Metric Analysis Framework (MAF)

Overview

Owen Boberg, Postdoc Research Associate, University of Washington

Large Synoptic Survey Telescope

My introduction

Chances are I am a new face…

- Relatively new to the project (~ 1 year at UW)
- Main duties
  - Work on scheduler and MAF development
  - Run OpSim simulations testing new observing strategies
    - Currently preparing runs for the call for white papers
  - Help communicate the results of new simulations to the project team and community
  - Help community use MAF to analyze and critique the simulations
  - Eventually run hundreds of survey simulations
  - Maintain docker images and documentation for MAF and OpSim
    - Please feel free to talk to me about Dockerizing other parts of LSST stack or any other software

What is MAF?

Python framework for understanding how telescope scheduling effects survey performance

- Developed to be used with Operations Simulator (OpSim) simulated surveys
  - minion_1016 is an OpSim survey
  - baseline2018a will soon be the official replacement
- MAF can be applied to a range of datasets
  - Just a matter of correctly formatting the data
- Comes with a large number of built in Metrics
  - Easily to write custom metrics using python Classes
  - Group similar metrics into batches
  - See Writing A New Metric.ipynb in tutorial notebooks
- Used to determine if a simulated survey meets the Science Requirements Document (SRD)
  - https://www.lsst.org/scientists/publications/science-requirements-document
  - Important when developing white papers
- Originally development by Lynne Jones et al.
  - https://spie.org/Publications/Proceedings/Paper/10.1117/12.2056835

OpSim database contents

sqlite3 baseline2018a.db

- List of tables in OpSim sqlite databases
  - Config, Field, ObsExposures, ObsHistory, ObsProposalHistory, Proposal, ProposalField, ScheduledDowntime, SessionSlewActivities, SlewFinalState, SlewHistory, SlewInitialState, SlewMaxSpeeds, TargetExposures, TargetHistory, TargetProposalHistory, UnscheduledDowntime, **SummaryAllProps**
  - MAF can access all of these for various metrics
- **SummaryAllProps** will likely be the most useful when creating new metrics

SummaryAllProps

Columns

observationId
night
observationStartTime
observationStartMJD
observationStartLST
numExposures
visitTime
visitExposureTime
proposalId
fieldId
fieldRA
fieldDec

altitude
azimuth
filter
airmass
skyBrightness
cloud
seeingFwhm500
seeingFwhmGeom
seeingFwhmEff
fiveSigmaDepth
slewTime
slewDistance
paraAngle

rotTelPos
rotSkyPos
moonRA
moonDec
moonAlt
moonAz
moonDistance
moonPhase
sunAlt
sunAz
solarElong

New columns can be added when running metrics using MAF Stackers

- Hour Angle, NormAirmassStacker, etc.
- See Stackers.ipynb in tutorital notebooks

What is MAF?

List of all Opsim Runs

Run List | Opsim Configuration | Metrics List | All Results | Multi Color | Summary Stats

OpsimRun	OpsimGroup	MafComment	OpsimComment	SQLite File	ResultsDb	MafDir
astro-lsst-01_2013	V4 features	Cadence	10 yr, HAbonus 0.5, HAmax 3, all props	astro-lsst-01_2013.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2013/ca
astro-lsst-01_2013	V4 features	SRD	10 yr, HAbonus 0.5, HAmax 3, all props	astro-lsst-01_2013.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2013/sr
astro-lsst-01_2013	V4 features	Standard	10 yr, HAbonus 0.5, HAmax 3, all props	astro-lsst-01_2013.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2013/al
astro-lsst-01_2016	V4 features	Cadence	All defaults again	astro-lsst-01_2016.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2016/ca
astro-lsst-01_2016	V4 features	SRD	All defaults again	astro-lsst-01_2016.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2016/sr
astro-lsst-01_2016	V4 features	Standard	All defaults again	astro-lsst-01_2016.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2016/al
astro-lsst-01_2020	V3-ish	Cadence	Turn off all new features	astro-lsst-01_2020.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2020/ca
astro-lsst-01_2020	V3-ish	SRD	Turn off all new features	astro-lsst-01_2020.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2020/sr
astro-lsst-01_2020	V3-ish	Sched (old)	Turn off all new features	astro-lsst-01_2020.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2020/sc
astro-lsst-01_2020	V3-ish	Sci (old)	Turn off all new features	astro-lsst-01_2020.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2020/sc
astro-lsst-01_2020	V3-ish	Standard	Turn off all new features	astro-lsst-01_2020.db.gz	ResultsDb	/local/lsst/opsim/baselines/astro-lsst-01_2020/al

- http://astro-lsst-01.astro.washington.edu:8081/
- Just a tool in MAF

What is MAF?

