pip install **skysurvey**

Fast python package to simulate ***targets*** observed by ***surveys***

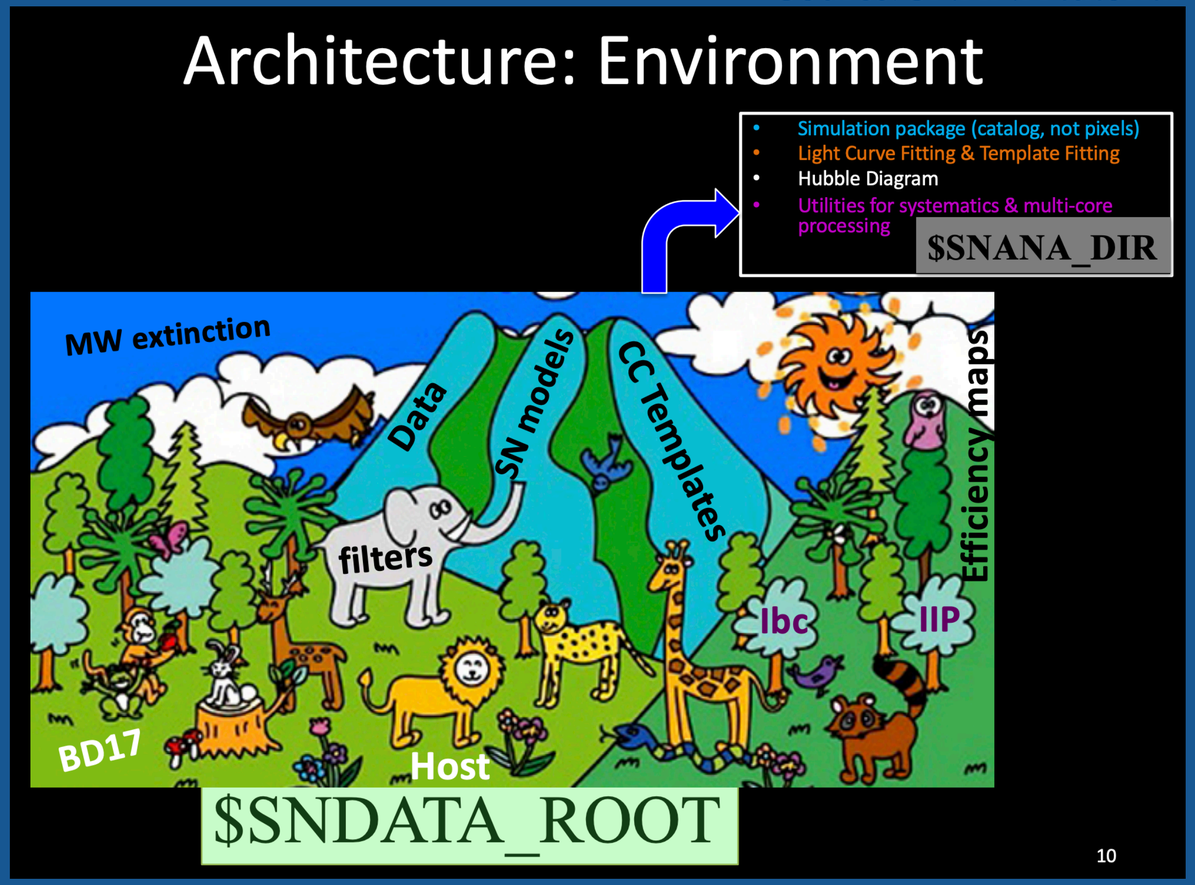
& see also pip install **modeldag**

m.rigault@ip2i.in2p3.fr

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n°759194 - USNAC)

Simulation: SNANA

Source: SNANA tutorial



Became the central tools SN cosmo:

Simulate Transient

Simulate effect of selection function

Correct for bias selection

Fit cosmology

Scolnic+18 ; Brout+22 ; Riess+22 ; Popovic 2024 ; Vincenzi+24

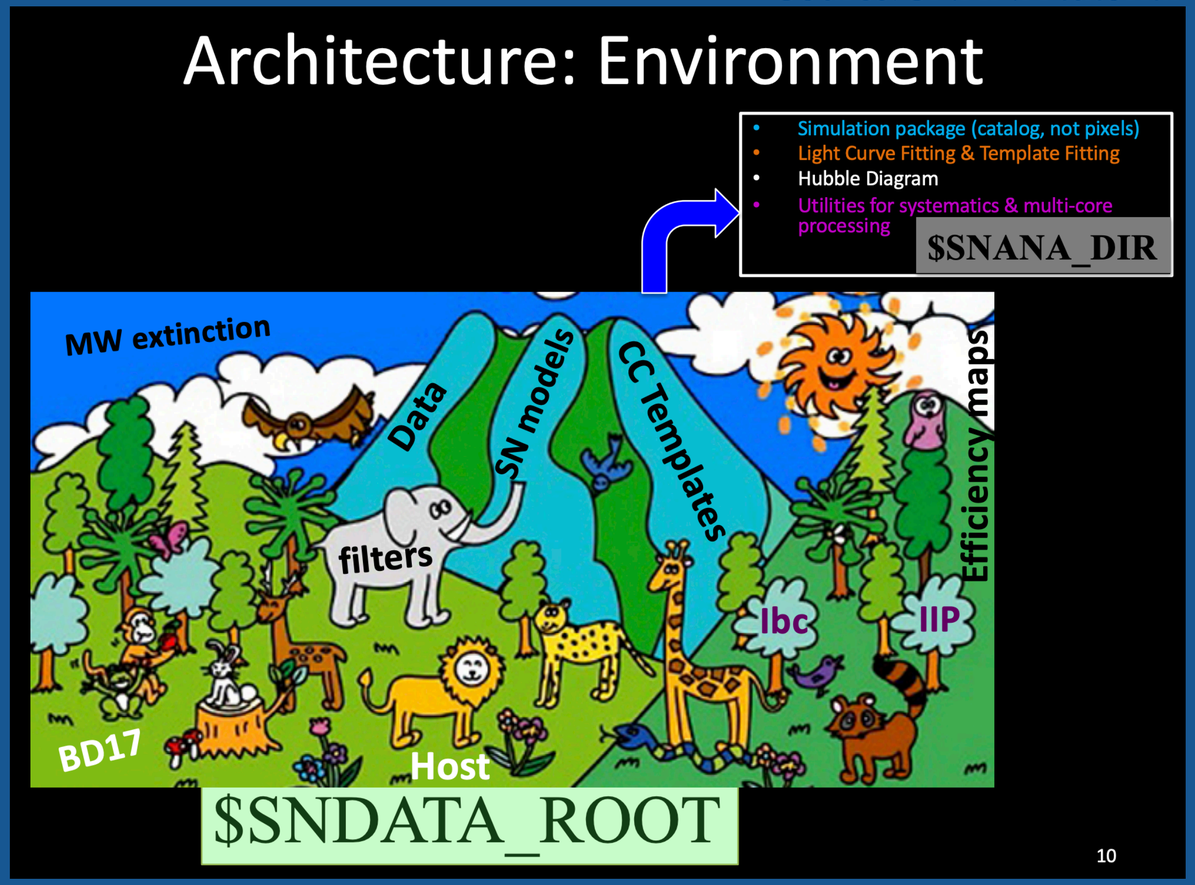
Preparing:

