Hackathon Demos Documentation

Release 1.0

Owen Boberg

Jun 05, 2021

Contents

Here you will find some basic instructions for running MAF and OpSim using Docker containers. Please see Docker and Directories sections before starting with MAF or OpSim.

CHAPTER 1

Docker installation and configuration

1.1 Installation

Please follow the install instructions provided by Docker that match your operating system.

1. Docker for Mac
2. Docker for Windows
3. Docker for Ubuntu

Note: Those using Linux also see think link

1.2 Memory Allocation for Mac and Windows

MAF and OpSim can use quite a lot of memory and CPU when running. To ensure your containers do not crash Docker will need to be configured to have access to more memory. Navigate to the Docker settings on your system, then the advanced tab. Increase the `Memory` bar to 6 - 8 GB if your system has that much.

Note: Running a full 10 year simulation typically requires `25GB` of memory so it is unlikely that you will be able to run long simulations on your laptop.

CHAPTER 2

Directories and GitHub repos

Setting up the following directory structure, and cloning these repos, will make it easier to follow the tutorials provided in this documentation. These paths can be changed to whatever you like, just be sure to update the examples accordingly.

1. Setup what will become our top working directory `~/flatiron`

```
mkdir flatiron
cd flatiron
mkdir maf_local
mkdir my_repos
mkdir -p opsimv4_data/run_local/output
```

2. Clone the following repos into these directories

```
cd maf_local
git clone https://github.com/LSST-nonproject/sims_maf_contrib.git
cd ..
cd my_repos
git clone https://github.com/lsst/sims_maf.git
git clone https://github.com/lsst-ts/scheduler_config
git clone https://github.com/lsst/sims_featureScheduler.git
```

3. If you are on a linux system you will need to open up the read/write permissions of any of the local directories that will be mounted inside of docker containers. For these tutorials you will need to do this for `maf_local`, `my_repos`, and `opsimv4_data`. Again, please see this link for additional setup steps for using docker on linux.

```
chmod a+rw maf_local
chmod a+rw my_repos
chmod a+rw opsimv4_data
```

CHAPTER 3

MAF

A simple introduction to using MAF with Docker. Before following these directions, make sure you have set up the directories as described in *Directories and GitHub repos*. If you have followed those directions, make sure your present working directory is `~/flatiron`.

3.1 Download an OpSim database

These commands are run on your local host in the `~/flatiron` directory.

```
$ wget http://astro-lsst-01.astro.washington.edu:8081/db_gzip/baseline2018a.db.gz
$ gunzip baseline2018a.db.gz
$ mv baseline2018a.db sims_maf_contrib/tutorials/baseline2018a.db
```

3.2 Start a Docker container

Running the following command will start a docker container using the `oboberg/maf:latest` image. This command will also mount local directories into the container so that the MAF output is saved.

```
docker run -it --rm -v ${PWD}/maf_local:/home/docmaf/maf_local \
                    -v ${PWD}/my_repos:/home/docmaf/my_repos \
                    -p 8888:8888 \
                     oboberg/maf:latest
```

Breakdown of command:

- `docker run` run a docker container
- `-it` give me an an interactive shell in the container
- `--rm` remove the container after it is stopped

- `-v ${PWD}/maf_local` mounts the local `maf_local` into the container at the path `/home/docmaf/maf_local`.
- `-v ${PWD}/my_repos` mounts the local `my_repos` into the container at the path `/home/docmaf/my_repos`.
- `-p 8888:8888` this is read as `port on host:port on container`. Meaning port `8888` in the container will be fed to port `8888` on your local host.This allows you to use things like `jupyter lab`.
- `oboberg/maf:latest` this is the name of the docker image. If you don't already have it from doing `docker pull oboberg/maf:latest`, it will automatically be pulled.

3.3 Now we are in the container

The terminal where you ran the docker command will now be a terminal inside the docker container. It will look something like this

```
(lsst-scipipe-10a4fa6) [docmaf@7b8b90333725 ~]$ ls
maf_local  my_repos  repo_pulls.sh  repos  stack  startup.sh
```

In these examples do not include `(lsst-scipipe-10a4fa6) [docmaf@7b8b90333725 ~]$` in your commands. This is included to illustrated when we are issuing commands in the docker container

3.4 Setup the sims_maf_contrib package

```
(lsst-scipipe-10a4fa6) [docmaf@7b8b90333725 ~]$ cd maf_local/sims_maf_contrib/
(lsst-scipipe-10a4fa6) [docmaf@7b8b90333725 sims_maf_contrib]$ eups declare sims_maf_
↪contrib -r . -t $USER
(lsst-scipipe-10a4fa6) [docmaf@7b8b90333725 sims_maf_contrib]$ setup sims_maf_contrib␣
↪-t $USER
```

At this point we will `cd` into the directory with the tutorial notebooks and then start up `jupyter lab`.

```
(lsst-scipipe-10a4fa6) [docmaf@7b8b90333725 ~]$ cd ~/maf_local/sims_maf_contrib/
↪tutorials/
```

3.5 Starting jupyter lab

```
(lsst-scipipe-10a4fa6) [docmaf@e6fe5279c797 tutorials]$ jupyter lab --ip=0.0.0.0
```

Note: If you prefer jupyter notebook just do: `jupyter notebook --ip=0.0.0.0`. Also, make sure nothing else is using port 8888 on your local machine.

You should see a dialog similar to this one, but the token will be some string of letters and numbers.

```
Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://(7b8b90333725 or 127.0.0.1):8888/?token=sometoken
```

When copying and pasting this URL into your local browser you will need to replace `(7b8b90333725 or 127.0.0.1)` with either `localhost`, `8888`, or `127.0.0.1`. So the actual URL you will use in your browser should look something like this:

```
http://localhost:8888/?token=sometoken
```

Once you copy that into your browser and hit enter you should see the familiar jupyter lab landing page.

Go ahead and click on `Introduction Notebook.ipynb` and start running through the cells. As long as you put `baseline2018a.db` in the correct directory the notebook will work right out of the box. See the first couple of steps of this document if you still need to get the database. To kill the jupyter lab/notebook go to the terminal where you started it and do control C twice. This will bring you back to the command prompt.

3.6 Quitting the container

You can now quit the container by simply typing exit.

```
(lsst-scipipe-10a4fa6) [docmaf@7b8b90333725 ~]$ exit
```

Any work that you did in the `maf_local` directory in the container, will be saved to the local directory `~/flatiron/maf_local`.

Note: Since we started the container with the `--rm` flag it will be deleted as soon as we exit. You certainly don't have to use this flag, but be sure to manage the running or stopped containers you having lying around.

CHAPTER 4

OpSim

Here we will go over how to run a very simple one night simulation. Before following these directions, make sure you have set up the directories as described in *Directories and GitHub repos*. If you have followed those directions, make sure your present working directory is `~/flatiron`.

4.1 Getting the OpSim Docker image

Use this command on your local host to pull the latest OpSim Docker image from the docker hub:

```
docker pull oboberg/opsim4_fbs_py3:latest
```

4.2 Start a Docker container

Running the following command will start a docker container using the `oboberg/opsim4_fbs_py3:latest` image. This command will also mount local directories into the container so that the OpSim output is saved.

```
docker run -it --rm -v ${PWD}/opsimv4_data/run_local:/home/opsim/run_local \
                    -v ${PWD}/my_repos:/home/opsim/my_repos \
                    -e OPSIM_HOSTNAME=opsim-docker\
                    -p 8888:8888 \
                     oboberg/opsim4_fbs_py3:latest
```

Breakdown of command:

- `docker run` run a docker container
- `-it` Give me an an interactive shell in the container
- `--rm` remove the container after it is stopped
- `-v ${PWD}/opsimv4_data/run_local` mounts the local `run_dir` into the container at the path `/home/opsim/run_local`.

- `-v ${PWD}/my_repos` mounts the local `my_repos` into the container at the path `/home/opsim/my_repos`.
- `-e OPSIM_HOSTNAME=opsim-docker` sets the `OPSIM_HOSTNAME` environment variable inside the container. This sets the name of the run tracking database and other output files. You can change this to whatever name you like.
- `-p 8888:8888` this is read as `port on host:port on container`. Meaning port `8888` in the container will be fed to port `8888` on your local host.This allows you to use things like `jupyter lab`.
- `oboberg/opsim4_fbs_py3:latest` this is the name of the docker image. If you don't already have it from doing `docker pull oboberg/opsim4_fbs_py3:latest`, it will automatically be pulled.

After running this command you will see a lot of dialog about all of the packages that are being setup. The final message will be about a `scons` test failing, but you do not need to worry about this.

4.3 Now we are in the container

The terminal where you ran the docker command will now be a terminal inside the docker container. It will look something like this

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 ~]
```

If you run an `ls` command you should see the following if you setup the directory structure previously mentioned.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 ~]$ ls
dds  default_configs  my_repos  pull_and_config.sh  pull_repos.sh  repos  run_and_
↪config.sh  run_local  sky_brightness_data  stack  startup_fbs.sh
```

4.4 Setup the OpSim tracking database

Run this command to setup the database that will be used to track OpSim runs. For this command make sure you give the full path for the `save-dir`.

```
manage_db --save-dir=/home/opsim/run_local/output/
```

If you `ls` the directory `run_local/output/` you will see the file `opsim-docker_sessions.db`.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 ~]$ ls run_local/output/
opsim-docker_sessions.db
```

Note: To get sense of how the volume mounting works, open a new terminal, or system browser, and navigate to `~/flatiron/opsimv4_data/run_local/output`. There you will also see the file `opsim-docker_sessions.db`.

4.5 Start a one day simulation with the feature based scheduler

In the docker container `cd` in the the `run_local` directory and run this command. (Note: do not include the `(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$` bit)

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ opsim4 --frac-duration=0.003
```

`--frac-duration` sets the length of the simulation and it is untis of fraction of a year. `(1 / 365)` is about `0.003`, for a full simulation `--frac-duration` is `10`.

Now if you `ls` in the the `run_local` directory you see that a log file has been produced. The actual OpSim database created by our one night run will be in `run_local/output`.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ ls
opsim-docker_2000.log  output
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ ls output/
opsim-docker_2000.db  opsim-docker_sessions.db
```

Note: You can see that the log file and output database share the same file root as the session database `opsim-docker`. The number `2000` will be autamotically increased by `1` as we run additional simulations.

4.6 Start a one day simulation with the proposal scheduler

To use the proposal scheduler we simply provide another command line option to the `opsim4` command.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ opsim4 --frac-duration=0.003 --
↪scheduler proposal
```

Again, if you `ls` in the the `run_local` directory you see that another log file and output database have been produced. Also note that the run number increased from `2000` to `2001`.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ ls
opsim-docker_2000.log  opsim-docker_2001.log  output
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ ls output/
opsim-docker_2000.db  opsim-docker_2001.db  opsim-docker_sessions.db
```

4.7 Useful information in the logs

Let's run the `head` command on the two logs to see some useful information about which scheduler was used, and the path to the default configuration.

- First the feature based run `opsim-docker_2000`

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ head -n5 opsim-docker_2000.log
2018-09-12 18:31:36,203 - INFO - root - Loading feature driver
2018-09-12 18:31:36,616 - DEBUG - matplotlib.backends - backend Qt5Agg version 5.9.2
2018-09-12 18:31:37,062 - DEBUG - kernel.Simulator - Using default configuration path:
↪ /home/opsim/repos/scheduler_config/config_run/
2018-09-12 18:31:37,198 - INFO - kernel.Simulator - Initializing simulation
2018-09-12 18:31:37,199 - INFO - kernel.Simulator - Simulation Session Id = 2000
```

- You can see in `line 1` that the feature scheduler was used `INFO - root - Loading feature driver`
- You can also see the path to the configuration that was used for the simulation `Using default configuration path: /home/opsim/repos/scheduler_config/config_run/`. We will come back to this in the next section.

- Now the proposal based run `opsim-docker_2001`

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ head -n5 opsim-docker_2001.log
2018-09-12 18:47:00,561 - INFO - root - Loading proposal driver
2018-09-12 18:47:00,813 - INFO - schedulerDriver - buildFieldsTable: 5292 fields
2018-09-12 18:47:00,818 - DEBUG - kernel.Simulator - Using default configuration path:
↪ /home/opsim/repos/scheduler_config/config_run/
2018-09-12 18:47:01,241 - DEBUG - matplotlib.backends - backend Qt5Agg version 5.9.2
2018-09-12 18:47:01,537 - INFO - kernel.Simulator - Initializing simulation
```

- You can see in `line 1` that the proposal scheduler was used `INFO - root - Loading proposal driver`
- You can also see that the same path was use for the configuration `Using default configuration path: /home/opsim/repos/scheduler_config/config_run/`.

4.8 Configuring Simulations

OpSim has recently been redesigned to read configurations from a GitHub repository called `scheduler_config` that can be found here. Within that repository there is a directory called `config_run`, which is where OpSim looks for the configuration for a simulation. From the two previous log files, we can see how this is set up in the docker container. OpSim is looking in `/home/opsim/repos/scheduler_config/config_run/` for the configuration.

When using the `feature based scheduler`, OpSim is reading `feature_scheduler.py` for how to run the simulation. For the `proposal based scheduler`, OpSim will look in this directory for `PexConfig` files that correspond to individual proposals in the simulation (e.g WFD, NES, SCP).

4.8.1 Setup up the repos for new configurations

We want to try a different configuration, but let's not use the `scheduler_config` repo that is built into the container, instead we will `eups declare` the cloned repo we mounted when we started the container.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 ~]$ cd /home/opsim/my_repos/scheduler_
↪config/
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 scheduler_config]$ eups declare scheduler_
↪config -r . -t $USER
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 scheduler_config]$ setup scheduler_config -
↪t $USER
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 scheduler_config]$ scons
```

Now if you run `eups list -v scheduler_config` you will see the correct repo is setup.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 scheduler_config]$ eups list -v scheduler_
↪config
git        /home/opsim/stack/stack/miniconda3-4.5.4-fcd27eb /home/opsim/repos/
↪scheduler_config                     current
tag:opsim  /home/opsim/.eups    /home/opsim/my_repos/scheduler_config               ␣
↪      user:opsim setup
```

We are now ready to edit the configurations. From the `/home/opsim/my_repos/scheduler_config` directory, go ahead and make a new branch in the repo.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 scheduler_config]$ git checkout -b my_
↪config_test
Switched to a new branch 'my_config_test'
```

Note: Any of the edits that we are about to do in `/home/opsim/my_repos/scheduler_config` can either be done in the docker container terminal using `vi`, or you can edit it them using your favorite local editor in the `~/flatiron/my_repos` directory.

