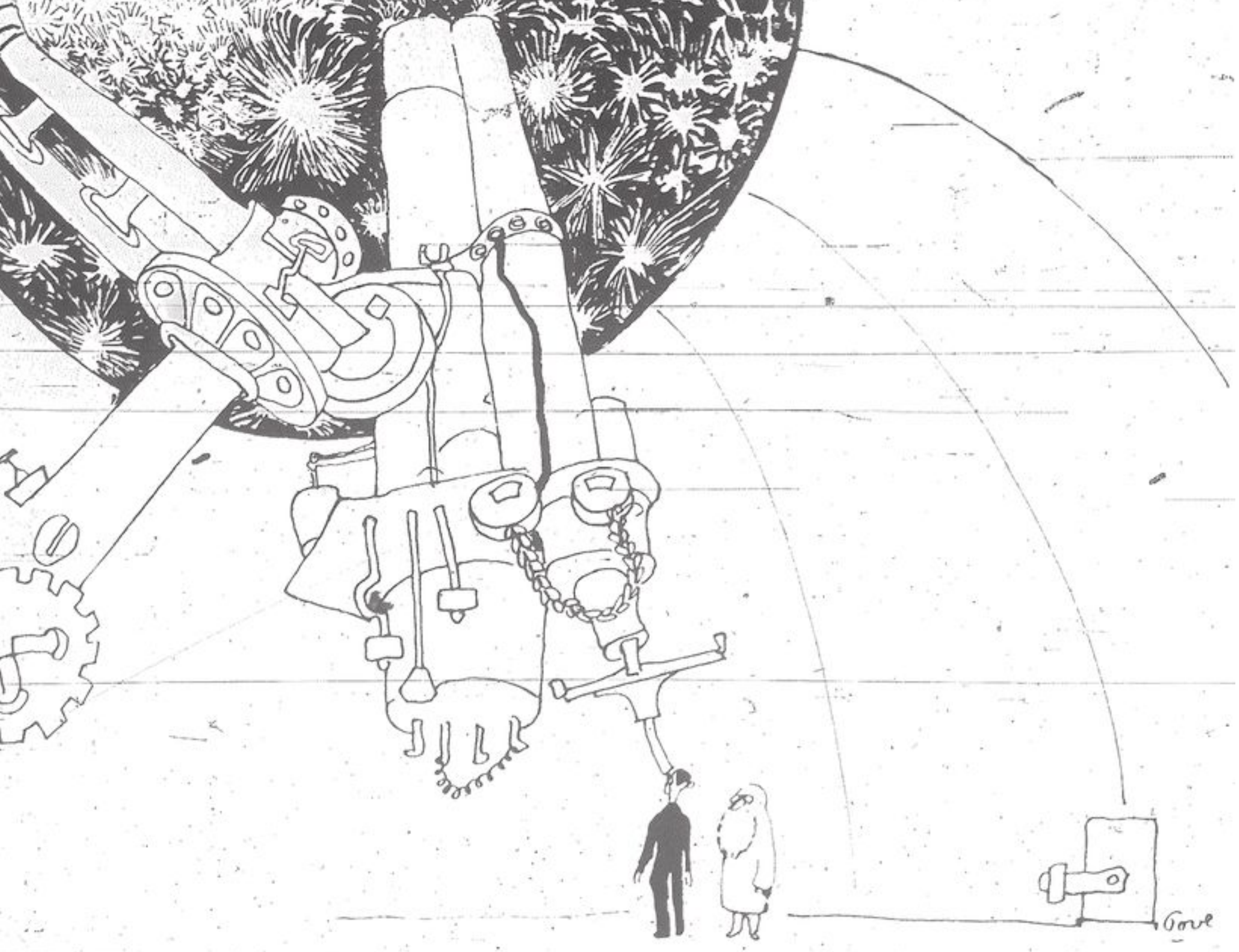
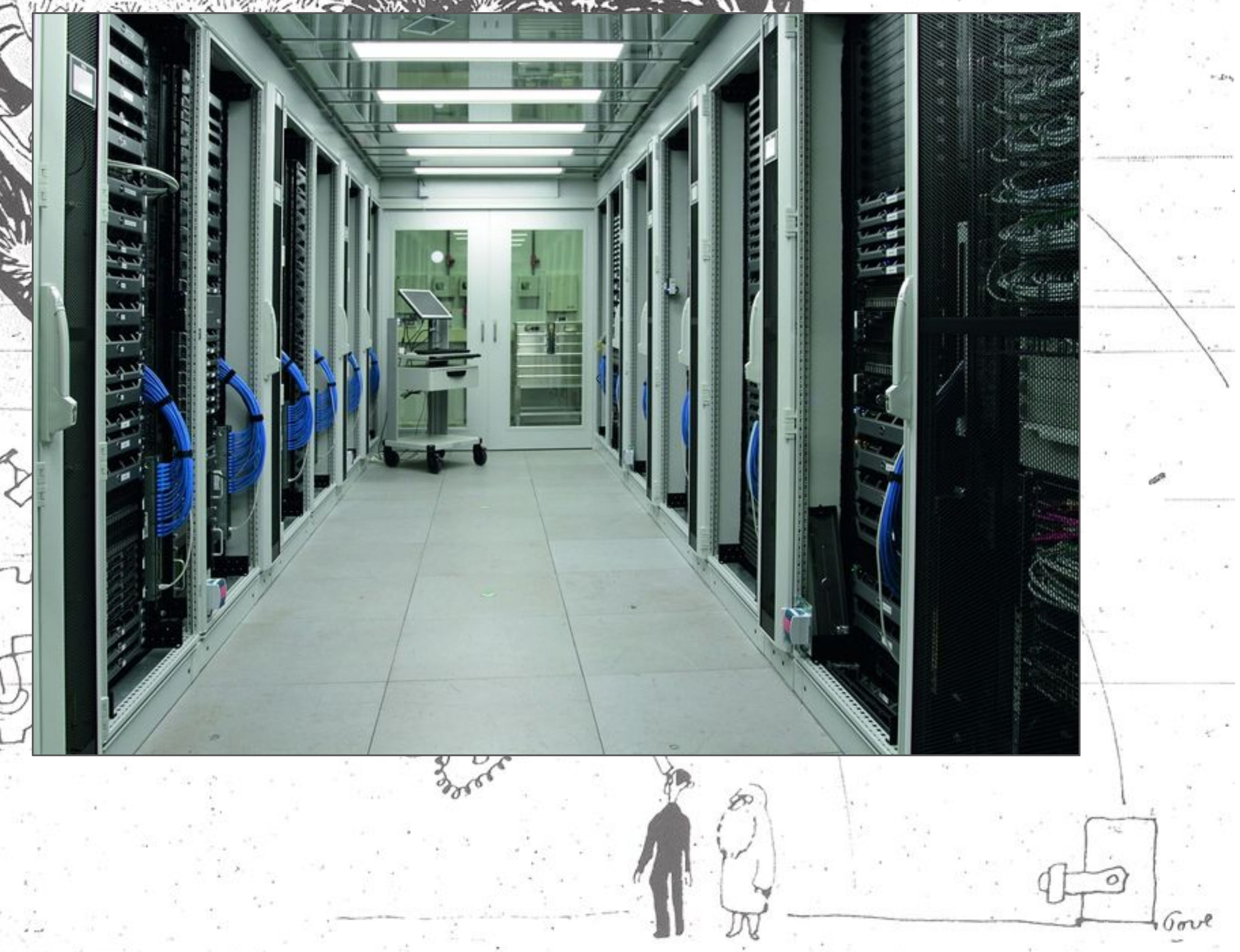