LSST, Roman

Plasstic / Elasticc

Simulation: SNANA

Source: SNANA tutorial



Became the central tools SN cosmo:

Simulate Transient

Simulate effect of selection function

Correct for bias selection

Fit cosmology

Scolnic+18 ; Brout+22 ; Riess+22 ; Vincenzi+24

SNANA is central for cosmological analyses…

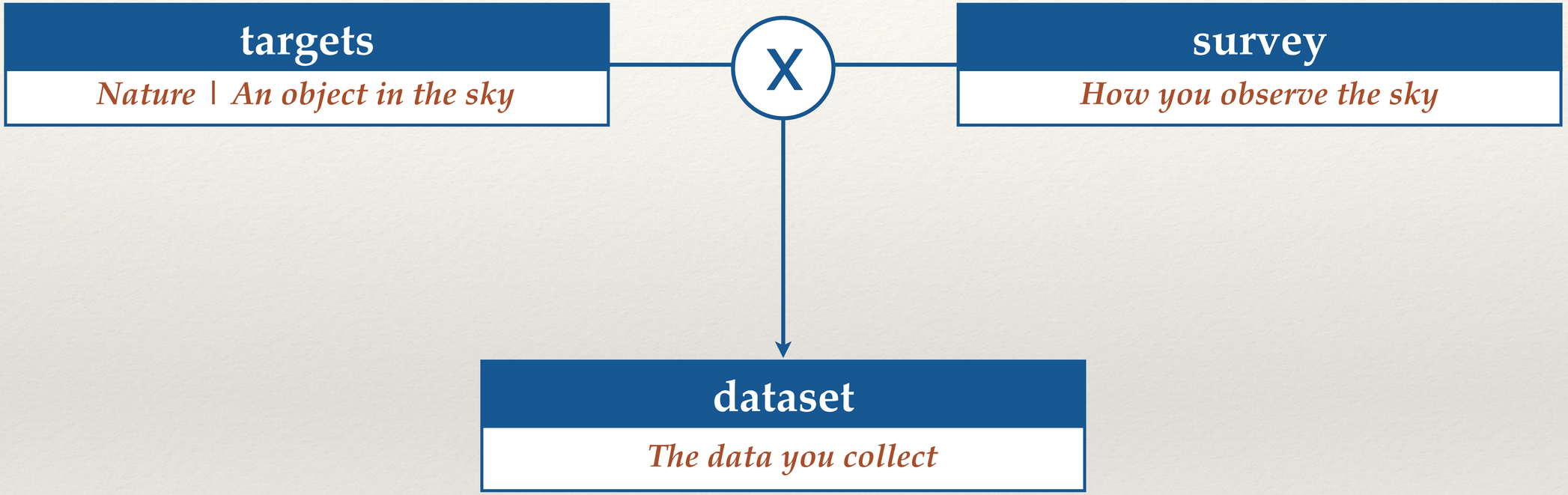
Yet, hard to install and complex to use.

pip install **skysurvey**

skysurvey | concept

aims to replace SNANA “step 1: simulation”

See also simsurvey | snsim



skysurvey | *goal: easy to use*

targets

Nature | An object in the sky

X

survey

How you observe the sky

```
import skysurvey
snia = skysurvey.SNeIa.from_draw(10_000, zmax=0.3,
                                 tstart = "2018-04-01",
                                 tstop = "2020-12-01")
```

Lots of pre-built targets, *easy* to customize

```
ztf =  skysurvey.ZTF.from_logs() # login requires
lsst = skysurvey.LSST.from_opsim('baseline_v3.3_10yrs.db')
```

Easy to build you own survey

dataset

The data you collect

```
dset = skysurvey.DataSet.from_targets_and_survey(snia, ztf)
```

Always as *easy* as that

skysurvey | *goal: easy to use*

targets

Nature | An object in the sky

```
prop = dict(zmax=0.3,
            tstart="2018-04-01", tstop="2020-12-01")

snia = skysurvey.SNeIa.from_draw(10_000, **prop)
snii = skysurvey.SNeII.from_draw(1_000, **prop)
```

Lots of pre-built targets, ***easy*** to customize

X

survey

How you observe the sky

```
ztf =  skysurvey.ZTF.from_logs() # login requires
lsst = skysurvey.LSST.from_opsim('baseline_v3.3_10yrs.db')
```

Easy to build you own survey

dataset

The data you collect

```
dset = skysurvey.DataSet.from_targets_and_survey([snia, snii], ztf)
```

Always as ***easy*** as that

skysurvey | *based* within the python environment

pandas NumPy SciPy

targets

Nature | An object in the sky

X

survey

How you observe the sky

`snia.data`

	z	x1	c	t0	ra	dec	magabs	mwebv	magobs	x0	template
0	0.07705	-0.030	1.221450	58487.214844	336.697937	7.605778	-15.358439	0.134286	22.429092	0.000017	salt2
1	0.06865	-0.700	-0.027583	58659.839844	60.706551	55.346775	-19.233326	0.719195	18.290983	0.000766	salt2
2	0.06905	0.320	0.351301	58782.136719	23.629887	-0.468616	-18.197527	0.036892	19.339998	0.000292	salt2