4.8.2 A new feature based configuration

For the `feature based scheduler` we will edit the file `~/my_repos/scheduler_config/config_run/feature_scheduler.py`. If you are doing this from the inside the container with `vi`, use the following, or edit it on your local host.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 scheduler_config]$ vi ~/my_repos/scheduler_
↪config/config_run/feature_scheduler.py
```

For this example we will remove the deep drilling fields, the pairs survey, and we won't take any observations in the `r filter`. Edit the file to look like this and save.

```
import numpy as np
import healpy as hp
import lsst.sims.featureScheduler as fs
from lsst.ts.scheduler.kernel import SurveyTopology

if __name__ == 'config':
    survey_topology = SurveyTopology()
    survey_topology.num_general_props = 4
    survey_topology.general_propos = ["NorthEclipticSpur", "SouthCelestialPole",
↪"WideFastDeep", "GalacticPlane"]
    survey_topology.num_seq_props = 1
    survey_topology.sequence_propos = ["DeepDrillingCosmology1"]

    target_maps = {}
    nside = fs.set_default_nside(nside=32)  # Required

    target_maps['u'] = fs.generate_goal_map(NES_fraction=0.,
                                            WFD_fraction=0.31, SCP_fraction=0.15,
                                            GP_fraction=0.15, nside=nside,
                                            generate_id_map=True)
    target_maps['g'] = fs.generate_goal_map(NES_fraction=0.2,
                                            WFD_fraction=0.44, SCP_fraction=0.15,
                                            GP_fraction=0.15, nside=nside,
                                            generate_id_map=True)
    #target_maps['r'] = fs.generate_goal_map(NES_fraction=0.46,
    #                                         WFD_fraction=1.0, SCP_fraction=0.15,
    #                                         GP_fraction=0.15, nside=nside,
    #                                         generate_id_map=True)
    target_maps['i'] = fs.generate_goal_map(NES_fraction=0.46,
                                            WFD_fraction=1.0, SCP_fraction=0.15,
                                            GP_fraction=0.15, nside=nside,
                                            generate_id_map=True)
    target_maps['z'] = fs.generate_goal_map(NES_fraction=0.4,
```

(continues on next page)

(continued from previous page)

```
                                                WFD_fraction=0.9, SCP_fraction=0.15,
                                                GP_fraction=0.15, nside=nside,
                                                generate_id_map=True)
    target_maps['y'] = fs.generate_goal_map(NES_fraction=0.,
                                                WFD_fraction=0.9, SCP_fraction=0.15,
                                                GP_fraction=0.15, nside=nside,
                                                generate_id_map=True)

    filters = ['u', 'g', 'i', 'z', 'y']
    surveys = []

    for filtername in filters:
        bfs = []
        bfs.append(fs.M5_diff_basis_function(filtername=filtername, nside=nside))
        bfs.append(fs.Target_map_basis_function(filtername=filtername,
                                                    target_map=target_maps[filtername][0],
                                                    out_of_bounds_val=hp.UNSEEN,␣
↪nside=nside))

        bfs.append(fs.MeridianStripeBasisFunction(nside=nside, width=(8.,)))
        bfs.append(fs.Slewtime_basis_function(filtername=filtername, nside=nside))
        bfs.append(fs.Strict_filter_basis_function(filtername=filtername))
        bfs.append(fs.Avoid_Fast_Revists(filtername=filtername, gap_min=240.,␣
↪nside=nside))

        weights = np.array([3.0, 0.5, 1., 3., 3., 3.])
        # surveys.append(fs.Greedy_survey_fields(bfs, weights, block_size=1,␣
↪filtername=filtername, dither=False,
        #                                        nside=nside, smoothing_kernel=9,
        #                                        tag_fields=True, tag_map=target_
↪maps[filtername][1]))
        surveys.append(fs.Greedy_survey_fields(bfs, weights, block_size=1,␣
↪filtername=filtername, dither=True,
                                                  nside=nside,
                                                  tag_fields=True,
                                                  tag_map=target_maps[filtername][1],
                                                  tag_names=target_maps[filtername][2]))


    scheduler = fs.Core_scheduler(surveys, nside=nside)  # Required
```

Now we are set to run a new feature based simulation and this configuration will be used.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ cd ~/run_local/
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ opsim4 --frac-duration=0.003
```

When it is done have a look at the new log file, which should be `opsim-docker_2002.log`. You see that the configuration was indeed read from the mounted repository.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ head -n5 opsim-docker_2002.log
2018-09-12 20:17:43,187 - INFO - root - Loading feature driver
2018-09-12 20:17:43,575 - DEBUG - matplotlib.backends - backend Qt5Agg version 5.9.2
2018-09-12 20:17:43,951 - DEBUG - kernel.Simulator - Using default configuration path:
↪ /home/opsim/my_repos/scheduler_config/config_run/
2018-09-12 20:17:44,272 - INFO - kernel.Simulator - Initializing simulation
2018-09-12 20:17:44,272 - INFO - kernel.Simulator - Simulation Session Id = 2002
```

If you want to check that no observations were taken in `r` you can quickly do so wtih `sqlite3`.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 output]$ cd ~/run_local/output/
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 output]$ sqlite3 opsim-docker_2002.db
SQLite version 3.23.1 2018-04-10 17:39:29
Enter ".help" for usage hints.
sqlite> sqlite> select * from summaryallprops where filter = 'r';
sqlite> .exit
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 output]$
```

This query will return nothing. For contrast, try the same thing with the first feature based run (`opsim-docker_2000.db`) that used the default configuration.

4.8.3 A new proposal based run

For proposal based runs there is not a single file that we edit, but rather a series of `PexConfig` python files. For this example we will configure the simulation to only do the Wide Fast Deep (WFD) area, plus deep drilling, and do single `30` second snaps, instead of two `15` second snaps.

First we will edit the `vi ~/my_repos/scheduler_config/config_run/survey.py` file already in the repository so it only includes WFD. This is easily done by adding this line `config.general_proposals=['WideFastDeep']` to the end of the file.

```
"""
This is an example configuration for some of the basic parameters for simulations.

07/2018 - Version 0
"""
import lsst.ts.schedulerConfig.survey
assert type(config)==lsst.ts.schedulerConfig.survey.Survey, 'config is of type %s.%s␣
↪instead of lsst.ts.schedulerConfig.survey.Survey' % (type(config).__module__,␣
↪type(config).__name__)
# The delay (units=seconds) to skip the simulation time forward when not receiving a␣
↪target.
config.idle_delay=60.0

# The start date (format=YYYY-MM-DD) of the survey.
config.start_date='2022-10-01'

# The fractional duration (units=years) of the survey.
config.duration=10.0

config.general_proposals=['WideFastDeep']
```

To edit the snaps we will need to create a new file called `widefastdeep_prop.py` in `~/my_repos/scheduler_config/config_run/`. Here we will do this with `touch`.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 config_run]$ touch widefastdeep_prop.py
```

Then edit that file to contain the following

```
import lsst.ts.schedulerConfig.science.wide_fast_deep
assert type(config)==lsst.ts.schedulerConfig.science.wide_fast_deep.WideFastDeep,
↪'config is of type %s.%s instead of lsst.ts.schedulerConfig.science.wide_fast_deep.
↪WideFastDeep' % (type(config).__module__, type(config).__name__)
config.filters['u'].exposures=[30]
```

(continues on next page)

(continued from previous page)

```
config.filters['g'].exposures=[30]
config.filters['r'].exposures=[30]
config.filters['i'].exposures=[30]
config.filters['z'].exposures=[30]
config.filters['y'].exposures=[30]
```

Now `cd` back to `~/run_local` and start a one night simulation.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local]$ opsim4 --frac-duration=0.003 --
↪scheduler proposal
```

We will use `sqlite3` again to illustrate that the configuration worked.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 run_local] cd ~/run_local/output
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 output]$ cp opsim-docker_2006.db opsim-
↪docker_2003.db
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 output]$ sqlite3 opsim-docker_2003.db
SQLite version 3.23.1 2018-04-10 17:39:29
Enter ".help" for usage hints.
sqlite> select * from Proposal;
1|2003|WideFastDeep|General
2|2003|DeepDrillingCosmology1|Sequence
sqlite> .exit
```

As you can see only `WideFastDeep` and `DeepDrillingCosmology1` are present in the `Proposal` table.

4.8.4 Switching back to the defaults

To switch back to the default configurations we can use `eups`.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 output]$ eups list -v scheduler_config
git        /home/opsim/stack/stack/miniconda3-4.5.4-fcd27eb /home/opsim/repos/
↪scheduler_config current setup
tag:opsim  /home/opsim/.eups    /home/opsim/my_repos/scheduler_config
```

4.9 Quitting the container

To quit the container simply type `exit` at the command promt.

```
(lsst-scipipe-fcd27eb) [opsim@9d54f5d124e1 ~] exit
```

Since we started the container with the `--rm` flag it will be deleted as soon as we exit. Therefore only edits and output in the mounted directories will be saved on your local host.

CHAPTER 5

Feature Based Scheduler Jupyter Notebooks

It is possible to run the feature based scheduler in jupyter notebooks. This is useful for examining the basis functions, and doing quick evaluations of how a given configuration is performing. These notebooks can also be found the `sims_featureScheduler` repository here.

5.1 Simple Feature Based Scheduler run with SOCS

This example notebook show how to do a simple 1 day FBS run using SOCS. In this example we use the default FBS configuration, a separate example will be given on how to provide a custom configuration.

Before running the notebook make sure you run `manage_db --save-dir $HOME/run_local/output/` on the command line to setup the SOCS database.

[1]:
```
import logging
import healpy as hp

from lsst.sims.ocs.database import SocsDatabase
from lsst.sims.ocs.kernel import Simulator
from lsst.sims.featureScheduler.driver import FeatureSchedulerDriver as Driver
from lsst.sims.ocs.setup import create_parser
from lsst.sims.ocs.setup import apply_file_config, read_file_config
```

[2]:
```
logging.getLogger().setLevel(logging.INFO)
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(name)-12s %(levelname)-8s %(message)s',
                    datefmt='%m-%d %H:%M')
```

This next cell loads default command line arguments. These are needed mainly to setup the database.

[3]:
```
parser = create_parser()
args = parser.parse_known_args()[0]
prog_conf = read_file_config()
if prog_conf is not None:
```

(continues on next page)

(continued from previous page)

```
    apply_file_config(prog_conf, args)
print(args.sqlite_save_dir, args.session_id_start, args.sqlite_session_save_dir)
```

```
/home/opsim/run_local/output/ None None
```

Setup socs database to store simulations results, if needed.

```
[4]: db = SocsDatabase(sqlite_save_path=args.sqlite_save_dir,
                       session_id_start=args.session_id_start,
                       sqlite_session_save_path=args.sqlite_session_save_dir)
```

```
[5]: session_id = db.new_session("FBS test on notebook")
```

We now define a driver for the simulation. In this case, we already imported the FBS driver as Driver so we simply call it.

```
[6]: driver = Driver()
```

By default the duration of a simulation is 10 years. Here we will run a single day.

```
[7]: args.frac_duration = 0.003
```

We now set the SOCS simulator

```
[8]: sim = Simulator(args, db, driver=driver)
```

```
[9]: sim.initialize()
```

```
09-13 17:51 kernel.Simulator INFO     Initializing simulation
09-13 17:51 kernel.Simulator INFO     Simulation Session Id = 2007
09-13 17:51 configuration.ConfigurationCommunicator INFO     Initializing␣
↪configuration communication
09-13 17:51 kernel.Simulator INFO     Finishing simulation initialization
```

And run the simulation

```
[10]: sim.run()
```

```
09-13 17:51 kernel.Simulator INFO     Starting simulation
09-13 17:51 kernel.Simulator INFO     run: rx scheduler config survey_duration=3650.0
09-13 17:51 kernel.Simulator INFO     run: rx driver config={'ranking': {'coadd_values
↪': 1, 'time_balancing': 1, 'timecost_time_max': 150.0, 'timecost_time_ref': 5.0,
↪'timecost_cost_ref': 0.3, 'timecost_weight': 1.0, 'filtercost_weight': 1.0,
↪'propboost_weight': 1.0, 'lookahead_window_size': 0, 'lookahead_bonus_weight': 0.0},
↪ 'constraints': {'night_boundary': -12.0, 'ignore_sky_brightness': 0, 'ignore_
↪airmass': 0, 'ignore_clouds': 0, 'ignore_seeing': 0}, 'darktime': {'new_moon_phase_
↪threshold': 20.0}, 'startup': {'type': 'HOT', 'database': ''}}
09-13 17:51 kernel.Simulator INFO     run: rx site config={'obs_site': {'name':
↪'Cerro Pachon', 'latitude': -30.2444, 'longitude': -70.7494, 'height': 2650.0}}
09-13 17:51 kernel.Simulator INFO     run: rx telescope config={'telescope': {
↪'altitude_minpos': 20.0, 'altitude_maxpos': 86.5, 'azimuth_minpos': -270.0,
↪'azimuth_maxpos': 270.0, 'altitude_maxspeed': 3.5, 'altitude_accel': 3.5, 'altitude_
↪decel': 3.5, 'azimuth_maxspeed': 7.0, 'azimuth_accel': 7.0, 'azimuth_decel': 7.0,
↪'settle_time': 3.0}}
09-13 17:51 kernel.Simulator INFO     run: rx dome config={'dome': {'altitude_maxspeed
↪': 1.75, 'altitude_accel': 0.875, 'altitude_decel': 0.875, 'altitude_freerange': 0.
↪0, 'azimuth_maxspeed': 1.5, 'azimuth_accel': 0.75, 'azimuth_decel': 0.75, 'azimuth_
↪freerange': 4.0, 'settle_time': 0.0}}
```