Scientific exploration

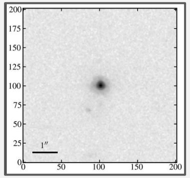
in the era of high throughput astronomical observatories.



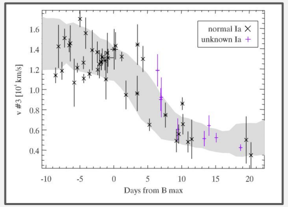
Scientific exploration

in the era of high throughput astronomical observatories.

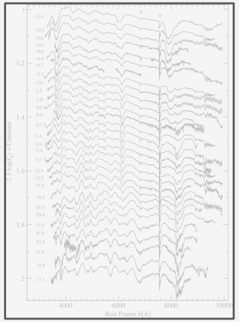
Field growing up (or was it OKC?)



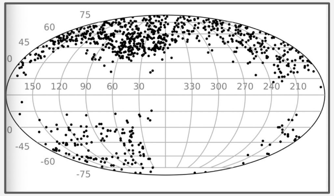
2008: 4 high-z SNe (**Master**)



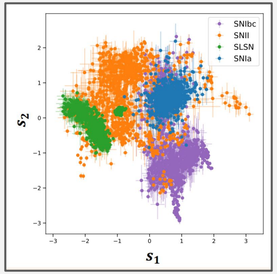
2012: ~40 SDSS SNe (OKC)



2014: ~400 SNfactory (they exist!)

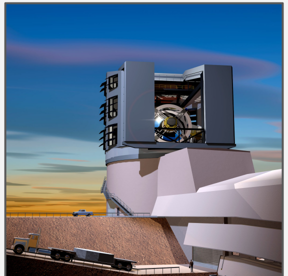


2023: ~2000 ZTF SNe Ia



2025: ~20000 ZTF SNe

Here be robots!



~2030: ~100000 LSST SNe

How to allow scientific creativity / diversity when pipelines need to be efficient and experts are needed to develop+train models?

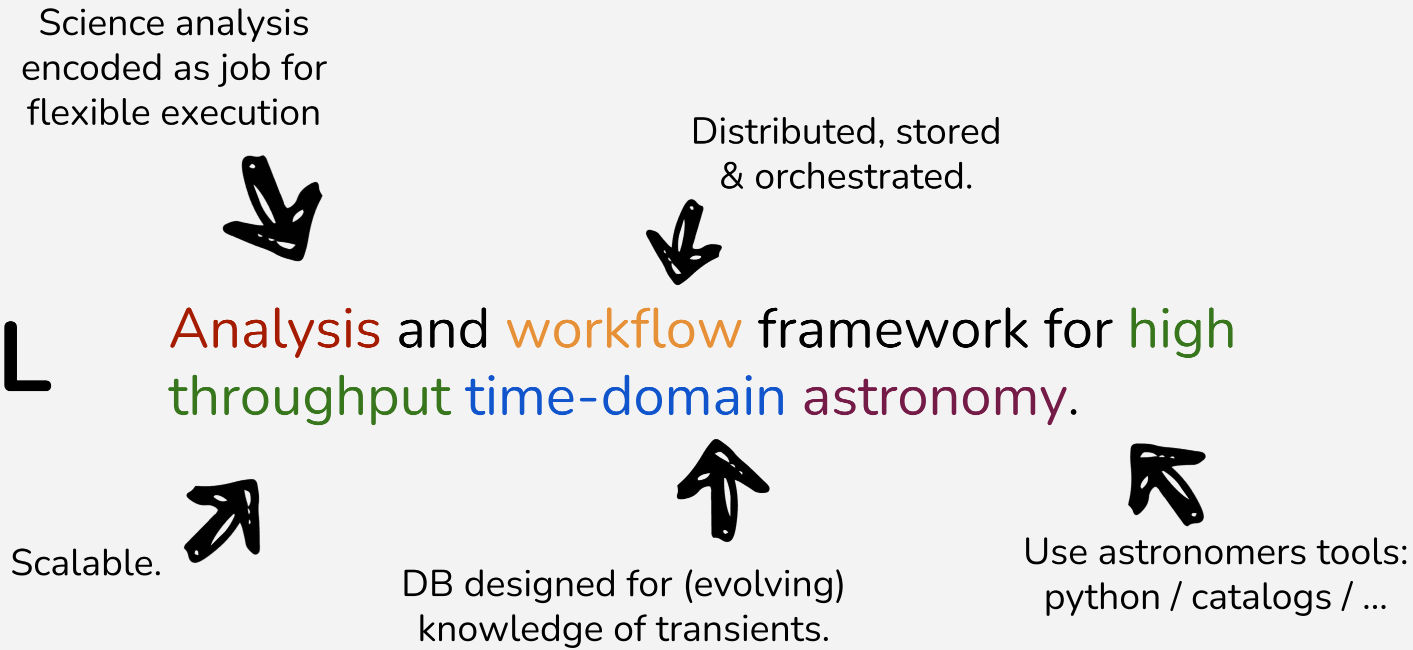
… related to:

1. Rerunning complex analysis with systematic modifications *(calibration!, how it started)*
2. Code-to-data
3. Reproducibility / FAIR workflows
4. Collaborative, sustainable software development

Requires structure where:

- A scientific idea can be encoded using "normal" tools (i.e. python)
- Domain specific software can be executed
- Method exists to run the above at high speed, while guaranteeing provenance and reproducibility.

