LIneA

Laboratório Interinstitucional de e-Astronomia

Reunião BPG-LSST

08/02/2021

Julia Gschwend

Atuação no LSST em 2020

- In-kind contributions
  - Escrita e revisão das propostas
  - Photo-z para o LSST Project
  - DESC pipeline scientist (0.25FTE)
    - RAIL evaluation*

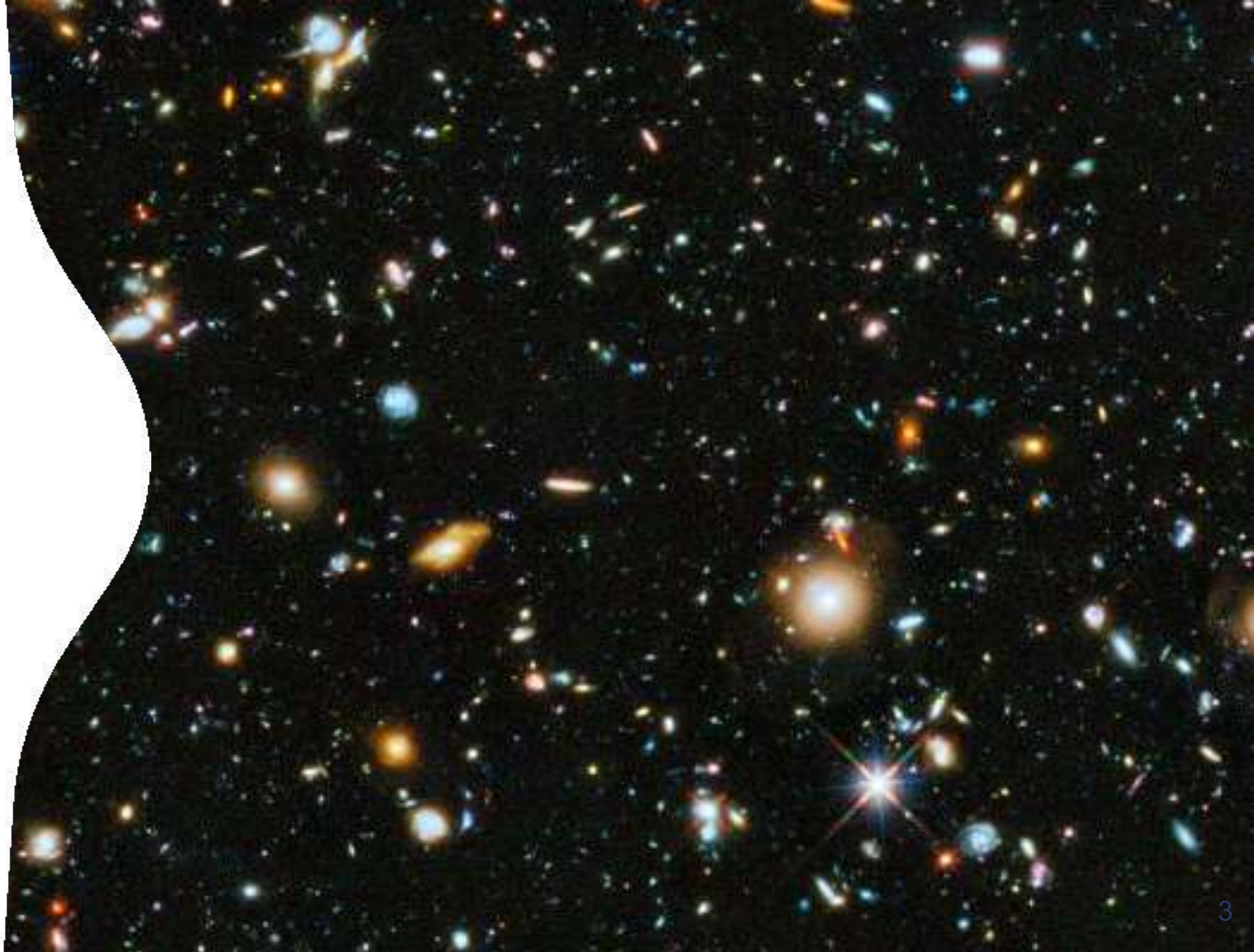
      LSSTDESC/RAIL

- Portal LSST @LIneA
  - Design e planejamento
  - Conversando sobre TI (apresentações)
  - Cursos (Jupyter, Python, SQL)

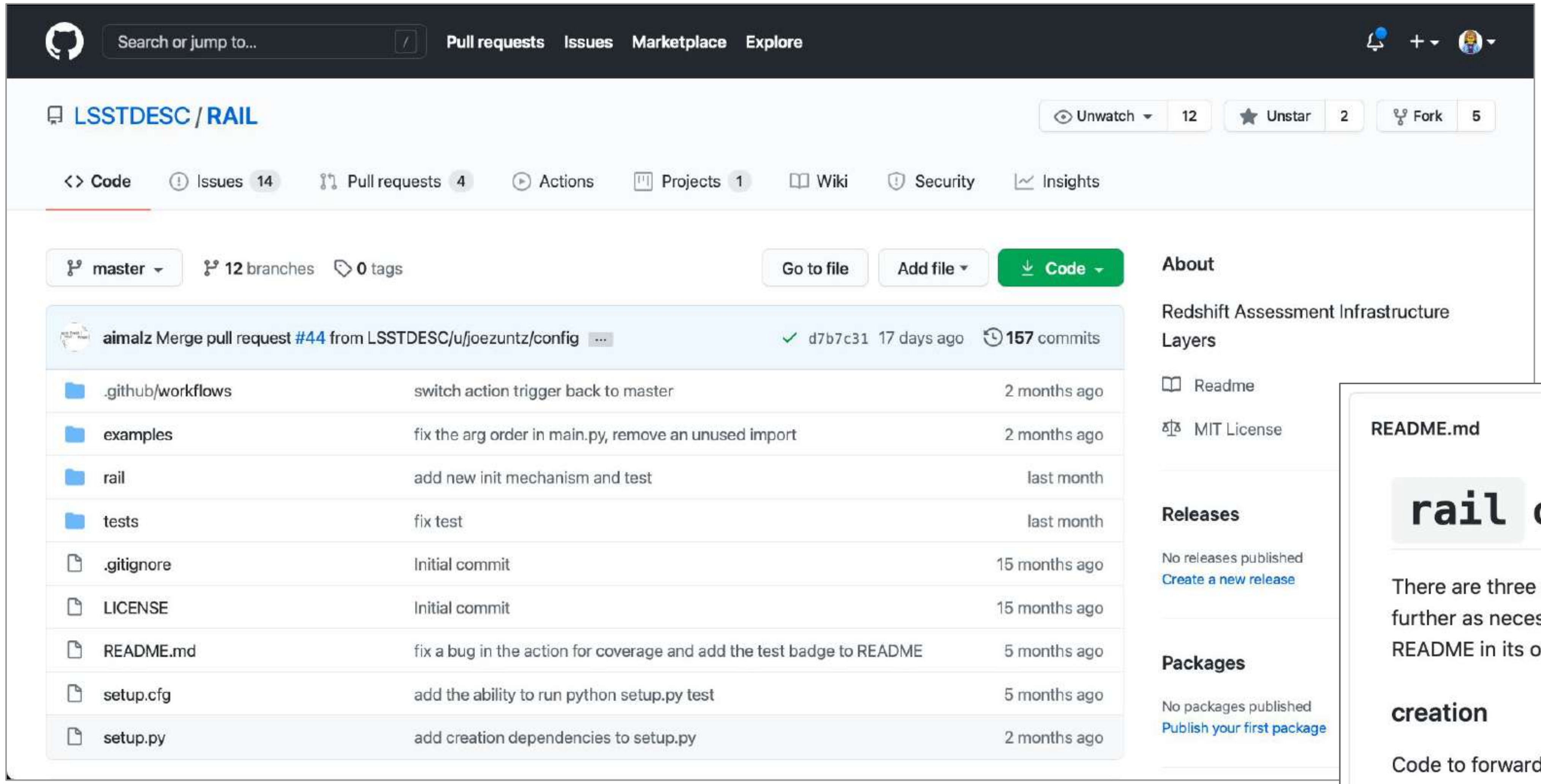
    linea-it/curso

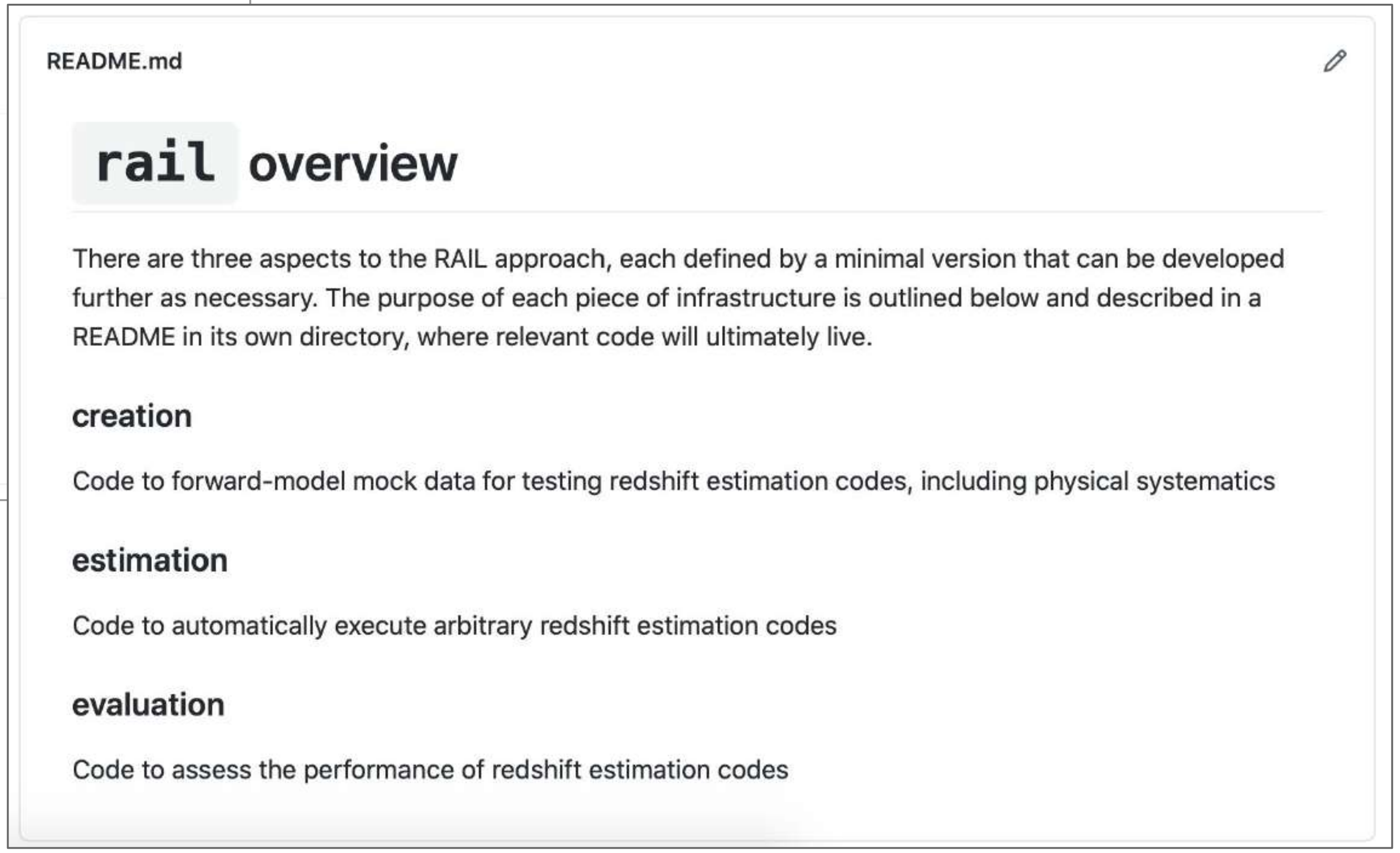
- Participação em eventos e telecons
  - DESC Meetings
  - DESC Sprint Week
  - PCW Workshop
  - EPO Workshop
  - DM Ops Bootcamp
  - DP0 virtual info sessions
  - Galaxy SC telecons
  - DESC PZ telecons
  - RAIL stand-up + co-working hours

