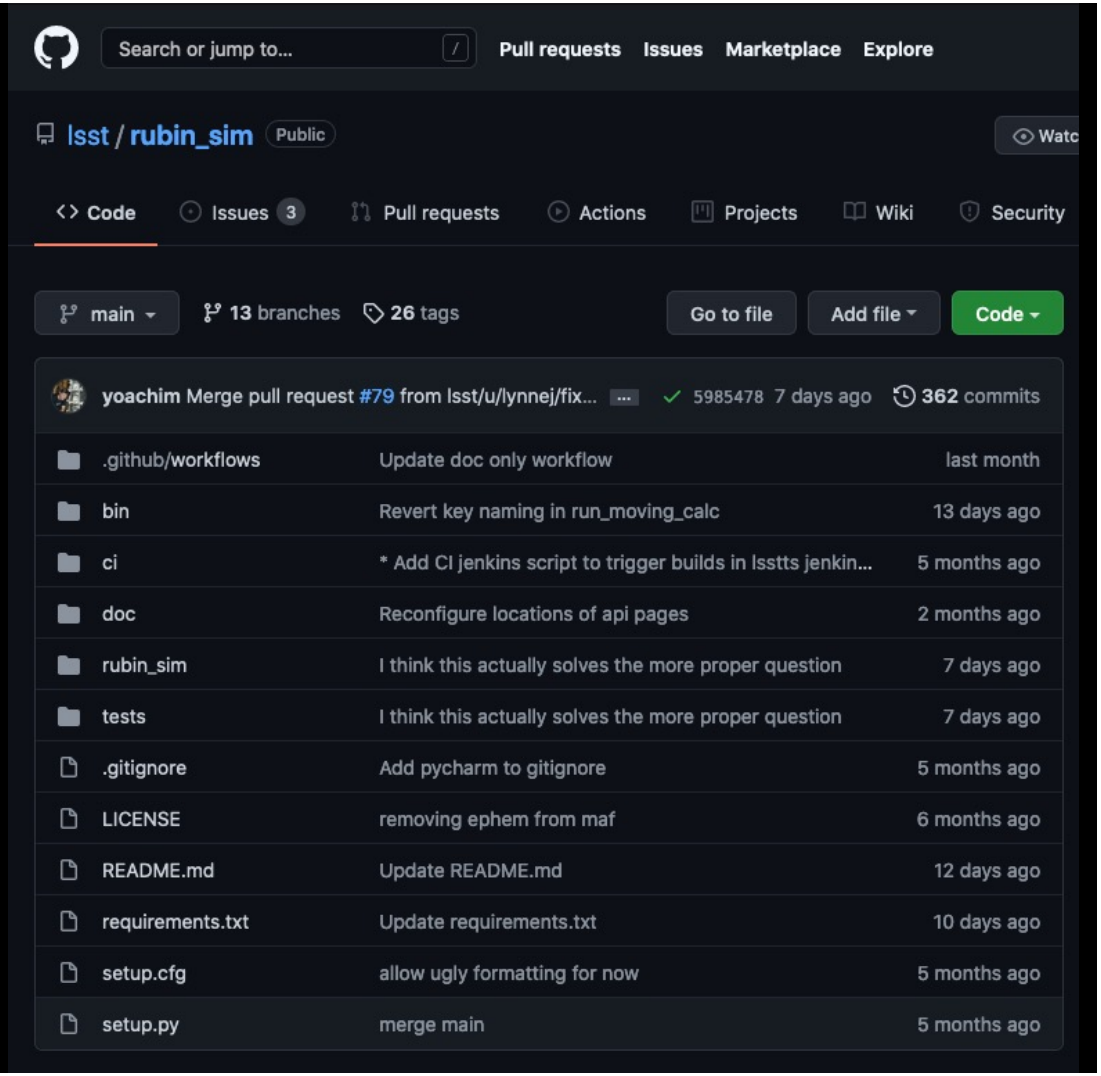
Introducing rubin_sim

Peter Yoachim
University of Washington

The Second SCOC-Science Collaborations Workshop, Nov 2021

https://github.com/lsst/rubin_sim



Our new package rubin_sim consolidates lots of repos that were previously scattered around the lsst sims

- No more DM stack dependencies, much easier to install
- Easier to work on the code, only one pull request needed
- We have CI for building docs and running unit tests

No more installing with EUPS!

