analysis_tools
setup guide

Rance Solomon (LAPP)

Commissioning du Plan Focal dans LSST-France

LAPP, Annecy
March 2024

VERA C. RUBIN OBSERVATORY

LAPP

purpose of analysis_tools

analysis_tools sont conçus pour fournir des outils de traçage QA informatifs et cohérents pour le flux de données Rubin.
(analysis_tools is designed to provide informative and consistent QA plotting tools for the Rubin data flow.)

L'objectif est de fournir les éléments de base d'une analyse très complexe, quel que soit l'ensemble de données.
(The purpose is to provide the building blocks for very complex analysis regardless of the dataset.)

Il contient déjà de nombreuses mesures et graphiques intéressants, mais à mesure que notre travail évolue et que de nouveaux problèmes surviennent, nous devrons compléter la library.
(It already contains many of the interesting metrics and plots, but, as our work evolves and new problems arise, we will have to add to the library.)

Plus important encore, c'est ainsi que les équipes de mise en service effectueront le QA sur la ciel. Donc si nous voulons contribuer, nous devons en être conscients.
(Most importantly, this is how commissioning teams will do on-sky QA. So if we want to contribute then we need to be on top of this.)

setup analysis_tools

Cela devrait faire fonctionner:
(This should get it running:)

```
$ source /cvmfs/sw.lsst.eu/linux-x86_64/lsst_distrib/[CHOSEN_WEEKLY_RELEASE]/loadLSST.bash
$ setup lsst_distrib
$ git clone https://github.com/lsst/analysis_tools.git
$ cd analysis_tools
$ setup -r . -t $USER
$ scons -Q -j 6 opt=3
```

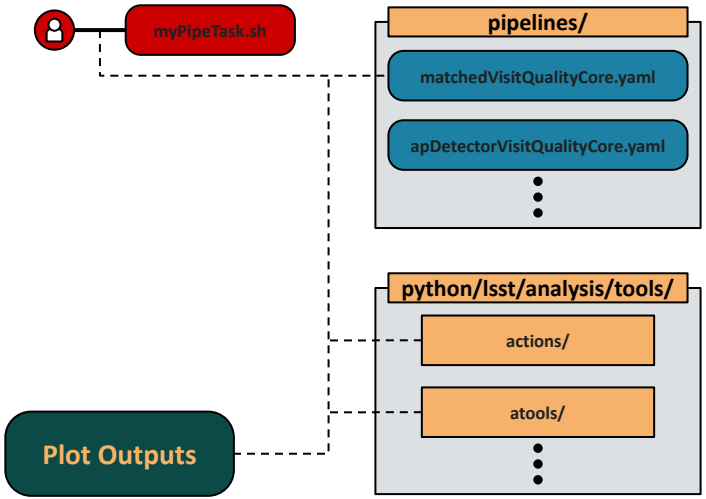
To do every time

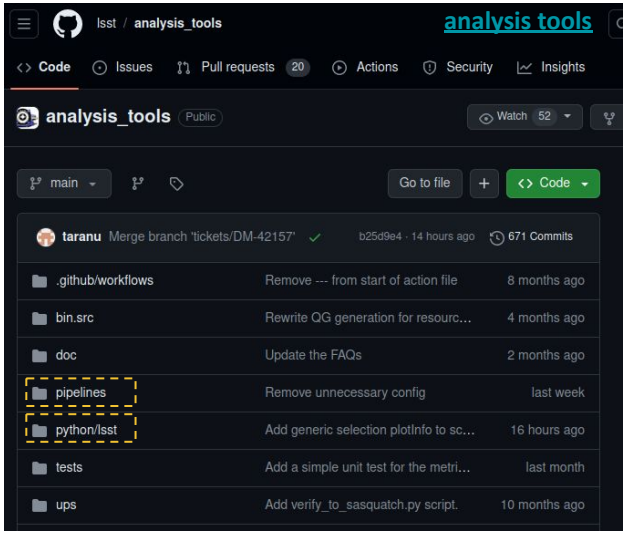
Only needed if running a version of analysis_tools not already in weekly release

Et voila! Vous avez maintenant votre propre analysis_tools.
(And voila! You now have your own analysis_tools.)

Pour exécuter analysis_tools, vous pouvez travailler dans Jupyter ou sur la ligne de commande. Nous examinerons la ligne de commande.
(To run analysis_tools, you can work in Jupyter or on the command line. We will look at the command line.)

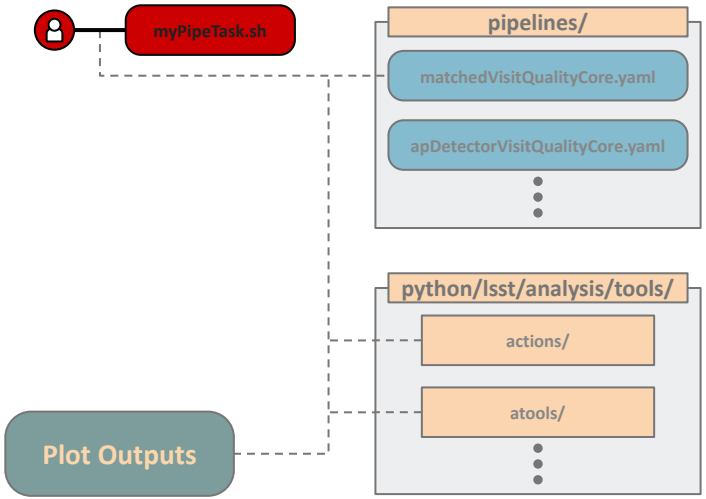
normal workflow





Dans la plupart des cas, il vous suffit d'accéder à `pipelines/` et `python/lsst/analysis/tools/` avec `actions/` ou `atools/`.
(For most cases you only need to access `pipelines/` and `python/lsst/analysis/tools/` with either `actions/` or `atools/`.)