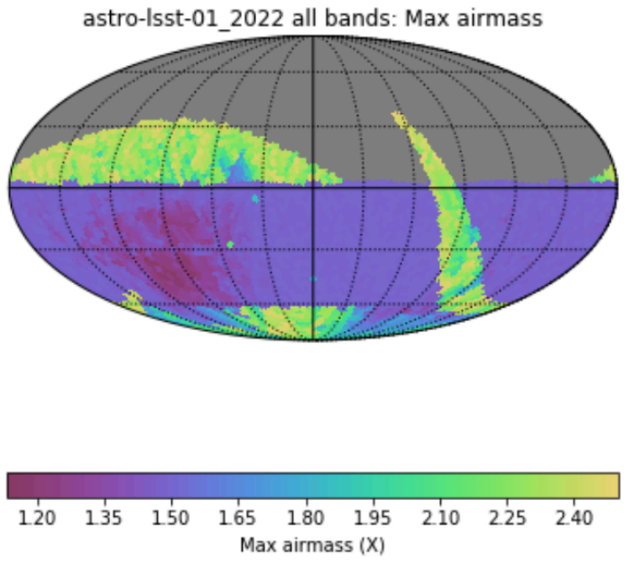
OpSim Run: baseline2018a

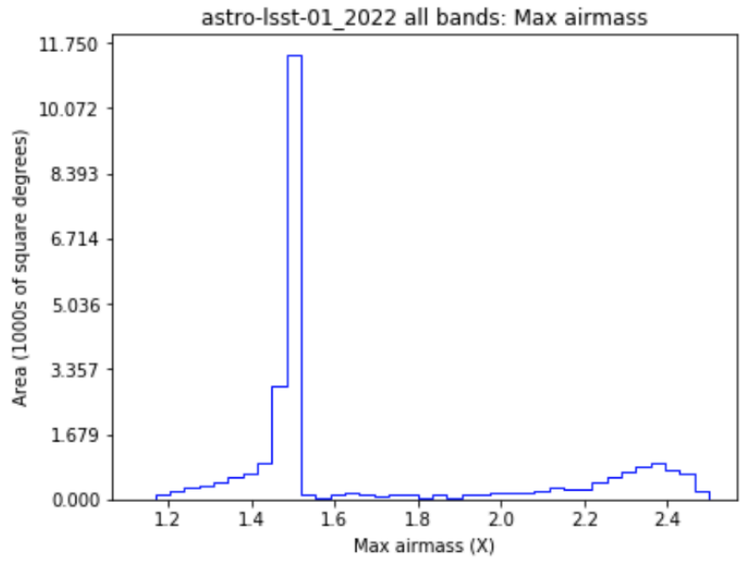
Run List | Opsim Configuration | Metrics List | All Results | Multi Color | Summary Stats

- Airmass
  - All visits
  - WFD
- Alt/Az
  - All Observations
  - Per filter
- Coadded m5 Maps
  - All visits
  - WFD
- Filter Changes
  - Per Night
  - Whole Survey
- Fivesigmadepth
  - All visits
  - WFD
- Ha
  - All visits
  - WFD
- Hourglass
  - Year 1
  - Year 10
  - Year 2
  - Year 3
  - Year 4
  - Year 5
  - Year 6
  - Year 7

Group: Airmass; Subgroup: All visits

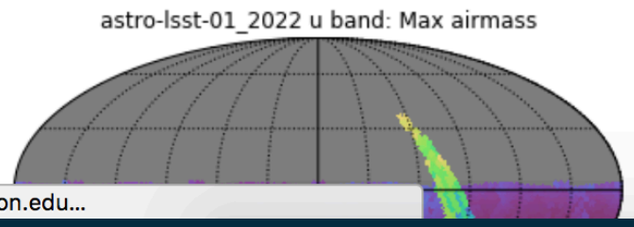
Max airmass HealpixSlicer all bands npz JSON

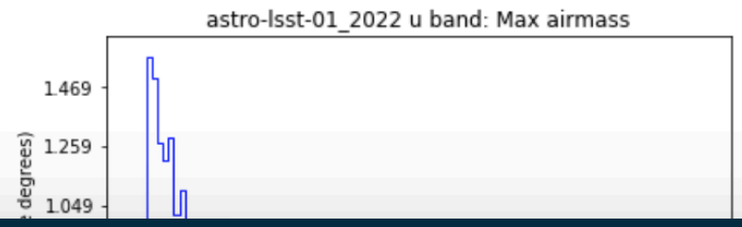




Map of Max airmass for all bands.

