RAIL: a platform for photometric redshift production and research Overview and Tutorial

LINCC Tech Talk, Nov 13 2024

Tianqing Zhang on behalf of RAIL team

The goal of the tutorial

- Understand a basic background of photometric redshift, and its relation to other topics in DESC
- Understand how RAIL is designed and its basic functionality
- Install RAIL or use RAIL in a pre-installed environment
- Get some photo-z results and make some plots!
- Basic understanding of the RAIL pipeline

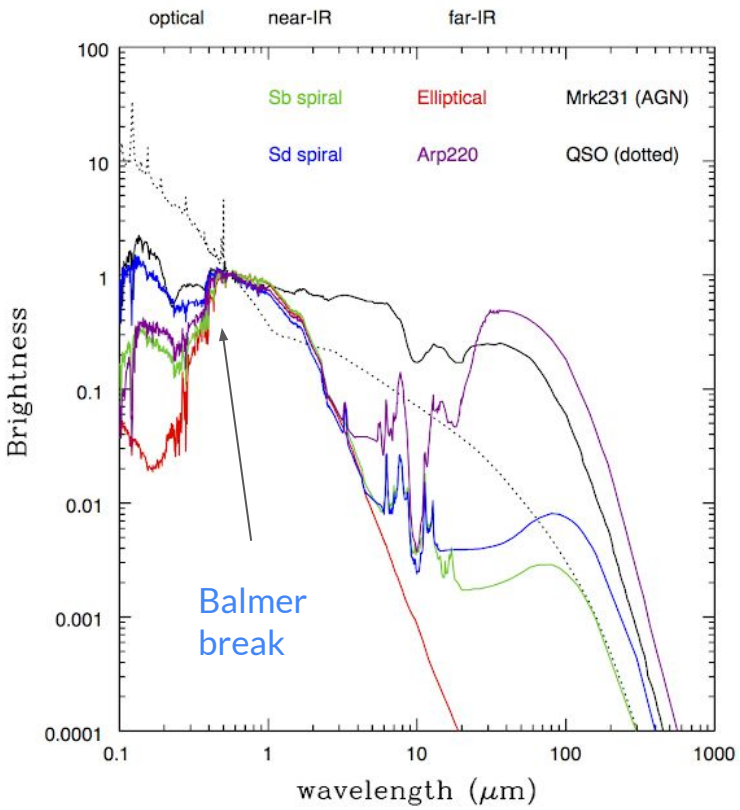
Photometric redshift

For LSST, we determine most galaxies' distance (redshift) by photometric information – photometric redshift (photo-z).

Photometric information: magnitude of a reference band, and colors (difference in magnitude between two bands)