If you have anaconda python, just clone and pip install

README.md

To install rubin_sim into a new conda environment (the typical use-case), set up a conda environment and pip install rubin_sim from source:

```
git clone https://github.com/lsst/rubin_sim.git
cd rubin_sim
conda create -n rubin   ### optional (but recommended)
conda activate rubin    ### optional (if new environment created above)
conda install -c conda-forge --file=requirements.txt
pip install -e .
```

```
export RUBIN_SIM_DATA_DIR=$HOME/rubin_sim_data # Optional. Set the data directory
rs_download_data  # Downloads ~2Gb of data to $RUBIN_SIM_DATA_DIR
```

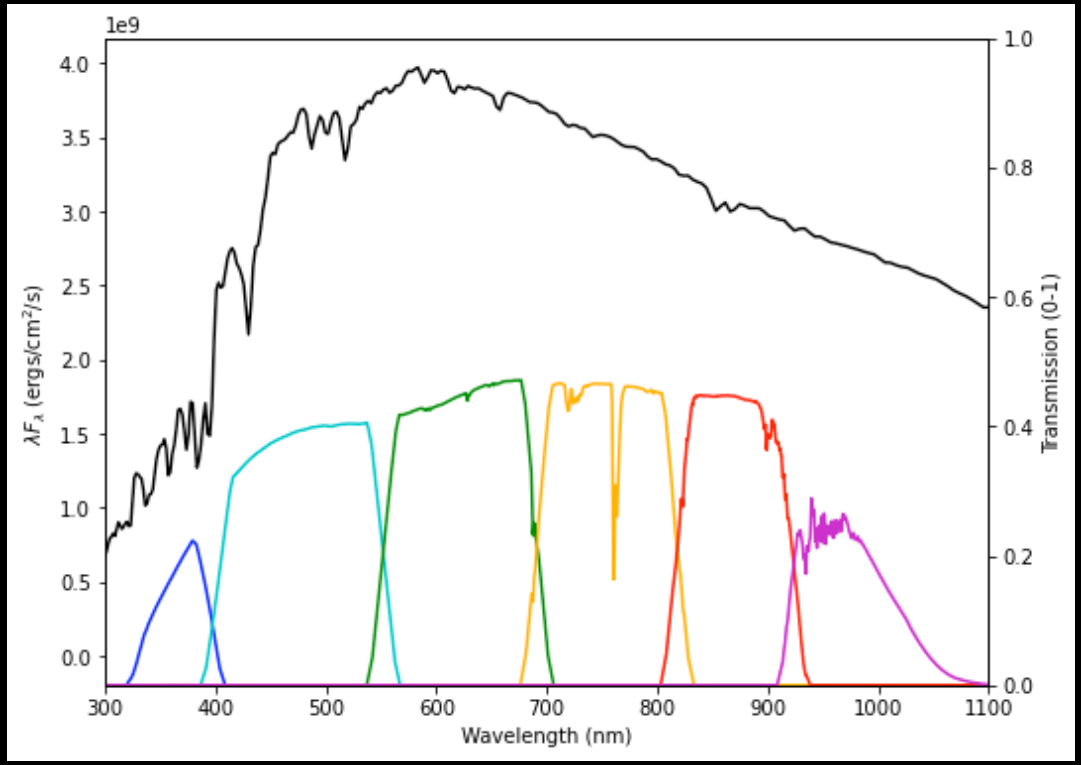
Data

- Dust maps
- Stellar density maps
- Solar system population orbits
- Filter curves
- Baseline survey strategy
- Sky brightness files
- MAF data—light curve shapes, etc