AMPEL



Science analysis encoded as job for flexible execution

Distributed, stored & orchestrated.

Analysis and workflow framework for high throughput time-domain astronomy.

Scalable.

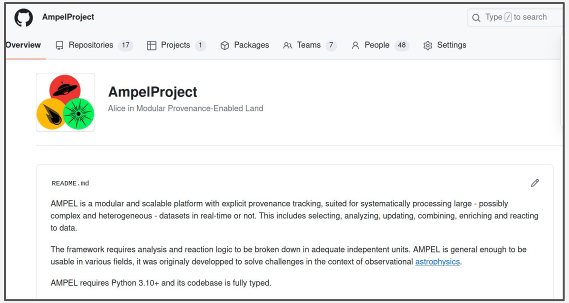
DB designed for (evolving) knowledge of transients.

Use astronomers tools: python / catalogs / …

Realizes code-to-data in astronomy.

https://github.com/AmpelAstro

AMPEL



AmpelProject

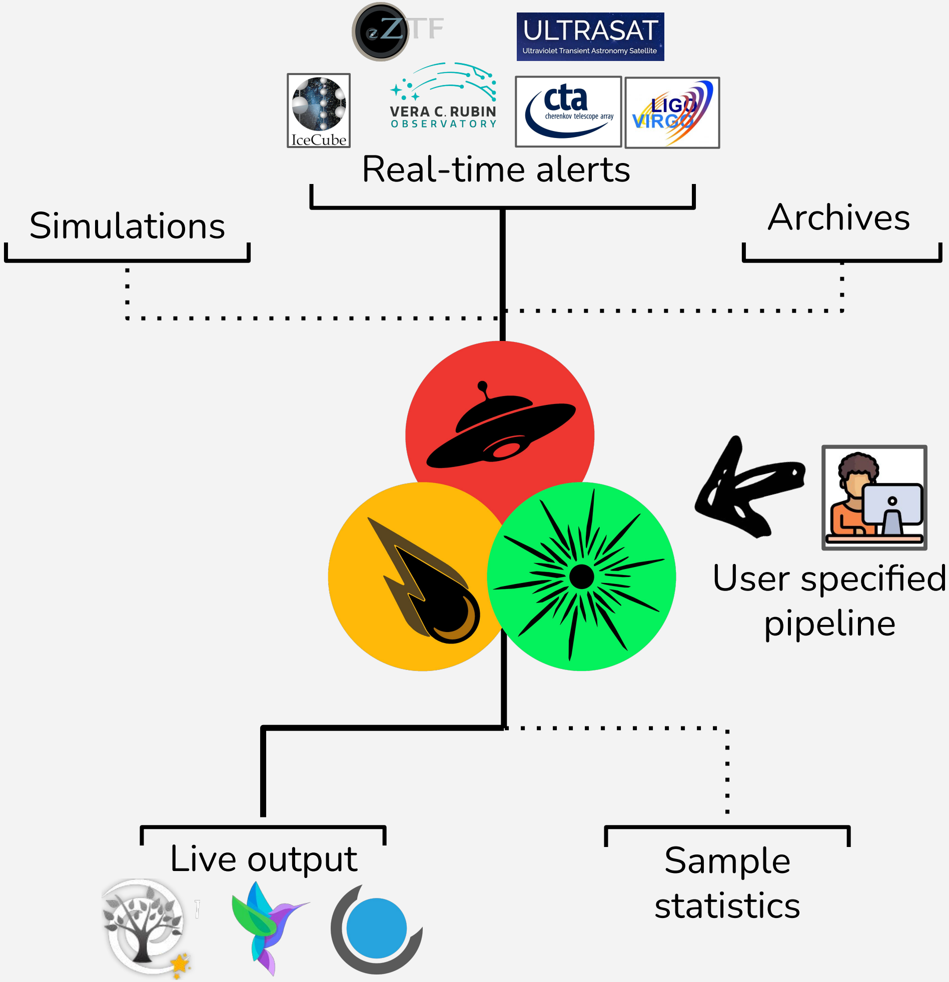
Alice in Modular Provenance-Enabled Land

README.md

AMPEL is a modular and scalable platform with explicit provenance tracking, suited for systematically processing large - possibly complex and heterogeneous - datasets in real-time or not. This includes selecting, analyzing, updating, combining, enriching and reacting to data.

The framework requires analysis and reaction logic to be broken down in adequate indepentent units. AMPEL is general enough to be usable in various fields, it was originaly developped to solve challenges in the context of observational astrophysics.

AMPEL requires Python 3.10+ and its codebase is fully typed.