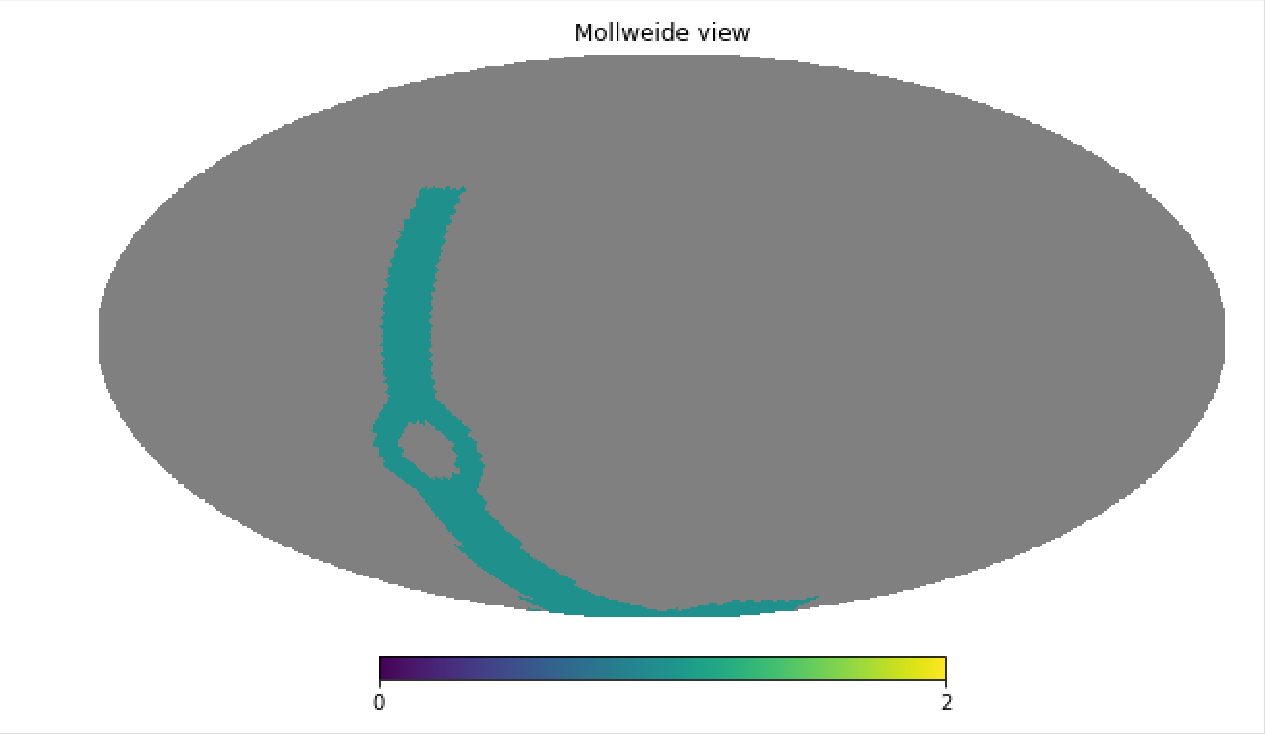
(continues on next page)

(continued from previous page)

```
09-13 17:51 kernel.Simulator INFO     run: rx rotator config={'rotator': {'minpos': -
↪90.0, 'maxpos': 90.0, 'maxspeed': 3.5, 'accel': 1.0, 'decel': 1.0, 'filter_change_
↪pos': 0.0, 'follow_sky': 0, 'resume_angle': 0}}
09-13 17:51 kernel.Simulator INFO     run: rx camera config={'camera': {'readout_time
↪': 2.0, 'shutter_time': 1.0, 'filter_change_time': 120.0, 'filter_max_changes_burst_
↪num': 1, 'filter_max_changes_burst_time': 0.0, 'filter_max_changes_avg_num': 3000,
↪'filter_max_changes_avg_time': 31557600.0, 'filter_removable': ['y', 'z'], 'filter_
↪mounted': ['g', 'r', 'i', 'z', 'y'], 'filter_unmounted': ['u']}}
09-13 17:51 kernel.Simulator INFO     run: rx slew config={'slew': {'prereq_domalt':␣
↪[], 'prereq_domaz': [], 'prereq_domazsettle': ['domaz'], 'prereq_telalt': [],
↪'prereq_telaz': [], 'prereq_telopticsopenloop': ['telalt', 'telaz'], 'prereq_
↪telopticsclosedloop': ['domalt', 'domazsettle', 'telsettle', 'readout',
↪'telopticsopenloop', 'filter', 'telrot'], 'prereq_telsettle': ['telalt', 'telaz'],
↪'prereq_telrot': [], 'prereq_filter': [], 'prereq_exposures': ['telopticsclosedloop
↪'], 'prereq_readout': []}}
09-13 17:51 kernel.Simulator INFO     run: rx optics config={'optics_loop_corr': {
↪'tel_optics_ol_slope': 0.2857142857142857, 'tel_optics_cl_alt_limit': [0.0, 9.0, 90.
↪0], 'tel_optics_cl_delay': [0.0, 36.0]}}
09-13 17:51 kernel.Simulator INFO     run: rx park config={'park': {'telescope_
↪altitude': 86.5, 'telescope_azimuth': 0.0, 'telescope_rotator': 0.0, 'dome_altitude
↪': 90.0, 'dome_azimuth': 0.0, 'filter_position': 'z'}}
09-13 17:51 featureSchedulerDriver INFO     Loading feature based scheduler␣
↪configuration from /home/opsim/repos/scheduler_config/config_run/feature_scheduler.
↪py.
09-13 17:52 featureSchedulerDriver INFO     Start up type is HOT, no state will be␣
↪read from the EFD.
09-13 17:52 kernel.Simulator INFO     Night 1
09-13 17:52 featureSchedulerDriver INFO     start_survey t=1664580953.883245
09-13 17:52 featureSchedulerDriver INFO     start_night t=1664580953.883245, night=1␣
↪timeprogress=0.00%
09-13 17:52 featureSchedulerDriver INFO     start_night t=1664580953.883245, night=1␣
↪timeprogress=0.00%
/home/opsim/repos/sims_skybrightness_pre/python/lsst/sims/skybrightness_pre/
↪SkyModelPre.py:363: UserWarning: Requested MJD between sunrise and sunset,␣
↪returning closest maps
  warnings.warn('Requested MJD between sunrise and sunset, returning closest maps')
/home/opsim/repos/sims_skybrightness_pre/python/lsst/sims/skybrightness_pre/
↪SkyModelPre.py:279: UserWarning: Requested MJD between sunrise and sunset,␣
↪returning closest maps
  warnings.warn('Requested MJD between sunrise and sunset, returning closest maps')
/home/opsim/repos/sims_seeingModel/python/lsst/sims/seeingModel/seeingModel.py:133:␣
↪RuntimeWarning: invalid value encountered in power
  airmass_correction = np.power(airmass, 0.6)
/home/opsim/repos/sims_skybrightness_pre/python/lsst/sims/skybrightness_pre/
↪SkyModelPre.py:49: RuntimeWarning: invalid value encountered in true_divide
  wterm = (x - xp[left])/baseline
09-13 17:53 featureSchedulerDriver INFO     end_night t=1664616576.277367, night=1␣
↪timeprogress=0.01%
09-13 17:53 featureSchedulerDriver INFO     end_night next moonphase=40.78%
09-13 17:53 featureSchedulerDriver INFO     end_night bright time waxing
```

We now have access to all the scheduler data structure to play with. In the cell bellow, we plot the TargetMapBasis function for the `g` filter.

[11]:
```
hp.mollview(sim.driver.scheduler.survey_lists[0][1].basis_functions[2]())
```




```
[ ]:
```

5.2 Feature Based Scheduler run with SOCS and custom scheduler configuration

This example notebook show how to do a slightly more complex 1 day FBS run using SOCS. In this example we use a custom FBS configuration done directly on the notebook.

Before running the notebook make sure you run `manage_db --save-dir $HOME/run_local/output/` on the command line to setup the SOCS database.

[1]:
```
import logging
import healpy as hp
import numpy as np

import lsst.sims.featureScheduler as fs

from lsst.sims.featureScheduler.driver import FeatureSchedulerDriver as Driver

from lsst.sims.ocs.database import SocsDatabase
from lsst.sims.ocs.kernel import Simulator
from lsst.sims.ocs.setup import create_parser
from lsst.sims.ocs.setup import apply_file_config, read_file_config

from lsst.ts.scheduler.kernel import SurveyTopology
```

[2]:
```
logging.getLogger().setLevel(logging.INFO)
logging.basicConfig(level=logging.INFO,
```

(continues on next page)

(continued from previous page)

```
                        format='%(asctime)s %(name)-12s %(levelname)-8s %(message)s',
                        datefmt='%m-%d %H:%M')
```

This next cell loads default command line arguments. These are needed mainly to setup the simulation database.

[3]:
```
parser = create_parser()
args = parser.parse_known_args()[0]
prog_conf = read_file_config()
if prog_conf is not None:
    apply_file_config(prog_conf, args)
print(args.sqlite_save_dir, args.session_id_start, args.sqlite_session_save_dir)
```

```
/home/opsim/run_local/output/ None None
```

Setup socs database to store simulations results, if needed.

[4]:
```
db = SocsDatabase(sqlite_save_path=args.sqlite_save_dir,
                  session_id_start=args.session_id_start,
                  sqlite_session_save_path=args.sqlite_session_save_dir)
```

[5]:
```
session_id = db.new_session("FBS test on notebook")
```

We now define a driver for the simulation. In this case, we already imported the FBS driver as Driver so we simply call it.

[6]:
```
driver = Driver()
```

By default the duration of a simulation is 10 years. Here we will run a single day.

[7]:
```
args.frac_duration = 0.003
```

We now set the SOCS simulator

[8]:
```
sim = Simulator(args, db, driver=driver)
```

Up to this point, the scheduler is still not configured. Let's make a custom configuration.

[9]:
```
survey_topology = SurveyTopology()
survey_topology.num_general_props = 4
survey_topology.general_propos = ["NorthEclipticSpur", "SouthCelestialPole",
↪"WideFastDeep", "GalacticPlane"]
survey_topology.num_seq_props = 1
survey_topology.sequence_propos = ["DeepDrillingCosmology1"]

target_maps = {}
nside = fs.set_default_nside(nside=32)  # Required

target_maps['u'] = fs.generate_goal_map(NES_fraction=0.,
                                        WFD_fraction=0.31, SCP_fraction=0.08,
                                        GP_fraction=0.004,
                                        WFD_upper_edge_fraction=0.0,
                                        nside=nside,
                                        generate_id_map=True)
target_maps['g'] = fs.generate_goal_map(NES_fraction=0.2,
                                        WFD_fraction=0.44, SCP_fraction=0.08,
                                        GP_fraction=0.004,
```

(continues on next page)

(continued from previous page)

```
                                                WFD_upper_edge_fraction=0.0,
                                                nside=nside,
                                                generate_id_map=True)
target_maps['r'] = fs.generate_goal_map(NES_fraction=0.46,
                                                WFD_fraction=1.0, SCP_fraction=0.08,
                                                WFD_upper_edge_fraction=0.0,
                                                GP_fraction=0.004,
                                                nside=nside,
                                                generate_id_map=True)
target_maps['i'] = fs.generate_goal_map(NES_fraction=0.46,
                                                WFD_fraction=1.0, SCP_fraction=0.08,
                                                GP_fraction=0.004,
                                                WFD_upper_edge_fraction=0.0,
                                                nside=nside,
                                                generate_id_map=True)
target_maps['z'] = fs.generate_goal_map(NES_fraction=0.4,
                                                WFD_fraction=0.9, SCP_fraction=0.08,
                                                GP_fraction=0.004,
                                                WFD_upper_edge_fraction=0.0,
                                                nside=nside,
                                                generate_id_map=True)
target_maps['y'] = fs.generate_goal_map(NES_fraction=0.,
                                                WFD_fraction=0.9, SCP_fraction=0.08,
                                                GP_fraction=0.004,
                                                WFD_upper_edge_fraction=0.0,
                                                nside=nside,
                                                generate_id_map=True)

cloud_map = fs.utils.generate_cloud_map(target_maps, filtername='r',
                                                wfd_cloud_max=0.7,
                                                scp_cloud_max=0.7,
                                                gp_cloud_max=0.7,
                                                nes_cloud_max=0.7)

# x1 = 30.
# x0 = 2.
# B = x1 / (x1 - x0)
# A = -B / x1
# width = np.arange(x0, x1, 0.5)
# z_pad = width + 8.
# weight = (A * width + B)
# height = np.zeros_like(width) + 80.

width = (10.,)
z_pad = (18.,)
weight = (1.,)
height = (80.,)

filters = ['u', 'g', 'r', 'i', 'z', 'y']
surveys = []

sb_limit_map = fs.utils.generate_sb_map(target_maps, filters)

filter_prop = {'u': 0.069,
               'g': 0.097,
               'r': 0.222,
               'i': 0.222,
```

(continues on next page)

(continued from previous page)

```
                'z': 0.194,
                'y': 0.194}

for filtername in filters:
    bfs = list()
    # bfs.append(fs.M5_diff_basis_function(filtername=filtername, nside=nside))
    bfs.append(fs.HourAngle_bonus_basis_function(max_hourangle=3.))
    bfs.append(fs.Skybrightness_limit_basis_function(nside=nside,
                                                      filtername=filtername,
                                                      min=sb_limit_map[filtername]['min
↪'],
                                                      max=sb_limit_map[filtername]['max
↪']))
    bfs.append(fs.Target_map_basis_function(filtername=filtername,
                                             target_map=target_maps[filtername][0],
                                             out_of_bounds_val=hp.UNSEEN, nside=nside))
    bfs.append(fs.MeridianStripeBasisFunction(nside=nside,width=width,
                                               weight=weight,
                                               height=height,
                                               zenith_pad=z_pad))
    # bfs.append(fs.HADecAltAzPatchBasisFunction(nside=nside,
    #                                             patches=patches[::-1]))
    bfs.append(fs.Aggressive_Slewtime_basis_function(filtername=filtername,␣
↪nside=nside, order=6., hard_max=20.))
    bfs.append(fs.Goal_Strict_filter_basis_function(filtername=filtername,
                                                     time_lag_min=90.,
                                                     time_lag_max=150.,
                                                     time_lag_boost=180.,
                                                     boost_gain=1.0,
                                                     unseen_before_lag=True,
                                                     proportion=filter_prop[filtername],
                                                     aways_available=filtername in 'zy'))
    bfs.append(fs.Avoid_Fast_Revists(filtername=None, gap_min=240., nside=nside))
    bfs.append(fs.Bulk_cloud_basis_function(max_cloud_map=cloud_map, nside=nside))
    bfs.append(fs.Moon_avoidance_basis_function(nside=nside, moon_distance=40.))
    # bfs.append(fs.CableWrap_unwrap_basis_function(nside=nside, activate_tol=70.,␣
↪unwrap_until=315,
    #                                                max_duration=90.))
    # bfs.append(fs.NorthSouth_scan_basis_function(length=70.))

    weights = np.array([2., 0.1, .1, 1., 3., 1.5, 1.0, 1.0, 1.0])
    surveys.append(fs.Greedy_survey_fields(bfs, weights, block_size=1,
                                            filtername=filtername, dither=True,
                                            nside=nside,
                                            tag_fields=True,
                                            tag_map=target_maps[filtername][1],
                                            tag_names=target_maps[filtername][2],
                                            ignore_obs='DD'))

# Set up pairs
pairs_bfs = []

pair_map = np.zeros(len(target_maps['z'][0]))
pair_map.fill(hp.UNSEEN)
wfd = np.where(target_maps['z'][1] == 3)
nes = np.where(target_maps['z'][1] == 1)
pair_map[wfd] = 1.
```

(continues on next page)