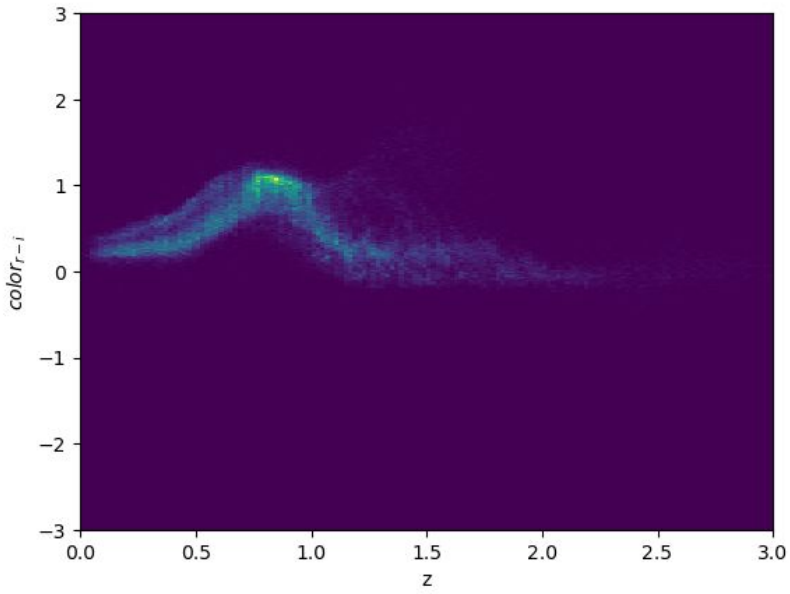
$\lambda_{obs} = \lambda_{rest} (1+z)$

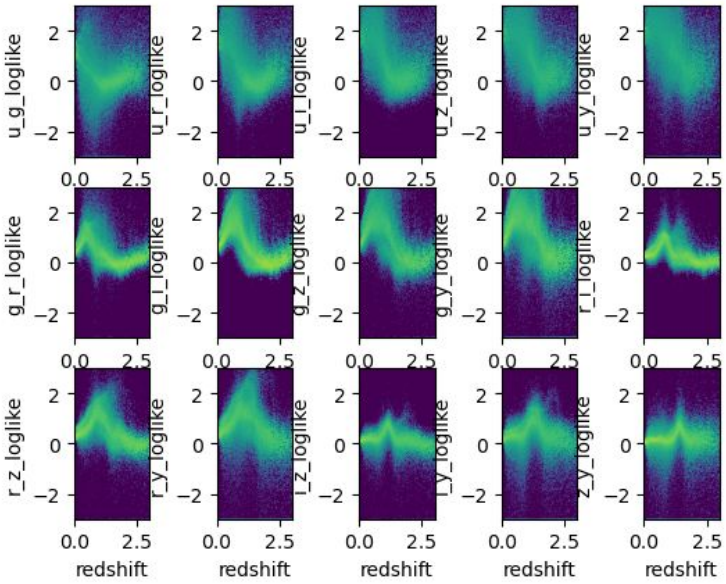
Colors in visible/IR bands are sensitive to redshift because the ~3600A Balmer break traverse through 400-1200 nm from z=0~2



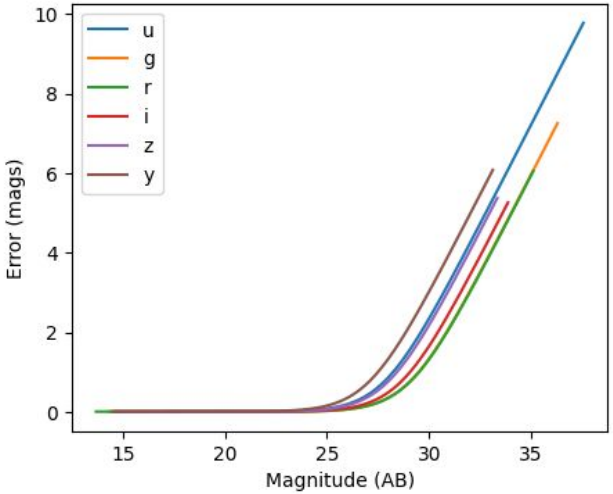
Credit: CANDLES, spectra of different objects

Redshift information from color





Photometric uncertainties



Photometric uncertainties blow up for dim objects

Quoting photo-z performance without stating what sample or limiting magnitude you are using is not useful

The challenges in photo-z studies

There are dozens of photo-z methods, categorized by whether they are data-driven or physics-driven (Salvato et al., Newman et al.). **They are written in different languages, expecting different format of input/output, different treatment of corner cases.**

This makes it very challenging to

1. Compare multiple methods on the same dataset (bias, variance, outliers, speed)
2. Efficiently produce photo-z products
3. Save configuration of multiple runs
4. Ensure safeguards are implemented

RAIL is designed to solve these issues.

TASK 0: A PHOTO-Z QUIZ

What are the normal input of a photo-z algorithm?

A. Ellipticity of galaxy
✅B. Magnitudes of a galaxy
✅C. Photometric noise of a galaxy
D. Location of a galaxy

What is the most important feature of a galaxy that makes photo-z works?

A. The emission line
B. The bulge and disk structure
✅C. The Balmer break
D. The star formation rate

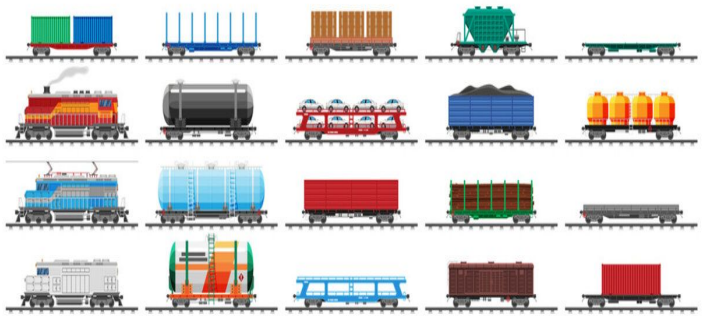
RAIL: Redshift Assessment Infrastructure Layers

RAIL is an **open-source software library** to provide a platform for photo-z production and study. It is originated and currently based in DESC, but is also envisioned to serve the a broader LSST community.

