P12

Preview of the Information Site for Subaru PFS Data Reduction Pipeline (DRP)

Yongming Liang , Masayuki Tanaka, Zhuoming Li, Kiyoto Yabe, Ayumi Takahashi, Sadman Ali, Sogo Mineo, Michitaro Koike, Paul Price, Robert Lupton, and PFS Pipeline TEAM.

(NAOJ, Princeton, LAM, Kavli IPMU, ASIAA, et. al.)

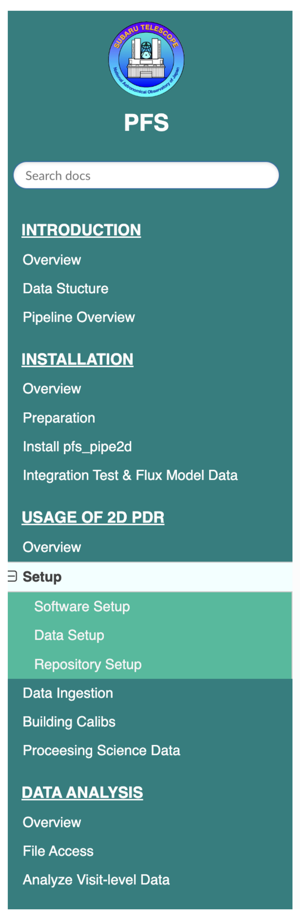
A Google form for us to learn what you need!

Abstract

Subaru Prime Focus Spectrograph (PFS) is the next-generation spectrograph mounted on the prime focus of the 8.2-meter Subaru Telescope. It is thus equipped by a wide field of view with a diameter of 1.3 deg and 2,400 fibers that can be adjusted by the “Cobra” positioners. After the exhaustive tests in engineering runs, the PFS scientific operation is coming in the 2025A semester. Besides the hardware instruments, the PFS team has also put a lot of effort into software, including the PFS data reduction pipeline (DRP), to produce high-quality products for the users once their queued observations finish. While in most cases, users can directly use the science-ready data delivered by the observatory, they can also try to improve the results by performing the data reduction for the raw data with specific optimization using the DRP. To meet this need, we are preparing a website to maintain the PFS DRP tutorials with timely information updates. This poster preview the key information of the website for the users, about what kind of content will be covered, how the tutorial can be used, and the potential direction for future upgrades. We will also collect feedback from users about what they will expect from the helpdesk, considering the upcoming PFS science operation.

1. Subaru PFS DRP

Subaru Prime Focus Spectrograph: A powerful spectrograph mounted on 8.2-m Subaru Telescope with 1.3 deg^2 FOV and 2,400 fibers.



PFS Data Reduction Pipeline (PDR): A joint effort between international institutions:
- 2D Pipeline led by Princeton for raw data reduction.
- 1D Pipeline led by LAM for analyzing data products.
- A PDR tutorial webpage hosted at NAOJ will be **online soon**!