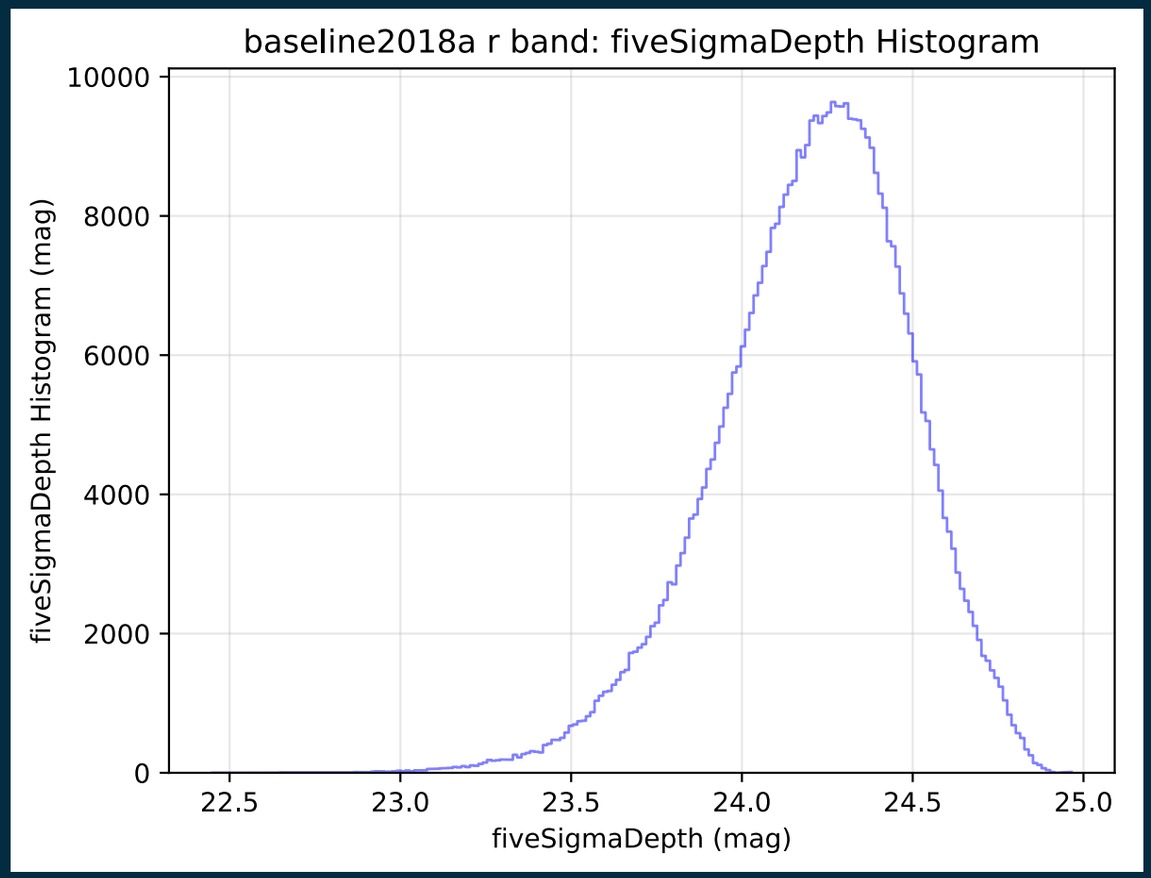
Max airmass HealpixSlicer u band npz JSON