RAIL evaluation



RAIL = Redshift Assessment Infrastructure Layers





README.md

rail overview

There are three aspects to the RAIL approach, each defined by a minimal version that can be developed further as necessary. The purpose of each piece of infrastructure is outlined below and described in a README in its own directory, where relevant code will ultimately live.

creation

Code to forward-model mock data for testing redshift estimation codes, including physical systematics

estimation

Code to automatically execute arbitrary redshift estimation codes

evaluation

Code to assess the performance of redshift estimation codes

RAIL modules

Creation:
forward model mock redshifts+photometry+posteriors

- Model p(z, data) from parameters or existing data set
- Draw redshifts, defining likelihoods p(data | z)
- Draw data, defining true posteriors p(z | data)
- p(z, data) for data need not be same for test and training sets!

Estimation:
infer photo-z posteriors from photometry

- Wrap any estimator for DESC computing environment
- Run wrapped estimators in parallel under controlled conditions

Evaluation:
assess performance metrics of photo-z posteriors

- Wrap any metric for automatic evaluation in experiments
- Add any probe-specific science metrics using in-RAIL information
- Baseline: DC1 paper metrics

DC1 photo-z validation metrics

- Cumulative Distribution Function

$$\mathrm{CDF}[f, q] \equiv \int_{-\infty}^{q} f(z)dz$$

- Probability Integral Transform

$$\mathrm{PIT} \equiv \mathrm{CDF}[\hat{p}, z_{true}]$$

- Conditional Density estimation (CDE) loss

$$L(f, \widehat{f}) \equiv \int\int (f(z|\mathbf{x}) - \widehat{f}(z|\mathbf{x}))^2 \mathrm{d}z\mathrm{d}P(\mathbf{x}).$$

true PDF (unknown)

observables (photometry)

$$\hat{L}(f, \widehat{f}) = \mathbb{E}_{\mathbf{X}}\left[\int \widehat{f}(z \mid \mathbf{X})^2 dz\right] - 2\mathbb{E}_{\mathbf{X},Z}\left[\widehat{f}(Z \mid \mathbf{X})\right] + K_f$$

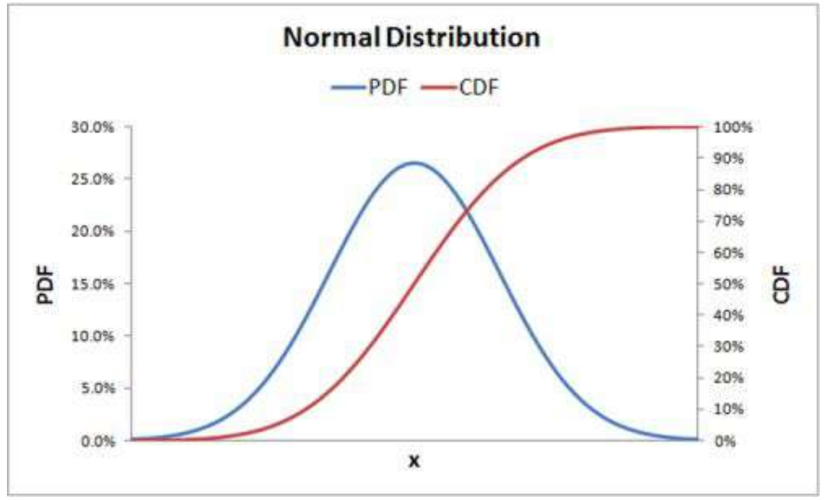
expectation value

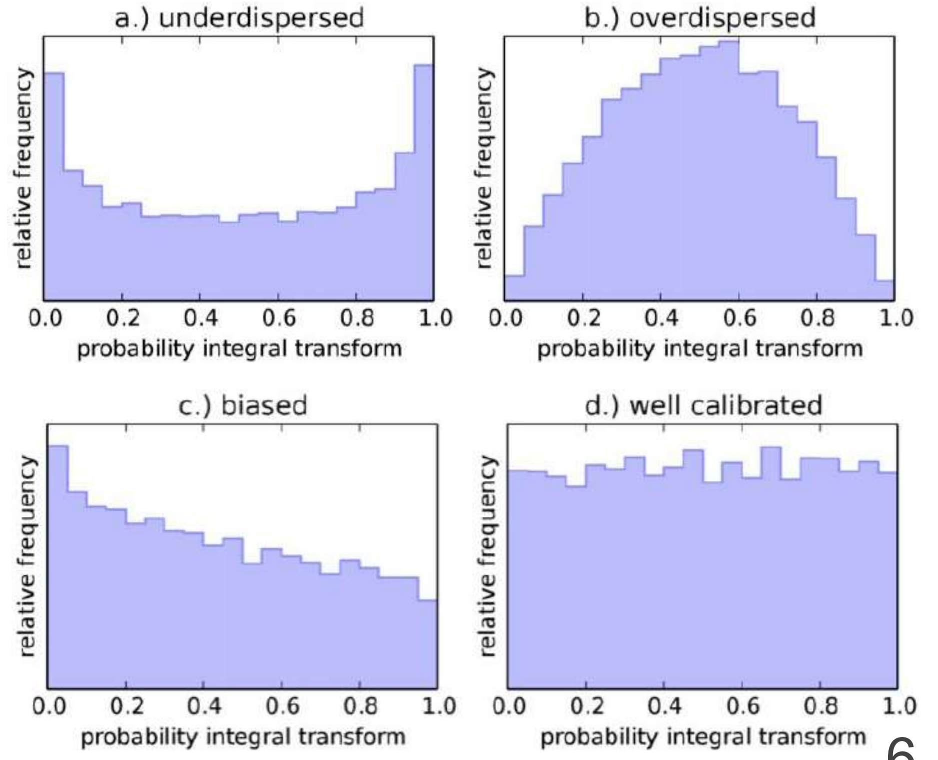
see Izbicki et al. 2017

MNRAS **000**, 000–000 (0000) Preprint 14 January 2020 Compiled using MNRAS LaTeX style file v3.0

Evaluation of probabilistic photometric redshift estimation approaches for LSST

S.J. Schmidt[1]*, A.I. Malz[2,3,4]†, J.Y.H. Soo[5,6], I.A. Almosallam[7,8], M. Brescia[9], S. Cavuoti[9,10], J. Cohen-Tanugi[11], A.J. Connolly[12], J. DeRose[13,14,15,16,17], P.E. Freeman[18], M.L. Graham[12], K.G. Iyer[19,20], M.J. Jarvis[21,22], J.B. Kalmbach[12], E. Kovacs[23], A.B. Lee[18], G. Longo[10], C.B. Morrison[12], J.A. Newman[24], E. Nourbakhsh[1], E. Nuss[11], T. Pospisil[18], H. Tranin[11], R.H. Wechsler[25,16,26], R. Zhou[15,24], R. Izbicki[27,28], and The LSST Dark Energy Science Collaboration
(Affiliations are listed at the end of the paper)