ZTF
ULTRASAT
Ultraviolet Transient Astronomy Satellite
IceCube
VERA C. RUBIN OBSERVATORY
cta
cherenkov telescope array
LIGO
VIRGO

Real-time alerts

Simulations

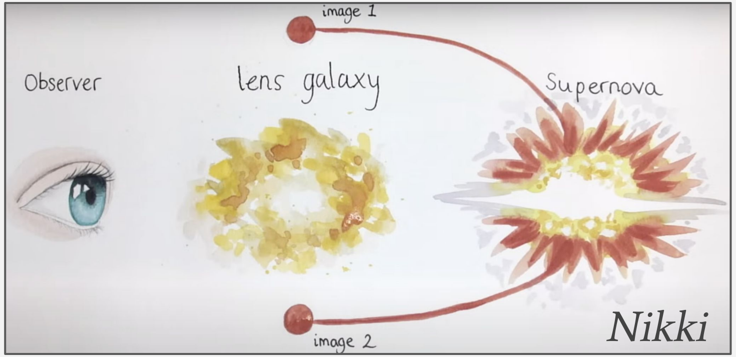
Archives

User specified pipeline

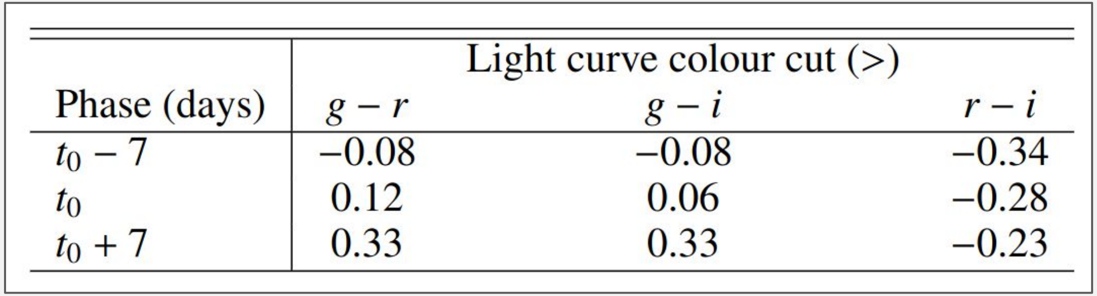
Live output

Sample statistics

Case of Searching for Gravitationally Lensed Supernovae

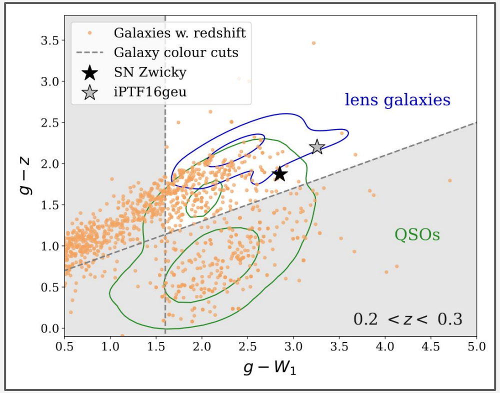


Nikki



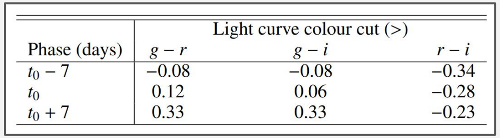
Phase (days)	Light curve colour cut (>)		
	$g-r$	$g-i$	$r-i$
t_0-7	−0.08	−0.08	−0.34
t_0	0.12	0.06	−0.28
t_0+7	0.33	0.33	−0.23

Ana

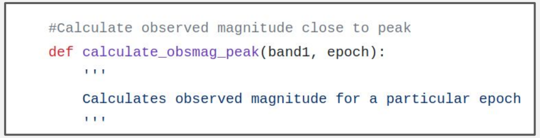


“Domain knowledge” suggest target region as combination of lightcurve and host galaxy properties.

1. Encode parameter space

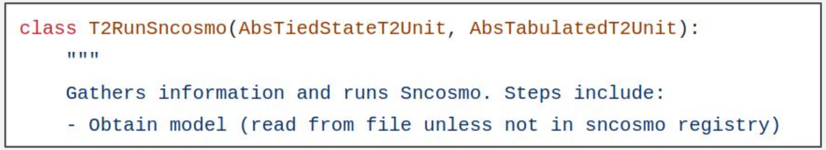


Phase (days)	Light curve colour cut (>) $g-r$	$g-i$	$r-i$
t_0-7	−0.08	−0.08	−0.34
t_0	0.12	0.06	−0.28
t_0+7	0.33	0.33	−0.23



```
#Calculate observed magnitude close to peak
def calculate_obsmag_peak(band1, epoch):
    '''
    Calculates observed magnitude for a particular epoch
    '''
```

2. Identify required units



```
class T2RunSncosmo(AbsTiedStateT2Unit, AbsTabulatedT2Unit):
    """
    Gathers information and runs Sncosmo. Steps include:
    - Obtain model (read from file unless not in sncosmo registry)
```

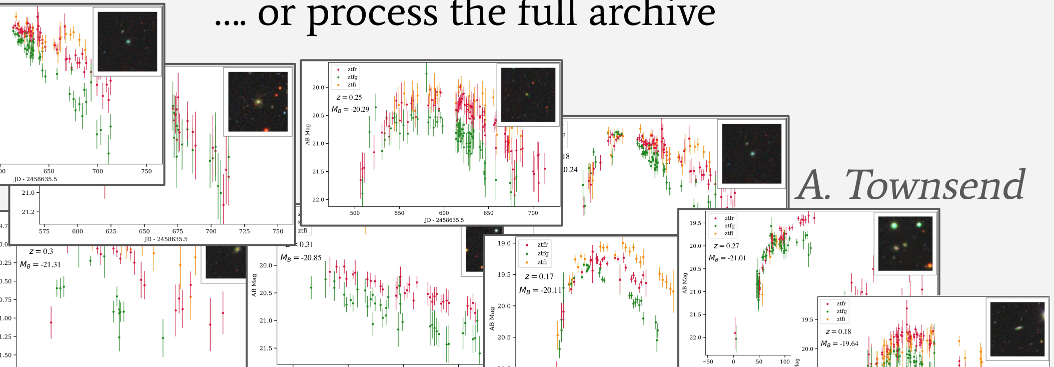
... or interface a new

Ampel-interface

`ampel-interface` provides type-hinted abstract base classes for Ampel.