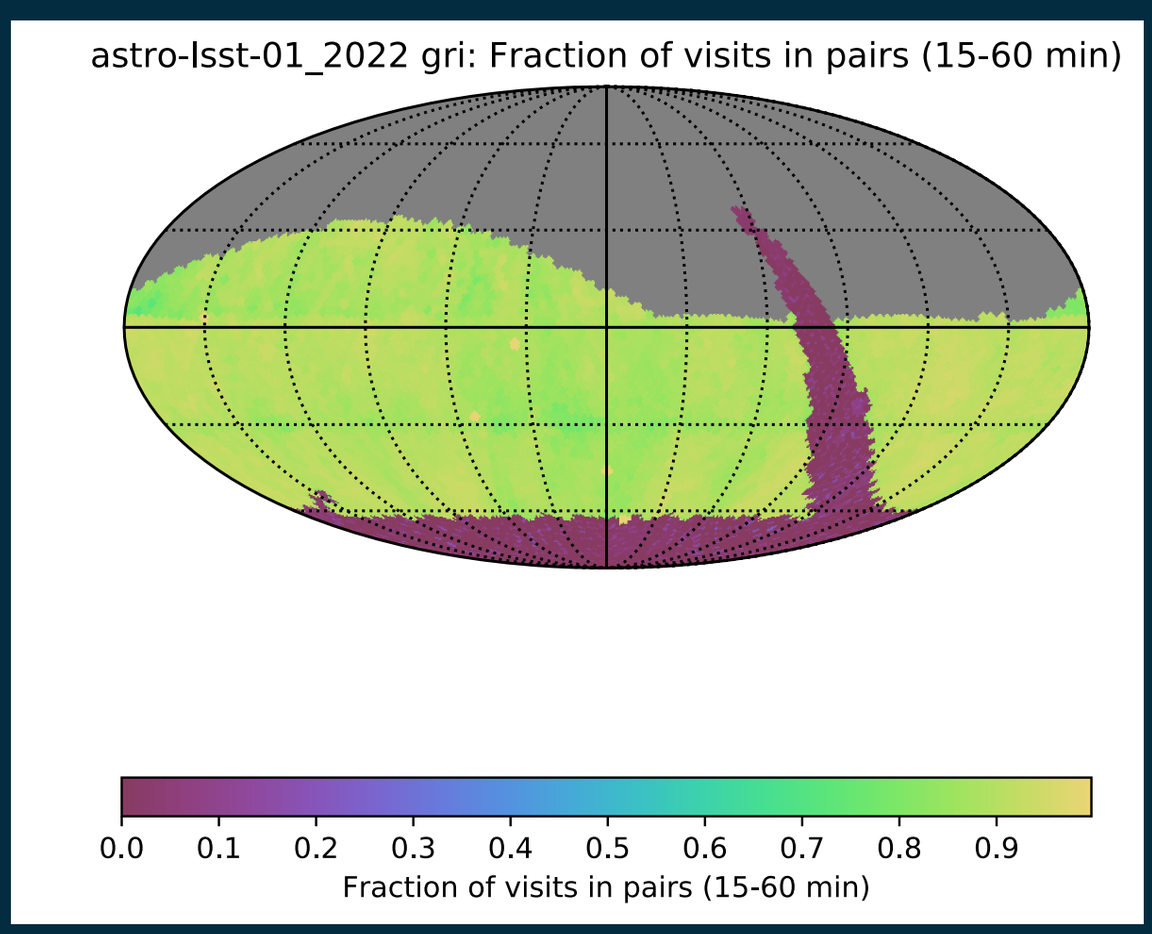


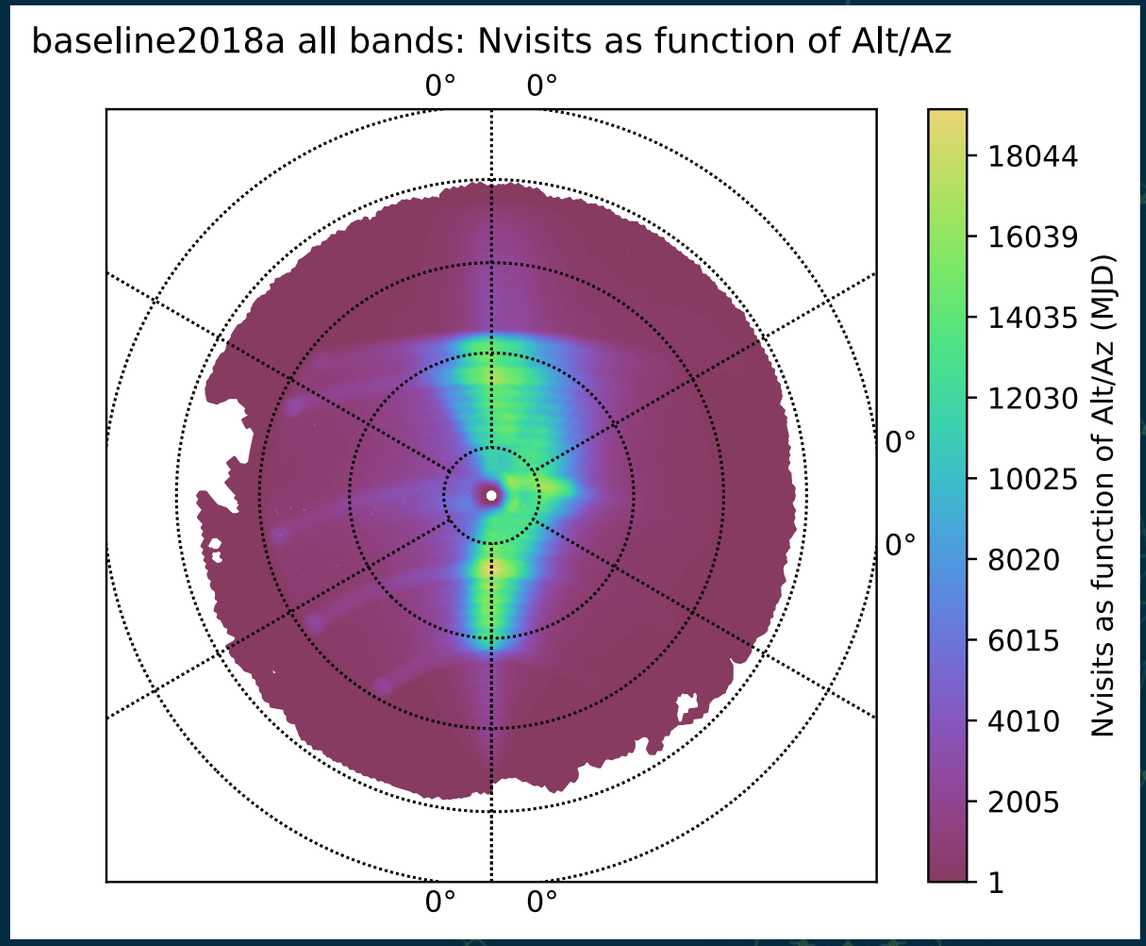


Waiting for astro-lsst-01.astro.washington.edu...

- http://astro-lsst-01.astro.washington.edu:8081/
- Just a tool in MAF


baseline2018a r band: fiveSigmaDepth Histogram
fiveSigmaDepth Histogram (mag)
10000
8000
6000
4000
2000
0
22.5
23.0
23.5
24.0
24.5
25.0
fiveSigmaDepth (mag)


astro-lsst-01_2022 gri: Fraction of visits in pairs (15-60 min)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Fraction of visits in pairs (15-60 min)


baseline2018a all bands: Nvisits as function of Alt/Az
0°
0°
0°
0°
0°
0°
18044
16039
14035
12030
10025
8020
6015
4010
2005
1
Nvisits as function of Alt/Az (MJD)

What does MAF do?

High level summary:

Connect to OpSim survey simulation databases

- OpSim = Operations Simulator
  - Combination of sims_ocs (simulated observatory control system) and ts_scheduler (scheduling algorithm deciding what to observe)
- Databases are sqlite

SQL query and slice the database once it is read into memory

- Query chooses the data (e.g filter is z)
- Slicing groups the queried data (e.g by night or position on the sky)

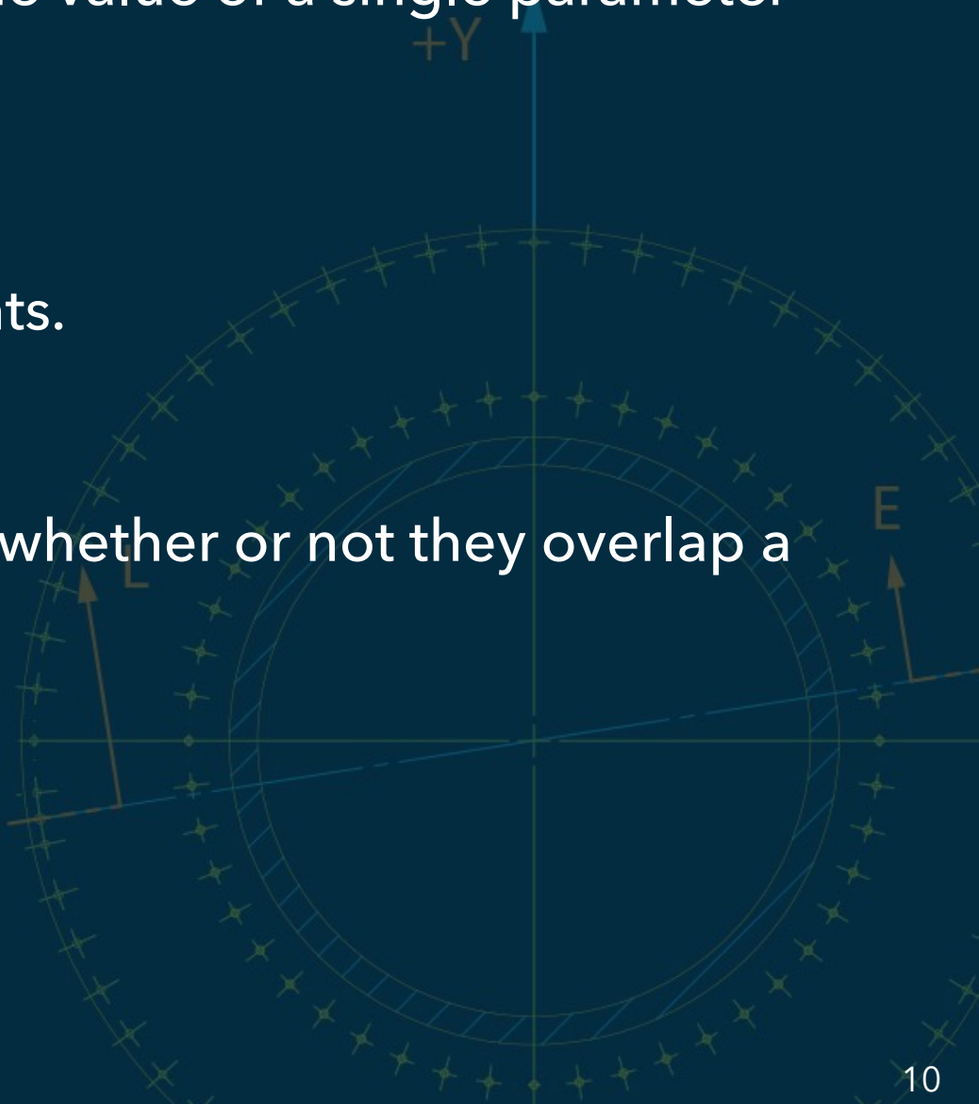
Calculate metrics and summary statistics on the selected and sliced data

- Use an empty sql query to select all data
- Slice the data based position in the sky (Healpix)
- Count the total number of visits at each HealPix over 10 year survey

What does MAF do?

3 main slicers

- The Unislicer simply clumps all visits into one group.
  - All the way down the index of the database
  - **Count total number of visits**
- The OneDSlicer groups visits into subsets based on the value of a single parameter from the OpSim data.
  - Example: night
  - Can also set a bin size, for example bins of 10 nights.
  - **Count number of visits per night bin**
- The HealpixSlicer groups visits into subsets based on whether or not they overlap a given healpixel.
  - Can set the resolution of the healpixels
  - **Count number of visits per healpix on sky**
- See Slicers.ipynb in tutorial notebooks



Let's walk through the building blocks of the MAF grammar

(See Introduction Notebooks.ipynb)

Large Synoptic Survey Telescope

The building blocks of MAF

Metric Bundle (Nvisits)

MetricBundle(**metric1, slicer1, sqlconstraint**)

metric1

CountMetric(observationStartMJD)

slicer1

HealpixSlicer(nside=64)

sqlcontstraint

sqlconstraint = ‘filter = "r"’

opsdb connection

opsdb = OpsimDatabase(`baseline2018a.db)

The building blocks of MAF

Metric Bundle Group

Metric Bundle (Nvisits)

metric1

slicer1

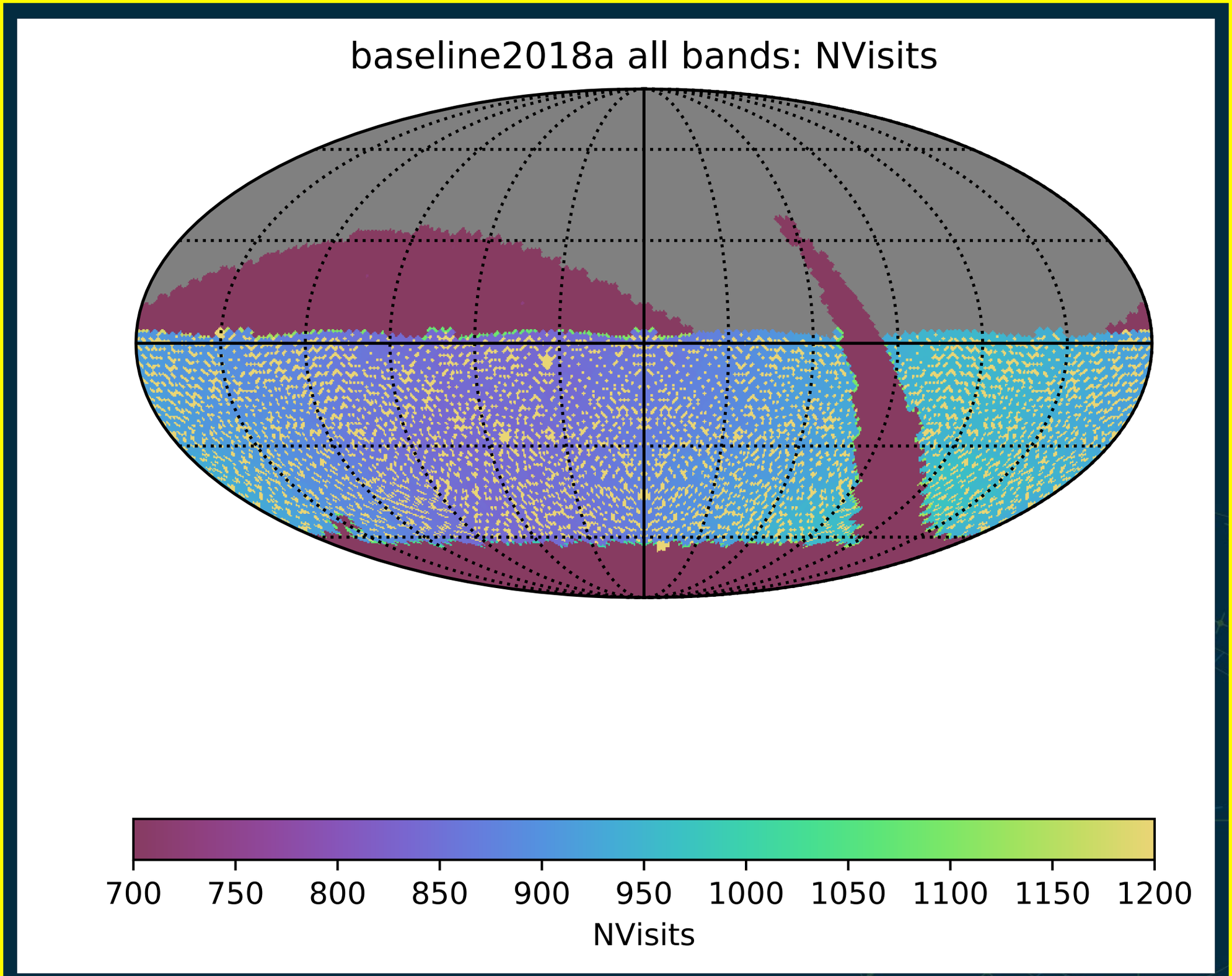
sqlcontstraint

opsdb connection

opsdb

Group = MetricBundleGroup({`**nvisits**`:**Nvisits**}, **opsdb**)

Group.runAll() Group.plotAll()



The building blocks of MAF

Metric Bundle Group

bundleDict

(Nvisits)

CountMetric

HealpixSlicer

sqlcontstraint

(maxairmass)

MaxMetric

HealpixSlicer

sqlcontstraint

opsdb

opsdb = db.OpsimDatabase(`baseline2018a.db)

bundleDict = {‘nvisits’**:Nvisits,**‘maxchi’**:maxairmass}**

Group = MetricBundleGroup(**bundleDict, opsdb**)

The building blocks of MAF

Metric Bundle Group

bundleDict

(Nvisits)

CountMetric

HealpixSlicer

sqlcontstraint

(NvisitsNight)