normal workflow – user level




```
#!/bin/bash

#HSC:
pipetask run -p ./pipelines/matchedVisitQualityCore.yaml \
    -b /sdf/group/rubin/repo/oga \
    -i LATISS/runs/AUXTEL_DRP_IMAGING_20230509_20240201/w_2024_05/PREOPS-4871 \
    -o u/rsolomon/atoolsScratch \
    -d "instrument='LATISS' AND skymap='latiss_v1' AND band='g' AND tract=3864 AND (patch=236 or patch=237)" \
    --register-dataset-types
```

`-p`: pipeline file
`-b`: butler repo
`-i`: input collection
`-o`: output collection
`-d`: data-id (optional; to plot only specific items in collection)
`--register-dataset-types`: only necessary if building a new dataset type

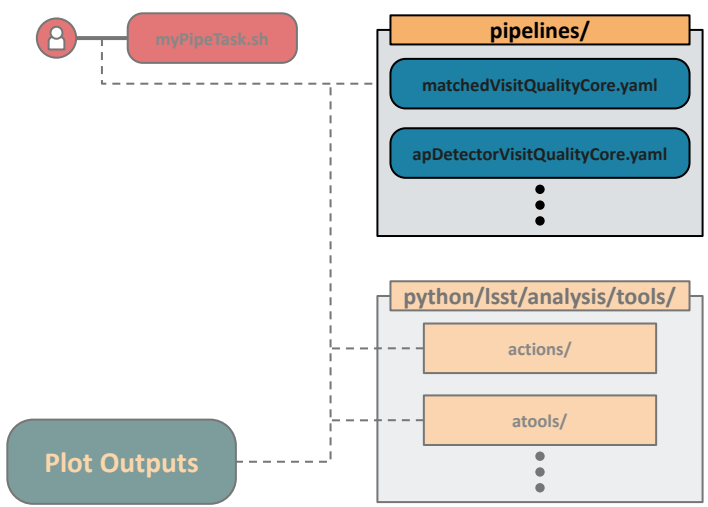
After running the first time, add these two lines so that the plots are updated:

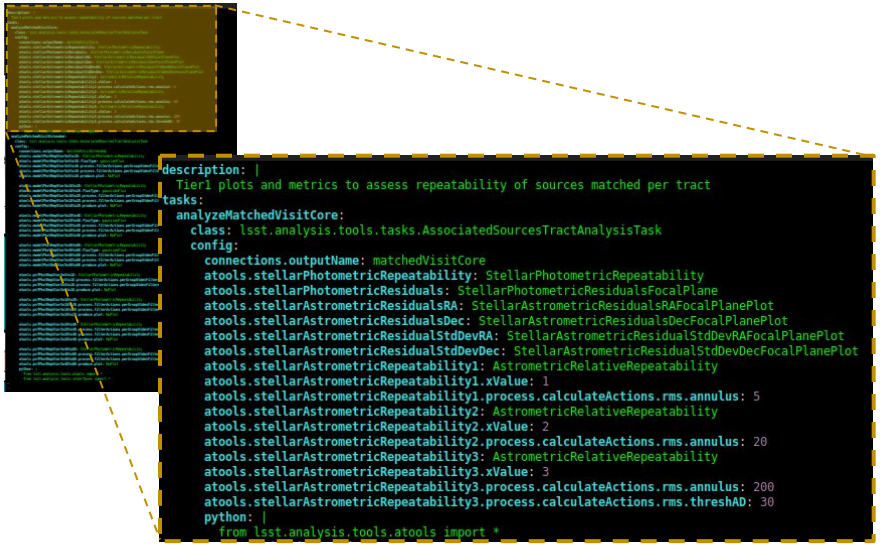
```
--prune-replaced=purge
--replace-run
```

L'utilisateur exécute un pipetask avec toutes les méta-informations nécessaires : le pipeline à exécuter, les emplacements des données, les emplacements de sortie, etc.