photUtils

SED and Bandpass classes

If you want to get the magnitude of an SED as observed by Rubin



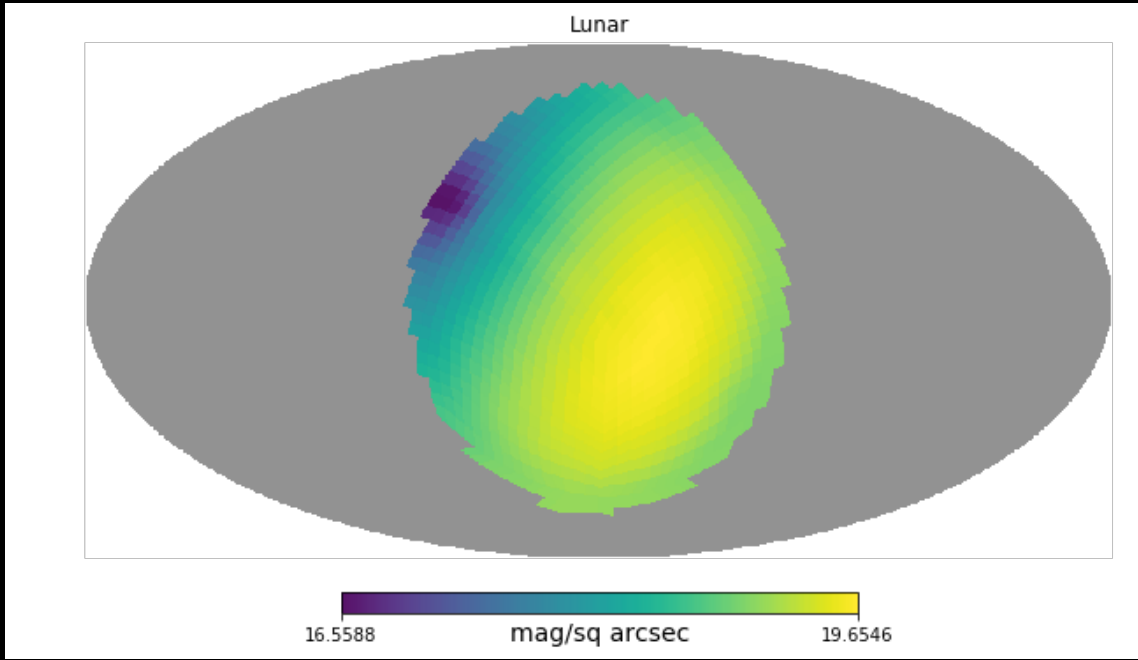
Site_models

- Cloud history
- Seeing history
- Downtime models
- Almanac info (sun/moon rise/set)

Skybrightness, skybrightness_pre

Uses ESO sky brightness plus twilight component to model sky brightness.

Computing the sky is the most computationally expensive part of scheduler simulations, so we have the pre-computed data as well (~200 Gb)