`survey.data`

	gain	zp	skynoise	mjd	band	ra	dec	fieldid_survey	fieldid
1867451	1	30	234.6674346923828	58800.0078125	desr	18.292482376098633	1.2328510284423828	11331	227158
1867554	1	30	234.6674346923828	58800.0078125	desr	18.292482376098633	1.2328510284423828	11331	227159
1867634	1	30	234.6674346923828	58800.0078125	desr	18.292482376098633	1.2328510284423828	11331	227160

dataset

The data you collect

`dset.data`

targets .index → **index** ; *survey .index* → **fieldid**

index	fieldid	rcid	time	band	flux	fluxerr	zp	zpsys
5821	**26009040**	1	3	59104.390625	ztfr	-39.302373	36.143482	25.233383	ab
	26054356	1	3	59106.328125	ztfg	8.518026	14.717361	24.559206	ab
	26331019	1	3	59112.300781	ztfg	24.598649	20.259558	26.025703	ab

In test development

jax dask polars

skysurvey | *DataFrame*

targets

```
import skysurvey
snia = skysurvey.SNeIa.from_draw(10_000, zmax=0.3,
                                 tstart = "2018-04-01",
                                 tstop = "2020-12-01")
```

X

survey

```
ztf = skysurvey.ZTF.from_logs()
```

47 M entries | 760 k exposures

dataset

```
dset = skysurvey.DataSet.from_targets_and_survey(snia, ztf)
```

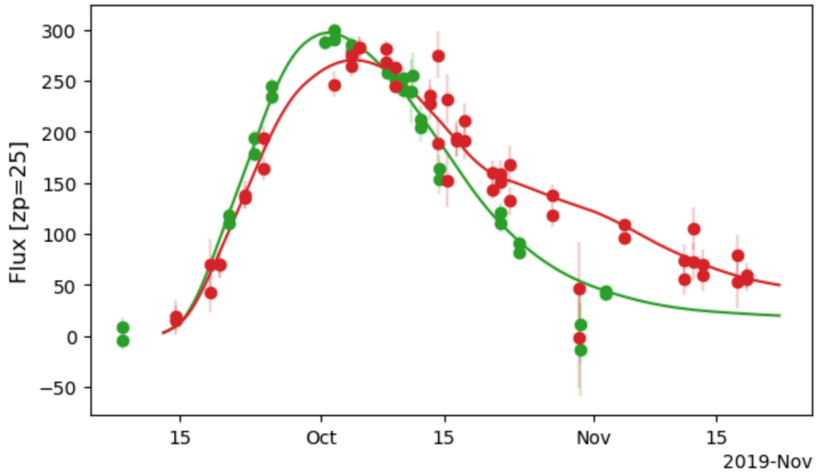
```
dset.data
```

targets .index → index; *survey* .index → fieldid

index	survey .index	fieldid	rcid	time	band	flux	fluxerr	zp	zpsys
5821	26009040	1	3	59104.390625	ztfr	-39.302373	36.143482	25.233383	ab
	26054356	1	3	59106.328125	ztfg	8.518026	14.717361	24.559206	ab
	26331019	1	3	59112.300781	ztfg	24.598649	20.259558	26.025703	ab
	26338533	1	3	59112.359375	ztfr	-3.302169	36.260182	26.127666	ab
	27121580	1	3	59128.250000	ztfr	-20.931607	46.604773	25.959711	ab
...	...	...	...	...	...	...	...	...	...
9727	30211656	1408	16	59203.179688	ztfr	14.887892	60.705860	26.204699	ab
	30211715	1408	16	59203.179688	ztfr	90.188222	61.295684	26.208666	ab
	30841192	1408	16	59221.167969	ztfr	34.804501	22.752532	26.142845	ab
	36571545	1408	16	59392.414062	ztfr	-9.636968	41.250546	26.149637	ab
	36571604	1408	16	59392.414062	ztfr	-18.475494	44.445759	26.151228	ab

8429351 rows × 8 columns

```
fig = dset.show_target_lightcurve(index=347, phase_window=[-40,100])
```



skysurvey | *easy & fast*

targets 0.5s

```
import skysurvey
snia = skysurvey.SNeIa.from_draw(10_000, zmax=0.3,
                                 tstart = "2018-04-01",
                                 tstop = "2020-12-01")
```