(The user runs a pipetask with all the necessary meta-information: the pipeline to run, data locations, output locations, …)

normal workflow – pipelines



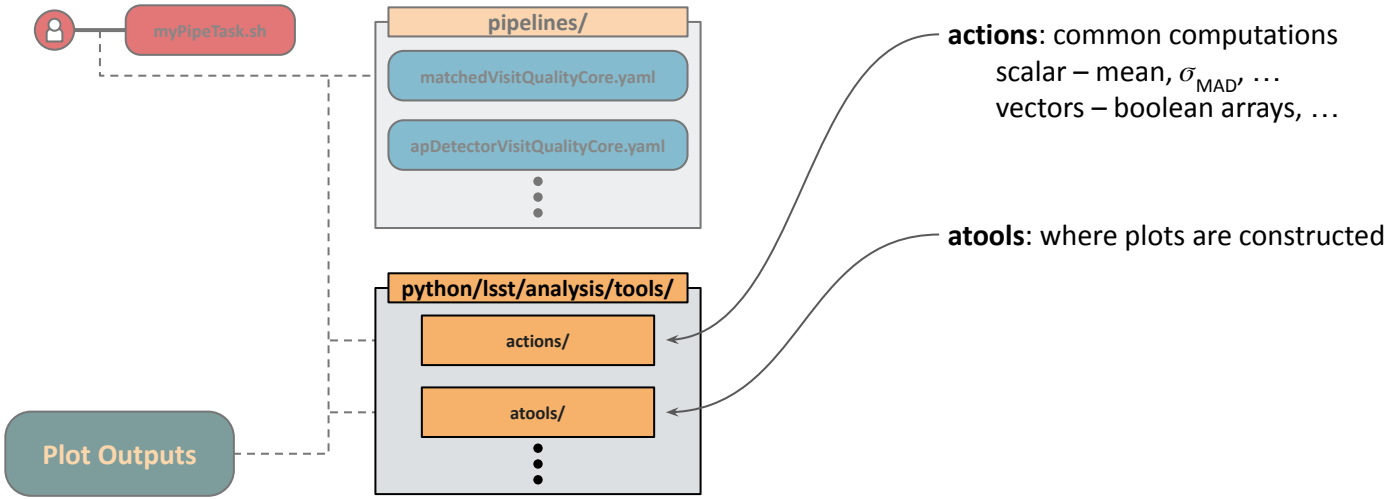


Le pipeline définit les éléments que vous souhaitez examiner. C'est ici que vous listez tous les calculs et tracés que vous souhaitez effectuer.
(The pipeline defines what things you want to look at. It is where you list all the computations and plots you want to be made.)

Les fichiers *Core.yaml contiennent les points d'intérêt habituels (e.g. des vérifications de base pour un ‘matched visit quality’).
*(*Core.yaml files contain the usual points of interest (e.g. base plots to check matched visit quality).)*

Les fichiers *Extended.yaml contiennent des vérifications supplémentaires à effectuer au cas où quelque chose semble étrange.
*(*Extended.yaml files contain extra checks to perform in case something looks strange.)*

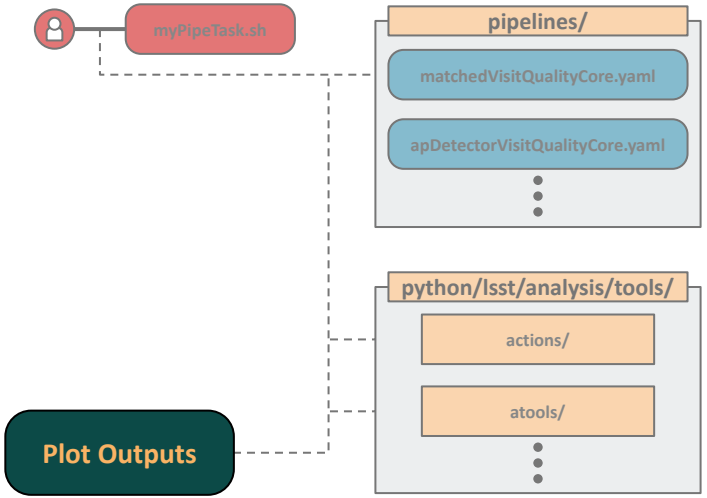
normal workflow – python/…/tools

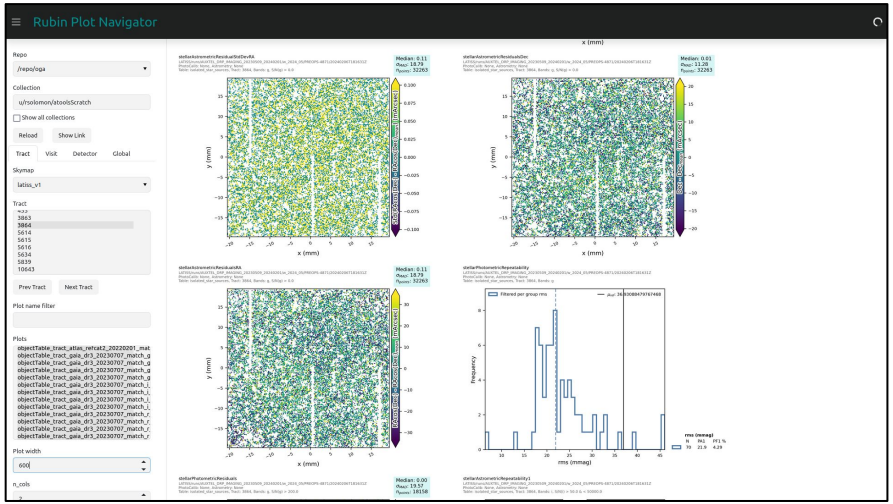


C'est ici que tous les calculs et tracés sont rassemblés.