2. Preparation

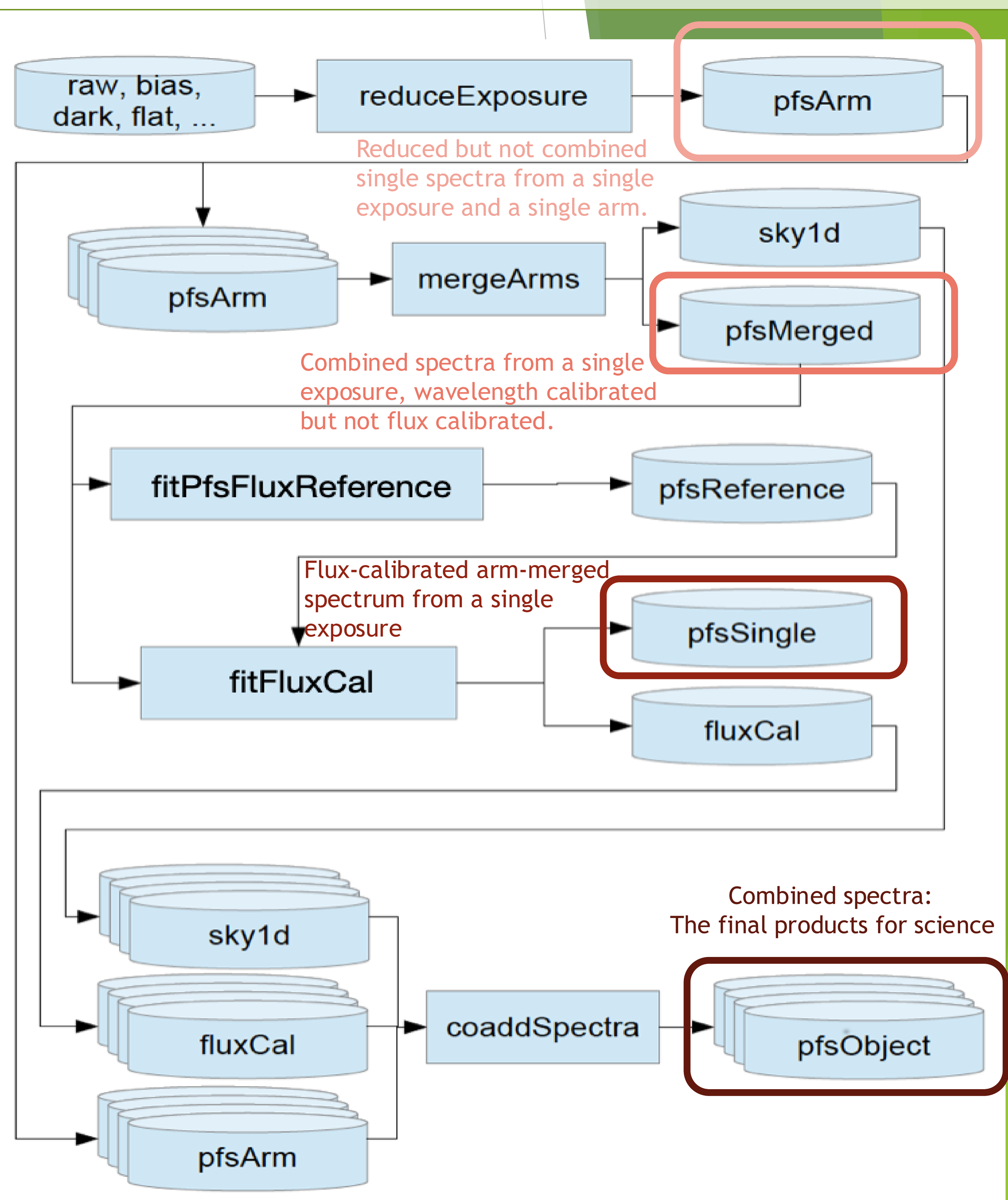
- ❑ Software Installation:
  - Install Dependencies;
    - Install Gen3 *pfs_pipe2d*;
      - Install Flux Model Data;
        - Perform Integration Test
- ❑ Environment Setup:
  - Software: *loadLSST.sh*, *setup pfs-pipe2d*;
    - Data: *checkPfsRawHeaders.py*, *checkPfsConfigHeaders.py*
      - Repository: *$DATASTORE* & *SQL Database*
- ❑ Data Ingestion with *butler*

You can see what raw datasets are in the datastore with the following command:

```
# butler query-datasets $DATASTORE --collections PFS-F/raw/all
butler query-datasets $DATASTORE --collections PFS/raw/all
```

The result looks something like this:

```
type     run                         id                     instrument arm dither pfs_design_id spectrograph detector exposure
---- ------------- ------------------------------------ ---------- --- ------ ------------- ------------ -------- --------
raw  PFS/raw/all 27217522-a357-5071-a32b-af97b5b8bee6        PFS    b   0.0            1            1        0        0
raw  PFS/raw/all 0ce0cbea-fe7c-589e-8259-30060bf20500        PFS    b   0.0            1            1        0        1
[...]
raw  PFS/raw/all 570092eb-f571-5631-8d20-11acbeabc640        PFS    r   0.0            3            1        1       26
raw  PFS/raw/all f8e3ae71-2cdf-5e55-bc42-4a4fb913770c        PFS    r   0.0            4            1        1       27
```



3. Running 2D DRP

- ❑ Build Calibration Products:

  Bias → Dark → Flat → detectorMap → fiberProfiles → fiberNorms

```
pipetask run \
--register-dataset-types \                      # register the dataset types from the pipeline
-j $CORES \                                     # number of cores to use in parallel
-b $DATASTORE \                                 # datastre directory to use
--instrument lsst.obs.pfs.PrimeFocusSpectrograph \  # the instrument PFS
-i PFS/raw/sps,PFS/calib \                      # input collection (comma-separated)
-o "$RERUN"/bias \                              # output CHAINED collection
-p $DRP_STELLA_DIR/pipelines/bias.yaml \        # pipeline configuration file to use
-d "instrument='PFS' AND exposure.target_name = 'BIAS'" \ # or, for example: -d "visit IN (123456..123466)" \
--fail-fast \                                   # immediately stop the ingestion process if error
-c isr:doCrosstalk=False                        # (optional) turn off the crosstalk correction
```

- ❑ Process Science Data:
  - Define Collection (a new concept from Gen3)

```
defineCombination.py $DATASTORE PFS object --where "exposure.target_name = 'OBJECT'"
defineCombination.py $DATASTORE PFS quartz --where "exposure.target_name = 'FLAT' AND dither = 0.0"
```

  - One Shot to Run the Pipeline

```
# Science pipeline
pipetask run \
--register-dataset-types -j $CORES -b $DATASTORE \
--instrument lsst.obs.pfs.PrimeFocusSpectrograph \
-i PFS/raw/all,PFS/raw/pfsConfig,PFS/calib \
-o "$RERUN"/science \
-p '$DRP_STELLA_DIR/pipelines/science.yaml' \
-d "combination = 'object'" \
--fail-fast \
-c isr:doCrosstalk=False \
-c fitFluxCal:fitFocalPlane.polyOrder=0 \
-c reduceExposure:doApplyScreenResponse=False \
-c reduceExposure:doBlackSpotCorrection=False \
-c 'reduceExposure:targetType=[SCIENCE, SKY, FLUXSTD]'
```

* **NOTE**: The flag-sets in this poster are shown as an example. The configuration may still be updated for better performance in future.

- ❑ Export the Products Following PFS Datamodel:

```
exportPfsProducts.py -b $DATASTORE -i PFS/raw/pfsConfig,"$RERUN"/science -o export
```

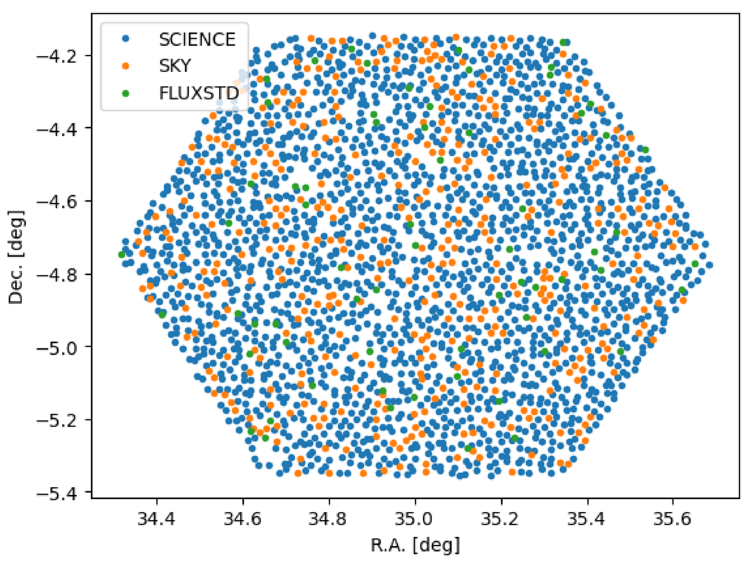
4. Data Analysis

- ❑ Access the data and pfsConfig with butler (same as on Science Platform)

```
from lsst.daf.butler import Butler
import pfs.datamodel as datamodel
```

```
$DATASTORE ='/work/pfs/datastore/'
$COLLECTION = 'user/tenmontaro/20250128'
butler =Butler($DATASTORE, collection=[$COLLECTION])
```

```
visit = 115893
pfsConfig = butler.get('pfsConfig', visit=visit)
for i in range (1,4):
    index = np.where(pfsConfig.targetType == i)
    plt.plot(pfsConfig.ra[index], pfsConfig.dec[index],
             '.', label=datamodel.TargetType(i))
plt.legend()
plt.xlabel('R.A. [deg]')
plt.ylabel('Dec. [deg]')
```