Developers of the RAIL package: DESC pipeline scientists, LINCC frameworks scientists and engineers, DESC in-kind contributors, etc.

The basic building blocks in RAIL are **stages**; the stages can be connected into **pipelines.**

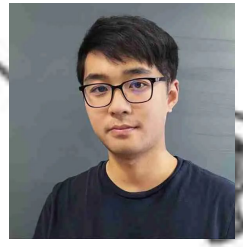
The RAIL's workflow is built upon **ceci**; the photo-z PDF format uses **qp**, a generic library for handling 1D PDFs. RAIL sub-packages are streamlined by **python-project-template**



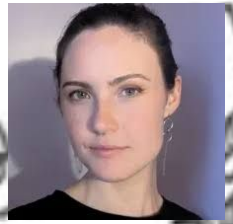
The RAIL Team

John-Franklin Crenshaw

Drew Oldag



Tianqing Zhang



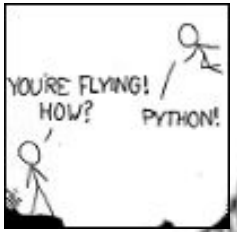
Olivia Lynn

Qianjun Hang

Luca Tortorelli

Eric Charles

Sam Schmidt



Alex Malz

Ziang Yan

Jan Luca van den Busch

And many others

TASK 1: Install RAIL / Use the NERSC environment

Installation how-to: https://rail-hub.readthedocs.io/en/latest/source/installation.html

Developer installation:

```
git clone https://github.com/LSSTDESC/rail.git

cd rail

conda env create -f environment.yml -n [env]  # or mamba env create, which is much faster

conda activate [env]

pip install -e .

rail clone-source --package-file rail_packages.yml

rail install --package-file rail_packages.yml --from-source
```

RAIL stages: getting photo-z's

Model
RAIL Stage
Data

	Name	Functionality	Examples
1.	Informer	Training/Preparing a photo-z model	Spec-z X photometry → Tree-Pz informer → Tree model
2.	Estimator	Estimate photo-z	Tree model, Photometry → Tree-Pz estimator → Tree photo-z PDFs
3.	Classifier	Assign tomographic binning	photo-z PDFs → Z-best cutter → Bin index
4.	Summarizer	Produce n(z)	photo-z PDFs → Hierarchical inference → n(z) samples

Available photo-z methods

Template fitting

BPZ (Bayesian Photometric Redshift) paper rail_bpz

LePhare paper rail_lephare

Machine Learning

CMNN paper rail_cmnn: Color-Matched Nearest Neighbor

FlexZBoost paper rail_flexzboost: xgboost

GPz paper rail_gpz: sparse Gaussian process

TPz paper rail_tpz: Tree PZ

PZFlow rail_pzflow: normalizing flow

SciKit Learn methods: KNN, Neural Net, RF, etc

Hybrid

Delight paper rail_delight

DNF

Image-based ML

DeepDISC rail_deepdisc

Inception rail_inception

Summarizers

Naive Stacking: Stack p(z)

NZ_DIR direct calibration n(z)

SOM: somoclu, SOMpz

YetAnotherWizz

LogGP

RAIL Stages: creating mock data

Model

RAIL Stage

Data

1. Creator — Create mock catalog: forward model e.g. normalizing flow → flow engine creator → Photometry

2. Degrader — Degrade mock catalog to observed catalog: True Photometry → Photometric error model → Observed Photometry

3. Evaluator — Evaluate performance: Spec-z X photo-z → PIT evaluator → PIT stats

Available engine and degraders

Engines

Pzflow engine

FSPS/DSPS

Noisifier

Photometric error model (add noise to photometry)

Line Confusion (change true redshift)

Reddening

Selector

Inverse redshift incompleteness

Spectroscopic selection (BOSS, DEEP2, VVDS, zCOSMOS)

Hybrid

Unrecognized blending degrader

RAIL Namespace stuff

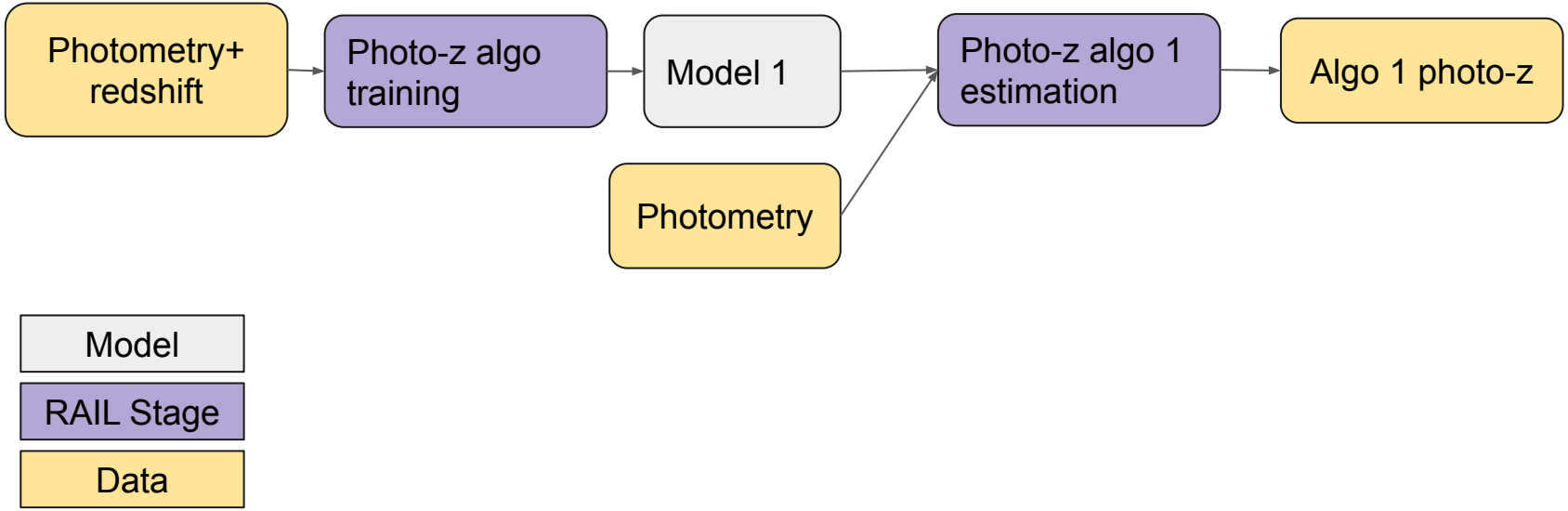
- RAIL has a lots of sub-packages, for the flexibility of installation
- But all RAIL packages share the same name space
  - A namespace is a system that has a unique name for each and every object in Python
- RAIL namespace is organized in the following way:
  - Informer, estimator and summarizers -> rail.estimation.algo
  - Engine -> rail.creation.engine
  - Degraders -> rail.creation.noisifier/selector
  - Evaluators -> rail.evaluation.metrics
  - Pipeline -> rail.pipeline
  - Basic classes -> rail.core
  - Utility (Tools) -> rail.utils (rail.tools)
    - Util are methods, tools are classes

TASK 2: GET SOME PHOTO-Z

Go to RAIL_estimation_demo.ipynb

RAIL pipeline

You can put multiple RAIL stages together to form a pipeline, as long as they can be represented by a Directed Acyclic Graph (meaning, no loop)



RAIL pipeline

The pipeline can be made in python and saved as yaml files

We are building pre-made pipelines in rail_pipelines

- estimate_all.py
- inform_all.py
- pz_all.py
- tomography.py

- apply_phot_errors.py
- blending.py
- spectroscopic_selection_pipeline.py

- evaluate_all.py

Pipeline YAML files

Ceci config

```yaml
config: config/all/inform_gpz_config.yml
inputs:
  input: ./data/dered_pdr3_wide_train_curated.pq
log_dir: ./log
modules: rail
output_dir: .
resume: false
site:
  max_threads: 2
  name: local
stages:
- classname: GPzInformer
  module_name: rail.estimation.algos.gpz
  name: inform_gpz
  nprocess: 1
```

To run a pipeline:
> ceci [ceci_config.yml]

RAIL config

```yaml
inform_gpz:
  aliases:
    model: model_inform_gpz
  bands:
  - HSCg_cmodel_dered
  - HSCr_cmodel_dered
  - HSCi_cmodel_dered
  - HSCz_cmodel_dered
  - HSCy_cmodel_dered
  config: null
  csl_binwidth: 0.1
  csl_method: normal
  err_bands:
  - g_cmodel_magerr
  - r_cmodel_magerr
  - i_cmodel_magerr
  - z_cmodel_magerr
  - y_cmodel_magerr
  gpz_method: VC
  hdf5_groupname: ''
  hetero_noise: true
  input: None
  learn_jointly: true
  log_errors: true
  mag_limits:
    HSCg_cmodel_dered: 27.88
    HSCi_cmodel_dered: 26.6
    HSCr_cmodel_dered: 27.05
    HSCy_cmodel_dered: 25.64
    HSCz_cmodel_dered: 26.6
  max_attempt: 100
  max_iter: 200
  model: model/estimator/model_gpz.pkl
  n_basis: 50
  name: inform_gpz
```

The RAIL Tutorial

DESC Sprint week @ SLAC Fall 2024

TQ, Eric on behalf of the RAIL team

The goal of the tutorial

- Understand a basic background of photometric redshift, and its relation to other topics in DESC
- Understand how RAIL is designed and its basic functionality
- Install RAIL or use RAIL in a pre-installed environment
- Get some photo-z results and make some plots!
- Basic understanding of the RAIL pipeline

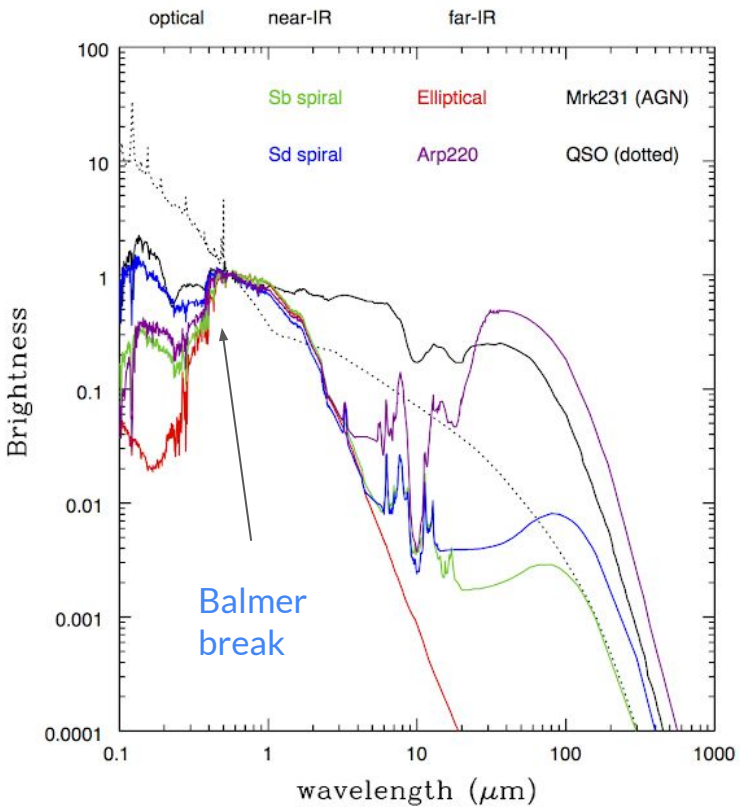
Photometric redshift

For LSST, we determine most galaxies' distance (redshift) by photometric information – photometric redshift (photo-z).

Photometric information: magnitude of a reference band, and colors (difference in magnitude between two bands)

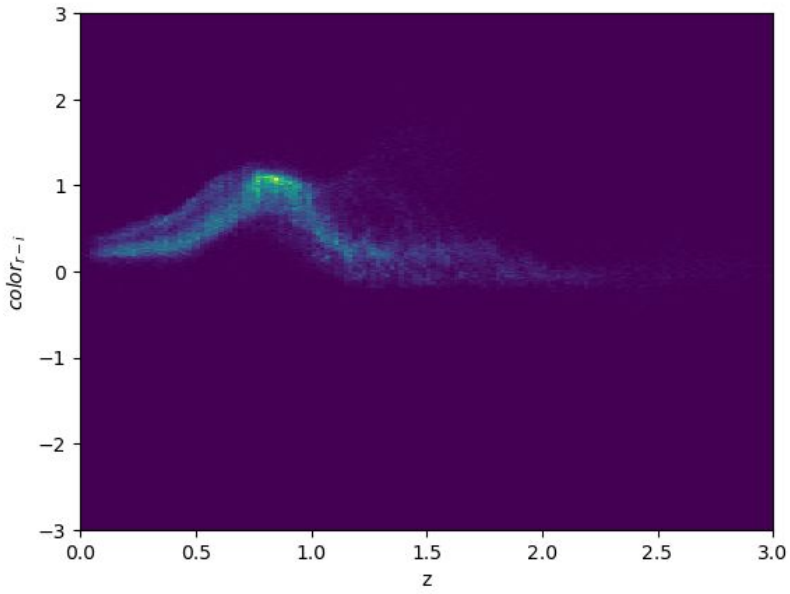
$\lambda_{obs} = \lambda_{rest} (1+z)$

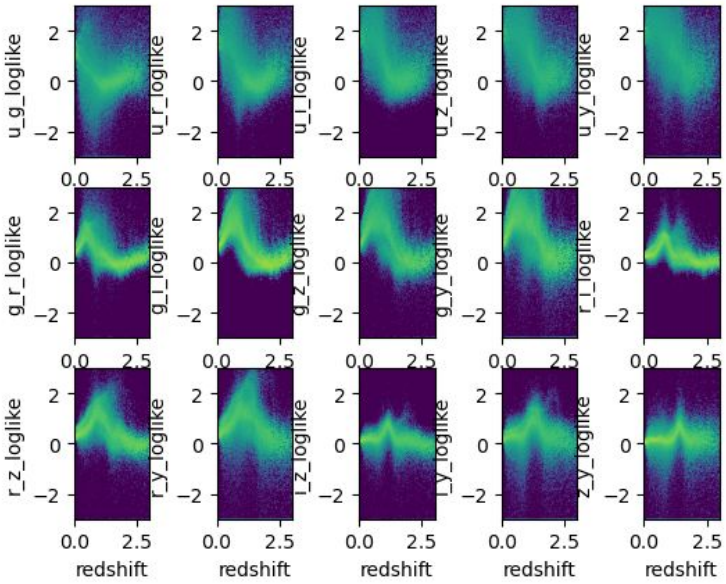
Colors in visible/IR bands are sensitive to redshift because the ~3600A Balmer break traverse through 400-1200 nm from z=0~2



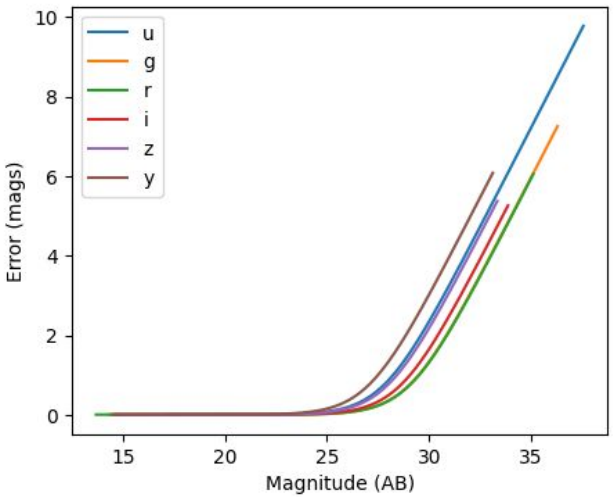
Credit: CANDLES, spectra of different objects

Redshift information from color





Photometric uncertainties



Photometric uncertainties blow up for dim objects

Quoting photo-z performance without stating what sample or limiting magnitude you are using is not useful

*Slide borrowed from Eric Charles

The challenges in photo-z studies

There are dozens of photo-z methods, categorized by whether they are data-driven or physics-driven (Salvato et al., Newman et al.). **They are written in different languages, expecting different format of input/output, different treatment of corner cases.**

This makes it very challenging to

1. Compare multiple methods on the same dataset (bias, variance, outliers, speed)
2. Efficiently produce photo-z products
3. Save configuration of multiple runs
4. Ensure safeguards are implemented

RAIL is designed to solve these issues.

TASK 0: A PHOTO-Z QUIZ

What are the normal input of a photo-z algorithm?

A. Ellipticity of galaxy
✅B. Magnitudes of a galaxy
✅C. Photometric noise of a galaxy
D. Location of a galaxy

What is the most important feature of a galaxy that makes photo-z works?

A. The emission line
B. The bulge and disk structure
✅C. The Balmer break
D. The star formation rate

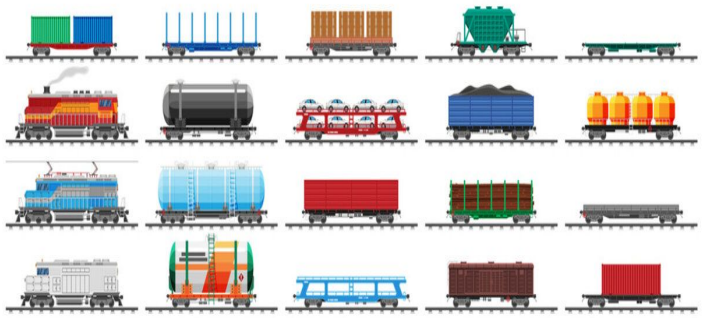
RAIL: Redshift Assessment Infrastructure Layers

RAIL is an **open-source software library** to provide a platform for photo-z production and study. It is originated and currently based in DESC, but is also envisioned to serve the a broader LSST community.

Developers of the RAIL package: DESC pipeline scientists, LINCC frameworks scientists and engineers, DESC in-kind contributors, etc.

The basic building blocks in RAIL are **stages**; the stages can be connected into **pipelines.**

The RAIL's workflow is built upon **ceci**; the photo-z PDF format uses **qp**, a generic library for handling 1D PDFs. RAIL sub-packages are streamlined by **python-project-template**



TASK 1: Install RAIL / Use the NERSC environment

Installation how-to: https://rail-hub.readthedocs.io/en/latest/source/installation.html

Developer installation:

```
git clone https://github.com/LSSTDESC/rail.git

cd rail

conda env create -f environment.yml -n [env]  # or mamba env create, which is much faster

conda activate [env]

pip install -e .

rail clone-source --package-file rail_packages.yml

rail install --package-file rail_packages.yml --from-source
```

TASK 1: Install RAIL / Use the NERSC environment

Log in NERSC:

```
python /global/common/software/lsst/common/miniconda/start-kernel-cli.py
desc-python-bleed
```

Or initialize a RAIL notebook in `desc-python-bleed`

Pull the example from rail-hub:

```
git clone git@github.com:LSSTDESC/rail.git
```

RAIL stages: getting photo-z's

Model
RAIL Stage
Data

	Name	Functionality	Examples
1.	Informer	Training/Preparing a photo-z model	Spec-z X photometry → Tree-Pz informer → Tree model
2.	Estimator	Estimate photo-z	Tree model, Photometry → Tree-Pz estimator → Tree photo-z PDFs
3.	Classifier	Assign tomographic binning	photo-z PDFs → Z-best cutter → Bin index
4.	Summarizer	Produce n(z)	photo-z PDFs → Hierarchical inference → n(z) samples

Available photo-z methods

Template fitting

BPZ (Bayesian Photometric Redshift) paper rail_bpz

LePhare paper rail_lephare

Machine Learning

CMNN paper rail_cmnn: Color-Matched Nearest Neighbor

FlexZBoost paper rail_flexzboost: xgboost

GPz paper rail_gpz: sparse Gaussian process

TPz paper rail_tpz: Tree PZ

PZFlow rail_pzflow: normalizing flow

SciKit Learn methods: KNN, Neural Net, RF, etc

Hybrid

Delight paper rail_delight

DNF

Image-based ML

DeepDISC rail_deepdisc

Inception rail_inception

Summarizers

Naive Stacking: Stack p(z)

NZ_DIR direct calibration n(z)

SOM: somoclu, SOMpz

YetAnotherWizz

LogGP

RAIL Stages: creating mock data

Model

RAIL Stage

Data

1. Creator — Create mock catalog: forward model e.g. normalizing flow → flow engine creator → Photometry

2. Degrader — Degrade mock catalog to observed catalog: True Photometry → Photometric error model → Observed Photometry

3. Evaluator — Evaluate performance: Spec-z X photo-z → PIT evaluator → PIT stats

Available engine and degraders

Engines

Pzflow engine

FSPS/DSPS

Noisifier

Photometric error model (add noise to photometry)

Line Confusion (change true redshift)

Reddening

Selector

Inverse redshift incompleteness

Spectroscopic selection (BOSS, DEEP2, VVDS, zCOSMOS)