CountMetric

OneDSlicer

sqlcontstraint

opsdb

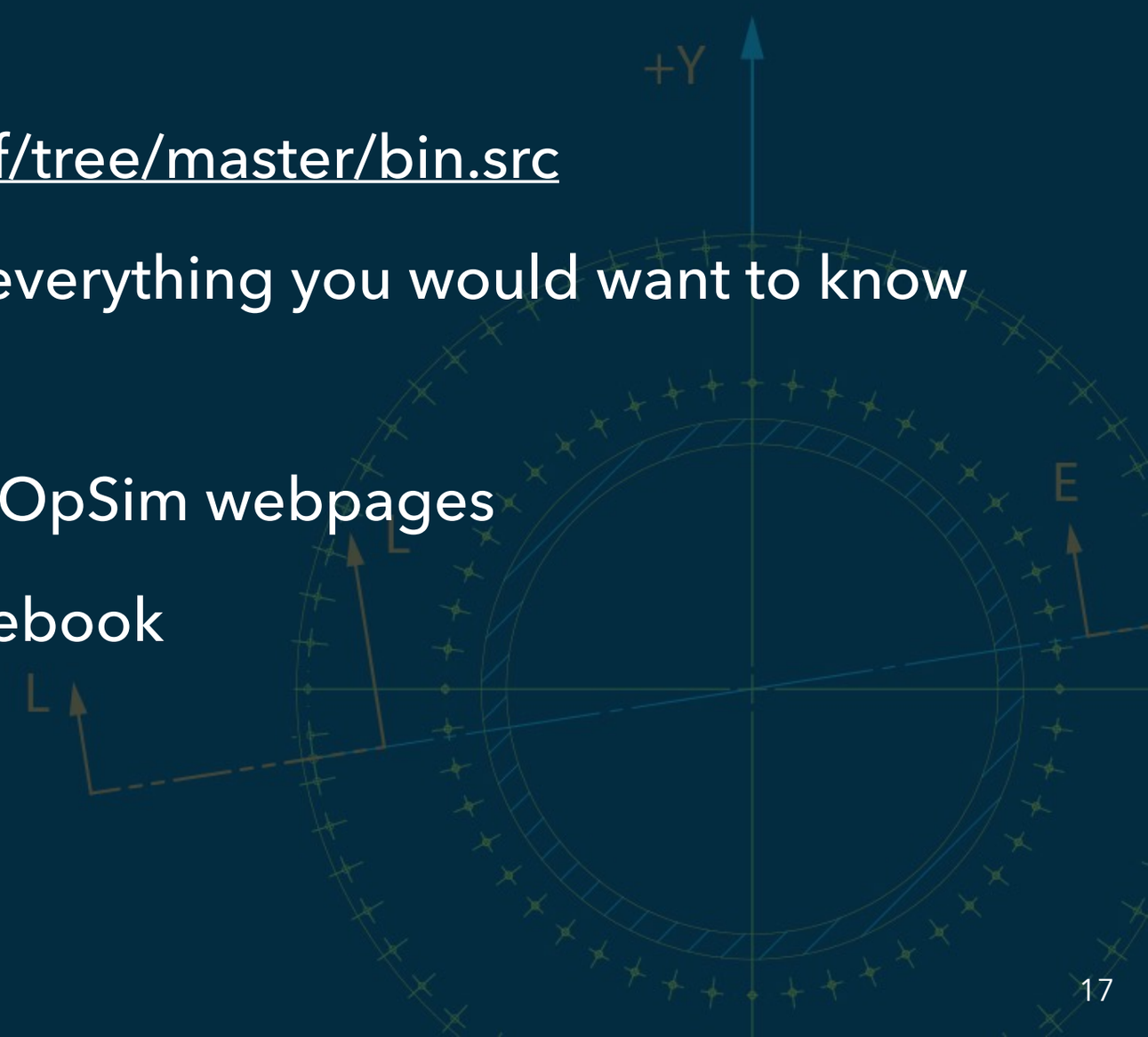
opsdb = db.OpsimDatabase(`baseline2018a.db)

bundleDict = {‘nvisits’**:Nvisits,**’nvisitsnight’**:NvisitsNight}**

Group = MetricBundleGroup(**bundleDict, opsdb**)

MAF batches

- We have recently put together batches of metric bundles to run small and/or large sets of related metrics.
  - https://github.com/lsst/sims_maf/tree/master/python/lsst/sims/maf/batches
  - https://github.com/lsst/sims_maf/tree/master/bin.src
- These batches calculate just about everything you would want to know about a simulated survey.
  - Produce the output seen on the OpSim webpages
- The batches still need a tutorial notebook