(This is where all of the calculations and plots are assembled.)

normal workflow – output (Rubin Plot Navigator)

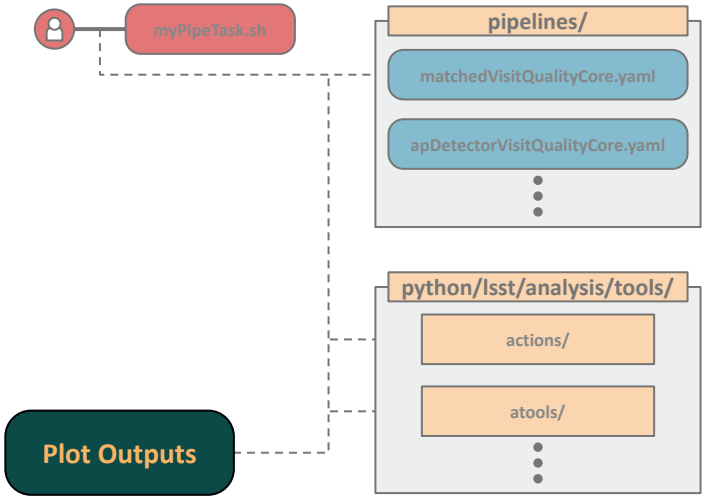


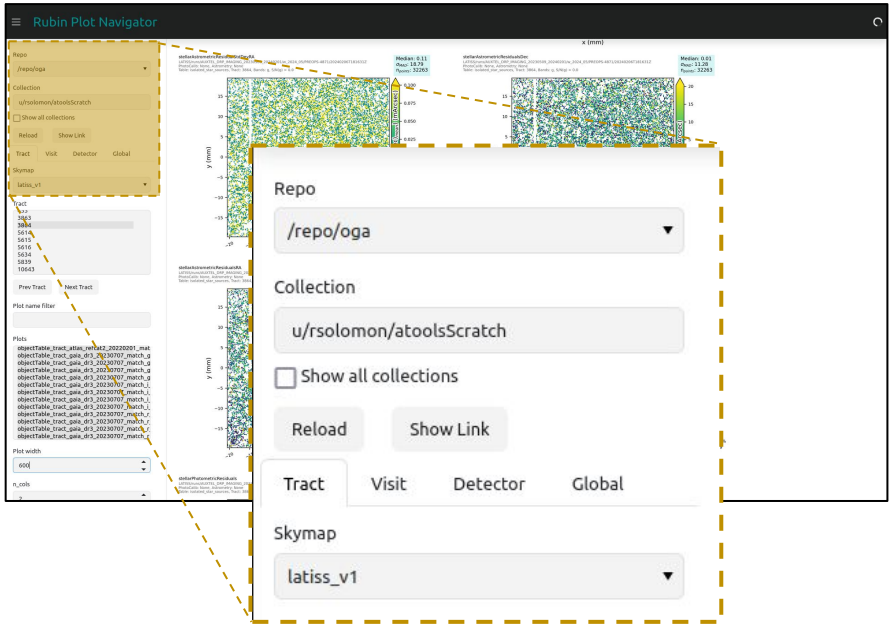


Une fois que vous avez exécuté la pipetask, tous les tracés et fichiers sont stockés dans votre emplacement de sortie. Et nous pouvons accéder à ces tracés à l'aide du *Rubin Plot Navigator*.

(Once you run the pipetask, all plots and files are stored in your -o output location. And we can access these plots using the Rubin Plot Navigator.)

normal workflow – output (Rubin Plot Navigator)