DC1 PIT histograms and QQ plots

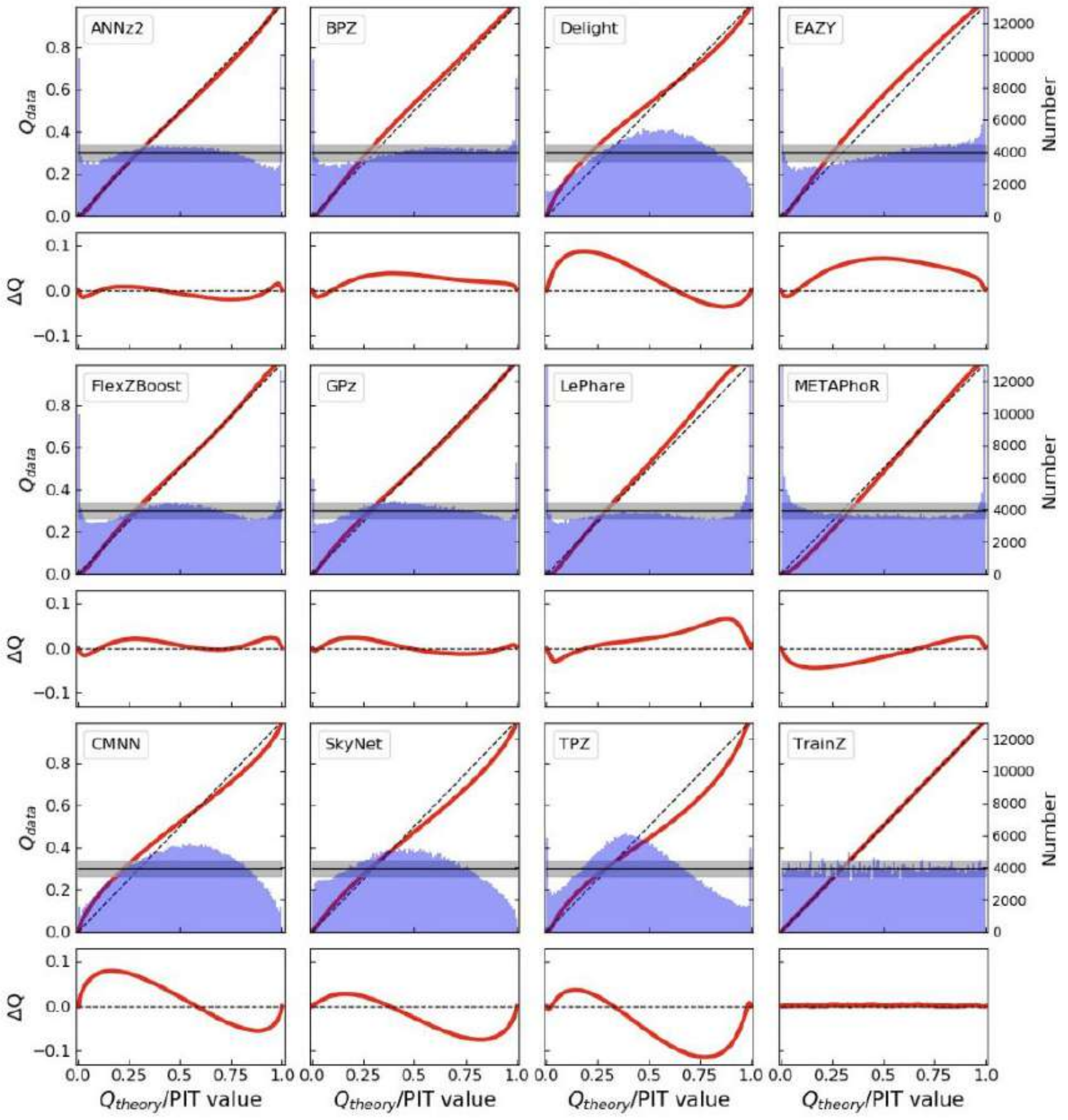


Figure 2. The QQ plot (red) and PIT histogram (blue) of the photo-z PDF codes (panels) along with the ideal QQ (black dashed diagonal) and ideal PIT (gray horizontal) curves, as well as a difference plot for the QQ difference from the ideal diagonal (lower inset). The gray shaded region indicates the 2σ range from a bootstrap resampling of the training set with a size of 30,000 galaxies using `trainZ`. The twelve codes exhibit varying degrees of four deviations from perfection: an overabundance of PIT values at the centre of the distribution indicate a catalogue of overly broad photo-z PDFs, an excess of PIT values at the extrema indicates a catalogue of overly narrow photo-z PDFs, catastrophic outliers manifest as overabundances at PIT values of 0 and 1, and asymmetry indicates systematic bias, a form of model misspecification. Values in excess of the 2σ shaded region show that for some codes these errors will be significant given expected training sample sizes.

Table 2. The catastrophic outlier rate as defined by extreme PIT values. We expect a value of 0.0002 for a proper Uniform distribution. An excess over this small value indicates true redshifts that fall outside the non-zero support of the $p(z)$.

Photo-z Code	fraction PIT$< 10^{-4}$ or >0.9999
`ANNz2`	0.0265
`BPZ`	0.0192
`Delight`	0.0006
`EAZY`	0.0154
`FlexZBoost`	0.0202
`GPz`	0.0058
`LePhare`	0.0486
`METAPhoR`	0.0229
`CMNN`	0.0034
`SkyNet`	0.0001
`TPZ`	0.0130
`trainZ`	0.0002

Table 3. CDE loss statistic of the individual photo-z PDFs for each code. A lower value of the CDE loss indicates more accurate individual photo-z PDFs, with `CMNN` and `FlexZBoost` performing best under this metric.

Photo-z Code	CDE Loss
`ANNz2`	−6.88
`BPZ`	−7.82
`Delight`	−8.33
`EAZY`	−7.07
`FlexZBoost`	−10.60
`GPz`	−9.93
`LePhare`	−1.66
`METAPhoR`	−6.28
`CMNN`	−10.43
`SkyNet`	−7.89
`TPZ`	−9.55
`trainZ`	−0.83

DC1 photo-z summary statistics

- Kolmogorov-Smirnov (KS) statistic

$$\mathrm{KS} \equiv \max_z \left(\left| \mathrm{CDF}[\hat{f}, z] - \mathrm{CDF}[\tilde{f}, z] \right| \right)$$

PIT dist. U(0,1)

- Cramer-von Mises (CvM) statistic

$$\mathrm{CvM}^2 \equiv \int_{-\infty}^{+\infty} \left(\mathrm{CDF}[\hat{f}, z] - \mathrm{CDF}[\tilde{f}, z] \right)^2 \mathrm{dCDF}[\tilde{f}, z]$$

- Anderson-Darling (AD) statistic

$$\mathrm{AD}^2 \equiv N_{tot} \int_{-\infty}^{+\infty} \frac{\left(\mathrm{CDF}[\hat{f}, z] - \mathrm{CDF}[\tilde{f}, z] \right)^2}{\mathrm{CDF}[\tilde{f}, z](1 - \mathrm{CDF}[\tilde{f}, z])} \mathrm{dCDF}[\tilde{f}, z]$$

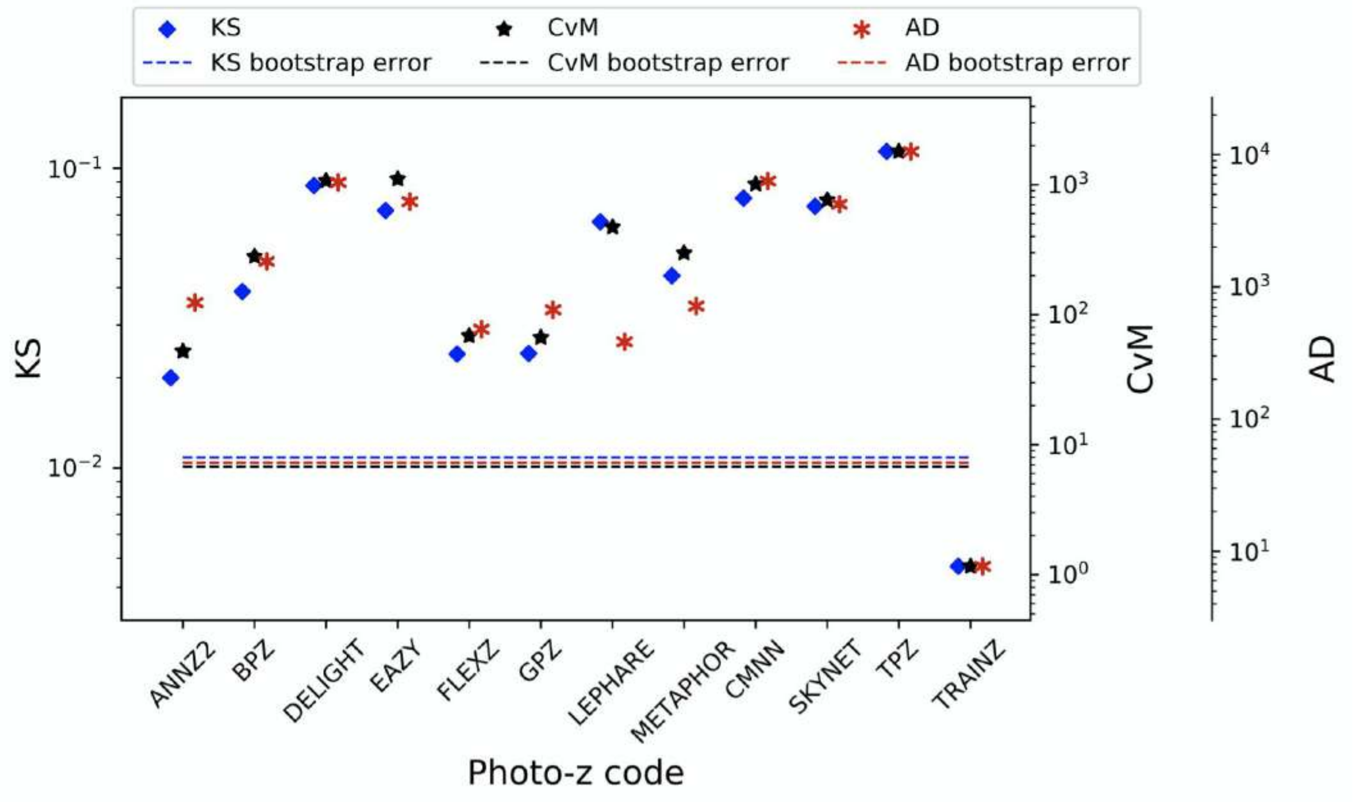
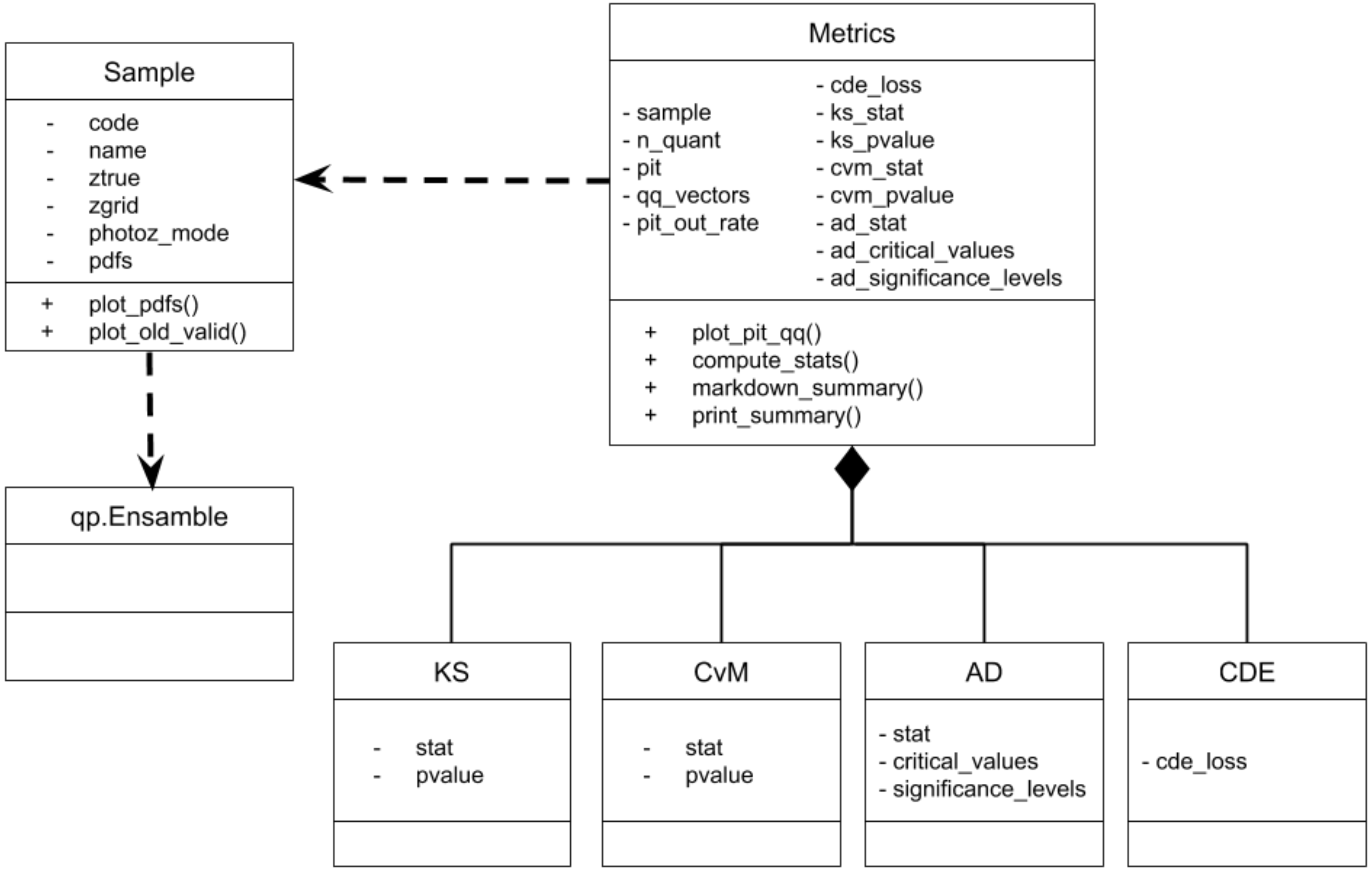