X

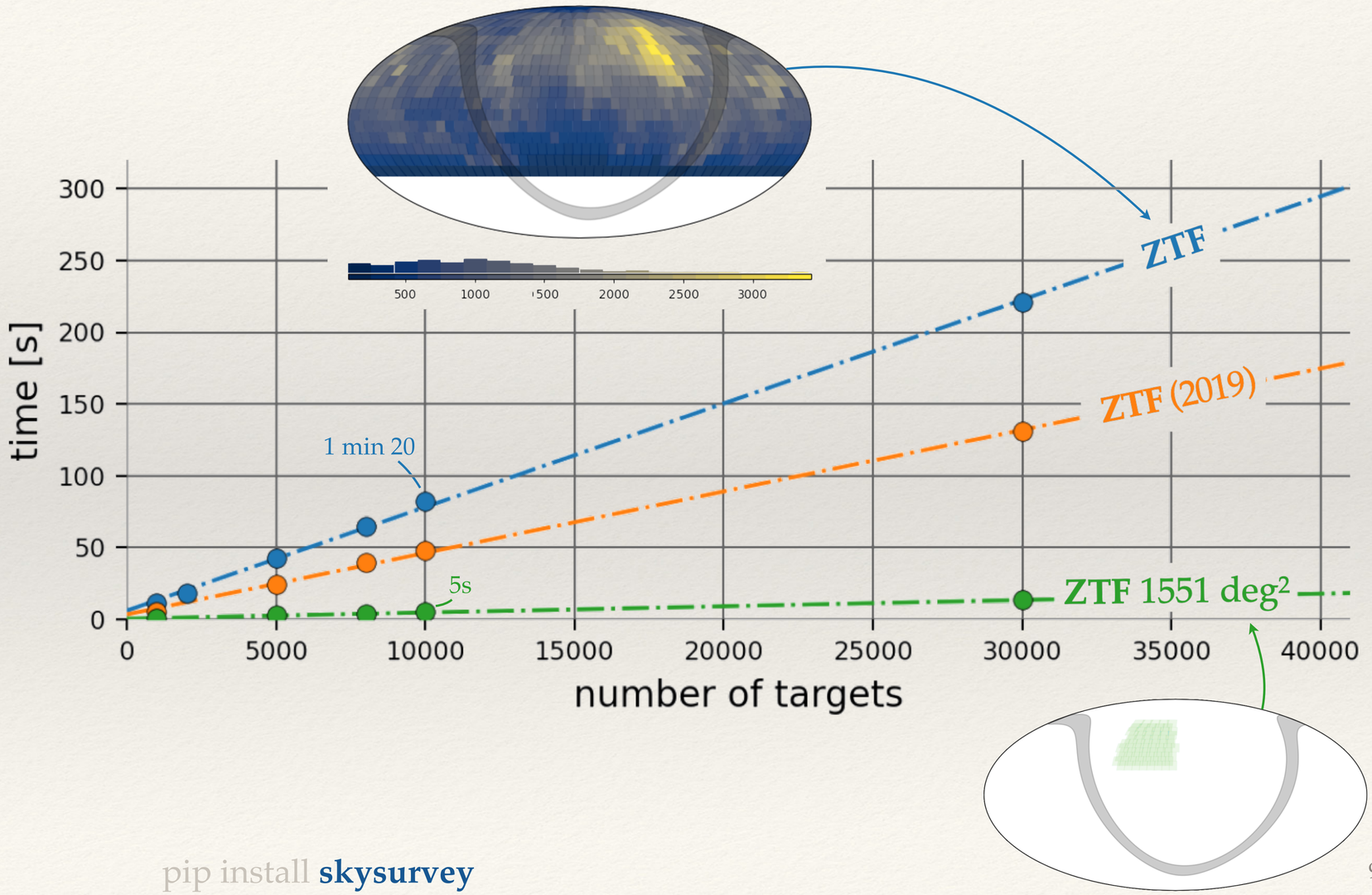
survey 1s

```
ztf = skysurvey.ZTF.from_logs()
```

47 M entries | 760 k exposures

dataset 80s

```
dset = skysurvey.DataSet.from_targets_and_survey(snia, ztf)
```



skysurvey | *easy & fast*

LSST opsim (dev)

survey 1min20

```
import skysurvey
lsst = skysurvey.LSST.from_opsim('baseline_v3.3_10yrs.db',
                                 sql_where="night<365*5")
```

119 962 089 lines DataFrame

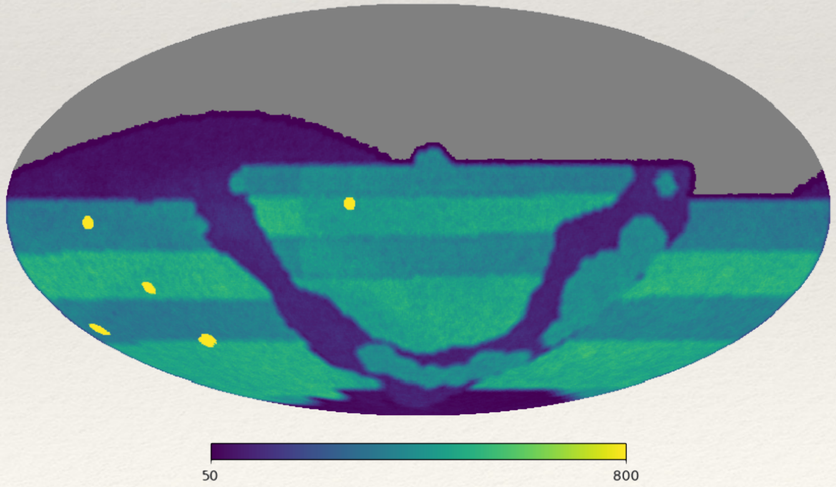
targets 1s

```
tstart, tstop = lsst.get_timerange()
snia = skysurvey.SNeIa.from_draw(100_000,
                                 template="salt2-extended",
                                 zmax=0.7,
                                 tstart=tstart, tstop=tstop)
```

dataset 3min20

```
dset = skysurvey.DataSet.from_targets_and_survey(snia, lsst)
```

100 000 SNe Ia | 5 years of LSST | 3min



skysurvey | *easy & fast*

LSST opsim (dev)

survey 1min20

```
import skysurvey
lsst = skysurvey.LSST.from_opsim('baseline_v3.3_10yrs.db',
                                 sql_where="night<365*5")
```

119 962 089 lines DataFrame

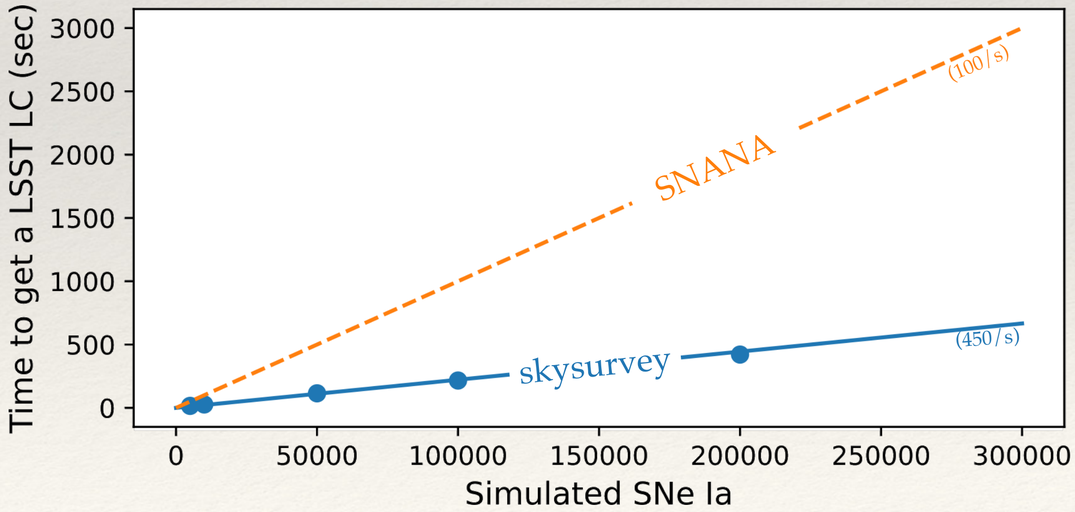
targets 1s

```
tstart, tstop = lsst.get_timerange()
snia = skysurvey.SNeIa.from_draw(100_000,
                                 template="salt2-extended",
                                 zmax=0.7,
                                 tstart=tstart, tstop=tstop)
```

