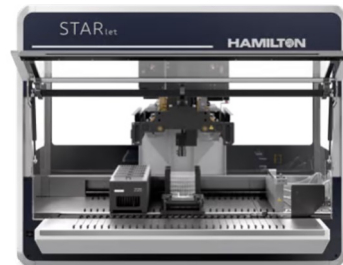
xkcd, David Dambman

LIMITATIONS: DRIVERS/APIs

API AI READINESS	FEATURES
★★★★	Robust documentation. As low level access as is permissible/safe. Data can be streamed/subscribed to. Interactive error recovery.
★★★	Errors are clear, sensors can be queried, control is direct. Documentation is OK.
★★	instrument is a bit of a black box. Legacy programming language. No API docs.
★	No or weak API.

GUI-BASED METHOD DEVELOPMENT

- An instrument and scheduler problem
- Hardware vendors should enable their devices to be used in the most un-restrictive way possible while ensuring quality and consistency
- Some are working around this (PyLabRobot), but it shouldn't be necessary



PROPRIETARY DATA FORMATS



FLOW CYTOMETERS (GREAT)

Industry-agreed standard file (.FCS) for data and metadata



IMAGERS (OK)

Industry-standard outputs (.png, .tif) automatically produced, but with more useful data often trapped in vendor-specific files/software



QPCR CYCLERS (BAD)

Getting anything useful in a machine-readable format requires heavy configuration in vendor software

Bio-Rad, Molecular Devices, Thermo Fisher

OPERATIONAL INSIGHTS

Most drivers / output files are focussed on the experimental outcome, but not what else happened along the way

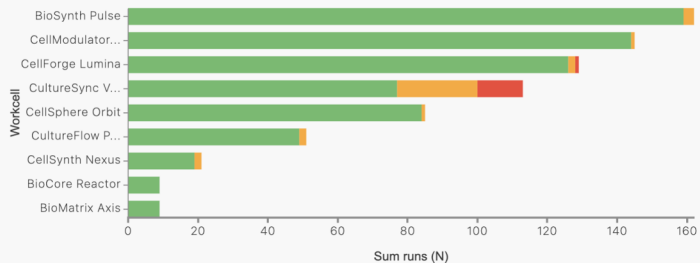
- Sample data
- Environmental conditions
- Instrument age/utilization

There's additional data to be captured

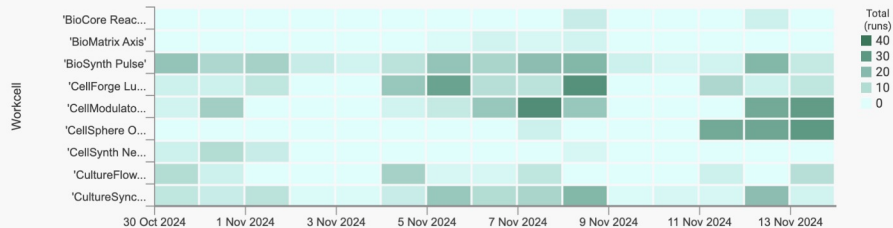
- LH deck cameras
- System-wide environmental sensors
- Tracking time on manual touchpoints

LIMITATIONS: OPERATIONAL INSIGHTS

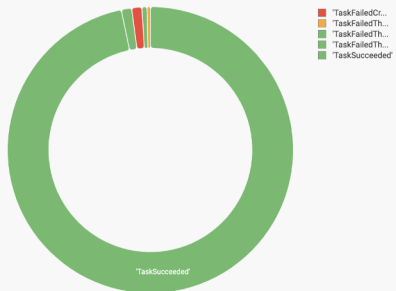
Workcell Runs by Outcome



Workcell Utilization Heatmap



Task Share by Outcome



Workcell Timeline by Outcome



LIBERATING LEGACY DATA

- A truly impressive amount of data is “trapped” in non-integrated instruments and proprietary data formats
- Unlike other LLM data sources, published data is sparse and oversimplified

Methods

Plant material. *Kalanchoë fedtschenkoi* ‘M2’ plants were purchased from Mass Spectrum Botanicals (Tampa, FL, USA) (Supplementary Method 2).

Estimation of DNA content. The DNA contents of young leaf tissue samples were analyzed using flow cytometry analysis service provided by Plant Cytometry Services (The Netherlands). The internal standard was *Vinca minor* (DNA = 1.51 pg/2 C = 1477 Mbp/2 C).

Chromosome counting. Images were collected using an Olympus FluoView FV1000 confocal microscope (Center Valley, PA, USA) with a 60× objective. Images were sharpened using Adobe Photoshop and chromosomes were counted using ImageJ software (Supplementary Method 3).

Illumina sequencing of genome. The genomic DNA libraries of *K. fedtschenkoi* were sequenced on a MiSeq instrument (Illumina, CA, USA) using MiSeq Reagent Kit v3 (600-cycle) (Illumina, CA, USA) (Supplementary Method 4).

Transcriptome sequencing. In order to capture mRNA abundance changes responsive to diel conditions, samples were collected in triplicate from mature *K. fedtschenkoi* leaves (i.e., the fifth and sixth mature leaf pairs counting from the top) every 2 h over a 24 h time course under 12 h light/12 h dark photoperiod. Additional tissues were sampled in triplicate including roots, flowers, shoot tips plus young leaves, and stems at one time point, 4 h after the beginning of the light period (Supplementary Method 5).

Genome assembly and improvement. The *K. fedtschenkoi* genome was initially assembled using platanus⁶⁶ from 70X Illumina paired-end reads (2 × 300 bp reads; unamplified 540 bp whole-genome shotgun fragment library), and three mate-libraries (3 kb, 14X; 6 kb, 12X; 11 kb, 11X). Further genome scaffolding was performed using MeDuSa⁶⁷ sequentially with the genome assemblies of *K. laxiflora* v1.1 (Phytozome), *Vitis vinifera* Genoscope.12X (Phytozome), and *Solanum tuberosum* v3.4 (Phytozome).

Protein-coding gene annotation. The genome annotation for *K. fedtschenkoi* was performed using homology-based predictors facilitated with transcript assemblies (Supplementary Method 6).

Construction of orthologous groups. The protein sequences of 26 plant species were selected for ortholog group construction (Supplementary Method 7).

Construction of species phylogeny. The phylogeny of plant species was constructed from the protein sequences of 210 single-copy genes identified through analysis of orthologous groups (see “Construction of orthologous groups” section). The details for species phylogeny construction are described in Supplementary Method 8.

Construction of protein tribes and phylogenetic analysis. The protein sequences used for ortholog analysis (see “Construction of orthologous groups”) were also clustered into tribes using TRIBE-MCL⁶⁸, with a BLASTp E-value cutoff of 1e-5 and an inflation value of 5.0. Phylogenetic analysis of the protein tribes is described in Supplementary Method 9.

WRAPPING UP

1. AI tools \neq AI strategy.
2. Lab Automation Fragmentation Strikes Again
3. Vendors: Keep it simple, keep it open

Have you tried implementing reactive, proactive, or generative AI?

What's working? What isn't?

Any additional limitations with existing platforms?

THE FULLY INTEGRATED LAB AUTOMATION PLATFORM POWERED BY:



HARDWARE

LINQ Bench - Built to Adapt, Ready to Scale



SOFTWARE

LINQ Cloud - Faster workflow creation. Simplified Running.
Deeper integration



SUPPORT

LINQ Service and Support - Expert support at any stage.