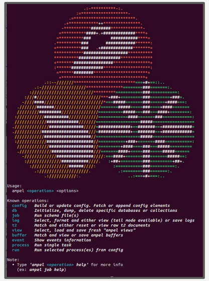
4. Upload to live instance

.... or process the full archive

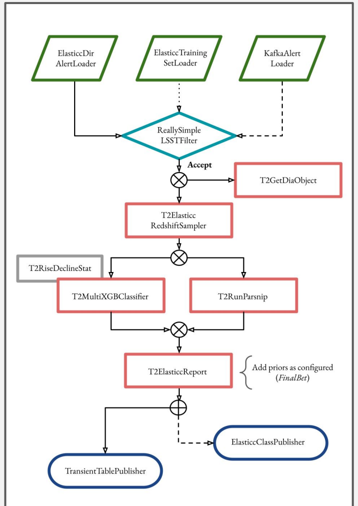


A. Townsend

3. Run job locally

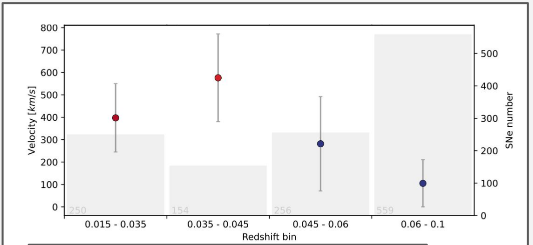


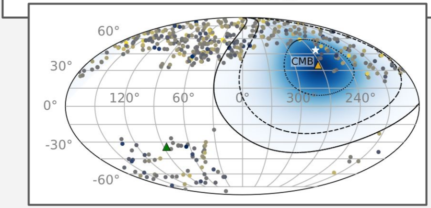
```
Usage:
  ampel <operation> <options>

Known operations:
  config   Build or update config. Fetch or append config elements
  db       Initialize, dump, delete specific databases or collections
  job      Run schema file(s)
  log      Select, format and either view (tail mode available) or save logs
  t2       Match and either reset or view raw t2 documents
  view     Select, load and save fresh "ampel views"
  buffer   Match and view or save ampel buffers
  event    Show events information
  process  Run single task
  run      Run selected process(es) from config

Note:
  • Type 'ampel <operation> help' for more info
    (ex: ampel job help)
```