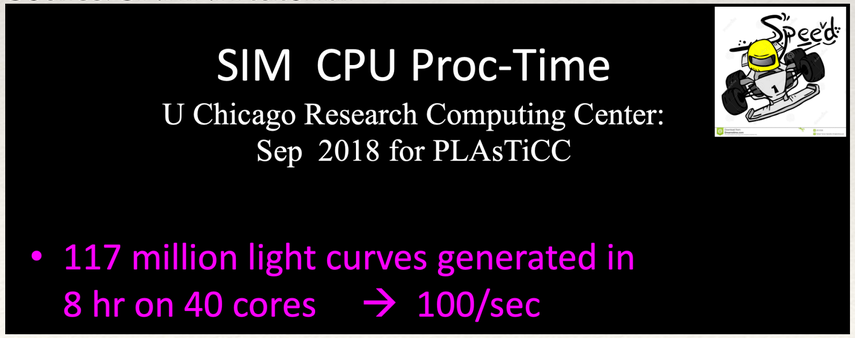
dataset 3min20

```
dset = skysurvey.DataSet.from_targets_and_survey(snia, lsst)
```

100 000 SNe Ia | 5 years of LSST | 3min



Source: SNANA tutorial



skysurvey: 450/sec | 1 laptop (M1)

skysurvey | *easy & fast*

LSST opsim (dev)

survey 1min20

```
import skysurvey
lsst = skysurvey.LSST.from_opsim('baseline_v3.3_10yrs.db',
                                 sql_where="night<365*5")
```

Customize these as you want !

targets 1s

```
tstart, tstop = lsst.get_timerange()
snia = skysurvey.SNeIa.from_draw(100_000,
                                 template="salt2-extended",
                                 zmax=0.7,
                                 tstart=tstart, tstop=tstop)
```

This will work

dataset 3min20

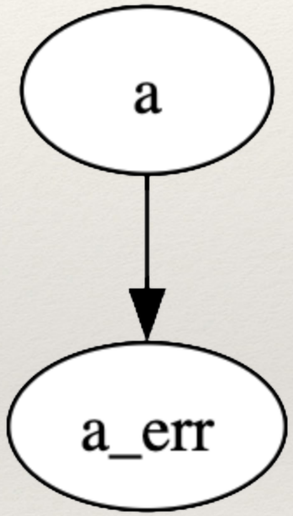
```
dset = skysurvey.DataSet.from_targets_and_survey(snia, lsst)
```

100 000 SNe Ia | 5 years of LSST | 3min

Easy to build *complex* transient

```
import numpy as np
import modeldag

def foo(value, scale=0.05, floor=0.2):
    """ the quadratic sum between value*scale and floor """
    return np.sqrt( (value*scale)**2 + floor**2)


model = {"a": {"func": np.random.normal,
               "kwargs": {"loc":10, "scale":2}
              },
         "a_err": {"func": foo,
                   "kwargs": {"value":"@a"} # leaving other param unchanged
                  }
        }

mdag = modeldag.ModelDAG(model)
mdag.visualize()
```