(continued from previous page)

```
pair_map[nes] = 1.

pairs_bfs.append(fs.Target_map_basis_function(filtername='',
                                              target_map=pair_map,
                                              out_of_bounds_val=hp.UNSEEN,␣
↪nside=nside))
pairs_bfs.append(fs.MeridianStripeBasisFunction(nside=nside, zenith_pad=(45.,),␣
↪width=(35.,)))
pairs_bfs.append(fs.Moon_avoidance_basis_function(nside=nside, moon_distance=30.))

# surveys.append(fs.Pairs_survey_scripted(pairs_bfs, [1., 1., 1.], ignore_obs='DD',␣
↪min_alt=20.))
surveys.append(fs.Pairs_different_filters_scripted(pairs_bfs, [1., 1., 1.], ignore_
↪obs='DD', min_alt=20.,
                                                   filter_goals=filter_prop))
# surveys.append(fs.Pairs_survey_scripted([], [], ignore_obs='DD'))

# Set up the DD
# ELAIS S1
surveys.append(fs.Deep_drilling_survey(9.45, -44., sequence='rgizy',
                                       nvis=[20, 10, 20, 26, 20],
                                       survey_name='DD:ELAISS1', reward_value=100,␣
↪moon_up=None,
                                       fraction_limit=0.148, ha_limits=([0., 0.5],␣
↪[23.5, 24.]),
                                       nside=nside,
                                       avoid_same_day=True,
                                       filter_goals=filter_prop))
surveys.append(fs.Deep_drilling_survey(9.45, -44., sequence='u',
                                       nvis=[7],
                                       survey_name='DD:u,ELAISS1', reward_value=100,␣
↪moon_up=False,
                                       fraction_limit=0.0012, ha_limits=([0., 0.5],␣
↪[23.5, 24.]),
                                       nside=nside))

# XMM-LSS
surveys.append(fs.Deep_drilling_survey(35.708333, -4 - 45 / 60., sequence='rgizy',
                                       nvis=[20, 10, 20, 26, 20],
                                       survey_name='DD:XMM-LSS', reward_value=100,␣
↪moon_up=None,
                                       fraction_limit=0.148, ha_limits=([0., 0.5],␣
↪[23.5, 24.]),
                                       nside=nside,
                                       avoid_same_day=True,
                                       filter_goals=filter_prop))
surveys.append(fs.Deep_drilling_survey(35.708333, -4 - 45 / 60., sequence='u',
                                       nvis=[7],
                                       survey_name='DD:u,XMM-LSS', reward_value=100,␣
↪moon_up=False,
                                       fraction_limit=0.0012, ha_limits=([0., 0.5],␣
↪[23.5, 24.]),
                                       nside=nside))

# Extended Chandra Deep Field South
# XXX--Note, this one can pass near zenith. Should go back and add better planning on␣
↪this.
```

(continues on next page)

(continued from previous page)

```
surveys.append(fs.Deep_drilling_survey(53.125, -28. - 6 / 60., sequence='rgizy',
                                       nvis=[20, 10, 20, 26, 20],
                                       survey_name='DD:ECDFS', reward_value=100, moon_
↪up=None,
                                       fraction_limit=0.148, ha_limits=[[0.5, 1.0],␣
↪[23., 22.5]],
                                       nside=nside,
                                       avoid_same_day=True,
                                       filter_goals=filter_prop))
surveys.append(fs.Deep_drilling_survey(53.125, -28. - 6 / 60., sequence='u',
                                       nvis=[7],
                                       survey_name='DD:u,ECDFS', reward_value=100,␣
↪moon_up=False,
                                       fraction_limit=0.0012, ha_limits=[[0.5, 1.0],␣
↪[23., 22.5]],
                                       nside=nside))
# COSMOS
surveys.append(fs.Deep_drilling_survey(150.1, 2. + 10. / 60. + 55 / 3600., sequence=
↪'rgizy',
                                       nvis=[20, 10, 20, 26, 20],
                                       survey_name='DD:COSMOS', reward_value=100,␣
↪moon_up=None,
                                       fraction_limit=0.148, ha_limits=([0., 0.5],␣
↪[23.5, 24.]),
                                       nside=nside,
                                       avoid_same_day=True,
                                       filter_goals=filter_prop))
surveys.append(fs.Deep_drilling_survey(150.1, 2. + 10. / 60. + 55 / 3600., sequence='u
↪',
                                       nvis=[7], ha_limits=([0., .5], [23.5, 24.]),
                                       survey_name='DD:u,COSMOS', reward_value=100,␣
↪moon_up=False,
                                       fraction_limit=0.0012,
                                       nside=nside))

# Extra DD Field, just to get to 5. Still not closed on this one
surveys.append(fs.Deep_drilling_survey(349.386443, -63.321004, sequence='rgizy',
                                       nvis=[20, 10, 20, 26, 20],
                                       survey_name='DD:290', reward_value=100, moon_
↪up=None,
                                       fraction_limit=0.148, ha_limits=([0., 0.5],␣
↪[23.5, 24.]),
                                       nside=nside,
                                       avoid_same_day=True,
                                       filter_goals=filter_prop))
surveys.append(fs.Deep_drilling_survey(349.386443, -63.321004, sequence='u',
                                       nvis=[7],
                                       survey_name='DD:u,290', reward_value=100, moon_
↪up=False,
                                       fraction_limit=0.0012, ha_limits=([0., 0.5],␣
↪[23.5, 24.]),
                                       nside=nside,
                                       filter_goals=filter_prop))

scheduler = fs.Core_scheduler(surveys, nside=nside)  # Required
```

Now we load the configuration to driver. We basically need to two two steps, load `scheduler`, `nside` and survey_topology.

[10]:
```
sim.driver.scheduler = scheduler
sim.driver.sky_nside = nside

sim.conf_comm.num_proposals = survey_topology.num_props
sim.conf_comm.survey_topology['general'] = survey_topology.general_propos
sim.conf_comm.survey_topology['sequence'] = survey_topology.sequence_propos
```

We now initialize the simulator.

[11]:
```
sim.initialize()
```

```
09-11 19:44 kernel.Simulator INFO     Initializing simulation
09-11 19:44 kernel.Simulator INFO     Simulation Session Id = 2002
09-11 19:44 configuration.ConfigurationCommunicator INFO     Initializing␣
↪configuration communication
09-11 19:44 kernel.Simulator INFO     Finishing simulation initialization
```

And we are ready to run the simulation.

[12]:
```
sim.run()
```

```
09-11 19:44 kernel.Simulator INFO     Starting simulation
09-11 19:44 kernel.Simulator INFO     run: rx scheduler config survey_duration=3650.0
09-11 19:44 kernel.Simulator INFO     run: rx driver config={'ranking': {'coadd_values
↪': 1, 'time_balancing': 1, 'timecost_time_max': 150.0, 'timecost_time_ref': 5.0,
↪'timecost_cost_ref': 0.3, 'timecost_weight': 1.0, 'filtercost_weight': 1.0,
↪'propboost_weight': 1.0, 'lookahead_window_size': 0, 'lookahead_bonus_weight': 0.0},
↪ 'constraints': {'night_boundary': -12.0, 'ignore_sky_brightness': 0, 'ignore_
↪airmass': 0, 'ignore_clouds': 0, 'ignore_seeing': 0}, 'darktime': {'new_moon_phase_
↪threshold': 20.0}, 'startup': {'type': 'HOT', 'database': ''}}
09-11 19:44 kernel.Simulator INFO     run: rx site config={'obs_site': {'name':
↪'Cerro Pachon', 'latitude': -30.2444, 'longitude': -70.7494, 'height': 2650.0}}
09-11 19:44 kernel.Simulator INFO     run: rx telescope config={'telescope': {
↪'altitude_minpos': 20.0, 'altitude_maxpos': 86.5, 'azimuth_minpos': -270.0,
↪'azimuth_maxpos': 270.0, 'altitude_maxspeed': 3.5, 'altitude_accel': 3.5, 'altitude_
↪decel': 3.5, 'azimuth_maxspeed': 7.0, 'azimuth_accel': 7.0, 'azimuth_decel': 7.0,
↪'settle_time': 3.0}}
09-11 19:44 kernel.Simulator INFO     run: rx dome config={'dome': {'altitude_maxspeed
↪': 1.75, 'altitude_accel': 0.875, 'altitude_decel': 0.875, 'altitude_freerange': 0.
↪0, 'azimuth_maxspeed': 1.5, 'azimuth_accel': 0.75, 'azimuth_decel': 0.75, 'azimuth_
↪freerange': 4.0, 'settle_time': 0.0}}
09-11 19:44 kernel.Simulator INFO     run: rx rotator config={'rotator': {'minpos': -
↪90.0, 'maxpos': 90.0, 'maxspeed': 3.5, 'accel': 1.0, 'decel': 1.0, 'filter_change_
↪pos': 0.0, 'follow_sky': 0, 'resume_angle': 0}}
09-11 19:44 kernel.Simulator INFO     run: rx camera config={'camera': {'readout_time
↪': 2.0, 'shutter_time': 1.0, 'filter_change_time': 120.0, 'filter_max_changes_burst_
↪num': 1, 'filter_max_changes_burst_time': 0.0, 'filter_max_changes_avg_num': 3000,
↪'filter_max_changes_avg_time': 31557600.0, 'filter_removable': ['y', 'z'], 'filter_
↪mounted': ['g', 'r', 'i', 'z', 'y'], 'filter_unmounted': ['u']}}
09-11 19:44 kernel.Simulator INFO     run: rx slew config={'slew': {'prereq_domalt':␣
↪[], 'prereq_domaz': [], 'prereq_domazsettle': ['domaz'], 'prereq_telalt': [],
↪'prereq_telaz': [], 'prereq_telopticsopenloop': ['telalt', 'telaz'], 'prereq_
↪telopticsclosedloop': ['domalt', 'domazsettle', 'telsettle', 'readout',
↪'telopticsopenloop', 'filter', 'telrot'], 'prereq_telsettle': ['telalt', 'telaz'],
↪'prereq_telrot': [], 'prereq_filter': [], 'prereq_exposures': ['telopticsclosedloop
↪'], 'prereq_readout': []}}
09-11 19:44 kernel.Simulator INFO     run: rx optics config={'optics_loop_corr': {
↪'tel_optics_ol_slope': 0.2857142857142857, 'tel_optics_cl_alt_limit': [0.0, 9.0, 90.
↪0], 'tel_optics_cl_delay': [0.0, 36.0]}}
```

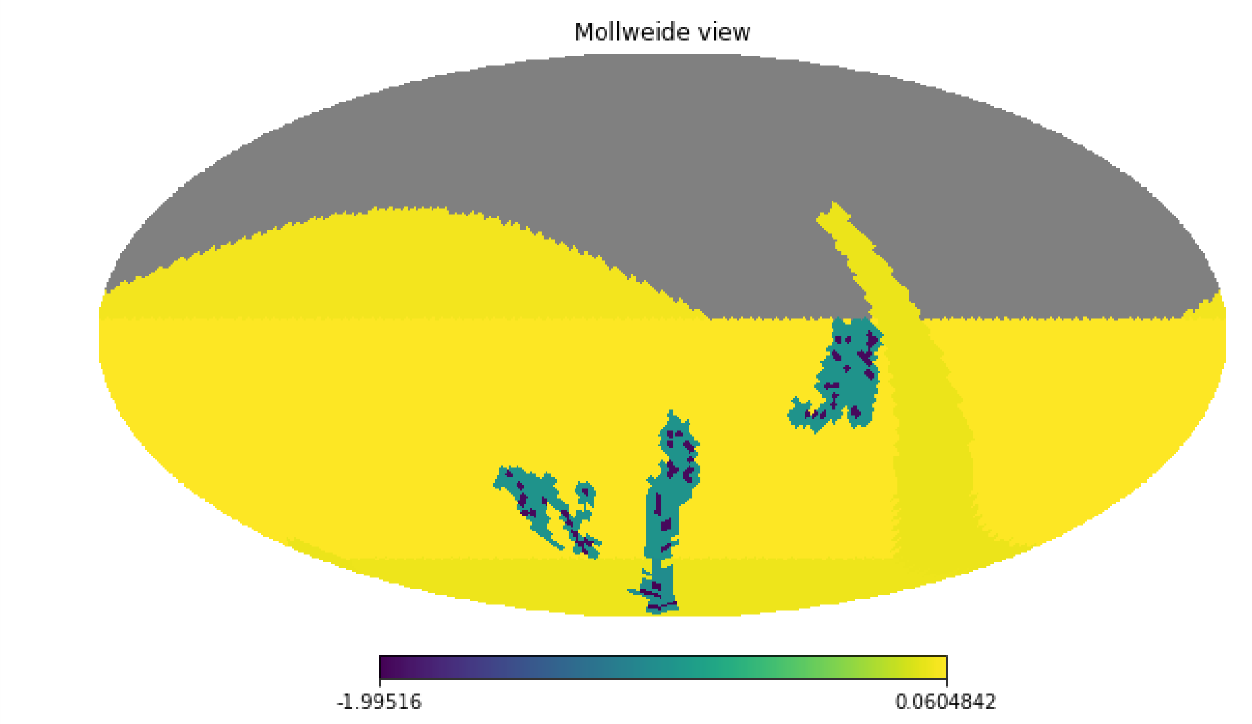
(continues on next page)

(continued from previous page)

```
09-11 19:44 kernel.Simulator INFO     run: rx park config={'park': {'telescope_
↪altitude': 86.5, 'telescope_azimuth': 0.0, 'telescope_rotator': 0.0, 'dome_altitude
↪': 90.0, 'dome_azimuth': 0.0, 'filter_position': 'z'}}
09-11 19:44 featureSchedulerDriver INFO     Scheduler already configured.
09-11 19:44 featureSchedulerDriver INFO     Start up type is HOT, no state will be␣
↪read from the EFD.
09-11 19:44 kernel.Simulator INFO     Night 1
09-11 19:44 featureSchedulerDriver INFO     start_survey t=1664580953.883245
09-11 19:44 featureSchedulerDriver INFO     start_night t=1664580953.883245, night=1␣
↪timeprogress=0.00%
09-11 19:44 featureSchedulerDriver INFO     start_night t=1664580953.883245, night=1␣
↪timeprogress=0.00%
/home/opsim/repos/sims_skybrightness_pre/python/lsst/sims/skybrightness_pre/
↪SkyModelPre.py:363: UserWarning: Requested MJD between sunrise and sunset,␣
↪returning closest maps
  warnings.warn('Requested MJD between sunrise and sunset, returning closest maps')
/home/opsim/repos/sims_skybrightness_pre/python/lsst/sims/skybrightness_pre/
↪SkyModelPre.py:279: UserWarning: Requested MJD between sunrise and sunset,␣
↪returning closest maps
  warnings.warn('Requested MJD between sunrise and sunset, returning closest maps')
/home/opsim/repos/sims_seeingModel/python/lsst/sims/seeingModel/seeingModel.py:133:␣
↪RuntimeWarning: invalid value encountered in power
  airmass_correction = np.power(airmass, 0.6)
/home/opsim/repos/sims_skybrightness_pre/python/lsst/sims/skybrightness_pre/
↪SkyModelPre.py:49: RuntimeWarning: invalid value encountered in true_divide
  wterm = (x - xp[left])/baseline
09-11 19:45 featureSchedulerDriver INFO     end_night t=1664616564.504870, night=1␣
↪timeprogress=0.01%
09-11 19:45 featureSchedulerDriver INFO     end_night next moonphase=40.78%
09-11 19:45 featureSchedulerDriver INFO     end_night bright time waxing
```

We now have access to all the scheduler data structure to play with. In the cell bellow, we plot the TargetMapBasis function for the `r` filter.