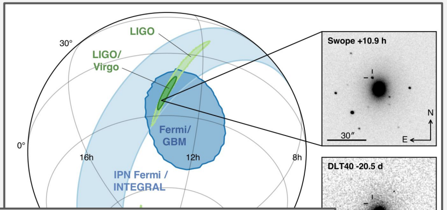
Sample projects

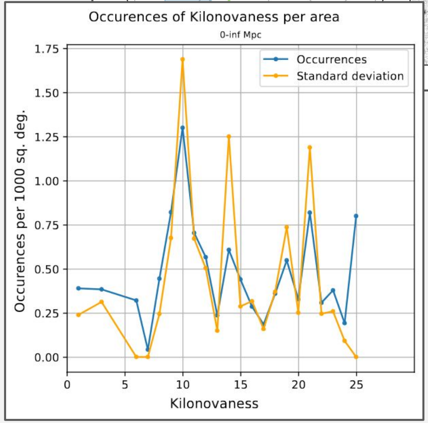
Mapping the Universe



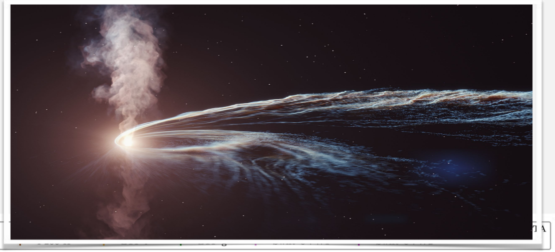


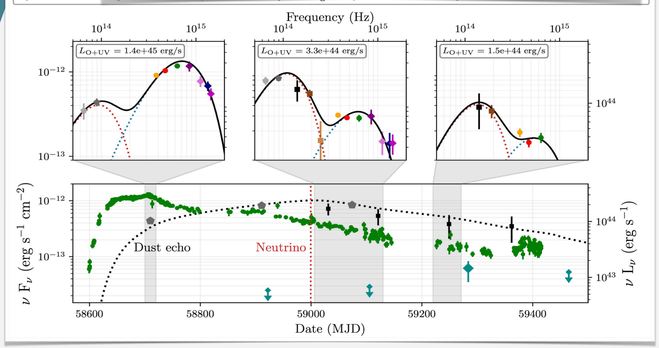
Kilonovaness





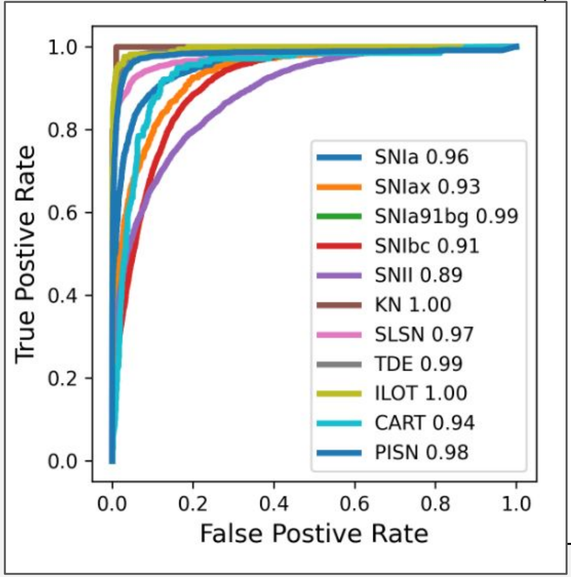
TDE/Neutrino

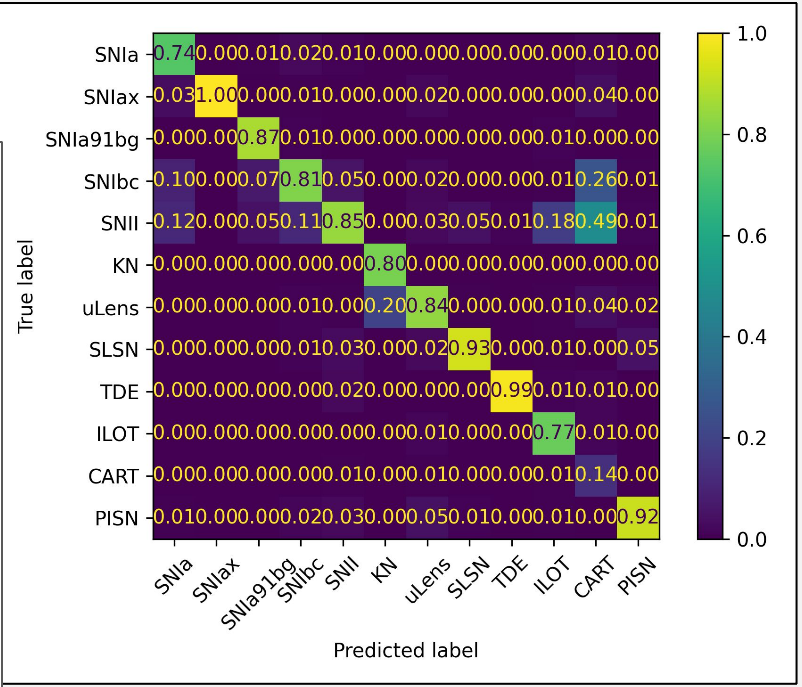



WISE W2
WISE W1
Frequency (Hz)
10^{14}
10^{15}
$L_{\rm O+UV}$ = 1.4e+45 erg/s
$L_{\rm O+UV}$ = 3.3e+44 erg/s
$L_{\rm O+UV}$ = 1.5e+44 erg/s
10^{-12}
10^{-13}
10^{44}
10^{43}
ν F$_\nu$ (erg s^{-1} cm^{-2})
ν L$_\nu$ (erg s^{-1})
Dust echo
Neutrino
Date (MJD)
58600
58800
59000
59200
59400

Getting ready for LSST: ELAsTiCC simulation

AMPEL provided best-in-class extragalactic classifications. Will be made available for users in real time.



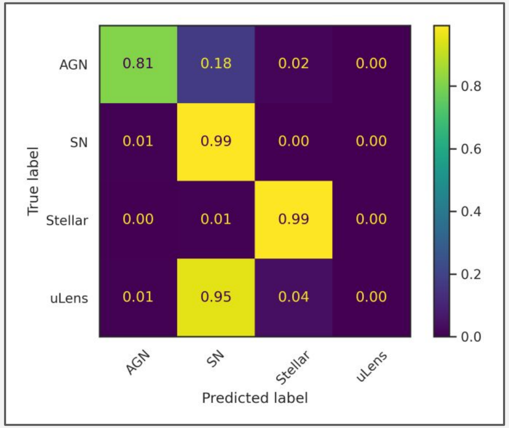


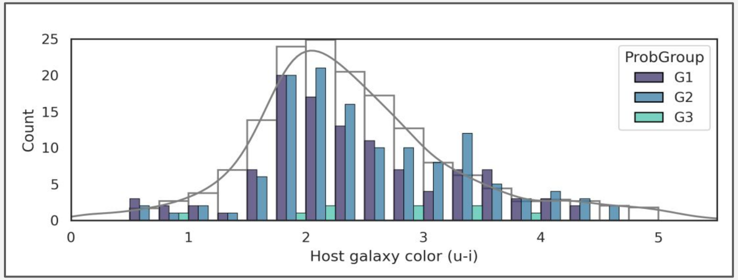
	SNIa	SNIax	SNIa91bg	SNIbc	SNII	KN	uLens	SLSN	TDE	ILOT	CART	PISN
SNIa	0.74	0.00	0.01	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00
SNIax	0.03	1.00	0.00	0.01	0.00	0.00	0.02	0.00	0.00	0.00	0.04	0.00
SNIa91bg	0.00	0.00	0.87	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
SNIbc	0.10	0.00	0.07	0.81	0.05	0.00	0.02	0.00	0.00	0.01	0.26	0.01
SNII	0.12	0.00	0.05	0.11	0.85	0.00	0.03	0.05	0.01	0.18	0.49	0.01
KN	0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00
uLens	0.00	0.00	0.00	0.01	0.00	0.20	0.84	0.00	0.00	0.01	0.04	0.02
SLSN	0.00	0.00	0.00	0.01	0.03	0.00	0.02	0.93	0.00	0.01	0.00	0.05
TDE	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.99	0.01	0.01	0.00
ILOT	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.77	0.01	0.00
CART	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.01	0.14	0.00
PISN	0.01	0.00	0.00	0.02	0.03	0.00	0.05	0.01	0.00	0.01	0.00	0.92

Getting ready for LSST: ELAsTiCC simulation

... but that is not what you will want to use.

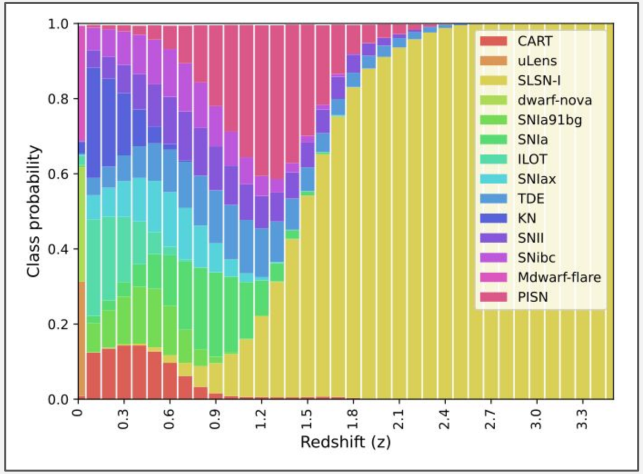
I. Infant SNe





II. Unbiased labeling

III. Max posterior



Three sample classifications, tuned for different goals. Each can be further optimized.

Run the AMPEL ELAsTiCC classifiers:

Try it out (with python3.10 + poetry):

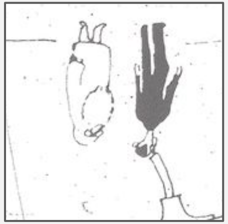
- `git clone --branch elasticc2 https://github.com/AmpelAstro/Ampel-HU-astro.git`
- `cd Ampel-HU-astro/`
- `poetry install -E "ztf sncosmo extcats notebook elasticc"`
- `ampel config build -out ampel_conf.yaml >& ampel_conf.log`
- `ampel job --config ampel_conf.yaml --schema examples/elasticc_alerttar.yml`

Will install a local AMPEL environment, download sample ELAsTiCC alerts and run three ML classifiers.