Batch output

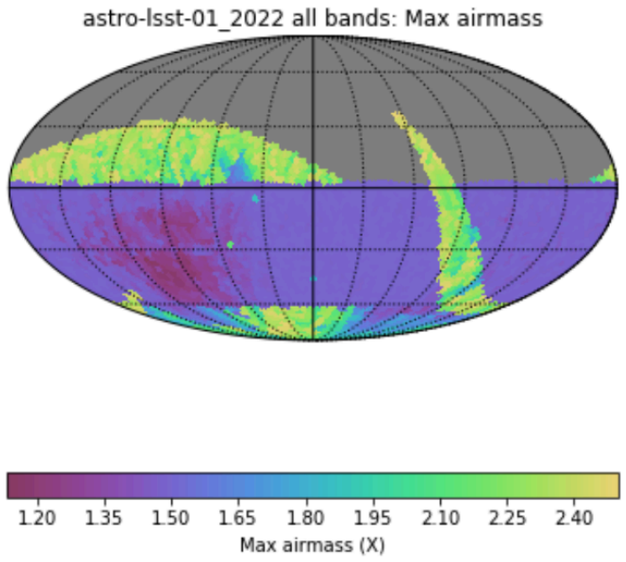
OpSim Run: baseline2018a

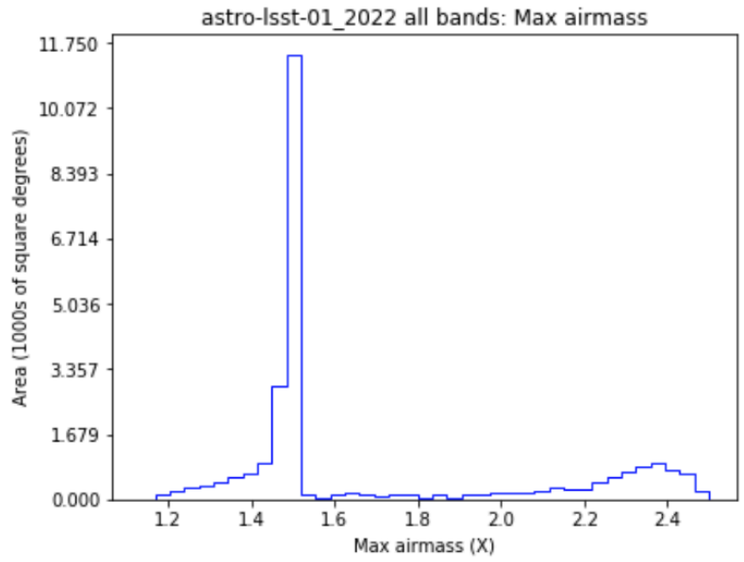
Run List | Opsim Configuration | Metrics List | All Results | Multi Color | Summary Stats

- Airmass
  - All visits
  - WFD
- Alt/Az
  - All Observations
  - Per filter
- Coadded m5 Maps
  - All visits
  - WFD
- Filter Changes
  - Per Night
  - Whole Survey
- Fivesigmadepth
  - All visits
  - WFD
- Ha
  - All visits
  - WFD
- Hourglass
  - Year 1
  - Year 10
  - Year 2
  - Year 3
  - Year 4
  - Year 5
  - Year 6
  - Year 7

Group: Airmass; Subgroup: All visits

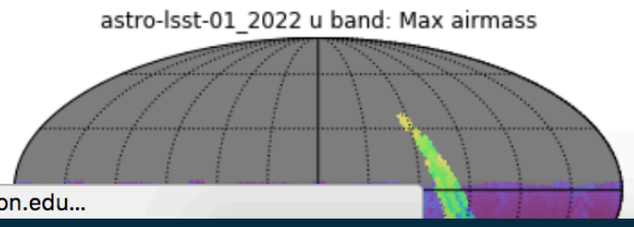
Max airmass HealpixSlicer all bands npz JSON

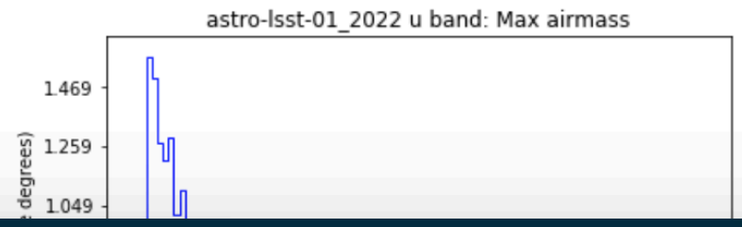




Map of Max airmass for all bands.

Max airmass HealpixSlicer u band npz JSON





Waiting for astro-lsst-01.astro.washington.edu...

- http://astro-lsst-01.astro.washington.edu:8081/

Why not just use something like pandas?

- Pandas can read sqlite
- Can use groupby and filtering to get similar functionality to slicers
- Separate from the LSST stack....
- MAF provides a high level of provenance and reproducibility
  - Slice once, measure multiple metrics
  - Metric results are saved as numpy npz files
    - Save info about the metric data, slicer used, and slice points
    - Can be read from disk for re-plotting and examining summary statistics
    - Easily compare different OpSim runs (tutorial notebook coming soon)
  - A results databased is produced when running the metrics
    - Store metric names, plots created, and summary statistics
- Trying to recreate MAF output with pandas can be a useful exercise

Useful links

We are currently improving and compiling documentation

MAF

- **Installation :** https://confluence.lsstcorp.org/display/SIM/Catalogs+and+MAF
- **Help :** https://sims-maf.lsst.io/index.html, https://github.com/LSST-nonproject/sims_maf_contrib/tree/master/tutorials
- **Github :** https://github.com/lsst/sims_maf
- **Docker :** https://hub.docker.com/r/oboberg/maf/
- **Results webpages :** http://astro-lsst-01.astro.washington.edu:8081/, http://astro-lsst-01.astro.washington.edu:8080/

OpSim

- **Installation :** https://lsst-sims.github.io/sims_ocs/
- **Help :** https://lsst-sims.github.io/sims_ocs/
- **Github :** https://github.com/lsst-sims/sims_ocs, https://github.com/lsst-ts/ts_scheduler
- **Docker :** https://hub.docker.com/r/oboberg/opsim4_fbs_py3/, https://github.com/oboberg/docker_readmes/blob/master/opsim4_docker/README.md

Thanks!

Thank you Federica and Rachel for organizing and the invite

- I am here to help people get up to speed with MAF through Wednesday.
- Looking forward to getting feedback on what other metrics need to be developed.
- I am also happy to give help with Docker for MAF and OpSim