[13]:
```
hp.mollview(sim.driver.scheduler.survey_lists[0][2].basis_functions[2]())
```


Mollweide view
-1.99516
0.0604842

```
[ ]:
```

CHAPTER 6

Information on currently available OpSim runs

- See http://astro-lsst-01.astro.washington.edu:8080/ for full MAF output.
- See https://github.com/lsst-pst/survey_strategy/blob/master/WPruns/comparisons/AlternateStrategiesComparison.md for additional comaprisons of available runs.

6.1 Run List

Run name	Note
baseline2018a	Current opsimv4 baseline
kraken_2026	Python 3 baseline2018a replacement (with dome crawl and OL)
colossus_2665	Python 3 baseline2018a replacement (with dome crawl and OL), WFD area increased by 1.5 degrees north an south
pontus_2002	Simulation of a PanSTARRs like survey
colossus_2664	WFD cadence in GP. GP proposal turned off
colossus_2667	Single visits per night survey
pontus_2489	“Many visits” 20s visits with single snap, 40s visits in u band
kraken_2035	9 Deep Drilling Fields (DDFs), 4 already decided + 5 additional
mothra_2045	2 alternating Dec bands switched every other year, WFD off
pontus_2502	2 alternating Dec bands switched every other year, WFD on at 25% level
kraken_2036	Full WFD first and last 2 years, 3 alternating dec bands in between
kraken_2042	Single 30 second snaps in all filters
kraken_2044	Simulation of a PanSTARRs like survey, no pairs
mothra_2049	Simulation of a PanSTARRs like survey, 2 alternating Dec bands switched every other year
nexus_2097	Simulation of a PanSTARRs like survey, Full WFD first and last 2 years, 3 alternating dec bands in between
astro-lsst-01_2039	No North Ecliptic Spur (NES) proposal

[1]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
# import lsst.syseng.throughputs as st
import lsst.sims.maf.db as db
import lsst.sims.maf.runComparison as rc
```

```
/astro/users/oboberg/userRepos/sims_maf/python/lsst/sims/maf/runComparison/
↪runComparison.py:25: UserWarning:
The generateDiffHtml method requires bokeh to be installed
but it is not needed to use the other methods in this class.
Run: pip install bokeh then restart your jupyter notebook kernel.
  'Run: pip install bokeh then restart your jupyter notebook kernel.')
```

[2]:
```
filterlist = ('u', 'g', 'r', 'i', 'z', 'y')
filtercolors = {'u':'b', 'g':'c', 'r':'g',
                'i':'y', 'z':'r', 'y':'m'}
```

[3]:
```
runlist = ['baseline2018a', 'kraken_2026', 'colossus_2665',
           'colossus_2664', 'astro-lsst-01_2039','colossus_2667', 'pontus_2489',
↪'kraken_2042', 'kraken_2035',
           'mothra_2045', 'pontus_2502', 'kraken_2036', 'pontus_2002', 'kraken_2044',
           'mothra_2049', 'nexus_2097']
rundirs = [r + '/all_combine' for r in runlist]
r = rc.RunComparison(baseDir='.', runlist=runlist, rundirs=rundirs)
```

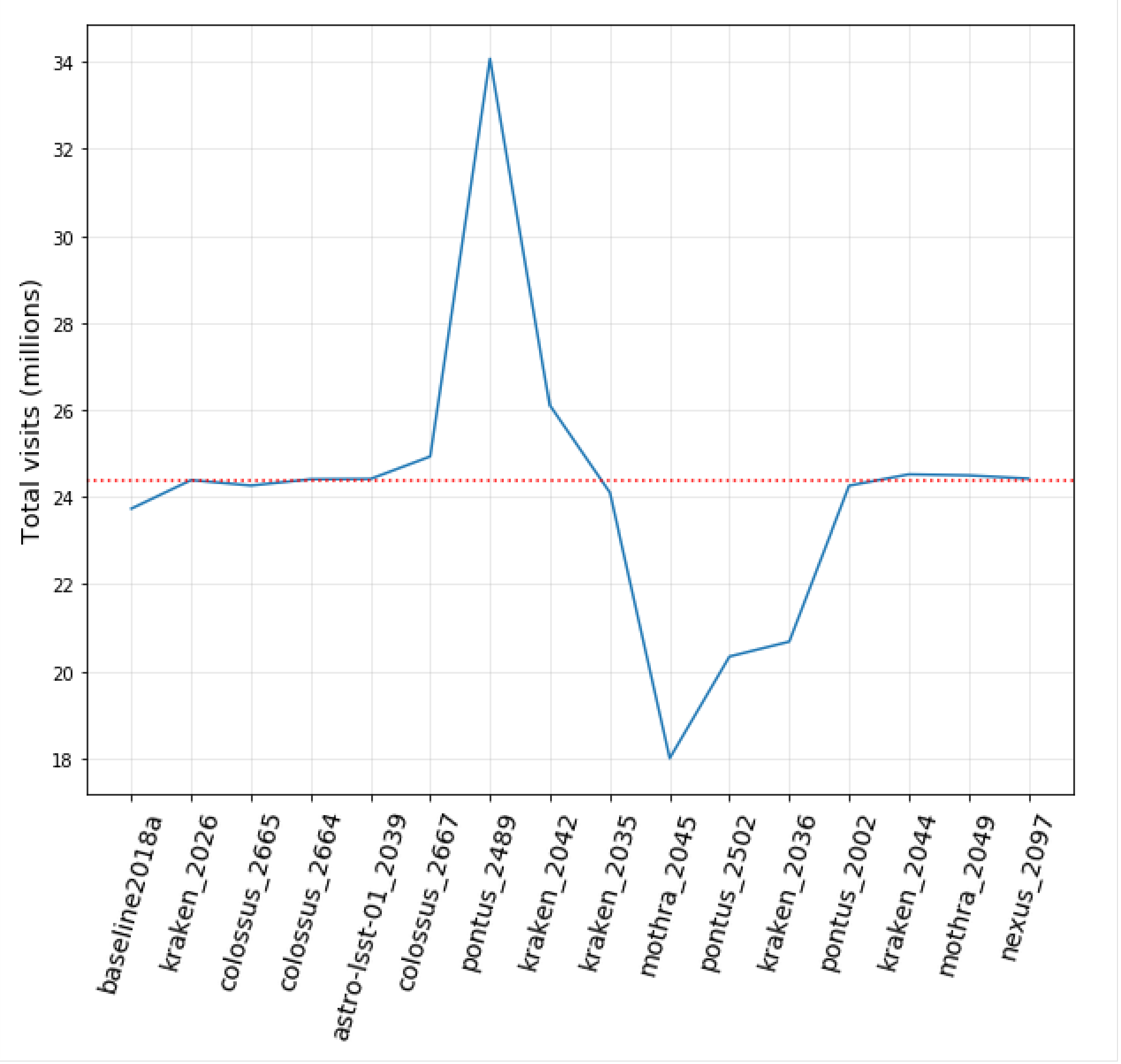
6.2 Total visits

[4]:
```
m = r.buildMetricDict(metricNameLike='Nvisits', metricMetadataLike='All props',␣
↪slicerNameLike='Unislicer')
r.addSummaryStats(m)
```

[5]:
```
plt.figure(figsize=(10, 8))
rx = np.arange(len(r.runlist))
plt.plot(rx, r.summaryStats['Nvisits All props']/100000)
plt.xticks(rx, r.runlist, rotation=75, fontsize='x-large')
plt.ylabel("Total visits (millions)", fontsize='x-large')
base = r.summaryStats['Nvisits All props']['kraken_2026']/100000
plt.axhline(base, color='r', linestyle=':')
plt.grid(True, alpha=0.3)

print(r.summaryStats['Nvisits All props']['kraken_2042']/r.summaryStats['Nvisits All␣
↪props']['kraken_2026'])
```