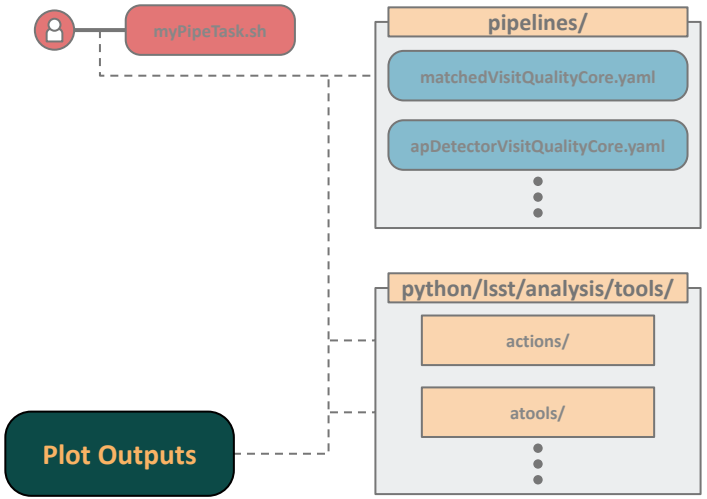


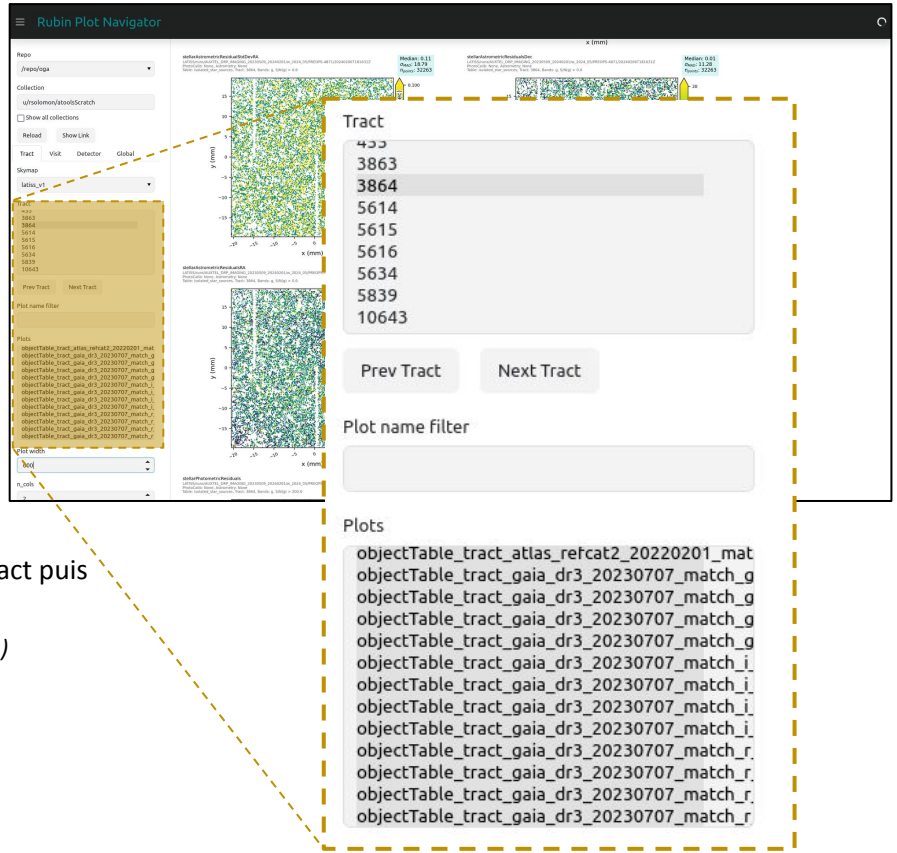


Sélectionnez le repo et la collection que vous souhaitez afficher.
(Select the repo and collection you want to look at.)

En fonction de l'analyse, vous pouvez voir les tracés d'analyse de la qualité au niveau tract, visit, détecteur, et global.
(Depending on the analysis you can see tract, visit, detector, global level QA plots.)

normal workflow – output (Rubin Plot Navigator)

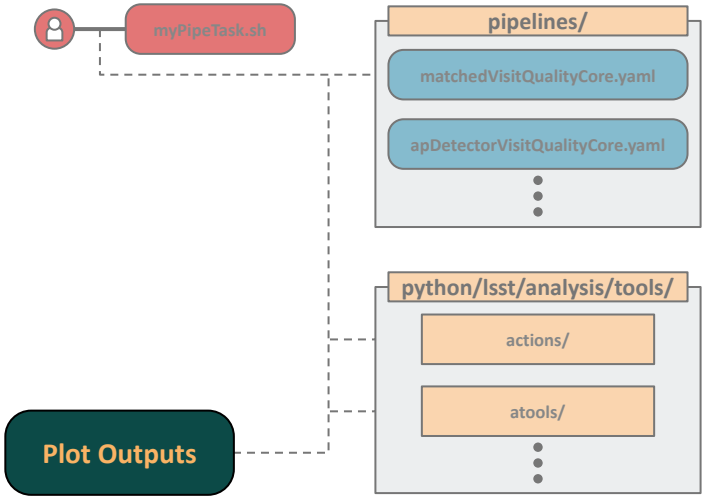


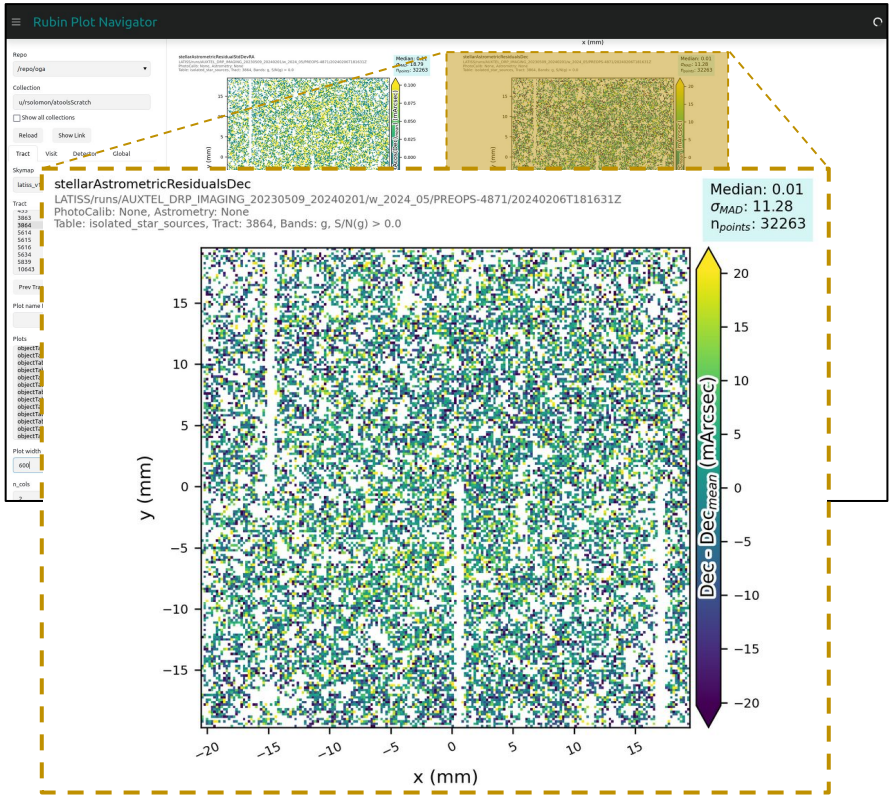


Ici, nous voyons le niveau du tract, nous sélectionnons donc notre tract puis les parcelles que nous voulons voir.