DAG

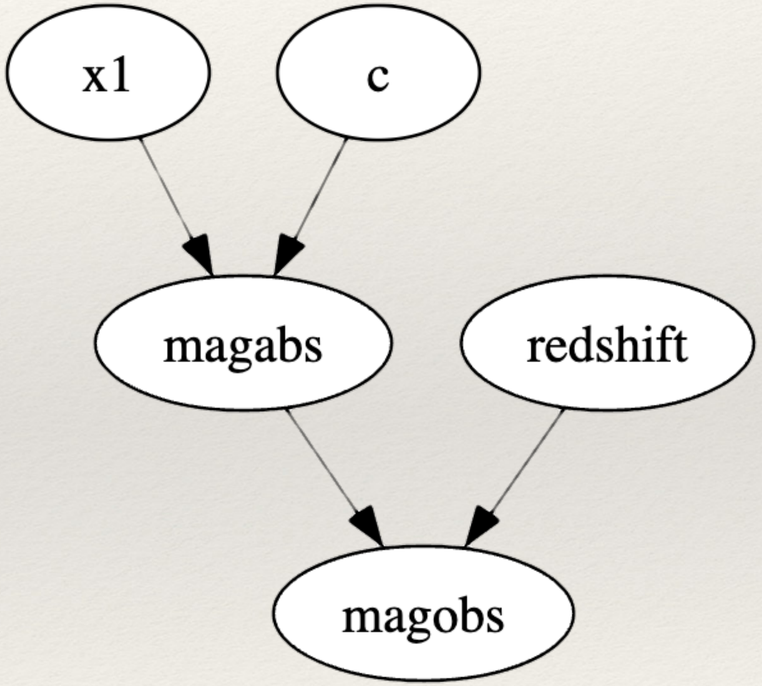


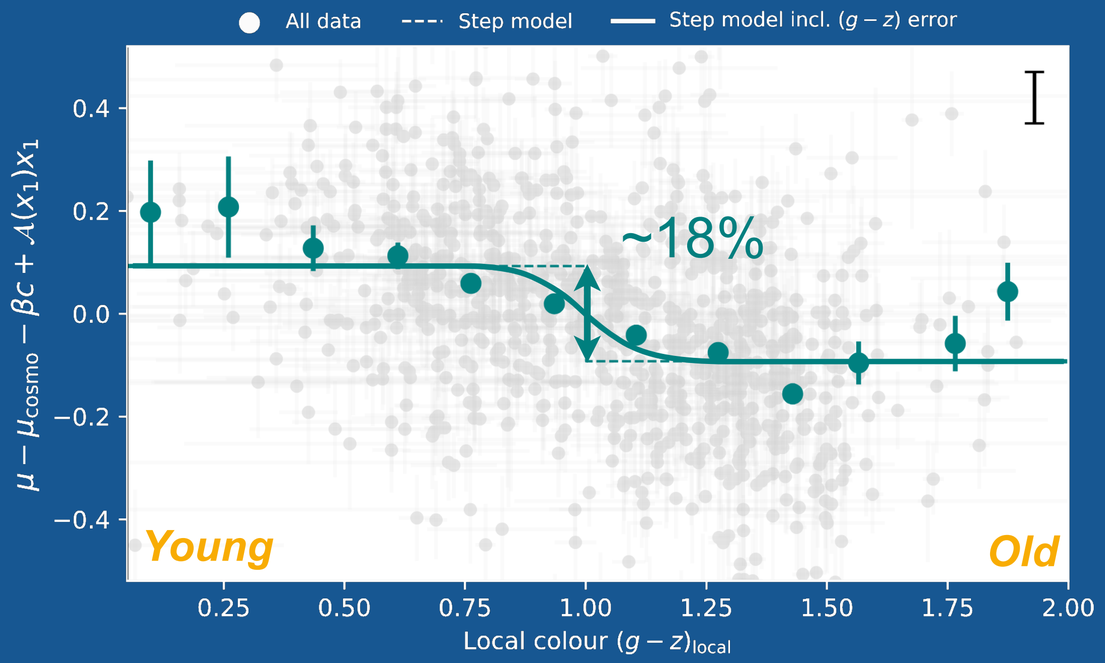
Easy to build *complex* transient

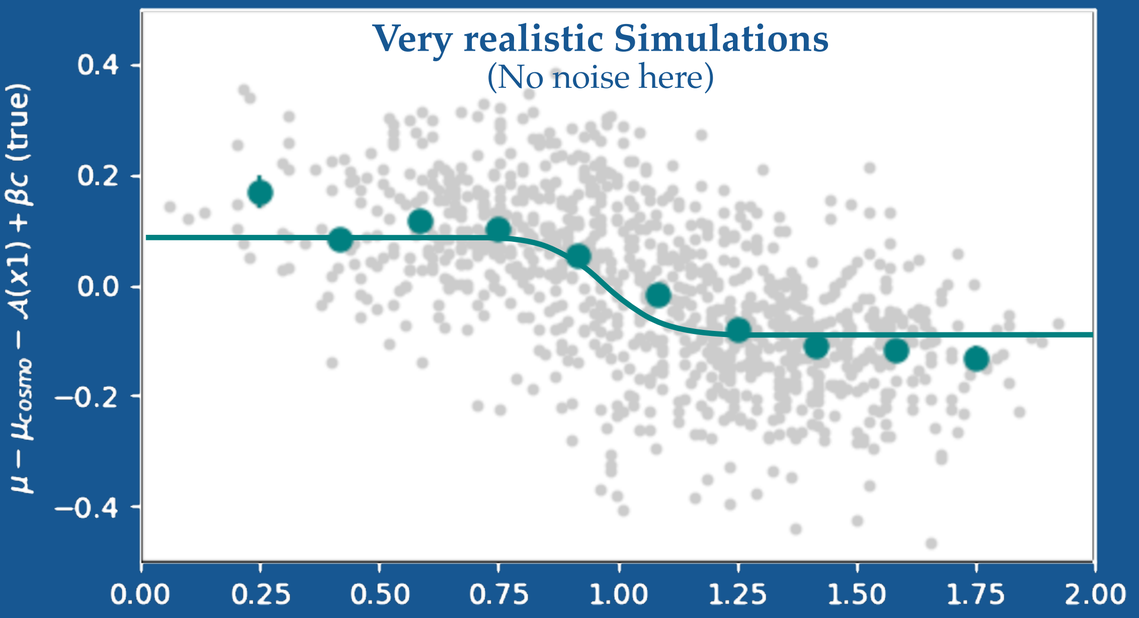
```python
# ---------------- #
#  Your functions  #
# ---------------- #
def tripp98(x1, c, alpha=-0.14, beta=3.3, sigmaint=0.1, mabs=-19.3):
    """ This is the Tripp 98 relation """
    magabs = x1*alpha + c*beta + mabs # natural SN Ia magnitude (non-standardized)
    magabs_scattered = np.random.normal(loc=magabs, scale=sigmaint) # add intrinsic scatter
    return magabs_scattered

def magabs_to_magobs(magabs, redshift, cosmology=Planck18):
    """ distance_modulus(cosmo) = m - M """
    mu = Planck18.distmod(redshift).value
    return mu + magabs

# ---------------- #
# Create the dict  #
# ---------------- #
modeldict = {"x1": {"func": np.random.normal, "kwargs": {"loc": 0, "scale":1}},
             "c": {"func": stats.lognorm.rvs, "kwargs": {"s": 0.7, "loc":-0.2, "scale":0.2}},
             "magabs": {"func": tripp98,
                        "kwargs": {"x1": "@x1", "c": "@c"}},

             "redshift": {"func": draw_from_rate},
             "magobs": {"func": magabs_to_magobs,
                        "kwargs": {"magabs":"@magabs", "redshift": "@redshift"}},
            }

# ---------------- #
#    Simulate      #
# ---------------- #
mdag = modeldag.ModelDAG(modeldict)
data_sim = mdag.draw(1_000)
```