Figure 3. A visualization of the Kolmogorov-Smirnoff (KS, blue diamond), Cramer-von Mises (CvM, black star), and Anderson-Darling (AD, red asterisk) statistics for the PIT distributions. There is generally good agreement between these statistics, with differences corresponding to the codes with outstanding catastrophic outlier rates, a reflection of the differences in how each statistic weights the tails of the distribution. Horizontal lines indicate the level of uncertainty found by bootstrapping a training set sample of 30,000 galaxies using `trainZ`; none of the codes reach this conservative ideal floor in expected uncertainty.

RAIL evaluation design



Roadmap: (issue #4)

- ✓ Refactor the metrics code from DC1 Paper (Py2→ Py3, OO, etc).
- ✓ Create a superclass for the generic concept of metric.
- ✓ Create one independent class for each metric.
- ✓ Create demo notebook using the toy data available in RAIL/examples/.
- ✓ Create a script to compute all metrics at once via command line.
- ● Validate against DC1 results.
- ○ Write unit tests for the new classes.
- ○ Update repo's documentation.
- ○ Pull request.

○	To do
●	Doing
✓	Done

RAIL evaluation directory

Git branch: issue/4/evaluation-baseline

Evaluation module

Script to call all classes and compute all metrics at once via command line.

Ancillary module to make plots.

```
% ls RAIL/rail/evaluation/
README.md      __pycache__    evaluator.py  plots.py   __init__.py
demo.ipynb     metrics.py     sample.py
```

Jupyter Notebook with demonstration

Base Metrics superclass and individual metrics classes. It receives sample object as input, computes PIT and QQ vectors and makes basic plots to display the metrics.

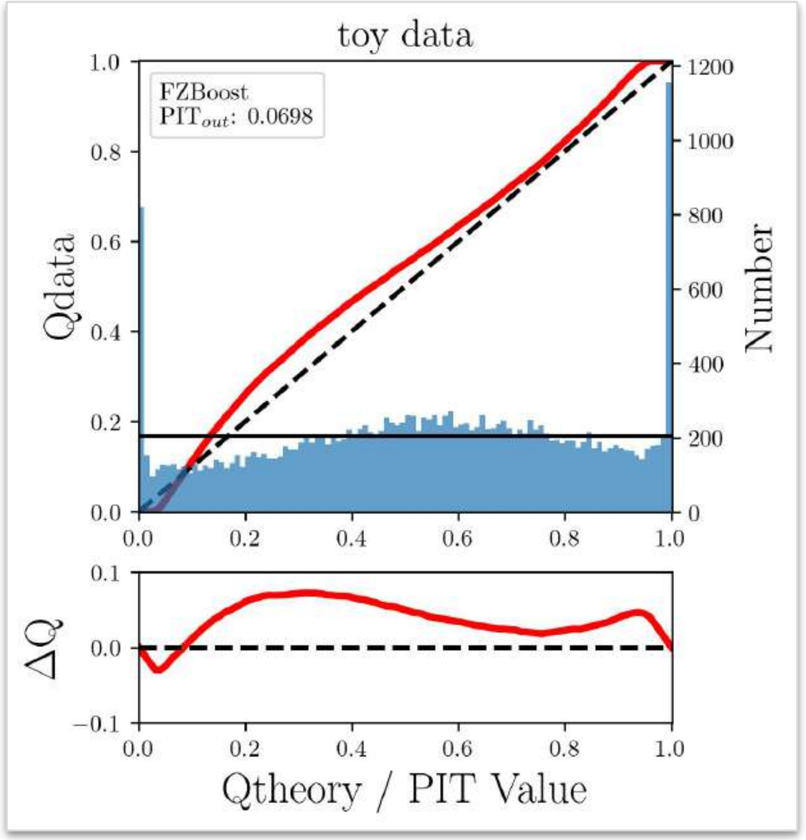
Class to handle input data. It combines the PDFs and the ztrue arrays. It inherits from qp.Ensamble() and makes basic plots for sample characterization.

evaluator.py (comand line mode)

```
% ls RAIL/examples/

base.yaml     configs    evaluator.py
main.py       results
```

```
(py37) julia@x86_64-apple-darwin13 examples % python evaluator.py
Found classifier FZBoost
Found classifier randomPZ
Found classifier simpleNN


        *** RAIL EVALUATION MODULE ***


Usage:
    python evaluator.py <code name> <PDFs file> <sample name> <z-spec file>

Example:
    python evaluator.py FZBoost ./results/FZBoost/test_FZBoost.hdf5 toy_data
../tests/data/test_dc2_validation_9816.hdf5


(py37) julia@x86_64-apple-darwin13 examples %
```

```
(py37) julia@x86_64-apple-darwin13 examples % python evaluator.py FZBoost ./results/
FZBoost/test_FZBoost.hdf5 toy_data ../tests/data/test_dc2_validation_9816.hdf5
Found classifier FZBoost
Found classifier randomPZ
Found classifier simpleNN


        *** RAIL EVALUATION MODULE ***


Photo-z results by: FZBoost
PDFs file: ./results/FZBoost/test_FZBoost.hdf5

Validation/test set: toy_data
z-true file: ../tests/data/test_dc2_validation_9816.hdf5


Reading data...

-----------------
Sample: toy data
Algorithm: FZBoost
-----------------
20449 PDFs
qp representation: interp
301 z bins edges from 0.0 to 3.0


Computing metrics...


   Metric     |   Value
--------------|----------
PIT out rate  |   0.0698
CDE loss      |  -6.7056
KS            |   0.0728
CvM           |  41.2865
AD            |  95.2651

Making plots...


Plots saved in:

plot_pit_qq_FZBoost_toy_data.png


Done!
```



demo.ipynb (in preparation)

Launcher | demo.ipynb | Code | Python 3

Demo: RAIL Evaluation

Contact: *Julia Gschwend* (julia@linea.gov.br), *Sam Schmidt, Alex Malz, Eric Charles*

The purpose of this notebook is to demonstrate the use of the metrics scripts to be used on the photo-z PDF catalogs produced by the PZ working group. The first implementation of the *evaluation* module is based on the refactoring of the algorithms used in Schmidt et al. 2020, available on Github repository PZDC1paper.

To run this code, you must install qp and have the notebook in the same directory as metrics.py. You must also install some run-of-the-mill Python packages: matplotlib, numpy, scipy, and skgof.

Contents

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import Markdown

#import warnings
#warnings.filterwarnings('ignore')

from sample import Sample
from metrics import *

%matplotlib inline
%load_ext autoreload
%autoreload 2
```

Sample

To compute the photo-z metrics of a given test sample, it is necessary to read the output of a photo-z code containing galaxies' photo-z PDFs. Let's use the toy data available in `tests/data/` (**test_dc2_training_9816.hdf5** and **test_dc2_validation_9816.hdf5**) and the configuration file available in `examples/configs/FZBoost.yaml` to generate a small samples of photo-z PDFs using the **FZBoost** algorithm available on RAIL's *estimation* module.

Run FZBoost

Go to dir `<your_path>/RAIL/examples/` and run the command:

```
python main.py configs/FZBoost.yaml
```

The photo-z output files (inputs for this notebook) will be writen at:

`<your_path>/RAIL/examples/results/FZBoost/test_FZBoost.hdf5`.

The new RAIL's version will produce output of the codes as qp files rather than the old format hdf5 files (Sam's message on Slack about RAIL's issue#33). TO DO: update the read() function of class Sample

[2]:
```
my_path = '/Users/julia/github/RAIL' # replace it by your path to RAIL's parent dir
pdfs_file = my_path + '/examples/results/FZBoost/test_FZBoost.hdf5'
ztrue_file = my_path + '/tests/data/test_dc2_validation_9816.hdf5'
```

Let's create a Sample object containing both the PDFs and true redshifts for each photo-z code.

[3]:
```
sample = Sample(pdfs_file, ztrue_file, code="FZBoost", name="toy data")
sample
```

[3]: `<sample.Sample at 0x7fac5bfce150>`

Launcher | demo.ipynb | Code | Python 3

[2]:
```
my_path = '/Users/julia/github/RAIL' # replace it by your path to RAIL's parent dir
pdfs_file = my_path + '/examples/results/FZBoost/test_FZBoost.hdf5'
ztrue_file = my_path + '/tests/data/test_dc2_validation_9816.hdf5'
```

Let's create a Sample object containing both the PDFs and true redshifts for each photo-z code.

[3]:
```
sample = Sample(pdfs_file, ztrue_file, code="FZBoost", name="toy data")
sample
```

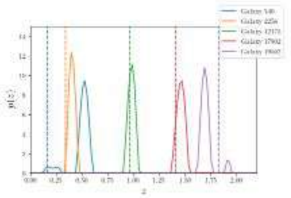
[3]: `<sample.Sample at 0x7fac5bfce150>`

[4]:
```
print(sample)
```

```
Sample: toy data
Algorithm: FZBoost

20449 PDFs
qp representation: interp
301 z bins edges from 0.0 to 3.0
```

PDFs of 5 galaxies for illustration. The function `plot_pdfs` calls a qp built-in plot function and returns the color codes of galaxies whose indexes are include in the list `gals`. The galaxies in the example were chosen arbitrarily to cover the sample's redshift space. The dashed lines represent their respective true redshifts.

[5]:
```
#gals = np.random.choice(len(ztrue), 5)
gals = [540, 2256, 12175, 17802, 19502]
colors = sample.plot_pdfs(gals)
```

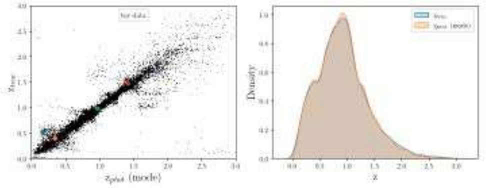


Validation plots

Traditional validation plots. The point colors (optional) follow the same color code as the PDFs above.

TO DO: update the plots below to look like Figure 4 from CHIPPR's paper (Malz & Hogg 2020).

[6]:
```
sample.plot_old_valid(gals=gals, colors=colors)
```



CDF-based metrics

The following metrics are computed based on the photo-z PDFs. Let's create a Metrics object to access the basic metrics (e.g., PIT outlier rate, defined below) and basic plots. It is the parent class of other particular metrics.

Instantiating a Metrics object can take a bit long, depending on the sample size.

[7]:
```
%%time
metrics = Metrics(sample)
```

```
CPU times: user 30.4 s, sys: 318 ms, total: 30.7 s
Wall time: 32.2 s
```

PIT

Launcher | demo.ipynb | Code | Python 3

PIT

The first metric we calculate is the Probability Integral Transform (PIT), which is the Cumulative Distribution Function (CDF)

$$\mathrm{CDF}(f, q) = \int_{-\infty}^{q} f(z)\, dz$$

evaluated at the galaxy's true redshift for every galaxy i in the catalog.

$$\mathrm{PIT}(p_i(z); z_i) = \int_{-\infty}^{z_i} p_i(z)\, dz$$

For instance, the PIT values for the 5 PDFs shown above are:

[8]:
```
metrics.pit[gals]
```

[8]: `array([0.02123863, 0.00962917, 0.38880144, 0.14467862, 0.93471813])`

PIT outlier rate

The PIT outlier rate is a global metric defined as the fraction of galaxies in the sample with extreme PIT values (PIT $< 10^{-4}$ or PIT > 0.9999). The lower and upper limits for considering a PIT as outlier are optional parameters set at the Metrics instantiation.

[9]:
```
print(f"PIT outlier rate of this sample: {metrics.pit_out_rate:.4f}")
```

```
PIT outlier rate of this sample: 0.0698
```

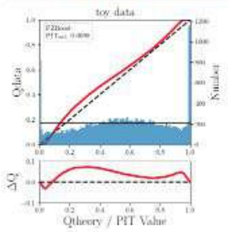
PIT-QQ plot

The histogram of PIT values is a useful tool for a qualitative assessment of PDFs quality. It shows whether the PDFs are:

- biased (tilted PIT histogram)
- under-dispersed (excess counts close to the boudaries 0 and 1)
- over-dispersed (lack of counts close the boudaries 0 and 1)
- well-calibrated (flat histogram)

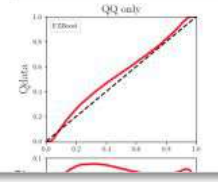
Following the standards in DC1 paper, the PIT histogram is accompanied by the quantile-quantile (QQ), which can be used to compare qualitatively the PIT distribution obtained with the PDFs agaist the ideal case (uniform distribution). The closer the QQ plot is to the diagonal, the better is the PDFs calibration.

[10]:
```
metrics.plot_pit_qq() #savefig=True)
```



By default, the function `plot_pit_qq` displays both PIT histogram and the QQ plots together. The title and label are retrieved from sample's attributed for sample name and the photo-z (if not informed as optional parameters). It is also possible to select one plot at a time.

[11]:
```
metrics.plot_pit_qq(show_pit=False, show_pit_out_rate=False, title="QQ only")
```



Launcher | demo.ipynb | Code | Python 3

$$\mathrm{AD}^2 \equiv N_{tot} \int_{-\infty}^{\infty} \frac{(\mathrm{CDF}[\hat{f}, z] - \mathrm{CDF}[\tilde{f}, z])^2}{\mathrm{CDF}[\hat{f}, z](1 - \mathrm{CDF}[\hat{f}, z])} d\mathrm{CDF}(\hat{f}, z)$$

where catastrophic outliers (default is PIT<0.01 or PIT>0.99) are removed for preserving numerical instability. Note that these values are not the same used to define the PIT outlier rate above (0.0001, and 0.9999).

The AD metric also inherits from scipy.stats library, returning three quantities: the AD stat value, a list of significance levels , and a list of critical values, and regarding those significance levels.

QUESTION: scipy.stats.anderson method supports the following distributions as reference: {'norm', 'expon', 'logistic', 'gumbel', 'gumbel_l', 'gumbel_r', 'extreme1'}. Apparently there is no way to compare the PIT distribution with a uniform distribution using this method. I'm using the default option ('norm'), but I think it's wrong.

[22]:
```
ad = AD(metrics._pit)
```

[23]:
```
ad.stat
```

[23]: `95.26506316495579`

[24]:
```
ad.significance_levels
```

[24]: `array([15. , 10. , 5. , 2.5, 1. ])`

[25]:
```
ad.critical_values
```

[25]: `array([0.576, 0.656, 0.787, 0.918, 1.092])`

TO DO: add sentence with interpretation to the numbers above

CDE Loss

In the absence of true photo-z posteriors, the metric used to evaluate individual PDFs is the **Conditional Density Estimate (CDE) Loss**, a metric analogue to the root-mean-squared-error:

$$L(f, \hat{f}) \equiv \int \int (f(z|x) - \hat{f}(z|x))^2 dz dP(x),$$

where $f(z|x)$ is the true photo-z PDF and $\hat{f}(z|x)$ is the estimated PDF in terms of the photometry x. Since $f(z|x)$ is unknown, we estimate the **CDE Loss** as described in Izbicki & Lee, 2017 (arXiv:1704.08095). :

$$\mathrm{CDE} = \mathbb{E}\left(\int \hat{f}(z|X)^2 dz \right) - 2\mathbb{E}_{X,Z}\left(\hat{f}(Z, X)\right) + K_f,$$

where the first term is the expectation value of photo-z posterior with respect to the marginal distribution of the covariates X, and the second term is the expectation value with respect to the joint distribution of observables X and the space Z of all possible redshifts (in practice, the centroids of the PDF bins), and the third term is a constant depending on the true conditional densities $f(z|x)$.

[26]:
```
cde_loss = CDE(sample).cde_loss
print(f"CDE loss of this sample: {cde_loss:.2f}")
```

```
CDE loss of this sample: -6.71
```

Summary

All metrics can be calculated at once and presented in a table by the main `metrics` object.

[27]:
```
metrics_table = metrics.markdown_summary()
Markdown(metrics_table)
```

[27]:

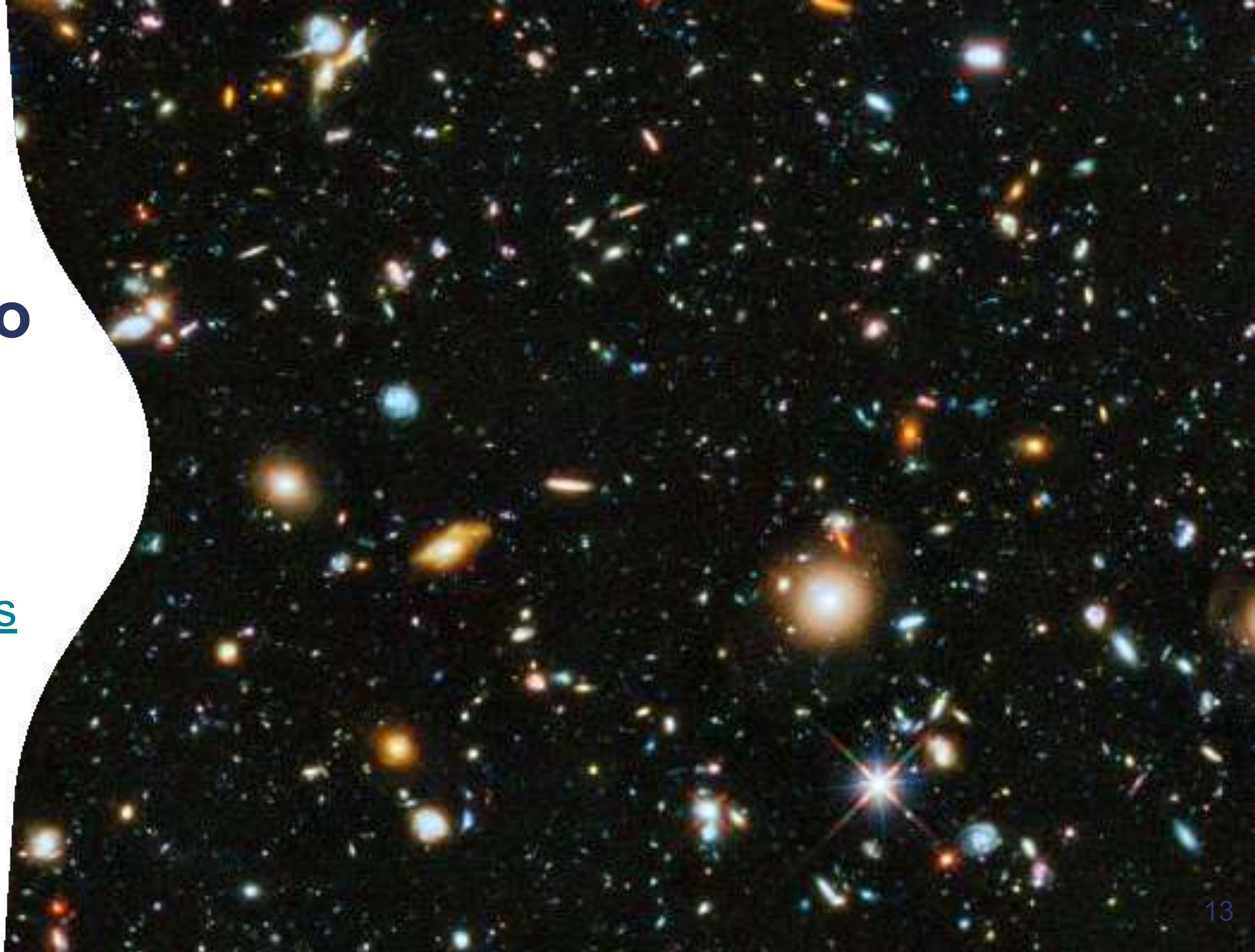
Metric	Value
PIT out rate	0.0698
CDE loss	-6.7056
KS	0.0728
CvM	41.2865

[ ]:

TO DO: Metrics plot (Figure 3 from DC1 Paper: Schmidt et al. 2020)

Conversando sobre TI

[pasta com apresentações]



Conversando sobre TI

Convidados

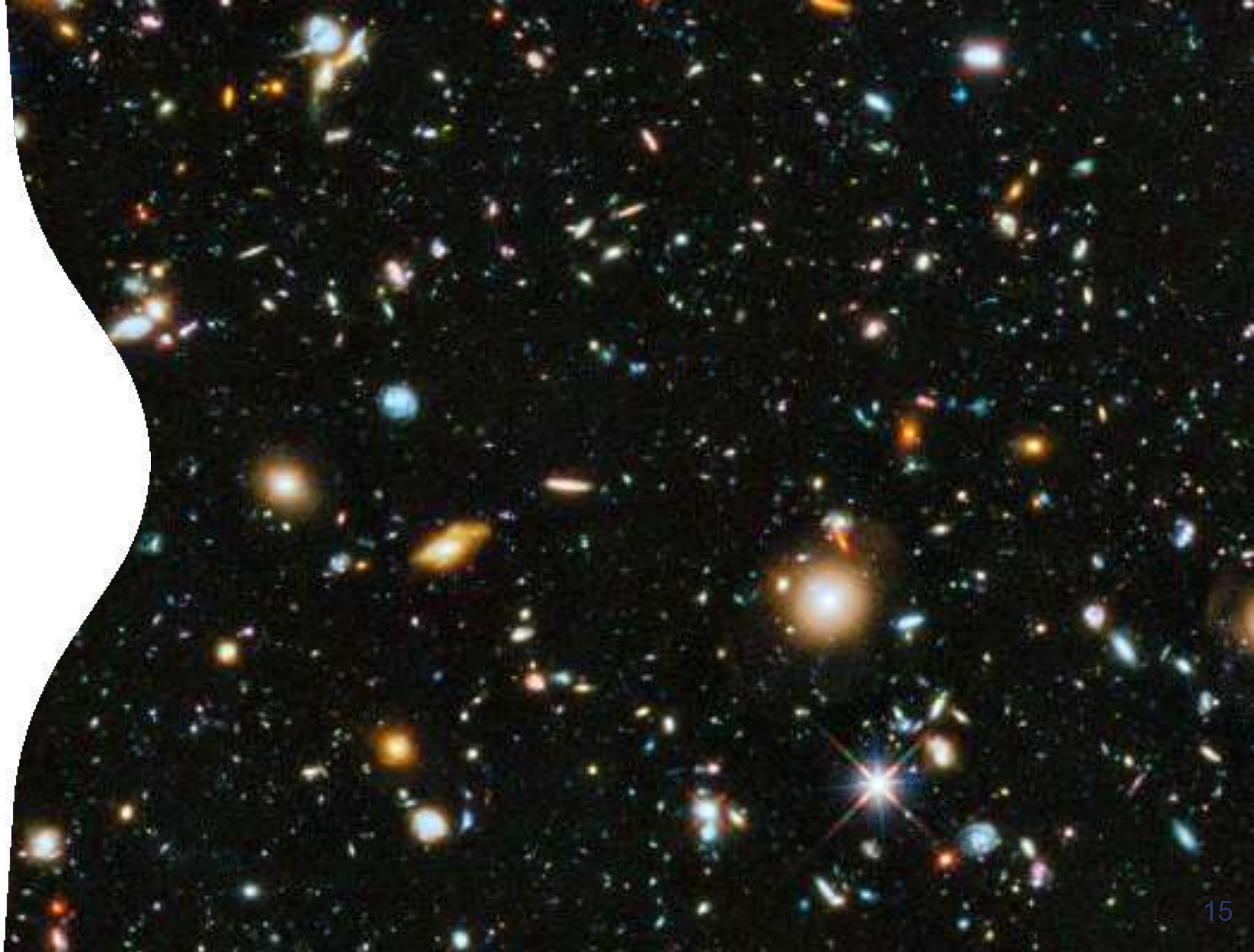
- DiRAC - Mario Juric et al.
- Patrícia Egeland
- Ricky Egeland
- Carla Osthoff
- Álvaro Coutinho
- Sérgio Novaes
- Eduardo Ogasawara
- Diego Carvalho
- Marta Mattoso
- Roberto Souto
- Daniel Oliveira
- Jacek Becla
- Fabio Hernandez

Temas, keywords

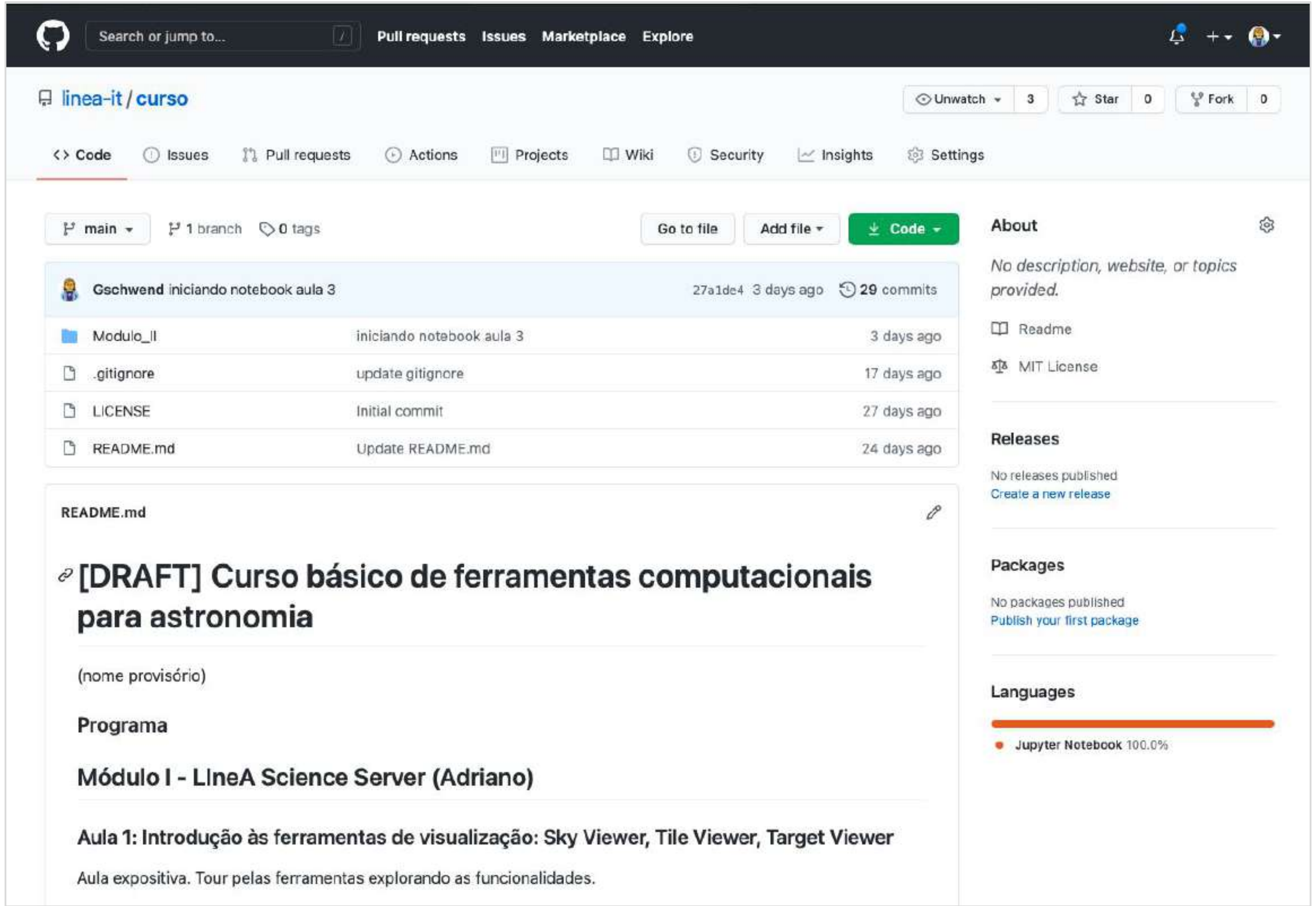
- Paralelização
- HPC
- Escalabilidade
- Spark
- AXS
- Parsl
- Jupyter
- Microservices
- Segurança
- Resenhas apresentações ADASS 2020

Curso

linea-it/curso



Curso

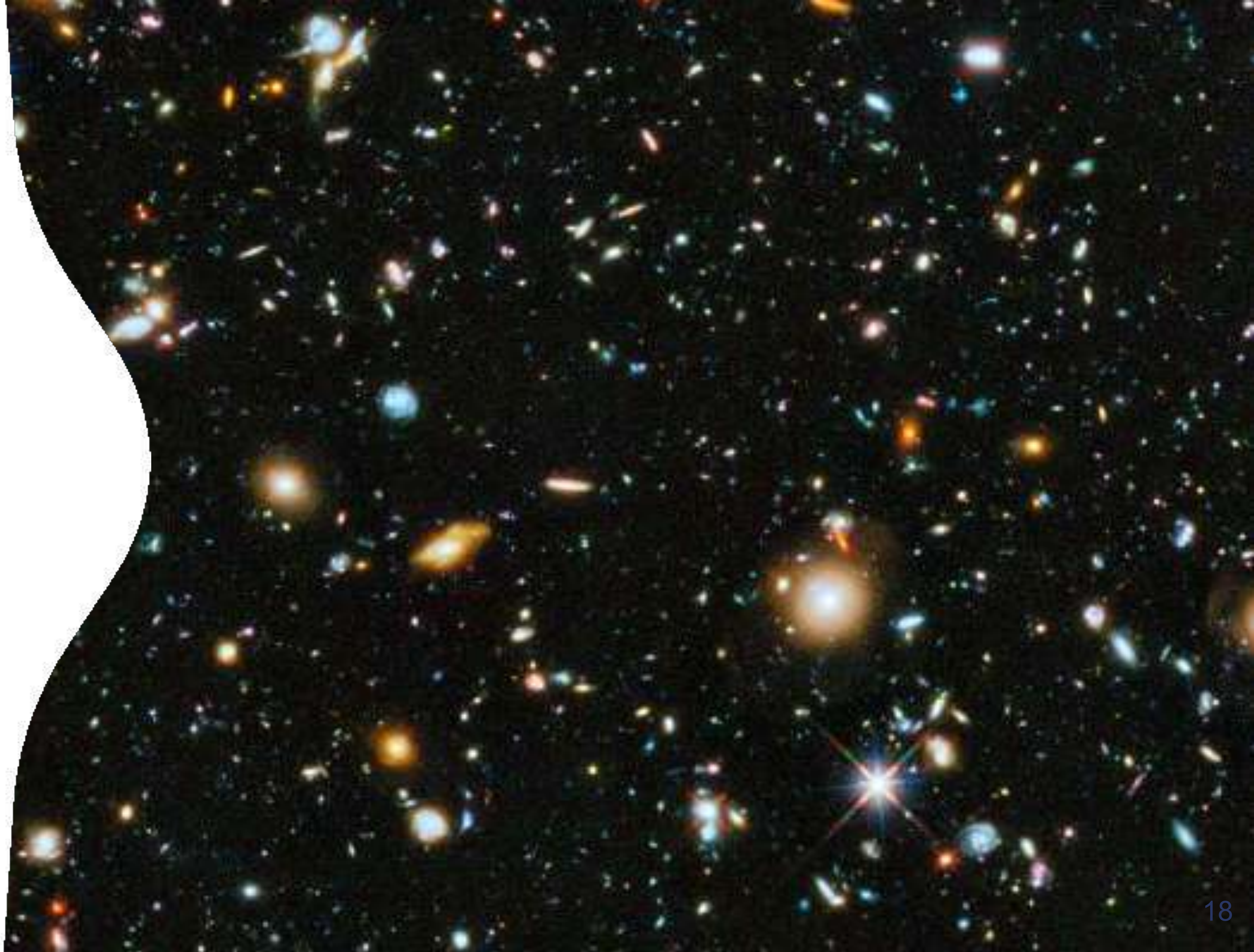