Can be modified to include new classifiers - would enter the DESC challenge!

Summary

- **LSST + LVK + SKA + CTA + Icecube/KM3NeT + ULTRASAT + ZTF/LS4/BlackGem/...**
  - Exciting times!
  - How do we “explore” these data floods?



- **Using the large transient counts requires dedicated fast pipelines**
  - Already throwing away most of the data
  - Not enough that an ML model exists - how do you use it?

- **No need to reinvent the wheel - tools exist**
  - But you have to learn how to drive

- **Ampel is one toolbox for time-domain data processing**
  - Photometric ZTF samples around the corner - 10x larger
  - User workflows can be applied to full alert streams

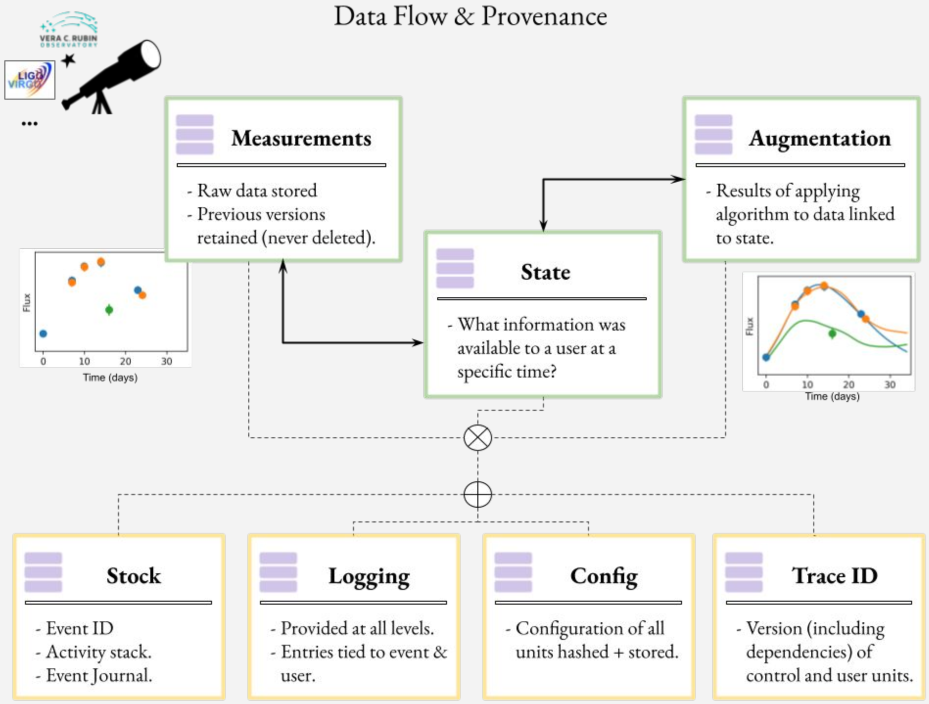
Provenance schema

AMPEL systematically records every transient *state** as an immutable object. Results always tied to a state.

The version (TraceID) and config of every operation is recorded in the DB, Logs and Journal trace the history of every Stock.

A workflow is specified by a job schema, which can distributed, mounted in a container and uploaded to a live instance.

* A *state* is the collection of data points associated with an (assumed) object at a specific time and visible to a specific user.



Data Flow & Provenance

Measurements

- Raw data stored
- Previous versions retained (never deleted).

Augmentation

- Results of applying algorithm to data linked to state.

State

- What information was available to a user at a specific time?

Stock

- Event ID
- Activity stack.
- Event Journal.

Logging

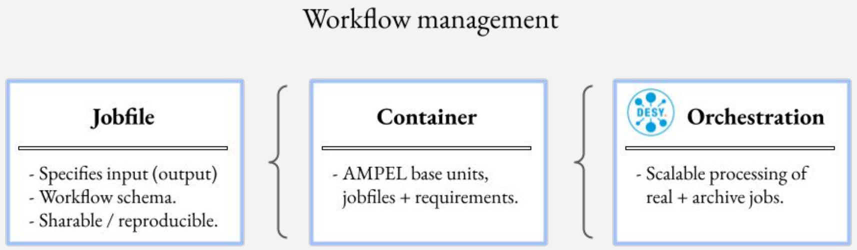
- Provided at all levels.
- Entries tied to event & user.

Config

- Configuration of all units hashed + stored.

Trace ID

- Version (including dependencies) of control and user units.



Workflow management

Jobfile

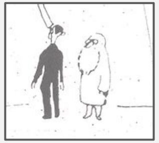
- Specifies input (output)
- Workflow schema.
- Sharable / reproducible.

Container

- AMPEL base units, jobfiles + requirements.

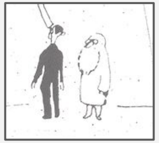
Orchestration

- Scalable processing of real + archive jobs.



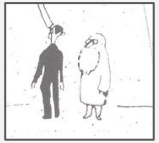
Exercise: From Idea to Implementation

Requirements: **Flexibility** **CS algorithms** **Scalable** **Provenance**



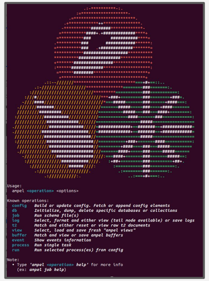
Exercise: From Idea to Implementation

Requirements:	Flexibility	CS algorithms	Scalable	Provenance
↓				
Implications:	Modularity	Interfaces, [python]	DB / unit independence	Immutability, workflow, ...