DAG

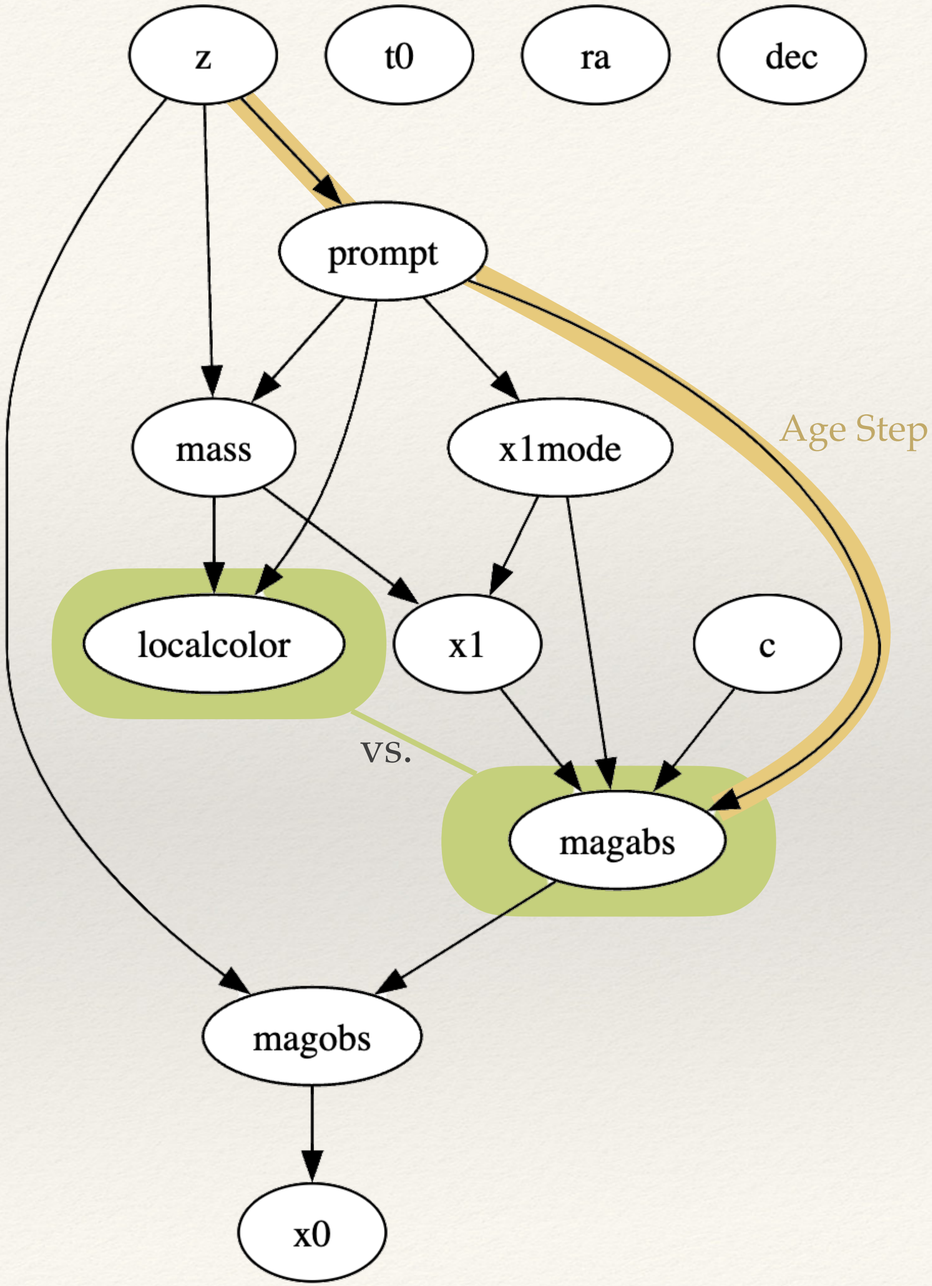
```python
mdag.visualize()
```







skysurvey | modeldag



```
snia.data
```

	z	x1	c	t0	ra	dec	magabs	mwebv	magobs	x0	template
0	0.07705	-0.030	1.221450	58487.214844	336.697937	7.605778	-15.358439	0.134286	22.429092	0.000017	salt2
1	0.06865	-0.700	-0.027583	58659.839844	60.706551	55.346775	-19.233326	0.719195	18.290983	0.000766	salt2
2	0.06905	0.320	0.351301	58782.136719	23.629887	-0.468616	-18.197527	0.036892	19.339998	0.000292	salt2

```
sn_07pk.data
```

	z	t0	magabs	ra	dec	magobs	amplitude	template
0	0.04905	56041.375000	-18.313118	213.854660	-78.848862	18.451269	1.207582e-15	v19-2007pk-corr
1	0.03315	56130.003906	-16.217342	310.827576	17.159157	19.671362	3.925357e-16	v19-2007pk-corr
2	0.04845	56179.964844	-14.626013	337.447815	37.145824	22.110720	4.150845e-17	v19-2007pk-corr

```
snii = skysurvey.SNeII.from_draw(1000)
snii.data
```

	template	z	t0	magabs	ra	dec	magobs	amplitude
0	v19-2016x-corr	0.02965	56173.551302	-17.000019	313.871430	-32.681907	18.640832	3.438910e-15
1	v19-2016bkv-corr	0.04335	56078.601439	-16.390584	294.182082	23.938362	20.096692	8.948369e-15
2	v19-asassn14jb-corr	0.02645	56139.120893	-16.300129	44.165477	-52.273437	19.087617	7.501353e-15

Customization is *easy*

rate, **redshift** range, absolute **magnitude** etc. when calling `[from_]draw()`

Pre-Built Transients

Transient template is built on top of `sncosmo`

Any `sncosmo` built-in source is available:

https://sncosmo.readthedocs.io/en/stable/source-list.html

Type Ia Supernovae

```
snia = skysurvey.SNeIa()
```

Transient Time Serie

```
sn_07pk = skysurvey.TSTransient("v19-2007pk-corr")
sn_07pk = skysurvey.TSTransient.from_draw(1000,
                                          template="v19-2007pk-corr")
```

Kilonova

```
kilonova = skysurvey.Kilonova()
```

Supernova Types

```
snii = skysurvey.SNeII.from_draw(1000)
```

"Collection Transient Time Series"

SNII	SNIIb	SNIIn	SNIb	SNIc	SNIcBL
23	12	6	12	7	6

Vincenzi+2019

See `skysurvey` doc for how to build a transient blackbody

pip install skysurvey

targets

```
import skysurvey
snia = skysurvey.SNeIa.from_draw(10_000, zmax=0.3,
                                 tstart = "2018-04-01",
                                 tstop = "2020-12-01")
```