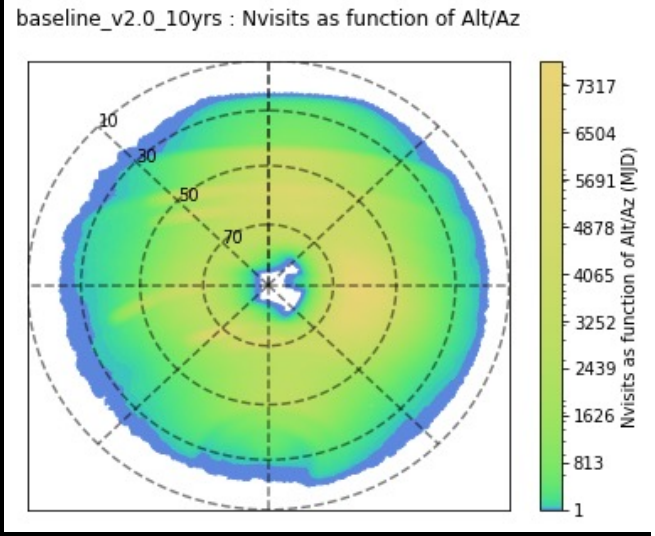


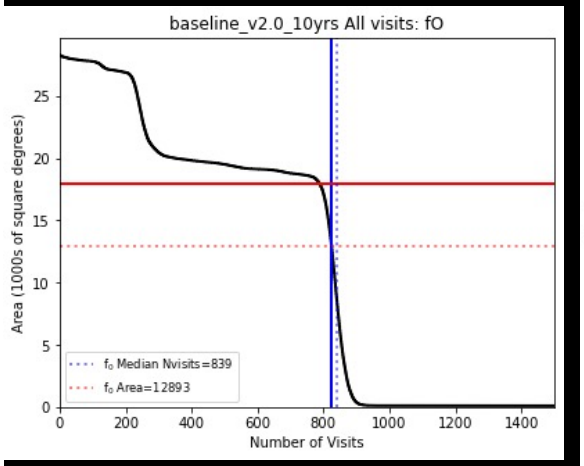
Scheduler

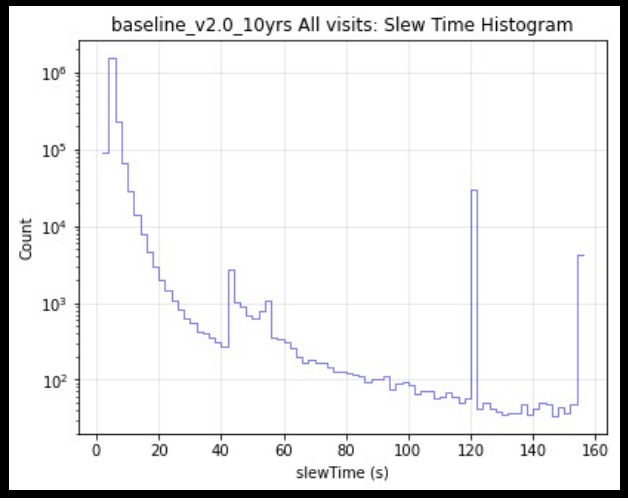
- Model observatory
- Scheduler
- Utils for running cadence simulations

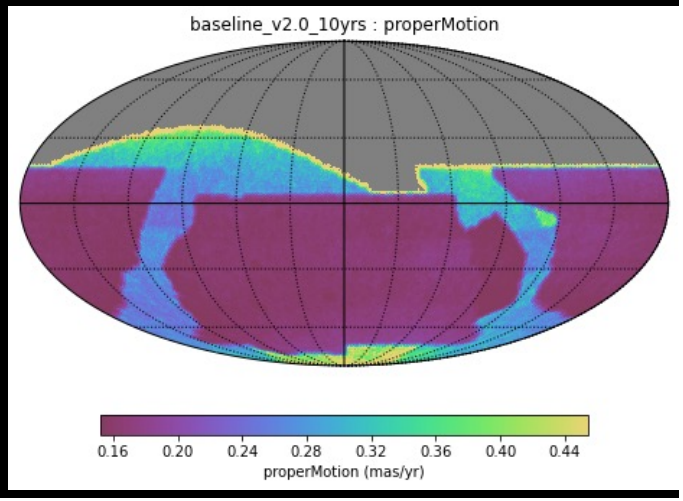
MAF

Our framework for analyzing survey pointing histories.









movingObjects

Tools for matching solar system objects to observations

Utils

- Coordinate transformations
- HEALpix utilities
- SNR functions
- Handy Rubin specific values
  - DDF locations
  - Site location
  - Camera footprint
  - Telescope zeropoints

What's changed in the migration to an independent package?

Not much.

- Now use the full focal plane with gaps by default in MAF
- Minor update to dithering in the scheduler (ensures proper dithering if we stop and restart)

This does mean you can't do perfect apples-to-apples comparison of old MAF analysis and old simulations with current MAF and sims

How to contribute

- Fork rubin_sim on github (creates your own copy of the repository)
- Clone the forked repo to your work area, edit code there
- Add your code to your forked repo (git add … ; git commit …) and push changes to your forked repo as usual (git push)
- Issue a pull request (PR) from your fork to the lsst/rubin_sim repository

We also have https://github.com/lsst/rubin_sim_notebooks

With example notebooks. It's great if folks can also add a notebook demonstrating their code. Jupyter notebooks are nice because you can add text, LaTeX, links, embed plots, etc.

Upcoming work on rubin_sim

- New metrics from the community
- Updates for survey operations
  - Faster 1-day simulation
  - swap out historical seeing, etc for real telemetry
- Always need more documentation
- Better formatting uniformity (pep8, camel_Case, etc)
- Check unit test coverage, deprecate unused scheduler classes

Hopefully being easier to install means the community doesn't have to rely as much on services like NOIRlab and can install and develop code locally.