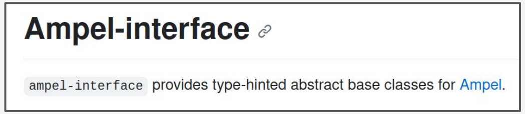


Exercise: From Idea to Implementation

Requirements:	Flexibility	CS algorithms	Scalable	Provenance
Implications:	Modularity	Interfaces, [python]	DB / unit independence	Immutability, workflow, ...
AMPEL:	Local job, common tools	Base classes, git integration	Four distinct action tiers	The State

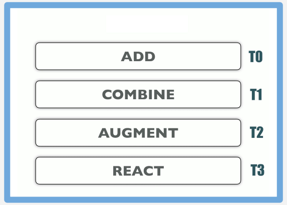


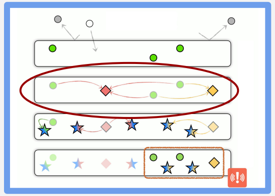
```
Usage:
  ampel <operation> <options>

Known operations:
  config   Build or update config. Fetch or append config elements
  db       Initialize, dump, delete specific databases or collections
  job      Run schema file(s)
  log      Select, format and either view (tail mode available) or save logs
  t2       Match and either reset or view raw t2 documents
  view     Select, load and save fresh "ampel views"
  buffer   Match and view or save ampel buffers
  event    Show events information
  process  Run single task
  run      Run selected process(es) from config

Note:
  • Type 'ampel <operation> help' for more info
    (ex: ampel job help)
```

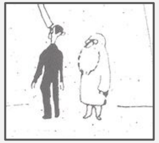


Ampel-interface

`ampel-interface` provides type-hinted abstract base classes for Ampel.







Exercise: From Idea to Implementation

Lessons:

Combining speed, flexibility and provenance is hard…

… mix of CS, domain knowledge & information theory…

… but possible. 30k lines of code (V. Brinnel!)

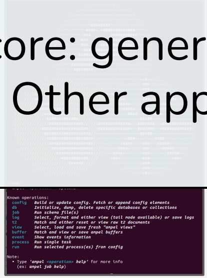
Not suitable for everything. Scope important.

Ampel-core: general tool for provenance focus analysis of data streams. Other applications?

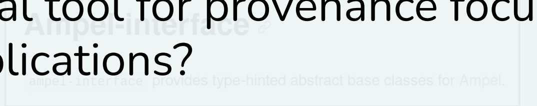
Requirements: Flexibility | CS algorithms | Scalable | Provenance

Implications: Modularity | Interfaces, [python] | DB / unit independence | Immutability, workflow, …

AMPEL: Local job, common tools | Base classes, git integration | Four distinct action tiers | The State

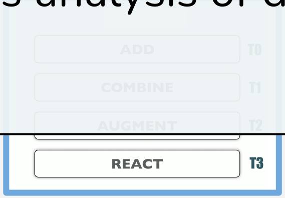


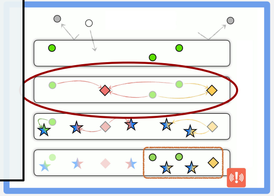
```
Usage:

Known operations:
  config   Build or update config. Fetch or append config elements
  db       Initialize, dump, delete specific databases or collections
  job      Run schema file(s)
  log      Select, format and either view (tail mode available) or save logs
  t2       Match and either reset or view raw t2 documents
  view     Select, load and save fresh "ampel views"
  buffer   Match and view or save ampel buffers
  event    Show events information
  process  Run single task
  run      Run selected process(es) from config

Note:
  • Type 'ampel <operation> help' for more info
    (ex: ampel job help)
```



Ampel-interface

ampel-interface provides type-hinted abstract base classes for Ampel.



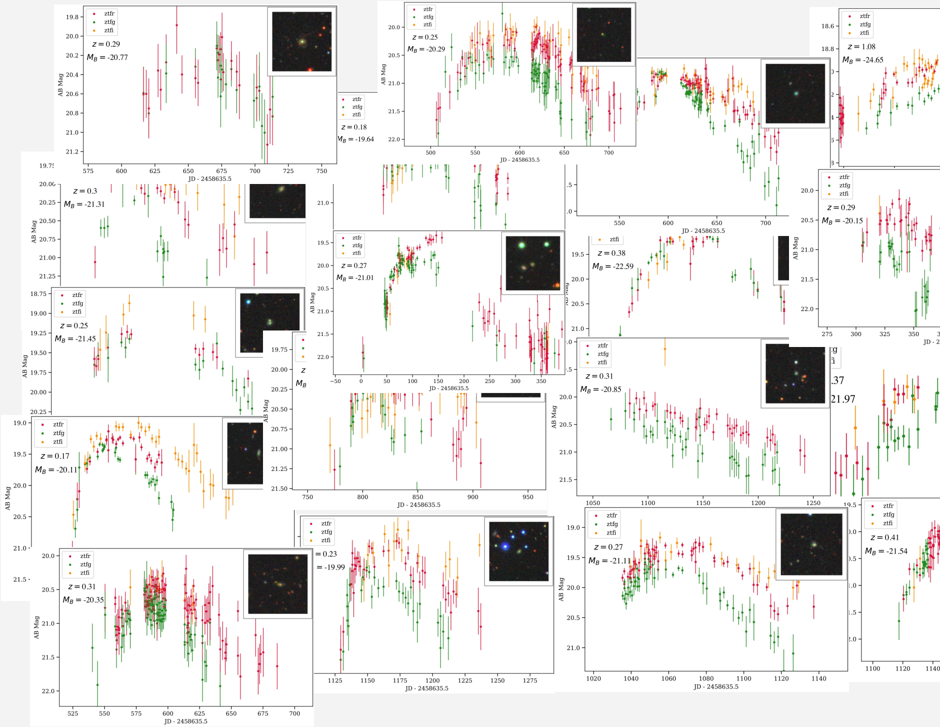


P2: **Searching for gravitationally lensed supernovae**

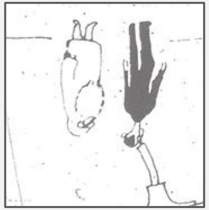
ZTF detects ~100 000 transients *each night*.
Data exists from 2018.

Systematically reprocess all archived data, incorporate astronomical catalogs + apply custom model.

A Townsend worked to subset of ~20 candidates. Many are Superluminous Supernovae, but not all...



Preparation is key



“Inverted” work order:

① Get some data

② Develop analysis (software)

③ Conclude based on results

→

③ Define project hypothesis

② Develop and test model

① Connect to data streams