X

survey

```
import pandas
from shapely import geometry

# footprint | 2-deg radius circle
footprint = geometry.Point(0,0).buffer(2)

# stored observing logs | ra, dec, zp, skynoise, gain, filter ...
data = pandas.read_parquet("observing_logs.parquet")

survey = skysurvey.Survey.from_pointings(data, footprint=footprint)
```

dataset

```
dset = skysurvey.DataSet.from_targets_and_survey(snia, ztf)
```

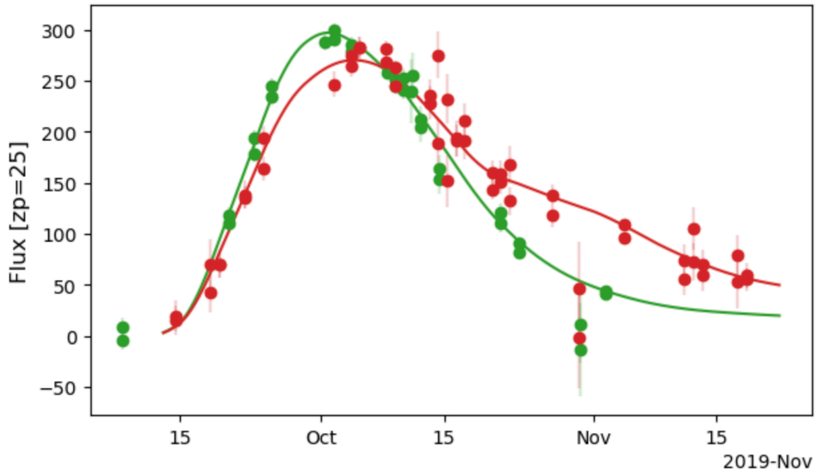
```
dset.data
```

targets .index → index; *survey* .index → fieldid

index	fieldid	rcid	time	band	flux	fluxerr	zp	zpsys
5821	26009040	1	3	59104.390625	ztfr	-39.302373	36.143482	25.233383	ab
	26054356	1	3	59106.328125	ztfg	8.518026	14.717361	24.559206	ab
	26331019	1	3	59112.300781	ztfg	24.598649	20.259558	26.025703	ab
	26338533	1	3	59112.359375	ztfr	-3.302169	36.260182	26.127666	ab
	27121580	1	3	59128.250000	ztfr	-20.931607	46.604773	25.959711	ab
...	...	...	...	...	...	...	...	...	...
9727	30211656	1408	16	59203.179688	ztfr	14.887892	60.705860	26.204699	ab
	30211715	1408	16	59203.179688	ztfr	90.188222	61.295684	26.208666	ab
	30841192	1408	16	59221.167969	ztfr	34.804501	22.752532	26.142845	ab
	36571545	1408	16	59392.414062	ztfr	-9.636968	41.250546	26.149637	ab
	36571604	1408	16	59392.414062	ztfr	-18.475494	44.445759	26.151228	ab

8429351 rows × 8 columns

```
fig = dset.show_target_lightcurve(index=347, phase_window=[-40,100])
```

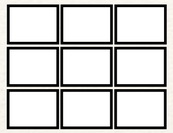


Mickael Rigault | CNRS | m.rigault@ip2i.in2p3.fr • github

Build your own Survey

footprint | *camera footprint in the sky* — `shapely`

Polygon



MultiPolygon

`skysurvey.get_footprint("des")`

data | *what was observed when, under which conditions* — `pandas`

ra, dec — Camera pointings

mjd — time in Modified Julian Date

band — name of the filter used

skynoise — level of the sky noise (unit of zp)

zp — depth of the image

gain — camera gain (e-/adu), used for noise

```python
import numpy as np
from shapely import geometry
from skysurvey.tools import utils

# footprint | 2-deg radius circle
footprint = geometry.Point(0,0).buffer(2)

# fake observations
size = 100_000 # n-pointings
data = {}
data["gain"] = 1
data["zp"] = 30
data["skynoise"] = np.random.normal(size=size, loc=200, scale=20)
data["mjd"] = np.random.uniform(58_800, 59_600, size=size)
data["band"] = np.random.choice(["desg","desr","desi"], size=size)

data["ra"], data["dec"] = utils.random_radec(size=size,
                                             ra_range=[10,350],
                                             dec_range=[-50,10])

# Load the survey
survey = skysurvey.Survey.from_pointings(data, footprint=footprint)
```

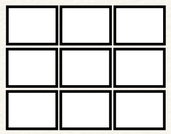
```python
# footprint | 2-deg radius circle
footprint = geometry.Point(0,0).buffer(2)

# fake observations
data = pandas.read_parquet("observing_logs.parquet")

# Load the survey
survey = skysurvey.Survey.from_pointings(data, footprint=footprint)
```

Will be used by `sncosmo` *for lightcurve realization*

Build your own GridSurvey

footprint | *camera footprint in the sky* — shapely

Polygon



MultiPolygon

`skysurvey.get_footprint("des")`

fields | *Coordinates of the grid center* — pandas

{**fieldid:** {**ra:** *value*, **dec:** *value*}}

data | *what was observed when, under which conditions* — pandas

ra, dec — Camera pointings

mjd — time in Modified Julian Date

band — name of the filter used

skynoise — level of the sky noise (unit of zp)

zp — depth of the image

gain — camera gain (e-/adu), used for noise

Will be used by `sncosmo` *for lightcurve realization*

```python
import numpy as np
import skysurvey
from shapely import geometry

# footprint
footprint = geometry.Point(0,0).buffer(2)

# fields
fields_radec = { 'C1': {'dec': -27.11161, 'ra': 54.274292+180},
                 'C2': {'dec': -29.08839, 'ra': 54.274292+180},
                 'C3': {'dec': -28.10000, 'ra': 52.648417+180},
                 'E1': {'dec': -43.00961, 'ra': 7.8744167+180},
                 'E2': {'dec': -43.99800, 'ra': 9.5000000+180},
                 'S1': {'dec':   0.00000, 'ra': 42.820000+180},
                 'S2': {'dec': -0.988389, 'ra': 41.194417+180},
                 'X1': {'dec': -4.929500, 'ra': 34.475708+180},
                 'X2': {'dec': -6.412111, 'ra': 35.664500+180},
                 'X3': {'dec': -4.600000, 'ra': 36.450000+180}
               }

# observing logs
size = 100_000
data = {}
data["gain"] = 1
data["zp"] = 30
data["skynoise"] = np.random.normal(size=size, loc=200, scale=20)
data["mjd"] = np.random.uniform(58_800, 59_600, size=size)
data["band"] = np.random.choice(["desg","desr","desi"],
                                size=size)

# - no ra, dec - #
data["fieldid"] = np.random.choice( list(fields_radec.keys()),
                                   size=size)

# Load a GridSurvey
survey = skysurvey.GridSurvey.from_pointings(data, fields_radec,
                                             footprint=footprint)
```