Hybrid

Unrecognized blending degrader

RAIL Namespace stuff

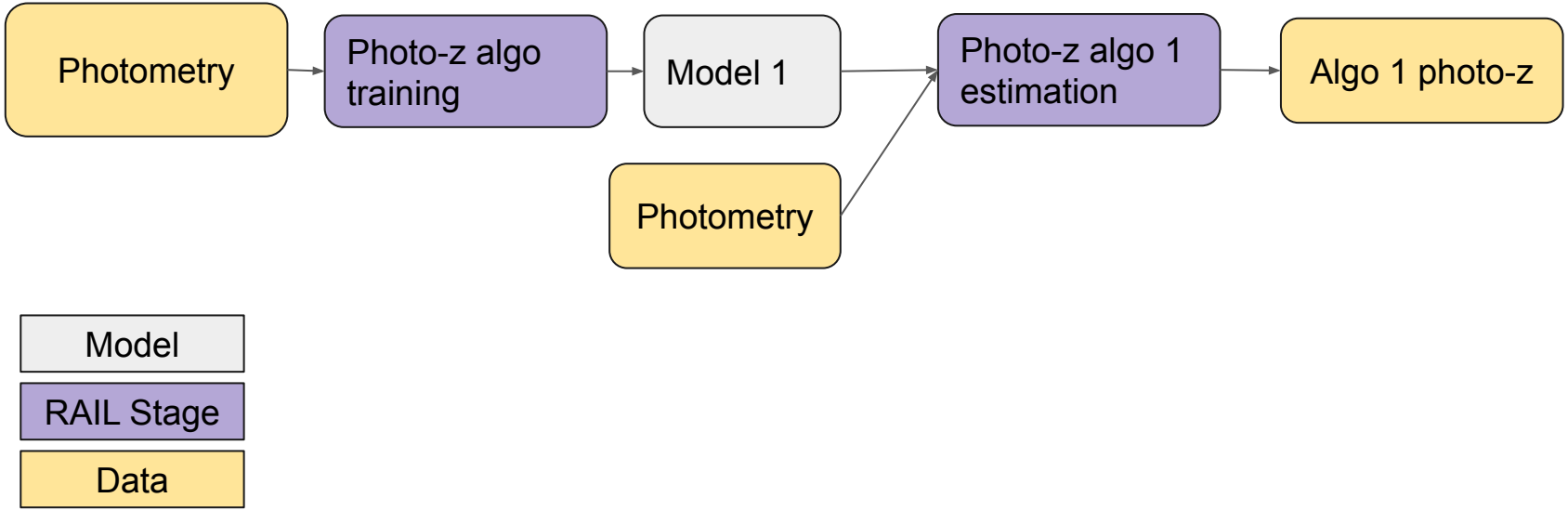
- RAIL has a lots of sub-packages, for the flexibility of installation
- But all RAIL packages share the same name space
  - A namespace is a system that has a unique name for each and every object in Python
- RAIL namespace is organized in the following way:
  - Informer, estimator and summarizers -> rail.estimation.algo
  - Engine -> rail.creation.engine
  - Degraders -> rail.creation.degrader
  - Evaluators -> rail.evaluation.metrics
  - Pipeline -> rail.pipeline
  - Basic classes -> rail.core
  - Utility (Tools) -> rail.utils (rail.tools)
    - Util are methods, tools are classes

TASK 2: GET SOME PHOTO-Z

Go to RAIL_estimation_demo.ipynb

RAIL pipeline

You can put multiple RAIL stages together to form a pipeline, as long as they can be represented by a Directed Acyclic Graph (meaning, no loop)



RAIL pipeline

The pipeline can be made in python and saved as yaml files

We are building pre-made pipelines in rail_pipelines

- estimate_all.py
- inform_all.py
- pz_all.py
- tomography.py

- apply_phot_errors.py
- blending.py
- spectroscopic_selection_pipeline.py

- evaluate_all.py

Pipeline YAML files

Ceci config

```yaml
config: config/all/inform_gpz_config.yml
inputs:
  input: ./data/dered_pdr3_wide_train_curated.pq
log_dir: ./log
modules: rail
output_dir: .
resume: false
site:
  max_threads: 2
  name: local
stages:
- classname: GPzInformer
  module_name: rail.estimation.algos.gpz
  name: inform_gpz
  nprocess: 1
```

To run a pipeline:
> ceci [ceci_config.yml]

RAIL config

```yaml
inform_gpz:
  aliases:
    model: model_inform_gpz
  bands:
  - HSCg_cmodel_dered
  - HSCr_cmodel_dered
  - HSCi_cmodel_dered
  - HSCz_cmodel_dered
  - HSCy_cmodel_dered
  config: null
  csl_binwidth: 0.1
  csl_method: normal
  err_bands:
  - g_cmodel_magerr
  - r_cmodel_magerr
  - i_cmodel_magerr
  - z_cmodel_magerr
  - y_cmodel_magerr
  gpz_method: VC
  hdf5_groupname: ''
  hetero_noise: true
  input: None
  learn_jointly: true
  log_errors: true
  mag_limits:
    HSCg_cmodel_dered: 27.88
    HSCi_cmodel_dered: 26.6
    HSCr_cmodel_dered: 27.05
    HSCy_cmodel_dered: 25.64
    HSCz_cmodel_dered: 26.6
  max_attempt: 100
  max_iter: 200
  model: model/estimator/model_gpz.pkl
  n_basis: 50
  name: inform_gpz
```