```
1.07012050584
```



6.3 CoaddM5 in WFD

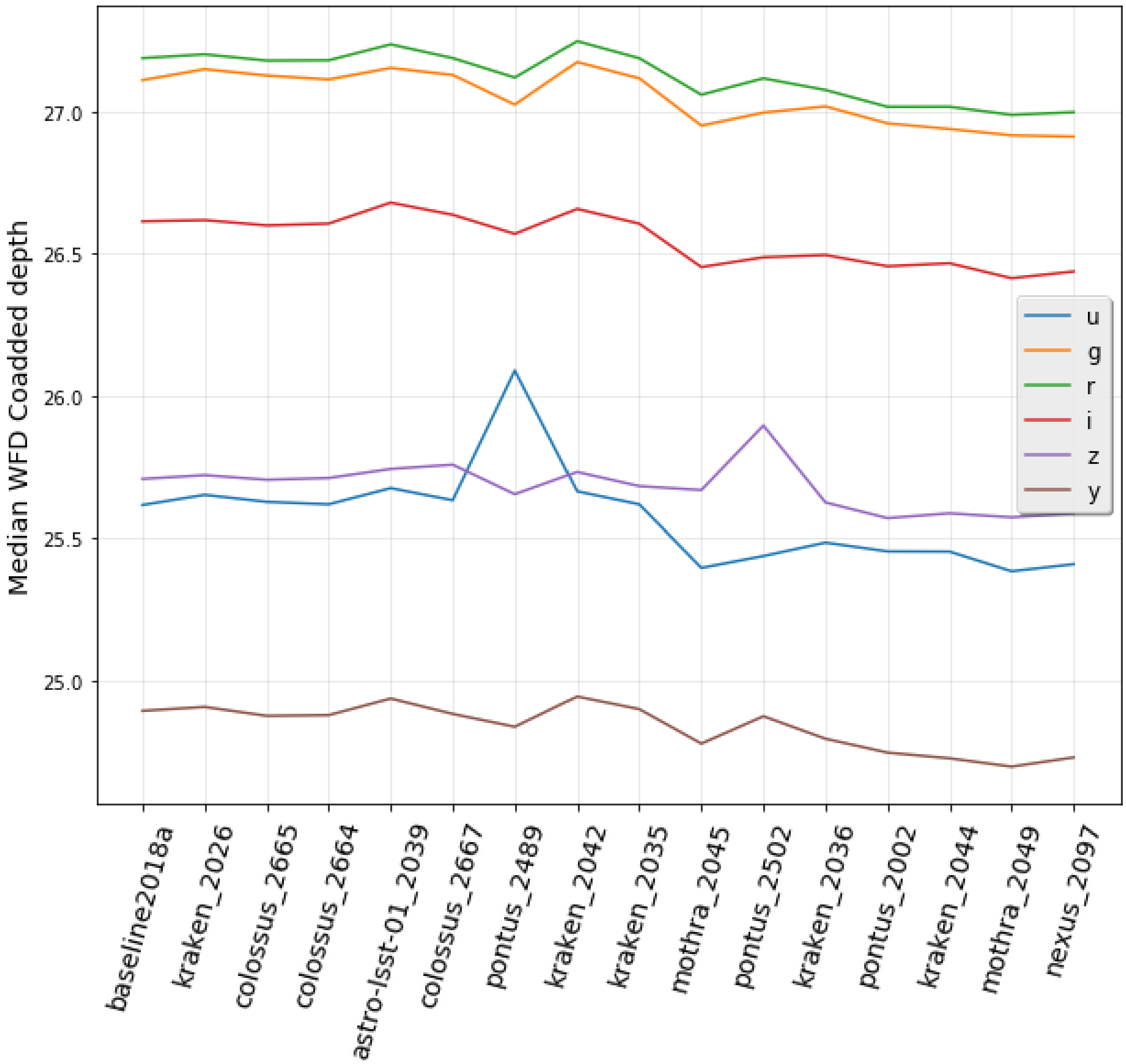
[6]:
```
for f in filterlist:
    metadata = 'WFD %s band' % f
    m = r.buildMetricDict(metricNameLike='CoaddM5',  metricMetadataLike=metadata)
    r.addSummaryStats(m)
```

[7]:
```
plt.figure(figsize=(10, 8))
for f in filterlist:
    metadata = 'WFD %s band' % f
    s = r.summaryStats['Median CoaddM5 %s HealpixSlicer' % metadata]
    plt.plot(rx, s, label=f)
plt.xticks(rx, r.runlist, rotation=75, fontsize='x-large')
```

(continues on next page)

(continued from previous page)

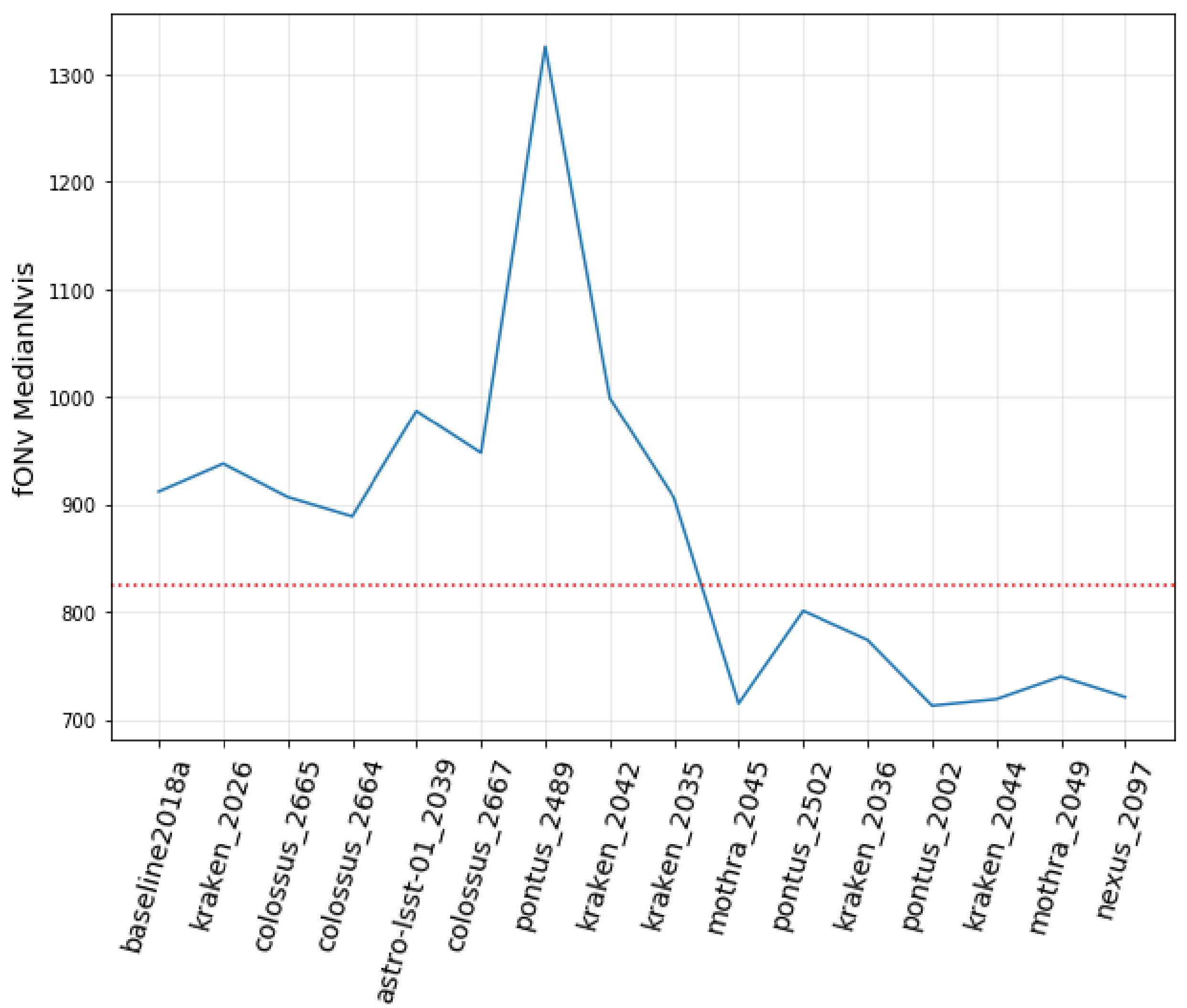
```
plt.ylabel("Median WFD Coadded depth", fontsize='x-large')
plt.legend(loc='right', fancybox=True, shadow=True, fontsize='large')
#base = r.summaryStats['Nvisits All props']['kraken_2026']/100000
#plt.axhline(base, color='r', linestyle=':')
plt.grid(True, alpha=0.3)
```

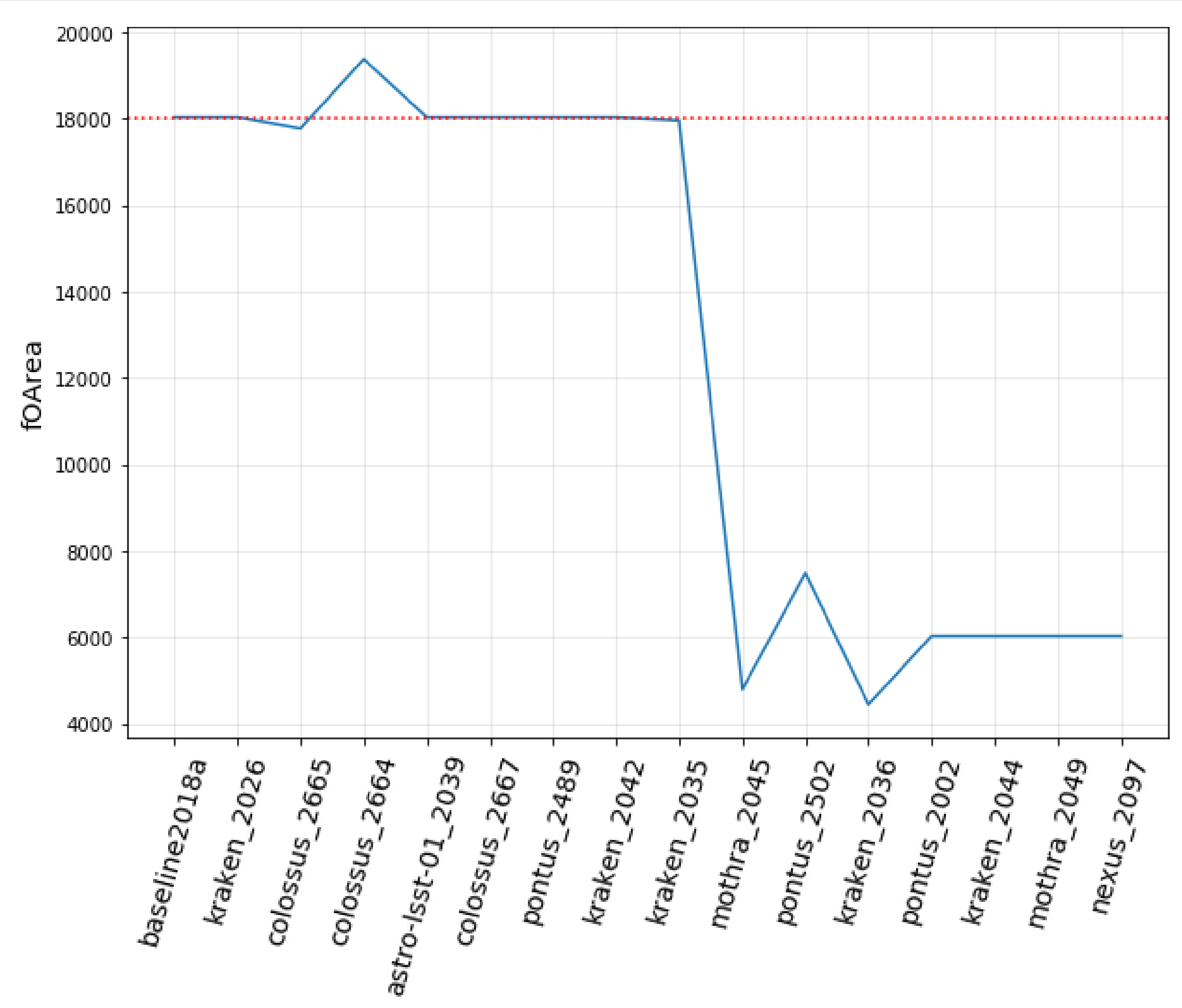


6.4 fO in WFD