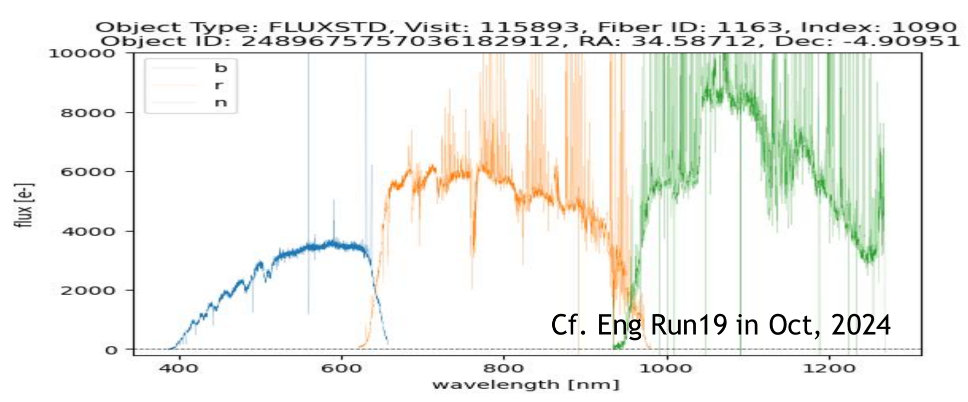
Cf. Eng Run19 in Oct, 2024

Data Structure after *exportPfsProducts.py*:

```
$EXPORT/
├── detectorMap : directory for detectorMap of different arm & spectrograph
├── images : directory for raw 2D images of each visit
├── pfsConfig : directory for pfsConfig files
├── pfsArm : directory for pfsArm products
├── pfsMerged : directory for pfsMerged products
│   ├── 20241024 : exposures on date 2024/10/24
│   ├── 20241025 : exposures on date 2024/10/24
│   └── 20241026 : exposures on date 2024/10/240
│       ├── pfsMerged-116413.fits : psfMerged spectra of visit=116413
│       ├── pfsMergedLsf-116413.pickle : psfMerged LSF of visit=116413
│       ├── ...
├── pfsSingle : directory for pfsSingle products
│   ├── -0001 : targets with catId=-1
│   ├── 01002 : targets with catId=1002, i.e., PS1
│   ├── 03006 : targets with catId=3006
│   └── 10086 : targets with catId=10086
│       └── 00001 : tract number, meaningless at the moment
│           └── 1,1 : patch number, meaningless at the moment
│               ├── pfsSingle-[catId]-[tract]-[patch]-[objId]-[visit].fits : psfSingle spectra of catId=10086
│               ├── pfsSingleLsf-[catId]-[tract]-[patch]-[objId]-[visit].fits : psfSingle LSF of catId=10086
│               ├── ...
└── pfsObject
    ├── -0001: (same as above)
```

- ❑ Play with single-visit data

```
# fiberID
fiberId = np.array([1163, ])
# Index of the fiber in the pfsConfig
index = np.where(pfsConfig.fiberId == fiberId)[0][0]
# Spectrograph; here, it is 2.
from pfs.utils.fibers import spectrographFromFiberId
spectrograph = spectrographFromFiberId(fiberId).item()
# for pfsArm, we need to know which spectrograph the object is observed with. We get the spectrograph ID
for arm in ('b','r','n'):
    pfsArm = butler.get('pfsArm', dataId=dict(visit=visit, arm=arm, spectrograph=spectrograph))
    idx_arm = np.where(pfsArm.fiberId == pfsConfig.fiberId[index])[0][0]
    plt.plot(pfsArm.wavelength[idx_arm], pfsArm.flux[idx_arm], '-', label=arm, linewidth=0.1)
# Skipped unrelated parts #
plt.show()
```



Cf. Eng Run19 in Oct, 2024

Contact to: PFS Helpdesk

❑ Yongming Liang: yongming.liang@nao.ac.jp

❑ Address:

National Astronomical Observatory of Japan
2-21-1 Osawa, Mitaka, Tokyo

5. Future Updates

1. Since the PFS PDR is still in a rapidly evolving phase, we will keep the information updated regularly after the website online. Please stay tuned!
2. We will include the data analysis for coadded pfsObject data, as well as the LAM 1D DRP usage.
3. We will collect the questions and problems from users once PFS is on board, and will try to deliver frequent Q&A to support observers.