(Here we see tract level so we select our tract and then the plots we want to see.)

normal workflow – output (Rubin Plot Navigator)

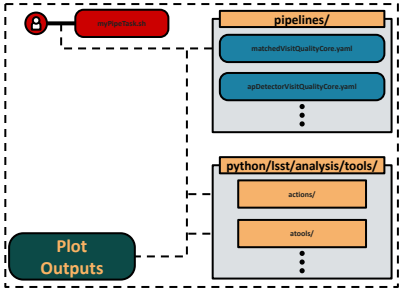




Les plots doivent contenir toutes les informations pertinentes ainsi que les mesures importantes.

(The plots should contain all relevant information along with important metrics.)

Example: adding a new analysis



Disons que nous voulons examiner la différence de magnitude entre Ap12 et PSF pour les étoiles coaddées en coordonnées RA et DEC.
(Let's imagine we want to look at the magnitude difference between Ap12 and PSF for coadded stars in RA and DEC coordinates.)

Nous utiliserons les données HSC pour la démonstration.
(We will use HSC data for demonstration.)

Il y a 3 étapes:
(There are 3 steps.)

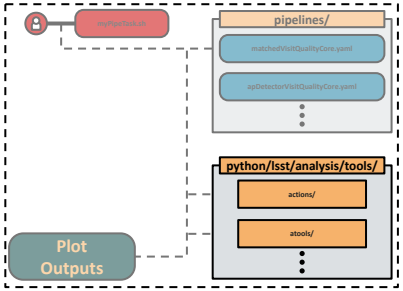
1. Ajouter la nouvelle classe plot dans `python/lsst/analysis/tools/atools/`.
(1. Add the new plot class in `python/lsst/analysis/tools/atools/`*.)*

2. Créez un pipeline pour la nouvelle analyse dans `pipelines/`.
(2. Create a pipeline for the new analysis in `pipelines/`*.)*

3. Exécutez votre pipetask.
(3. Run your pipetask.)

Cela créera une version différente analysis_tools, vous devrez donc exécuter `setup -r . -t $USER` et `scons -Q -j 6 opt=3` étapes dans la configuration initiale.
(This will create a different build of analysis_tools so you will need to run the `setup -r . -t $USER` *and* `scons -Q -j 6 opt=3` *steps in the initial setup.)*

Example: adding a new analysis



Puisque nous examinons les étoiles coaddées, cette classe s'intègre parfaitement dans le fichier **skyObjects.py**.

(Since we are looking at coadded stars, this class fits well in the skyObjects.py file.)

Nous effectuons des coupes de sélection : coadded, SN, extendedness.

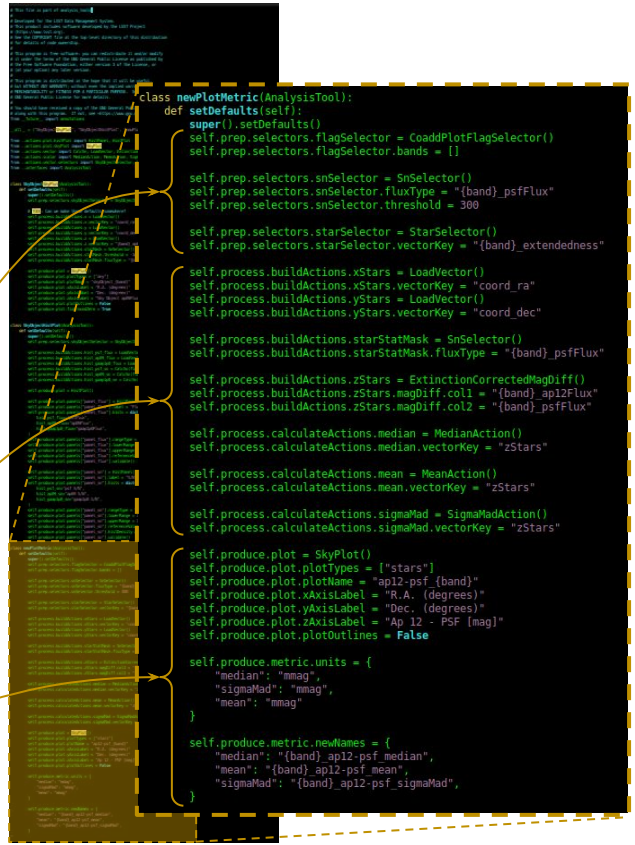
(Make selection cuts: coadded, SN, extendedness)

Ajoutez des coordonnées et des magnitudes et calculez les métriques avec la coupe SN appliquée.

(Add the coordinates and magnitudes and compute the metrics with a SN cut applied.)

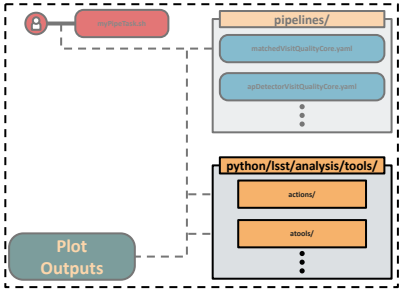
Déclarez le type de tracé, les étiquettes et les unités à afficher.

(Declare the plot type, labels, and units to be displayed.)



```
class newPlotMetric(AnalysisTool):
    def setDefaults(self):
        super().setDefaults()
        self.prep.selectors.flagSelector = CoaddPlotFlagSelector()
        self.prep.selectors.flagSelector.bands = []

        self.prep.selectors.snSelector = SnSelector()
        self.prep.selectors.snSelector.fluxType = "{band}_psfFlux"
        self.prep.selectors.snSelector.threshold = 300

        self.prep.selectors.starSelector = StarSelector()
        self.prep.selectors.starSelector.vectorKey = "{band}_extendedness"

        self.process.buildActions.xStars = LoadVector()
        self.process.buildActions.xStars.vectorKey = "coord_ra"
        self.process.buildActions.yStars = LoadVector()
        self.process.buildActions.yStars.vectorKey = "coord_dec"

        self.process.buildActions.starStatMask = SnSelector()
        self.process.buildActions.starStatMask.fluxType = "{band}_psfFlux"

        self.process.buildActions.zStars = ExtinctionCorrectedMagDiff()
        self.process.buildActions.zStars.magDiff.col1 = "{band}_ap12Flux"
        self.process.buildActions.zStars.magDiff.col2 = "{band}_psfFlux"

        self.process.calculateActions.median = MedianAction()
        self.process.calculateActions.median.vectorKey = "zStars"

        self.process.calculateActions.mean = MeanAction()
        self.process.calculateActions.mean.vectorKey = "zStars"

        self.process.calculateActions.sigmaMad = SigmaMadAction()
        self.process.calculateActions.sigmaMad.vectorKey = "zStars"

        self.produce.plot = SkyPlot()
        self.produce.plot.plotTypes = ["stars"]
        self.produce.plot.plotName = "ap12-psf_{band}"
        self.produce.plot.xAxisLabel = "R.A. (degrees)"
        self.produce.plot.yAxisLabel = "Dec. (degrees)"
        self.produce.plot.zAxisLabel = "Ap 12 - PSF [mag]"
        self.produce.plot.plotOutlines = False

        self.produce.metric.units = {
            "median": "mmag",
            "sigmaMad": "mmag",
            "mean": "mmag"
        }

        self.produce.metric.newNames = {
            "median": "{band}_ap12-psf_median",
            "mean": "{band}_ap12-psf_mean",
            "sigmaMad": "{band}_ap12-psf_sigmaMad",
        }
```

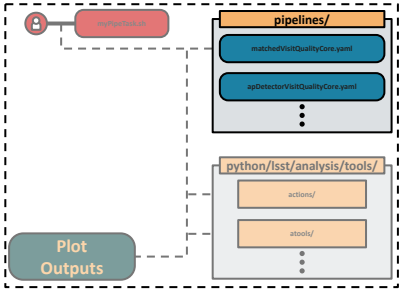
Example: adding a new analysis



N'oubliez pas de mettre à jour les importations nécessaires.

(Don’t forget to update the necessary imports.)

Example: adding a new analysis




```yaml
description: |
  An example pipeline to run our new plot
tasks:
  testNewPlot:
    class: lsst.analysis.tools.tasks.ObjectTableTractAnalysisTask
    config:
      connections.outputName: testNewPlot
      atools.newPlot: newPlotMetric
      python: |
        from lsst.analysis.tools.atools import *
```

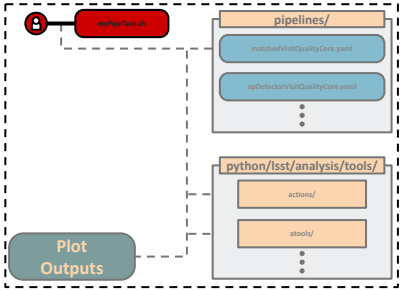
Maintenant, nous préparons le pipeline. Nous allons créer un nouveau fichier: `pipelines/myNewPipeline.yaml`

(Now we prepare the pipeline. We will create a new file: `pipelines/myNewPipeline.yaml`.)

Ici, nous voulons exécuter sur un ensemble de données objecTable_tract que nous pouvons voir par la `class`. La plupart des types d'ensembles de données ont déjà une tâche existante, mais dans certains cas, vous devrez peut-être créer la vôtre.

(Here we are wanting to run on an objectTable_tract dataset which we can see by the `class`. Most dataset types already have an existing Task, but in some cases you may have to create your own.)

Example: adding a new analysis




```
#!/bin/bash

#HSC:
pipetask run \
    -p pipelines/myNewPipeline.yaml \
    -b /sdf/group/rubin/repo/main/butler.yaml \
    -i HSC/runs/RC2/w_2022_28/DM-35609 \
    -o u/rsolomon/newPlotTest \
    --instrument lsst.obs.subaru.HyperSuprimeCam \
    -d "instrument='HSC' AND (band='g' or band='r' or band='i' or band='z' or band='y') AND skymap='hsc_rings_v1' AND tract=9813 AND patch=68" \
    --register-dataset-types ##--prune-replaced=purge --replace-run
```

Ensuite, nous construisons la pipetask pour exécuter le travail. Ici, je l'ai écrit dans un script bash.

(Then we construct the pipetask to run the job. Here I have written it in a bash script.)

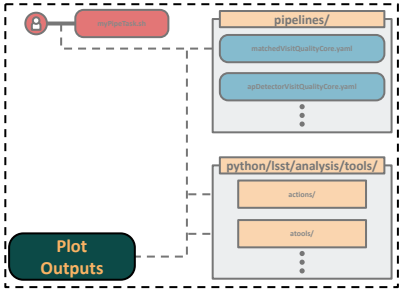
Ici, nous opérons sur les données HSC. Certains noms de paramètres pour HSC diffèrent de ceux de Rubin, nous pouvons donc ajouter la section `--instrument` pour garantir que le schéma est lu correctement.

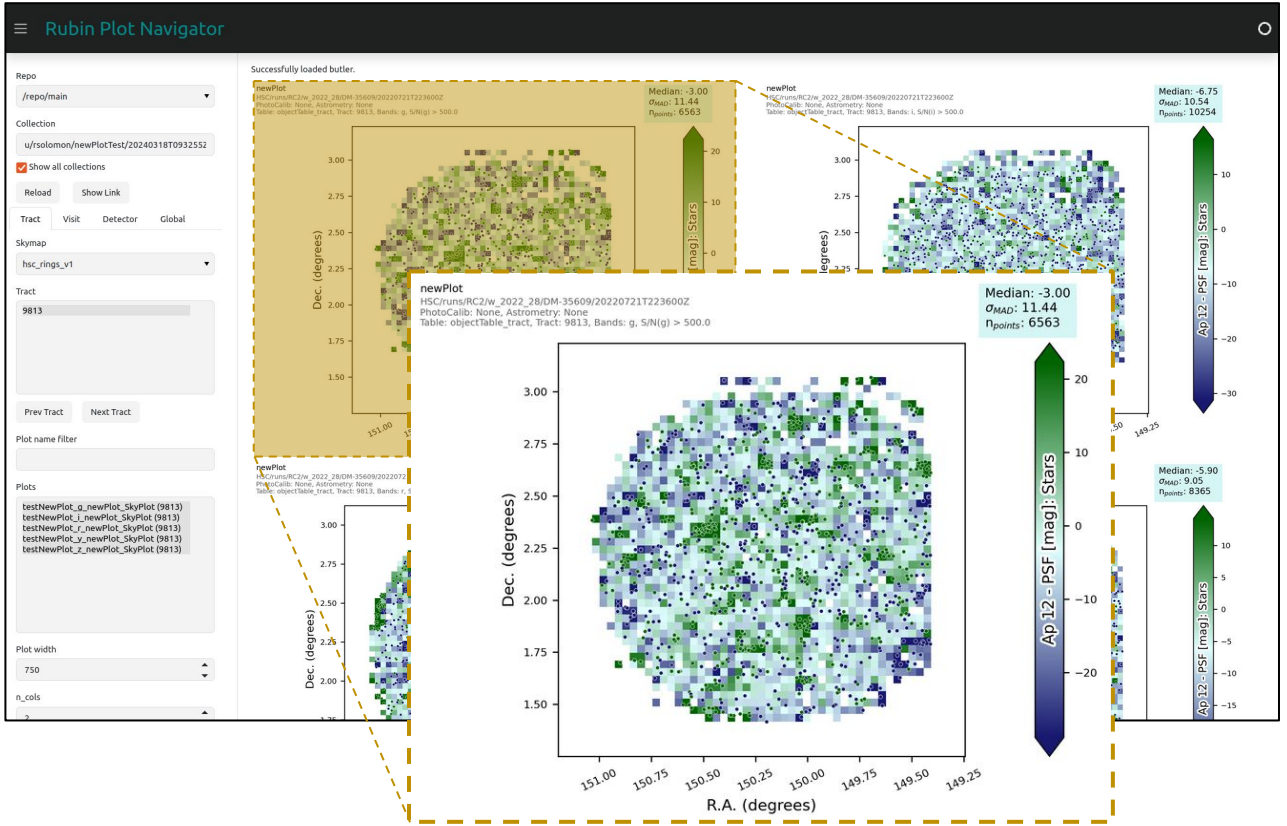
(Here we are running on HSC data. Some of the parameter names for HSC differ from Rubin so we can add the `--instrument` *section to ensure the schema is read properly.)*

N'oubliez pas d'inclure `--prune-replaced=purge` et `--replace-run` dans les exécutions ultérieures afin que les tracés soient mis à jour dans votre collection.

(Do not forget to include the `--prune-replaced=purge` *and* `--replace-run` *in subsequent runs so that the plots are updated in your collection.)*

Example: adding a new analysis





C'est plus ou moins ça.

(That is more or less it.)