```
[8]: # SRD metrics:
m = r.buildMetricDict(metricNameLike='fO', metricMetadataLike='WFD')
r.addSummaryStats(m)
```

[9]:
```
fostats = ['fONv MedianNvis', 'fOArea']
base = [825, 18000]
for f, b in zip(fostats, base):
    plt.figure(figsize=(10, 7))
    s = r.summaryStats[f + ' fO WFD HealpixSlicer']
    plt.plot(rx, s, label=f)
    plt.xticks(rx, r.runlist, rotation=75, fontsize='x-large')
    plt.ylabel("%s" % f, fontsize='x-large')
    #plt.legend(loc='right', fancybox=True, shadow=True, fontsize='large')
    plt.axhline(b, color='r', linestyle=':')
    plt.grid(True, alpha=0.3)
```

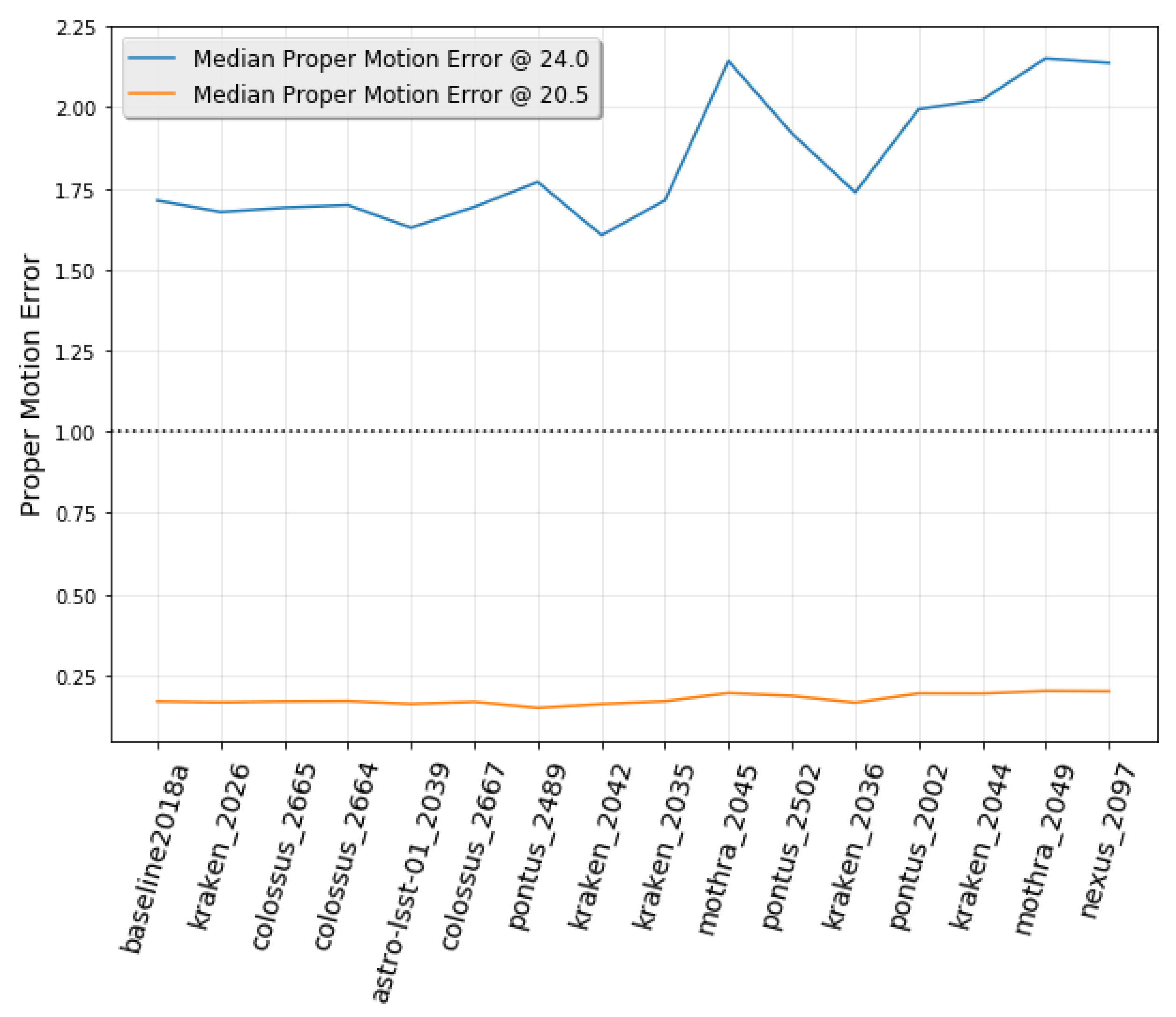




6.5 Proper Motion Error in WFD

[10]:
```
# SRD metrics:
m = r.buildMetricDict(metricNameLike='Proper Motion Error', metricMetadataLike='WFD')
r.addSummaryStats(m)
```

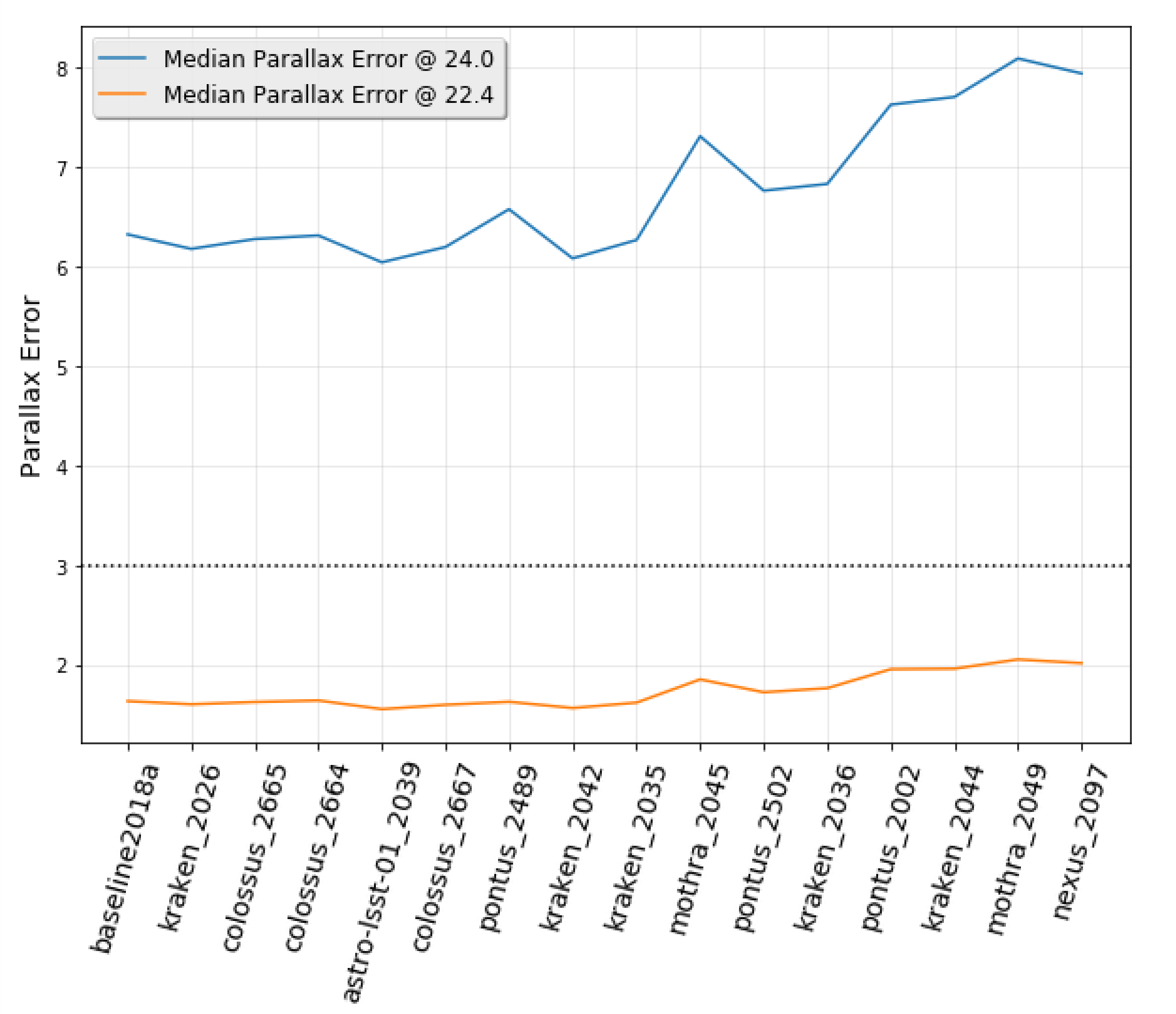
[11]:
```
pm = ['Median Proper Motion Error @ 24.0', 'Median Proper Motion Error @ 20.5']
plt.figure(figsize=(10, 7))
for f in pm:
    s = r.summaryStats[f + ' WFD HealpixSlicer']
    plt.plot(rx, s, label=f)
plt.xticks(rx, r.runlist, rotation=75, fontsize='x-large')
plt.ylabel("Proper Motion Error", fontsize='x-large')
plt.legend(loc='upper left', fancybox=True, shadow=True, fontsize='large')
plt.axhline(1, color='k', linestyle=':')
plt.grid(True, alpha=0.3)
```



6.6 Parallax Error in WFD

[12]:
```
# SRD metrics:
m = r.buildMetricDict(metricNameLike='Parallax Error', metricMetadataLike='WFD')
r.addSummaryStats(m)
```

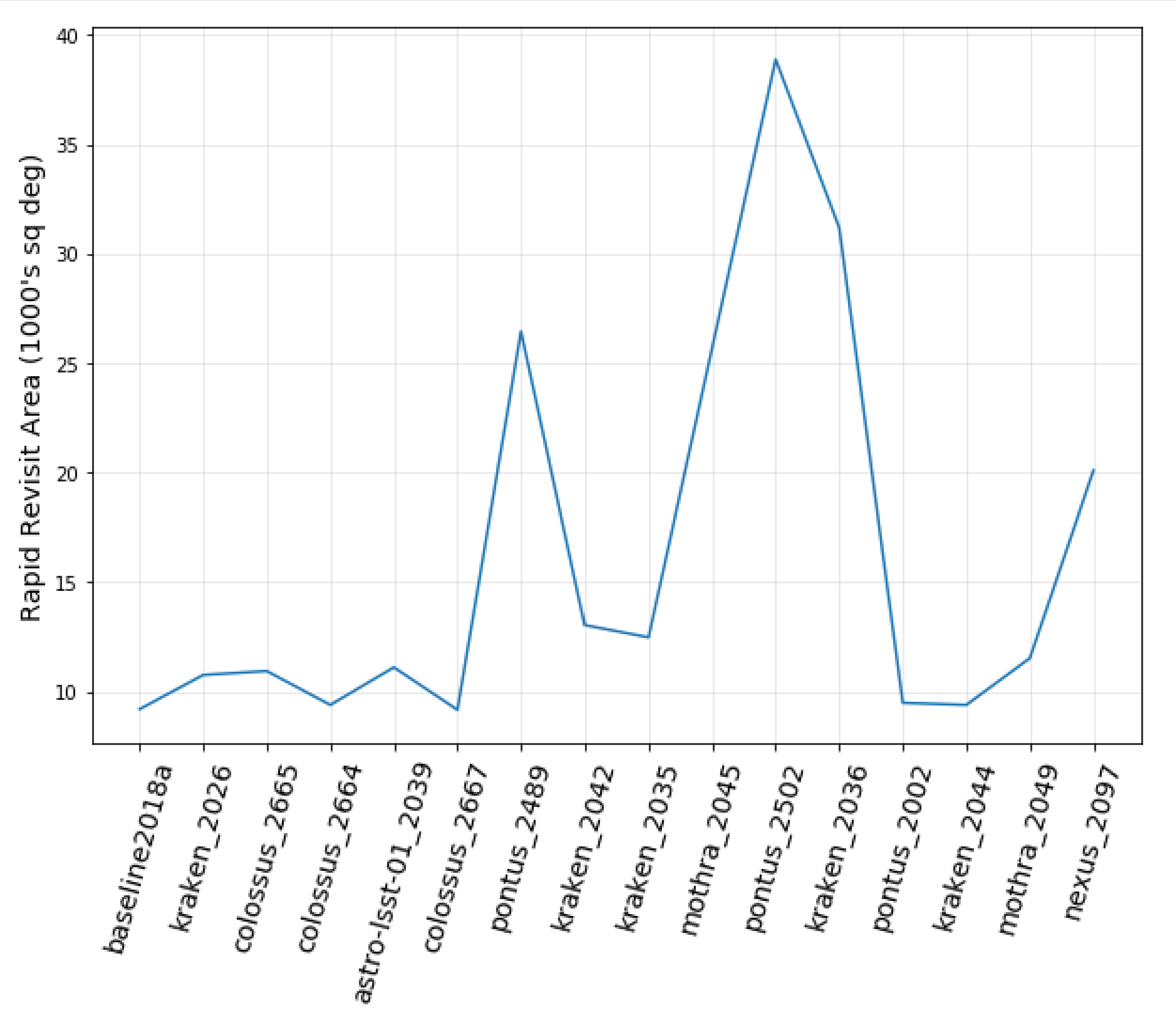
[13]:
```
pm = ['Median Parallax Error @ 24.0', 'Median Parallax Error @ 22.4']
plt.figure(figsize=(10, 7))
for f in pm:
    s = r.summaryStats[f + ' WFD HealpixSlicer']
    plt.plot(rx, s, label=f)
plt.xticks(rx, r.runlist, rotation=75, fontsize='x-large')
plt.ylabel("Parallax Error", fontsize='x-large')
plt.legend(loc='upper left', fancybox=True, shadow=True, fontsize='large')
plt.axhline(3, color='k', linestyle=':')
plt.grid(True, alpha=0.3)
```



6.7 Rapid Revisit in WFD

[14]:
```
# SRD metrics:
m = r.buildMetricDict(metricNameLike='RapidRevisit', metricMetadataLike='WFD')
r.addSummaryStats(m)
```

[15]:
```
plt.figure(figsize=(10, 7))
s = r.summaryStats['Area (sq deg) RapidRevisits WFD HealpixSlicer']
plt.plot(rx, s/1000, label=f)
plt.xticks(rx, r.runlist, rotation=75, fontsize='x-large')
plt.ylabel("Rapid Revisit Area (1000's sq deg)", fontsize='x-large')
#plt.legend(loc='upper left', fancybox=True, shadow=True, fontsize='large')
#plt.axhline(2000, color='k', linestyle=':')
plt.grid(True, alpha=0.3)
```

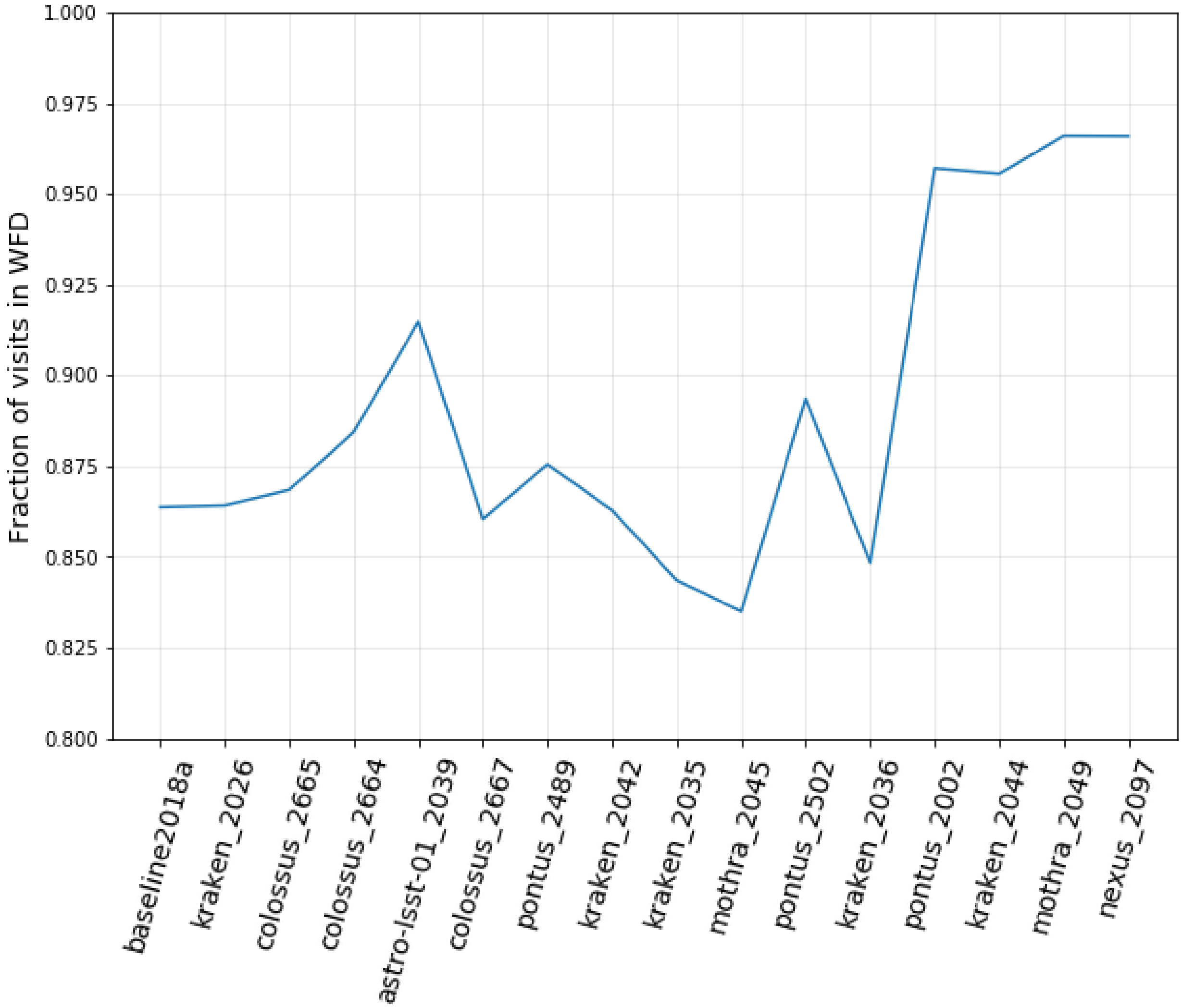


6.8 Fraction spent in WFD

[16]:
```
# Fraction of time spent on WFD?
m = r.buildMetricDict(metricNameLike='Nvisits', metricMetadataLike='WFD',␣
↪slicerNameLike='UniSlicer')
r.addSummaryStats(m)
```

[17]:
```
plt.figure(figsize=(10, 7))
s = r.summaryStats['Fraction of total Nvisits WFD']
plt.plot(rx, s, label=f)
plt.xticks(rx, r.runlist, rotation=75, fontsize='x-large')
plt.ylabel("Fraction of visits in WFD", fontsize='x-large')
#plt.legend(loc='upper left', fancybox=True, shadow=True, fontsize='large')
#plt.axhline(2000, color='k', linestyle=':')
plt.grid(True, alpha=0.3)
plt.ylim(0.8, 1.0)
```

[17]: (0.8, 1.0)



6.9 Median Inter-Night Gap in WFD (per filter)

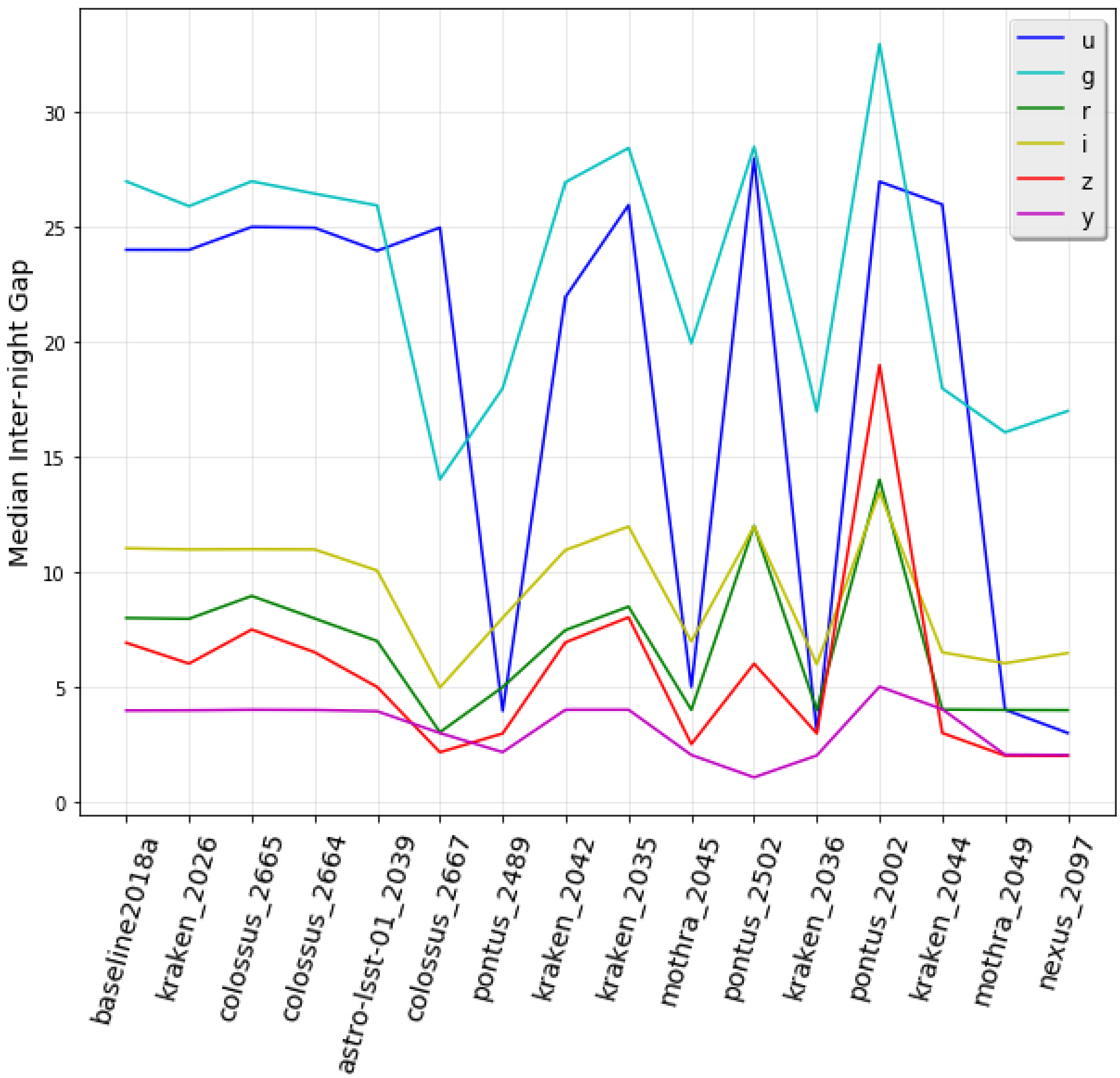
[18]:
```
# rolling cadence internight gaps
m = r.buildMetricDict(metricNameLike='Median Inter-Night Gap', metricMetadataLike='WFD
↪')
r.addSummaryStats(m)
```

[19]:
```
plt.figure(figsize=(10, 8))
for f in filterlist:
    metadata = 'WFD %s band' % f
    s = r.summaryStats['Median Median Inter-Night Gap %s HealpixSlicer' % metadata]
    plt.plot(rx, s, color=filtercolors[f], label=f)
plt.xticks(rx, r.runlist, rotation=75, fontsize='x-large')
plt.ylabel("Median Inter-night Gap", fontsize='x-large')
plt.legend(loc='upper right', fancybox=True, shadow=True, fontsize='large')
```

(continues on next page)

(continued from previous page)