Search or jump to...
Pull requests
Issues
Marketplace
Explore
linea-it / curso
Unwatch 3
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
Gschwend iniciando notebook aula 3
27a1de4 3 days ago
29 commits
Modulo_II
iniciando notebook aula 3
3 days ago
.gitignore
update gitignore
17 days ago
LICENSE
Initial commit
27 days ago
README.md
Update README.md
24 days ago
About
No description, website, or topics provided.
Readme
MIT License
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Jupyter Notebook 100.0%
README.md
[DRAFT] Curso básico de ferramentas computacionais para astronomia
(nome provisório)
Programa
Módulo I - LIneA Science Server (Adriano)
Aula 1: Introdução às ferramentas de visualização: Sky Viewer, Tile Viewer, Target Viewer
Aula expositiva. Tour pelas ferramentas explorando as funcionalidades.


LIneA
Laboratório Interinstitucional de e-Astronomia
Curso básico de ferramentas computacionais para astronomia
Módulo II - Python & Jupyter
Julia Gschwend

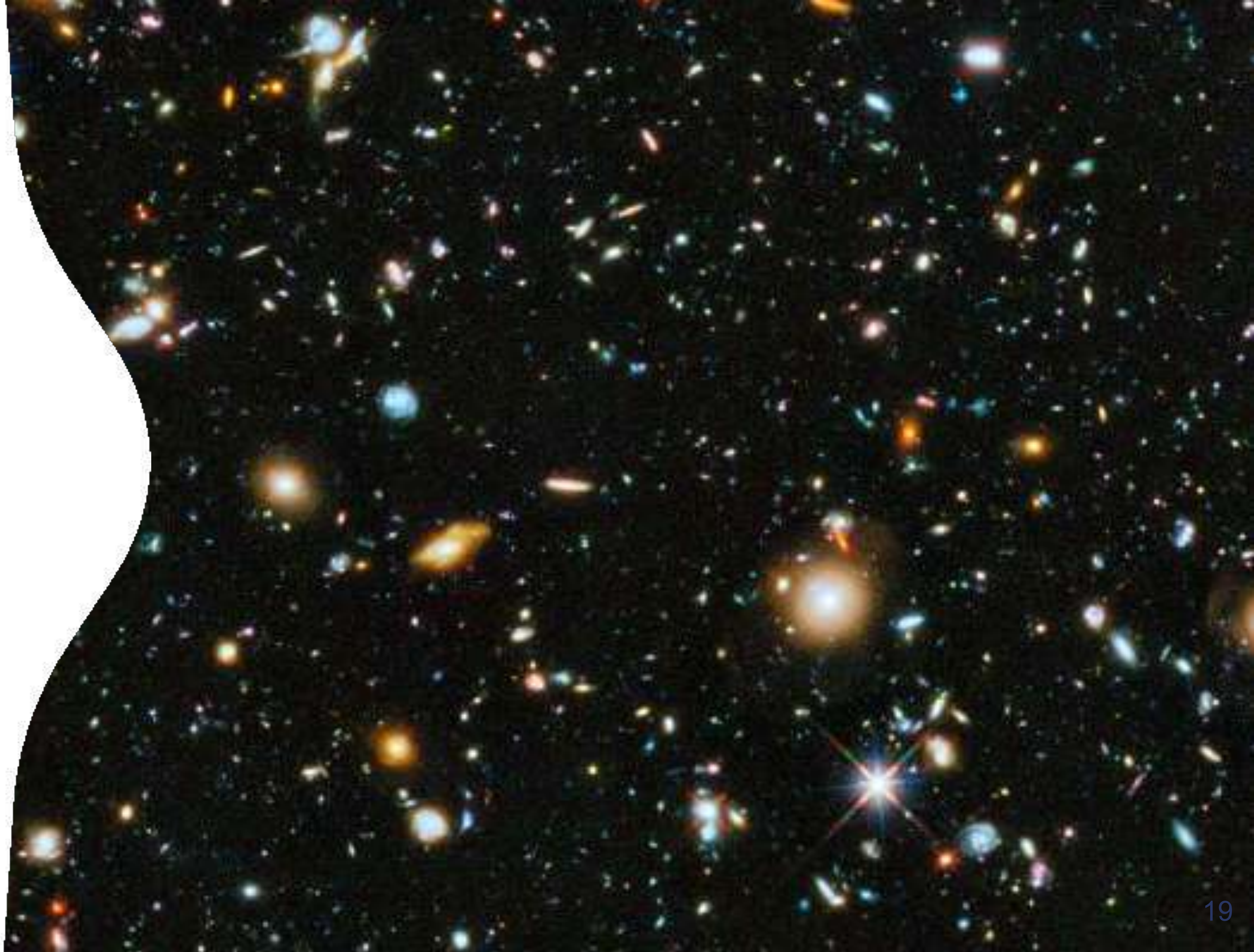
Próximos passos

- Continuar trabalho como Pipeline Scientist (RAIL)
- Concluir planejamento 2021-2024
- Iniciar desenvolvimento do portal LSST
- Concluir preparação do curso e iniciar as aulas
- Participar do DP0
- Análise DC2 com pipelines do DES (PZ valid, Gal. Properties, LF, MF)

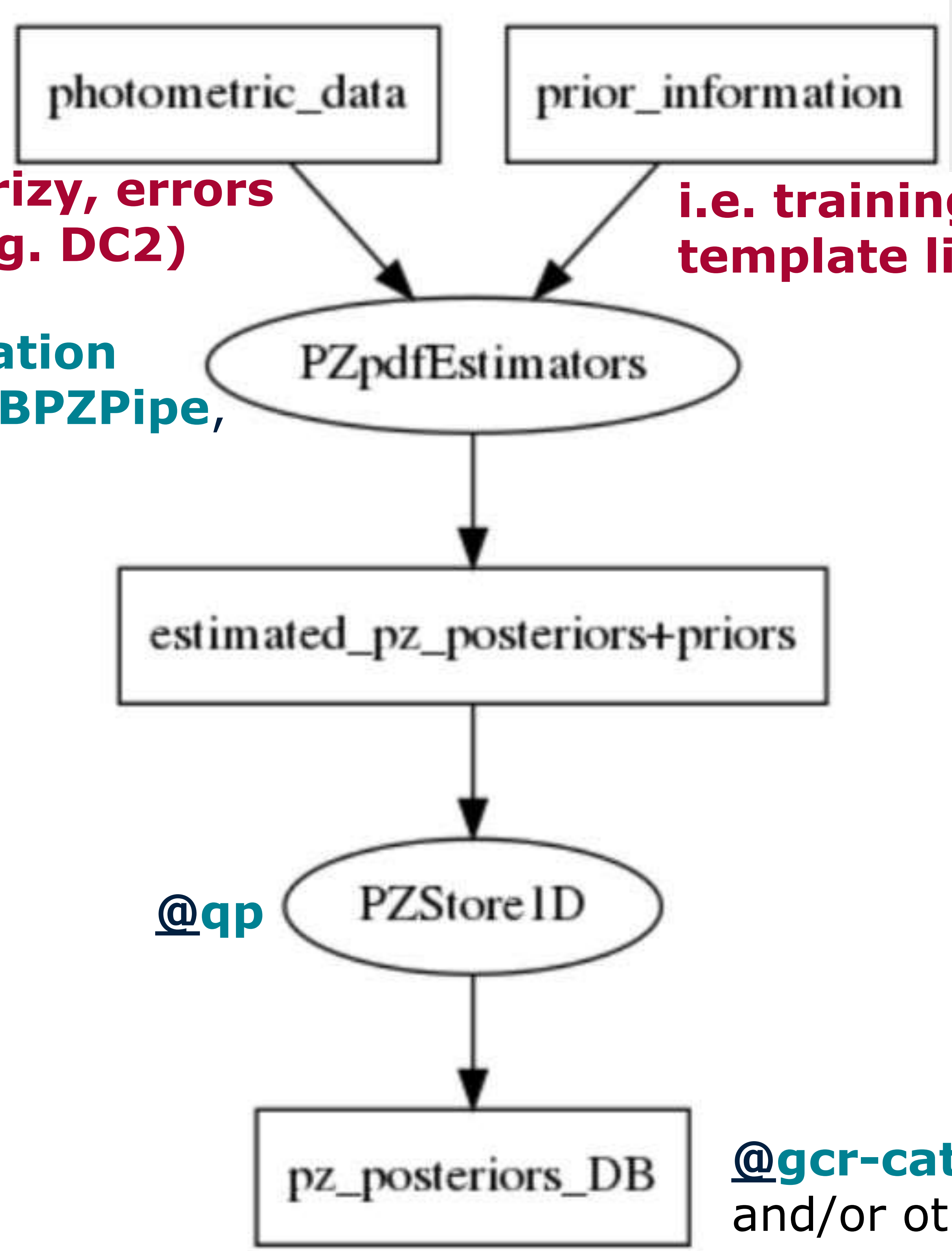
Obrigada.



Extra



PZEstimate



ugrizy, errors (e.g. DC2)

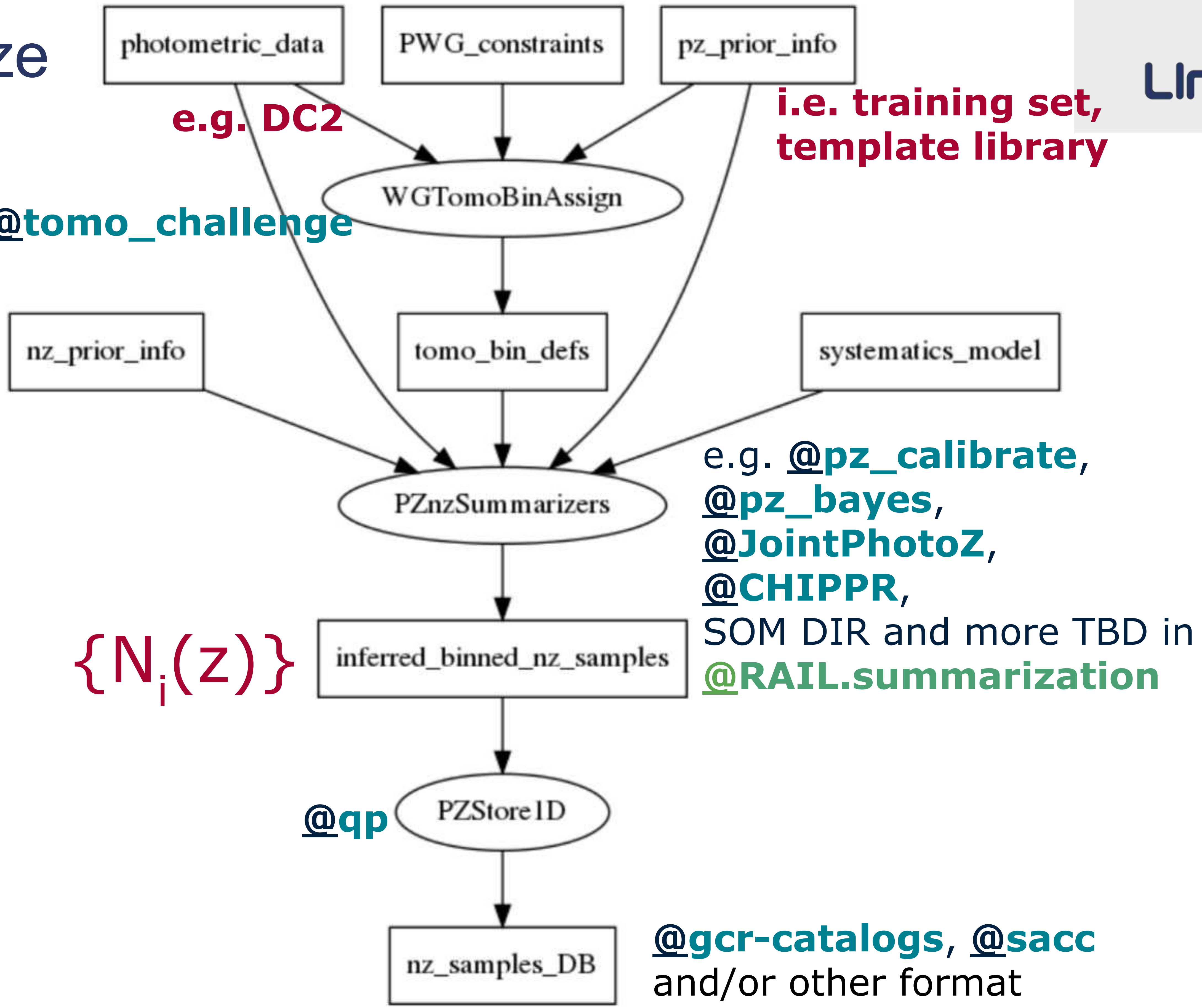
i.e. training set, template library

@RAIL.estimation
(supersedes **@BPZPipe**, **@FlexZPipe**)

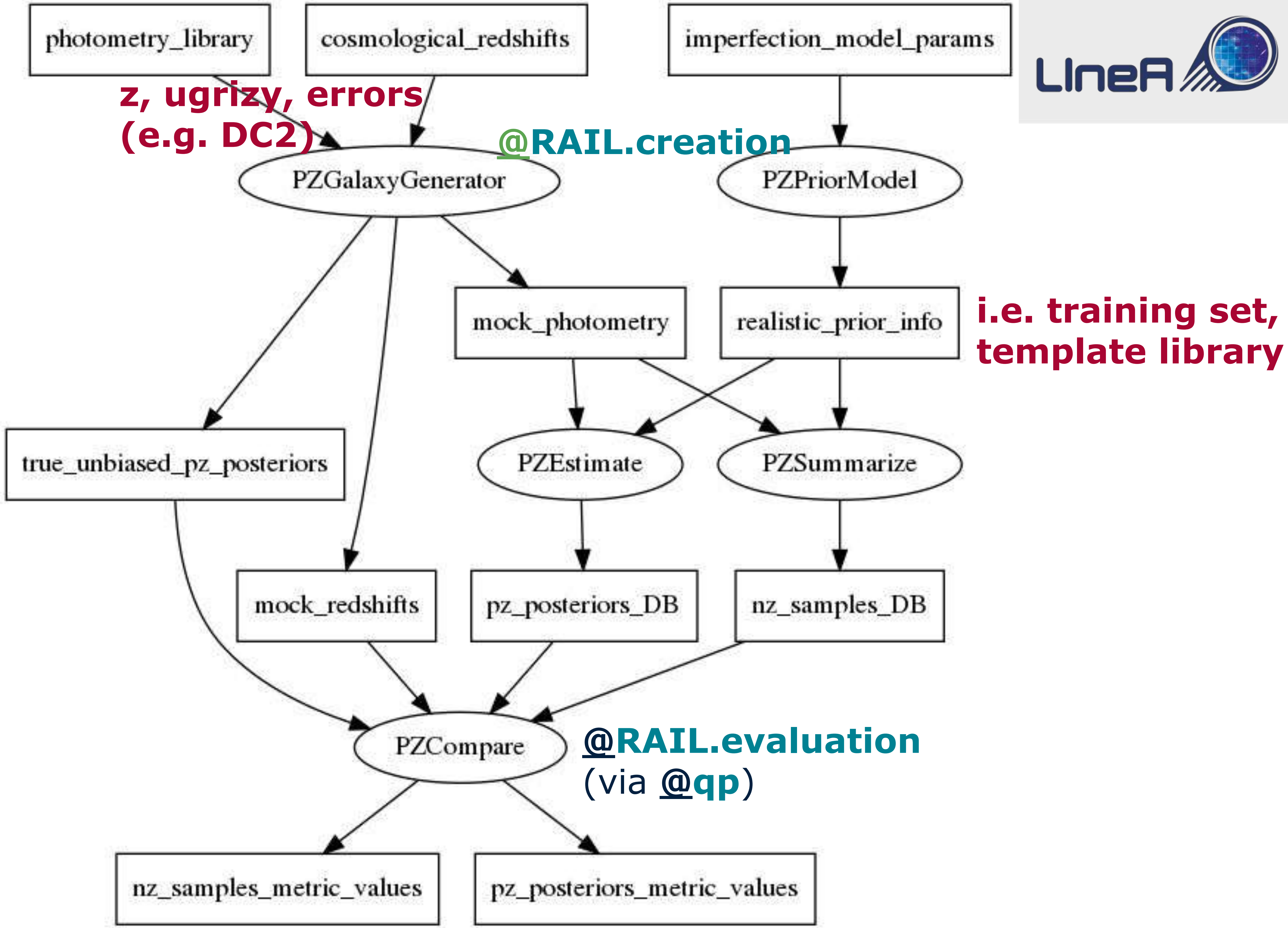
@qp

@gcr-catalogs, **@sacc**
and/or other format

PZSummarize



PZValidate


photometry_library
cosmological_redshifts
imperfection_model_params
z, ugrizy, errors
(e.g. DC2)
@RAIL.creation
PZGalaxyGenerator
PZPriorModel
mock_photometry
realistic_prior_info
i.e. training set,
template library
true_unbiased_pz_posteriors
PZEstimate
PZSummarize
mock_redshifts
pz_posteriors_DB
nz_samples_DB
PZCompare
@RAIL.evaluation
(via @qp)
nz_samples_metric_values
pz_posteriors_metric_values