```
#base = r.summaryStats['Nvisits All props']['kraken_2026']/100000
#plt.axhline(base, color='r', linestyle=':')
plt.grid(True, alpha=0.3)
```



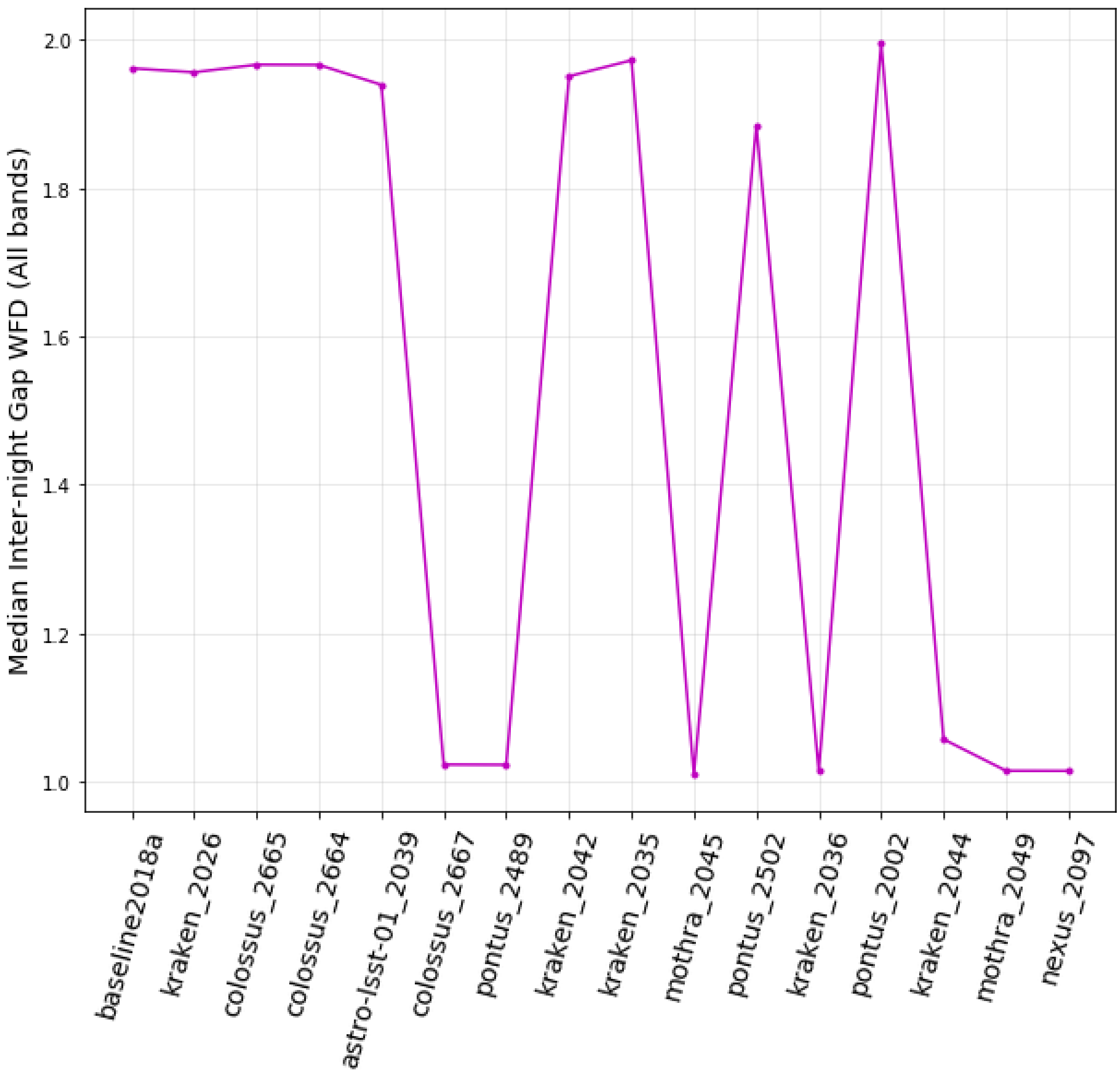
6.10 Median Inter-Night Gap in WFD (all filters)

[20]:
```
plt.figure(figsize=(10, 8))
metadata = 'WFD all bands'
s = r.summaryStats['Median Median Inter-Night Gap %s HealpixSlicer' % metadata]
plt.plot(rx, s, marker='.', color=filtercolors[f], label=f)
plt.xticks(rx, r.runlist, rotation=75, fontsize='x-large')
plt.ylabel("Median Inter-night Gap WFD (All bands)", fontsize='x-large')
#plt.legend(loc='upper right', fancybox=True, shadow=True, fontsize='large')
```

(continues on next page)

(continued from previous page)

```
#base = r.summaryStats['Nvisits All props']['kraken_2026']/100000
#plt.axhline(base, color='r', linestyle=':')
plt.grid(True, alpha=0.3)
```



6.11 Median Season Length in WFD

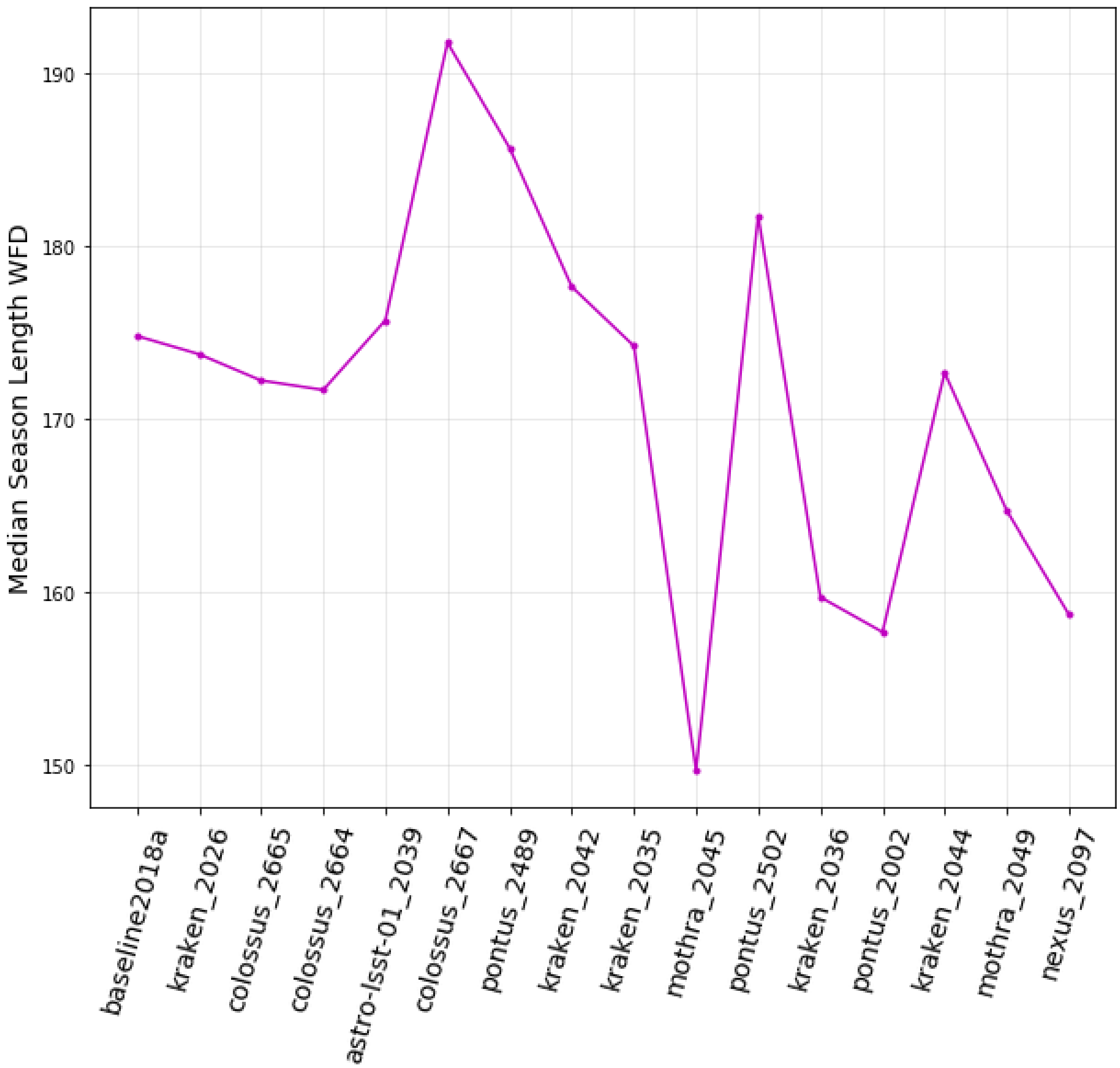
```
[21]: # season length
m = r.buildMetricDict(metricNameLike='Season Length', metricMetadataLike='WFD')
r.addSummaryStats(m)
```

```
[22]: plt.figure(figsize=(10, 8))
metadata = 'WFD all bands'
```

(continues on next page)

(continued from previous page)

```
s = r.summaryStats['Median Median Season Length %s HealpixSlicer' % metadata]
plt.plot(rx, s, marker='.', color=filtercolors[f], label=f)
plt.xticks(rx, r.runlist, rotation=75, fontsize='x-large')
plt.ylabel("Median Season Length WFD", fontsize='x-large')
#plt.legend(loc='upper right', fancybox=True, shadow=True, fontsize='large')
#base = r.summaryStats['Nvisits All props']['kraken_2026']/100000
#plt.axhline(base, color='r', linestyle=':')
plt.grid(True, alpha=0.3)
```



6.12 HA histograms

[23]:
```
import lsst.sims.maf.metricBundles as mb
ha = {}
```

(continues on next page)

(continued from previous page)

```
for f in filterlist:
    ha[f] = mb.createEmptyMetricBundle()
    # kraken_2026_HA_Histogram_g_band_ONED
    ha[f].read(os.path.join('kraken_2026', 'all_combine', 'kraken_2026_HA_Histogram_
↪%s_band_ONED.npz' % f))
```

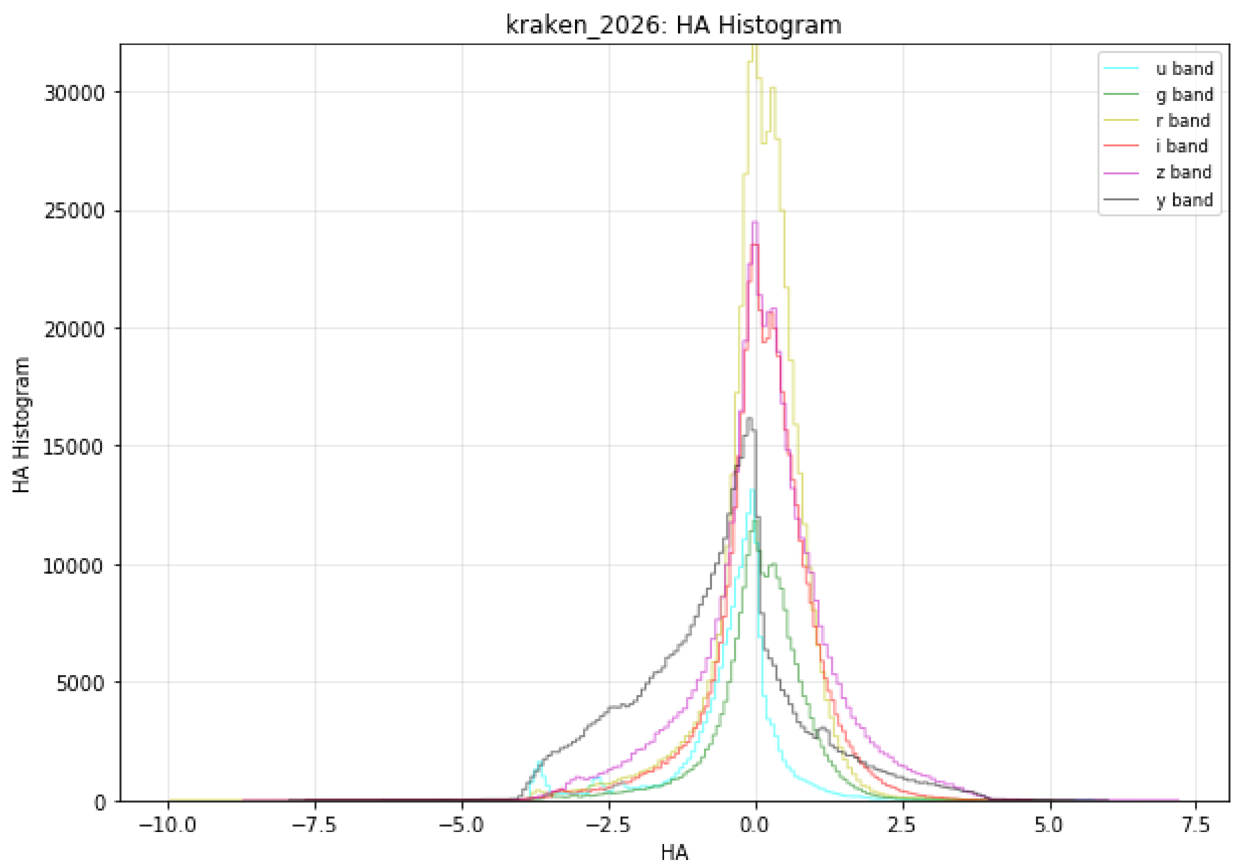
[24]:
```
import lsst.sims.maf.plots as plots
```

[25]:
```
ph = plots.PlotHandler()
```

[26]:
```
ph.setMetricBundles(ha)
```

[27]:
```
ph.plot(plotFunc=plots.OneDBinnedData(), plotDicts={'yMin': 0, 'yMax':32000, 'figsize
↪': (10, 7)})
```

[27]: 1



[ ]: