desisim Documentation

Release 0.31.1

DESI

Dec 14, 2018

Contents

CHAPTER 1

Introduction

This is the documentation for desisim.

CHAPTER 2

Contents

The `toctree` directive can be used to link to other files in this directory and even display their sections. See the toctree documentation.

2.1 Quality Assurance

2.1.1 Overview

The desisim pacakge includes a few scripts for running QA on the outputs.

2.1.2 Scripts

desi_qa_zfind

Generate redshift accuracy QA based on the truth vs. RedRock outputs.

usage

Here is the usage:

```
usage: desi_qa_zfind [-h] [--verbose] [--load_simz_table LOAD_SIMZ_TABLE]
                     [--reduxdir PATH] [--rawdir PATH] [--yaml_file YAML_FILE]
                     [--qafig_path QAFIG_PATH]
                     [--write_simz_table WRITE_SIMZ_TABLE]

Generate QA on redshift for a production [v0.2.1]

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  --verbose             Provide verbose reporting of progress. (default:
                        False)
  --load_simz_table LOAD_SIMZ_TABLE
                        Load an existing simz Table to remake figures
                        (default: None)
  --reduxdir PATH       Override default path ($DESI_SPECTRO_REDUX/$SPECPROD)
                        to processed data. (default: None)
  --rawdir PATH         Override default path ($DESI_SPECTRO_REDUX/$SPECPROD)
                        to processed data. (default: None)
  --yaml_file YAML_FILE
                        YAML file for debugging (primarily). (default: None)
  --qafig_path QAFIG_PATH
                        Path to where QA figure files are generated. Default
                        is specprod_dir+/QA (default: None)
  --write_simz_table WRITE_SIMZ_TABLE
                        Write simz to this filename (default: None)
```

example

Here is the typical execution:

```
desi_qa_zfind
```

desi_qa_s2n

Generate S/N QA for all object types for all of the nights in a production. The object types and redshifts are taken from *truth*.

usage

Here is the usage:

```
usage: desi_qa_s2n [-h] [--reduxdir PATH] [--qafig_path QAFIG_PATH]

Generate S/N QA for a production [v0.2.1]

optional arguments:
  -h, --help            show this help message and exit
  --reduxdir PATH       Override default path ($DESI_SPECTRO_REDUX) to
                        processed data. (default: None)
  --qafig_path QAFIG_PATH
                        Path to where QA figure files are generated. Default
                        is specprod_dir+/QA (default: None)
```

examples

Generate the figures:

```
desi_qa_s2n
```

A series of PNG files are created for the various cameras and object types.

2.2 desisim change log

2.2.1 0.31.2 (unreleased)

- No changes yet.

2.2.2 0.31.1 (2018-12-14)

- quickquasars updates:
  - support eBOSS (PR #450).
  - mimic redshift fitter uncertainties (PR #452).
  - adding shift to redshift (PR #454).
  - fix error in size of Z_noFOG (PR #455).
  - Fix quickquasars targetid truth (PR #457).
- Precompute colors for star and galaxy templates. (PR #453).
- Refactor S/N qa to load cframes only once (also updates OII for new TRUTH table) (PR #459, PR #465).
- Use basis_templates v3.1 and matching desisim-testdata 0.6.1 (PR #464).

2.2.3 0.31.0 (2018-11-08)

- Update to new fibermap format for consistency with targeting and fiber assignment; requires desispec >= 0.26.0 (PR #446).
- Update *desisim.templates.BGS* to use latest selection cuts (PR #439).
- Fix quickquasar to work with Saclay mocks (PR #435).
- Add support for >v3.0 stellar templates, with notebook to boot (PR #434).
- Update notebook describing the construction of the LRG templates (PR #433).
- Fix quicksurvey (PR #431).
- Update quickcat model (PR #430, PR #427).
- Fix archetype computation for redrock (PR #429).
- Change `electron` to `count` for FITS compliance (PR #428).
- Do not include Mg II emission by default (PR #426).
- Add and adjust the nebular emission line spectra added to galaxy templates (PR #424).
- quickquasar options for random z, ignoring transmission, random seeds, desisim.templates.SIMQSO vs. QSO (PR #419, PR #408, PR #406, PR #401).
- Read and write *select_mock_targets* style *simspec* file (PR #416).
- Restore *quickquasars* to a functioning state, after being broken in PR #409 (PR #413).
- Add optional *nside* and *overwrite* arguments to *wrap-newexp* and *obs.new_exposure*, respectively (PR #412).
- Major (and backwards-incompatible) refactor of how the template/simulated metadata are returned by desisim.templates (PR #409).

- Adding reading metals from LyA transmission files (PR #407).

2.2.4 0.30.0 (2018-08-09)

- Update templates to DR7+ standard-star designation (FSTD–>STD) (PR #400).
- Update standard star bit name again STD -> STD_FAINT; requires desitarget 0.23.0 (PR #402).

2.2.5 0.29.0 (2018-07-26)

- Option in quickspectra to write the full sim table (PR #392).
- Option to use Gaussian instead of Poisson for QSO DLA. Requires specsim >= v0.12 (PR #393).
- Use *overwrite* instead of *clobber* for *astropy.io.fits* (PR #395).

2.2.6 0.28.0 (2018-07-18)

- Add BALs to templates.QSO class (PR #321).
- Enable redshift QA using input summary catalogs of truth and redshifts (PR #349).
- Add zstats-like good/fail/miss/list QA method from desitest mini notebook and refactor previous code to enable it (PR #351).
- quickquasar mags and random seed (PR #350 and PR #352).
- New pixsim and pixsim_nights (PR #353, PR #354, and PR #358).
- Generate confusion matrix related to Spectype (PR #359).
- Update QA to use qaprod_dir
- Fix newexp-mock wrapper when first expid != 0 (PR #361).
- newexp-mock options for production running (PR #363).
- Add BALs to QSO spectra outside of desisim.templates (PR #370).
- Add rest-frame option to templates.SIMQSO (PR #377).
- Optionally change output wave vector in templates.SIMQSO when noresample=True or restframe=True (PR #383).
- Fix `newexp-mock` and `wrap-fastframe` file parsing for `NIGHT/EXPID/*.*` vs. `NIGHT/*.*`.
- Speed up emission line simulation when using `MKL >= 2018.0.2` (PR #390).

2.2.7 0.27.0 (2018-03-29)

- Fix pixsim_mpi; make it faster with scatter/gather (PR #329, PR #332, and PR #344).
- Fix PSF convolution for newexp-mock (PR #331).
- BGS redshift bug fix (PR #333).
- Astropy 2 compatibility (PR #334).
- Fix newexp-mock –nspec option (PR #340).
- Fix fibermap EXTNAME (PR #340).

- Fix PSF convolution for newexp_mock (PR #331).
- Match desispec renaming and relocating of of pix -> preproc (PR #337 and PR #339).
- More robust handling of unassigned fiber inputs (PR #341).

2.2.8 0.26.0 (2018-02-27)

Requires desitarget >= 0.19.0

- Update BGS fiber acceptance vs. z (PR #326)
- Update desitarget imports for desitarget/0.19.0 (PR #328)

2.2.9 0.25.1 (2018-02-23)

Requires desitarget < 0.19.0

- Fix set_xscale(...) nonposy -> nonposx for qa_zfind

2.2.10 0.25.0 (2018-02-23)

- Fix double PSF convolution in pixsims (PR #320).
- Additional edits to QA scripts and doc to run with mini Notebook (PR #322).
- Optional specsim config for simulating spectra (PR #325)

2.2.11 0.24.0 (2018-01-30)

- Support new LRG templates (v2.0). (PR #302).
- Bug fixes and additional features added to SIMQSO template maker. (PR #303).
- Fixes quickspectra (broken by desispec change) (PR #306).
- Fixes quickspectra random seed (never worked?) (PR #306).
- Improves pixsim_mpi performance (PR #312).
- Optionally do not wavelength resample simqso templates (PR #310).
- Default to basis templates v2.4 instead of 2.3
- Minor edits to QA scripts and doc (PR #311).
- Adds quickspectra –skyerr option (PR #313).
- Correct fastframe output BUNIT (PR #317).

2.2.12 0.23.0 (2017-12-20)

- Fixed crash in newexp-mock success print message.
- Refactor DLA code into its own module (PR #294).
- Adds reader for LyA skewer v2.x format (PR #297).
- Removed deprecated brick output from quickgen.

- Preliminary support for simqso based QSO templates (PR #293).
- fastframe can directly output cframes (PR #287).
- adds BGS efficiency notebooks (PR #285 and PR #286).

2.2.13 0.22.0 (2017-11-10)

- Scaling updates to wrap-fastframe and wrap-newexp (PR #274).
- Fix a minor units scaling bug in lya_spectra (PR #264).
- newexp takes exposures list with EXPID and arcs/flats (PR #275).
- lyman alpha QSOs with optional DLAs (PR #275).
- Update arc lamp line list (PR #272).
- Fix MPI pixsim wrappers (PR #265 and PR #262).
- quicksurvey updats for latest surveysim outputs (PR #270).
- Adds fastfiber method of fiber input loss calculations (PR #261).
- Fix quickgen moon input parameters (PR #263).
- Adds quickspectra script (PR #259).

2.2.14 0.21.0 (2017-09-29)

- Major refactor of newexp to add connection to upstream mocks, surveysims, and fiber assignment (PR #250).
- Support latest (>DR4) data model in the templates metadata table and also scale simulated templates by 1e17 erg/s/cm2/Angstrom (PR #252).
- Add desi_qa_s2n script (PR #254)
- Refactor desi_qa_zfind script (PR #254)
- Refactor redshift QA for new data model (PR #254)
- Refactor shared QA methods to desisim.spec_qa.utils (PR #254)
- New plots for S/N of spectra for various objects (ELG, LRG, QSO) (PR #254)
- Add BGS, MWS to z_find QA
- Miscellaneous polishing in QA (velocity, clip before RMS, extend [OII] flux, S/N per Ang)
- Bug fix: correctly select both "bright" and "faint" BGS templates by default (PR #257).
- Updates for newexp/fastframe wrappers for end-to-end sims (PR #258).

2.2.15 0.20.0 (2017-07-12)

- Adds tutorial on simulating spectra (PR #244).
- Fixes QSO template wavelength extrapolation (PR #247); requires desispec > 0.15.1.
- Uses `desitarget.cuts.isLRG_colors`; requires desitarget >= 0.14.0 (PR #246).
- Uses `desiutil.log` instead of `desispec.log`.

2.2.16 0.19.0 (2017-06-15)

- "FLAVOR" keyword is arc/flat/science but not dark/bright/bgs/mws/etc to match desispec usage (PR #243).
- Add `nocolorcuts` option for LyA spectra (PR #242).
- Fixes for `targets.dat` to `targets.yaml` change (PR #240).
- Changed refs to `desispec.brick` to its new location at `desiutil.brick` (PR #241).
- Remove LyA absorption below the LyA limit (PR #236).
- Refactor and speed-up of QSO templates; add Lya forest on-the-fly (PR #234).

2.2.17 0.18.3 (2017-04-13)

- Add DLAs to lya spectra (PR #220)
- Fix quickgen for specsim v0.8 (PR #226).
- Add verbose output to templates code (PR #230).
- Much faster quickcat (PR #233).

2.2.18 0.18.2 (2017-03-27)

- Fixed a number of documentation errors (PR #224).
- Removed unneeded Travis scripts in `etc/`.
- Fixed N^2 scaling of *`desisim.templates.QSO.make_templates()`*.
- Speed up *`desisim.templates.GALAXY`* by factor of 8-12 by caching velocity dispersions (PR #229)

2.2.19 0.18.1 (2016-03-05)

- Update `desisim.module` to use `DESI_BASIS_TEMPLATES` v2.3.

2.2.20 0.18.0 (2016-03-04)

- pixsims add new required keywords DOSVER, FEEVER, DETECTOR.
- Small bug fixes in quickcat; drop unused truth,targets columns to save memory in quicksurvey loop (PRs #198, #199).
- quickgen update to support white dwarf templates (PR #204)
- several enhancements of the templates code
  - optionally output rest-frame templates (PR #208)
  - rewrite of lya_spectra to achieve factor of 10 speedup; use COSMO (astropy.cosmology setup) as a new optional keyword for qso_desi_templates; updated API (PRs #210, #212)
  - various small changes to desisim.templates (PR #211)
  - support for DA and DB white dwarf subtypes (PR #213)
- update test dependencies (PR #214)

2.2.21 0.17.1 (2016-12-05)

- Fix bug when obsconditions contain tiles that don't overlap catalog
- Add `surveysim --start_epoch` option

2.2.22 0.17.0 (2016-12-02)

- fixes tests for use with latest desitarget master
- Refactor quickgen and quickbrick to reduce duplicated code (PR #184)
- Makes BGS compatible with desitarget master after isBGS -> isBGS_faint vs. isBGS_bright
- Refactor quickcat to include dependency on observing conditions
- Update quicksurvey to use observing conditions from surveysim
- Fixes use of previous zcatalog when updating catalog with new observations

2.2.23 0.16.0 (2016-11-10)

- Requires specsim >= v0.6
- Add integration test for quickgen (PR #179)
- Cache specsim Simulator for faster testing (PR #178)
- Add lya_spectra.get_spectra (PR #156)
- Add quickgen and quickbrick unit tests and bug fixes (PR #176, #177)

2.2.24 0.15.0 (2016-10-14)

- Fix some `build_sphinx` errors.
- Run coverage tests under Python 2.7 for now.
- Update template Module file to new DESI+Anaconda infrastructure.
- quickbrick unit tests and bug fixes (#166)
- new quickgen features (PR #173 and #175)
  - fix exptime and airmass for specsim v0.5
  - new –frameonly option
  - moon phase, angle, and zenith options
  - misc cleanup and unit tests

2.2.25 0.14.0 (2016-09-14)

- updates for python 3.5

2.2.26 0.13.1 (2016-08-18)

- fix batch.pixsim seeds vs. seed typo

2.2.27 0.13.0 (2016-08-18)

- desi_qa_zfind: fixed –reduxdir option; improved plots
- PR#132: major refactor of template generation, including ability to give input redshifts, magnitudes, or random seeds from metadata table.
- desisim.batch.pixsim functions propagate random seeds for reproducibility

2.2.28 0.12.0 (2016-07-14)

- desi_qa_zfind options to override raw and processed data directories
- PRODNAME -> SPECPROD and TYPE -> SPECTYPE to match latest desispec
- remove unused get_simstds.py
- fix #142 so that pixsim only optionally runs preprocessing
- fix #141 to avoid repeated TARGETIDs when simulating both bright and dark tiles together
- add io.load_simspec_summary() convenience function to load and merge truth information from fibermap and simspec files.
- adjusts which magnitudes were plotted for each target class

2.2.29 0.11.0 (2016-07-12)

Pixsim updates:

- simulate fully raw data, then call preprocessing
- bug fix for simulating tiles in parallel
- fix pixsim loading of non-default PSFs

2.2.30 0.10.0 and prior

- No changes.rst yet

2.3 desisim API

2.3.1 desisim

Tools for DESI instrument simulations, including input templates. It does not cover cosmology simulations.

2.3.2 desisim.archetypes

Archetype routines for desisim.

class desisim.archetypes.**ArcheTypes**(*chi2*)

Object for generating archetypes and determining their responsibility.

Parameters **chi2** (*numpy.ndarray*) – Chi^2 matrix computed by desisim.archetypes.compute_chi2().

get_archetypes(*chi2_thresh=0.1*, *responsibility=False*)
Solve the SCP problem to get the final set of archetypes and, optionally, their responsibility.

Note: We assume that each template has uniform "cost" but a more general model in principle could be used / implemented.

Parameters

- **chi2** (*numpy.ndarray*) – Chi^2 matrix computed by archetypes.compute_chi2().
- **chi2_thresh** (*float*) – Threshold chi2 value to differentiate "different" templates.
- **responsibility** (*bool*) – If True, then compute and return the responsibility of each archetype.

Returns

- *If responsibility==True then returns a tuple of (iarch, resp, respindx) where –*

  iarch [integer numpy.array] Indices of the archetypes [N].

  resp [integer numpy.array] Responsibility of each archetype [N].

  respindx [list of] Indices the parent sample each archetype is responsible for [N].

- *If responsibility==False then only iarch is returned.*

responsibility(*iarch*, *a_matrix*)
Method to determine the responsibility of each archetype.

In essence, the responsibility is the number of templates described by each archetype.

Parameters

- **iarch** (*indices of the archetypes*) –
- **a_matrix** (*distance matrix*) –

Returns

- **resp** (*responsibility of each archetype (number of objects represented by each archetype)*)
- **respindx** (*list containing the indices of the parent objects represented by each archetype*)

desisim.archetypes.**compute_chi2**(*flux*, *ferr=None*)
Compute the chi2 distance matrix.

Parameters

- **flux** (*numpy.ndarray*) – Array [Nspec, Npix] of spectra or templates where Nspec is the number of spectra and Npix is the number of pixels.
- **ferr** (*numpy.ndarray*) – Uncertainty spectra ccorresponding to flux (default None).

Returns

chi2 [numpy.ndarray] Chi^2 matrix [Nspec, Nspec] between all combinations of normalized spectra.

amp [numpy.ndarray] Amplitude matrix [Nspec, Nspec] between all combinations of spectra.

Return type Tuple of (chi2, amp) where

2.3.3 desisim.batch

Batch scripts. Why exactly is this sub-package different from *desisim.scripts*?

desisim.batch.**calc_nodes**(*ntasks*, *tasktime*, *maxtime*)
Return a recommended number of nodes to use to process *ntasks* within *maxtime* if each one takes *tasktime* minutes.

2.3.4 desisim.batch.pixsim

Provides utility functions for batch processing of pixel-level simulations at NERSC. This is a temporary pragmatic package – after desispec.pipeline code is merged and vetted, this should use that infrastructure for more rigorous logging, environment setup, and scaling flexibility.

Example:

#- From python import desisim.batch.pixsim flavors = ['arc', 'flat', 'dark', 'dark', 'gray', 'gray', 'bright', 'bright'] expids = range(len(flavors)) desisim.batch.pixsim.batch_newexp('newexp-blat.sh', flavors, expids=expids) desisim.batch.pixsim.batch_pixsim('pixsim-blat.sh', flavors, expids=expids)

#- then from the command line [edison] sbatch newexp-batch.sh Submitted batch job 233895 [edison] sbatch -d afterok:233895 pixsim-batch.sh Submitted batch job 233901

desisim.batch.pixsim.**batch_newexp**(*batchfile*, *flavors*, *nspec=5000*, *night=None*, *expids=None*, *nodes=None*, *pixprod=None*, *desi_spectro_sim=None*, *tileids=None*, *seed=None*)
Write a slurm batch script for run newexp-desi for the list of flavors

desisim.batch.pixsim.**batch_pixsim**(*batchfile*, *flavors*, *nspec=5000*, *night=None*, *expids=None*, *nodes=None*, *pixprod=None*, *desi_spectro_sim=None*, *seed=None*)
Write a slurm batch script for run newexp-desi for the list of flavors

2.3.5 desisim.cosmology

All cosmology related routines of desisim should be put here for consistency.

2.3.6 desisim.dla

Functions and methods for inserting DLAs into QSO spectra.

desisim.dla.**calc_lz**(*z*, *boost=1.6*)

Parameters

- **z** (*ndarray*) – redshift values for evaluation
- **boost** (*float*) – boost for SLLS (should be 1 if only running DLAs)

Returns l(z) aka dN/dz values of DLAs

Return type ndarray

desisim.dla.**calculate_lox**(*model*, *NHI_min*, *NHI_max=None*, *neval=10000*, *cumul=False*)
Calculate l(X) over an N_HI interval

Parameters

- **z** (*float*) – Redshift for evaluation

- **NHI_min** (*float*) – minimum log NHI value
- **NHI_max** (*float, optional*) – maximum log NHI value for evaluation (Infinity)
- **neval** (*int, optional*) – Discretization parameter (10000)
- **cumul** (*bool, optional*) – Return a cumulative array? (False)
- **cosmo** (*astropy.cosmology, optional*) – Cosmological model to adopt (as needed)

Returns **lX** – l(X) value

Return type float

desisim.dla.**dla_spec**(*wave*, *dlas*)

Generate spectrum absorbed by dlas :param wave: observed wavelengths :type wave: ndarray :param dlas: DLA dicts :type dlas: list

Returns ndarray of absorbed flux

Return type abs_flux

desisim.dla.**evaluate_fN**(*model*, *NHI*)

Evaluate an f(N,X) model at a set of NHI values

Parameters **NHI** (*array*) – log NHI values

Returns **log_fN** – f(NHI,X) values

Return type array

desisim.dla.**init_fNHI**(*slls=False*, *mix=True*)

Parameters

- **slls** (*bool*) – SLLS only?
- **mix** (*bool*) – Mix of DLAs and SLLS?

Returns fNHI model

Return type model

desisim.dla.**insert_dlas**(*wave*, *zem*, *rstate=None*, *seed=None*, *fNHI=None*, *debug=False*, ***kwargs*)

Insert zero, one or more DLAs into a given spectrum towards a source with a given redshift :param wave: wavelength array in Ang :type wave: ndarray :param zem: quasar emission redshift :type zem: float :param rstate: for random numberes :type rstate: numpy.random.rstate, optional :param seed: :type seed: int, optional :param fNHI: f_NHI object :type fNHI: spline :param **kwargs: Passed to init_fNHI()

Returns List of DLA dict's with keys z,N dla_model (ndarray): normalized specrtrum with DLAs inserted

Return type dlas (list)

desisim.dla.**voigt_tau**(*wave*, *par*)

Find the optical depth at input wavelengths Taken from linetools.analysis.voigt

This is a stripped down routine for calculating a tau array for an input line. Built for speed, not utility nor with much error checking. Use wisely. And take careful note of the expected units of the inputs (cgs)

Parameters

- **wave** (*ndarray*) – Assumed to be in cm
- **parm** (*list*) –

Line parameters. All are input unitless and should be in cgs par[0] = logN (cm^-2) par[1] = z par[2] = b in cm/s par[3] = wrest in cm par[4] = f value par[5] = gamma (s^-1)

Returns **tau** – Optical depth at input wavelengths

Return type ndarray

desisim.dla.**voigt_wofz**(*vin*, *a*)
Uses scipy function for calculation. Taken from linetools.analysis.voigt

Parameters

- **vin** (*ndarray*) – u parameter
- **a** (*float*) – a parameter

Returns **voigt**

Return type ndarray

2.3.7 desisim.io

I/O routines for desisim

class desisim.io.**SimSpec**(*flavor*, *wave*, *flux*, *skyflux*, *fibermap*, *truth*, *obsconditions*, *header*, *objtruth=None*)
Lightweight wrapper object for simspec data

Object has properties flavor, nspec, wave, flux, skyflux, fibermap, truth, obsconditions, header, cameras.

cameras is dict, keyed by camera name, of SimSpecCameras objects with properies wave, phot, skyphot.

add_camera(*camera*, *wave*, *phot*, *skyphot=None*)
Add per-camera photons to this SimSpec object, using SimSpecCamera

Parameters

- **camera** – camera name, e.g. b0, r1, z9
- **wave** – 1D[nwave] array of wavelengths
- **phot** – 2D[nspec, nwave] array of photons per bin (not per Angstrom)

Optional: skyphot: 2D[nspec, nwave] array of sky photons per bin

class desisim.io.**SimSpecCamera**(*camera*, *wave*, *phot*, *skyphot=None*)
Wrapper of per-camera photon data from a simspec file

desisim.io.**_parse_filename**(*filename*)
Parse filename and return (prefix, camera, expid)

camera=None if the filename isn't camera specific

e.g. /blat/foo/simspec-00000003.fits -> ('simspec', None, 3) e.g. /blat/foo/preproc-r2-00000003.fits -> ('preproc', 'r2', 3)

desisim.io.**_qso_format_version**(*filename*)
Return 1 or 2 depending upon QSO basis template file structure

desisim.io.**_resize**(*image*, *shape*)
Resize input image to have new shape, preserving its 2D arrangement

Parameters

- **image** – 2D ndarray
- **shape** – tuple (ny,nx) for desired output shape

Returns new image with image.shape == shape

desisim.io.**empty_metatable**(*nmodel=1*, *objtype='ELG'*, *subtype=''*, *simqso=False*, *input_meta=False*)

Initialize template metadata tables depending on the given object type.

Parameters

- **nmodel** (int) – Number of rows in output table. Defaults to 1.
- **objtype** (str) – Object type. Defaults to ELG.
- **subtype** (str) – Subtype for the given object type (e.g., LYA is objtype=QSO). Defaults to .
- **simqso** (bool) – Initialize a templates.SIMQSO-style objmeta table rather than a templates.QSO one. Defaults to False.
- **input_meta** (bool) – Initialize an input_meta table for use with the various desisim.templates classes (see its use in, e.g., desitarget.mock.mockmaker) Defaults to False.

Returns

- **meta** (astropy.table.Table) – Metadata table which is agnostic about the object type.
- **objmeta** (astropy.table.Table) – Objtype-specific supplemental metadata table (e.g., containing the [OII] flux for ELG targets and surface gravity for stars.

desisim.io.**empty_snemetatable**(*nmodel=1*)

Initialize a metadata table for SNE.

Parameters **nmodel** (int) – Number of rows in output table. Defaults to 1.

Returns **snemeta** – Metadata table.

Return type astropy.table.Table

desisim.io.**fibers2cameras**(*fibers*)

Return a list of cameras covered by an input array of fiber IDs

desisim.io.**find_basis_template**(*objtype*, *indir=None*)

Return the most recent template in $DESI_BASIS_TEMPLATE/{objtype}_template*.fits

desisim.io.**find_cosmics**(*camera*, *exptime=1000*, *cosmics_dir=None*)

Return full path to cosmics template file to use

Parameters

- **camera** (*str*) – e.g. 'b0', 'r1', 'z9'
- **exptime** (*int, optional*) – exposure time in seconds
- **cosmics_dir** (*str, optional*) – directory to look for cosmics templates; defaults to $DESI_COSMICS_TEMPLATES if set or otherwise $DESI_ROOT/spectro/templates/cosmics/v0.2 (note HARDCODED version)

Exposure times <120 sec will use the bias templates; otherwise they will use the dark cosmics templates

desisim.io.**findfile**(*filetype*, *night*, *expid*, *camera=None*, *outdir=None*, *mkdir=True*)

Return canonical location of where a file should be on disk

Parameters

- **filetype** (*str*) – file type, e.g. 'preproc' or 'simpix'
- **night** (*str*) – YEARMMDD string
- **expid** (*int*) – exposure id integer
- **camera** (*str*) – e.g. 'b0', 'r1', 'z9'
- **outdir** (*Optional[str]*) – output directory; defaults to $DESI_SPECTRO_SIM/$PIXPROD
- **mkdir** (*Optional[bool]*) – create output directory if needed; default True

Returns full file path to output file

Return type str

Also see desispec.io.findfile() which has equivalent functionality for real data files; this function is only be for simulation files.

desisim.io.**get_tile_radec**(*tileid*)
Return (ra, dec) in degrees for the requested tileid.

If tileid is not in DESI, return (0.0, 0.0) TODO: should it raise an exception instead?

desisim.io.**load_simspec_summary**(*indir*, *verbose=False*)
Combine fibermap and simspec files under indir into single truth catalog

Parameters **indir** – path to input directory; search this and all subdirectories

Returns astropy.table.Table with true Z catalog

desisim.io.**read_basis_templates**(*objtype*, *subtype=''*, *outwave=None*, *nspec=None*, *infile=None*, *onlymeta=False*, *verbose=False*)
Return the basis (continuum) templates for a given object type. Optionally returns a randomly selected subset of nspec spectra sampled at wavelengths outwave.

Parameters

- **objtype** (*str*) – object type to read (e.g., ELG, LRG, QSO, STAR, STD, WD, MWS_STAR, BGS).
- **subtype** (*str, optional*) – template subtype, currently only for white dwarfs. The choices are DA and DB and the default is to read both types.
- **outwave** (*numpy.array, optional*) – array of wavelength at which to sample the spectra.
- **nspec** (*int, optional*) – number of templates to return
- **infile** (*str, optional*) – full path to input template file to read, over-riding the contents of the $DESI_BASIS_TEMPLATES environment variable.
- **onlymeta** (*Bool, optional*) – read just the metadata table and return
- **verbose** – bool Be verbose. (Default: False)

Returns Tuple of (outflux, outwave, meta) where outflux is an Array [ntemplate,npix] of flux values [erg/s/cm2/A]; outwave is an Array [npix] of wavelengths for FLUX [Angstrom]; meta is a Meta-data table for each object. The contents of this table varies depending on what OBJTYPE has been read.

Raises

- EnvironmentError – If the required $DESI_BASIS_TEMPLATES environment variable is not set.

- IOError – If the basis template file is not found.

desisim.io.**read_cosmics**(*filename*, *expid=1*, *shape=None*, *jitter=True*)
Reads a dark image with cosmics from the input filename.

The input might have multiple dark images; use the *expid%n* image where *n* is the number of images in the input cosmics file.

Parameters

- **filename** – FITS filename with EXTNAME=IMAGE-, *IVAR*-, MASK-* HDUs
- **expid** – integer, use *expid % n* image where *n* is number of images
- **shape** – (ny, nx, optional) tuple for output image shape
- **jitter** (*bool, optional*) – If True (default), apply random flips and rolls so you don't get the exact same cosmics every time

Returns *desisim.image.Image* object with attributes pix, ivar, mask

desisim.io.**read_simspec**(*filename*, *cameras=None*, *comm=None*, *readflux=True*, *readphot=True*)
Read a simspec file and return a SimSpec object

Parameters **filename** – input simspec file name

Options: cameras: camera name or list of names, e.g. b0, r1, z9 comm: MPI communicator readflux: if True (default), include flux readphot: if True (default), include per-camera photons

desisim.io.**simdir**(*night=None*, *expid=None*, *mkdir=False*)
Return $DESI_SPECTRO_SIM/$PIXPROD/{night} If mkdir is True, create directory if needed

desisim.io.**write_simpix**(*outfile*, *image*, *camera*, *meta*)
Write simpix data to outfile.

Parameters

- **outfile** – output file name, e.g. from io.findfile('simpix', ...)
- **image** – 2D noiseless simulated image (numpy.ndarray)
- **meta** – dict-like object that should include FLAVOR and EXPTIME, e.g. from HDU0 FITS header of input simspec file

desisim.io.**write_simspec**(*sim*, *truth*, *fibermap*, *obs*, *expid*, *night*, *objmeta=None*, *outdir=None*, *filename=None*, *header=None*, *overwrite=False*)
Write a simspec file

Parameters

- **sim** (*Simulator*) – specsim Simulator object
- **truth** (*Table*) – truth metadata Table
- **fibermap** (*Table*) – fibermap Table
- **obs** (*dict-like*) – dict-like observation conditions with keys SEEING (arcsec), EXPTIME (sec), AIRMASS, MOONFRAC (0-1), MOONALT (deg), MOONSEP (deg)
- **expid** (*int*) – integer exposure ID
- **night** (*str*) – YEARMMDD string
- **objmeta** (*dict*) – objtype-specific metadata
- **outdir** (*str, optional*) – output directory

- **filename** (*str, optional*) – if None, auto-derive from envvars, night, expid, and outdir
- **header** (*dict-like*) – header to include in HDU0
- **overwrite** (*bool, optional*) – overwrite pre-existing files

Notes

Calibration exposures can use `truth=None` and `obs=None`.

desisim.io.**write_simspec_arc**(*filename*, *wave*, *phot*, *header*, *fibermap*, *overwrite=False*)
Alternate writer for arc simspec files which just have photons

class desisim.lya_mock_p1d.**MockMaker**(*N2=15*, *dv_kms=10.0*, *seed=666*, *white_noise=False*)
Class to generate 1D mock Lyman alpha skewers.

get_density(*var_delta*, *z*, *delta*)
Transform Gaussian field delta to lognormal density, at each z.

get_gaussian_fields(*Ns=1*, *new_seed=None*)
Generate Ns Gaussian fields at redshift z_c.

If new_seed is set, it will reset random generator with it.

get_lya_skewers(*Ns=10*, *new_seed=None*)
Return Ns Lyman alpha skewers (wavelength, flux).

If new_seed is set, it will reset random generator with it.

get_redshifts()
Get redshifts for each cell in the array (centered at z_c).

desisim.lya_mock_p1d.**get_tau**(*z*, *density*)
transform lognormal density to optical depth, at each z

desisim.lya_mock_p1d.**power_amplitude**(*z*)
Add redshift evolution to the Gaussian power spectrum.

desisim.lya_mock_p1d.**power_kms**(*z_c*, *k_kms*, *dv_kms*, *white_noise*)
Return Gaussian P1D at different wavenumbers k_kms (in s/km), fixed z_c.

Other arguments: dv_kms: if non-zero, will multiply power by top-hat kernel of this width white_noise: if set to True, will use constant power of 100 km/s

2.3.8 desisim.lya_spectra

Function to simulate a QSO spectrum including Lyman-alpha absorption.

desisim.lya_spectra.**apply_lya_transmission**(*qso_wave*, *qso_flux*, *trans_wave*, *trans*)
Apply transmission to input flux, interpolating if needed. Note that the transmission might include Lyman-beta and metal absorption, so we should probably change the name of this function.

Parameters

- **qso_wave** – 1D[nwave] array of QSO wavelengths
- **qso_flux** – 2D[nqso, nwave] array of fluxes
- **trans_wave** – 1D[ntranswave] array of transmission wavelength samples
- **trans** – 2D[nqso, ntranswave] transmissions [0-1]

Returns output_flux[nqso, nwave]

This routine simply apply the transmission the only thing besides multiplication is a wavelength interpolation of transmission to the QSO wavelength grid

desisim.lya_spectra.**apply_metals_transmission**(*qso_wave*, *qso_flux*, *trans_wave*, *trans*, *metals*)

Apply metal transmission to input flux, interpolating if needed. The input transmission should be only due to lya, if not has no meaning. This function should not be used in London mocks with version > 2.0, since these have their own metal transmission already in the files, and even the "TRANSMISSION" HDU includes already Lyman beta.

Parameters

- **qso_wave** – 1D[nwave] array of QSO wavelengths
- **qso_flux** – 2D[nqso, nwave] array of fluxes
- **trans_wave** – 1D[ntranswave] array of lya transmission wavelength samples
- **trans** – 2D[nqso, ntranswave] transmissions [0-1]
- **metals** – list of metal names to use

Returns output_flux[nqso, nwave]

desisim.lya_spectra.**get_spectra**(*lyafile*, *nqso=None*, *wave=None*, *templateid=None*, *normfilter='sdss2010-g'*, *seed=None*, *rand=None*, *qso=None*, *add_dlas=False*, *debug=False*, *nocolorcuts=True*)

Generate a QSO spectrum which includes Lyman-alpha absorption.

Parameters

- **lyafile** (*str*) – name of the Lyman-alpha spectrum file to read.
- **nqso** (*int, optional*) – number of spectra to generate (starting from the first spectrum; if more flexibility is needed use TEMPLATEID).
- **wave** (*numpy.ndarray, optional*) – desired output wavelength vector.
- **templateid** (*int numpy.ndarray, optional*) – indices of the spectra (0-indexed) to read from LYAFILE (default is to read everything). If provided together with NQSO, TEMPLATEID wins.
- **normfilter** (*str, optional*) – normalization filter
- **seed** (*int, optional*) – Seed for random number generator.
- **rand** (*numpy.RandomState, optional*) – RandomState object used for the random number generation. If provided together with SEED, this optional input superseeds the numpy.RandomState object instantiated by SEED.
- **qso** (*desisim.templates.QSO, optional*) – object with which to generate individual spectra/templates.
- **add_dlas** (*bool*) – Inject damped Lya systems into the Lya forest These are done according to the current best estimates for the incidence dN/dz (Prochaska et al. 2008, ApJ, 675, 1002) Set in calc_lz These are *not* inserted according to overdensity along the sightline
- **nocolorcuts** (*bool, optional*) – Do not apply the fiducial rzW1W2 color-cuts cuts (default True).

Returns (flux, wave, meta, dla_meta) where:

- flux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (erg/s/cm2/A).

- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum with columns defined in desisim.io.empty_metatable *plus* RA, DEC.
- objmeta (astropy.Table): Table of additional object-specific meta-data [nmodel] for each output spectrum with columns defined in desisim.io.empty_metatable.
- dla_meta (astropy.Table): Table of meta-data [ndla] for the DLAs injected into the spectra. Only returned if add_dlas=True

Note: *dla_meta* is only included if add_dlas=True.

desisim.lya_spectra.**read_lya_skewers**(*lyafile*, *indices=None*, *read_dlas=False*, *add_metals=False*)

Reads Lyman alpha transmission skewers (from CoLoRe, format v2.x.y)

Parameters **lyafile** – full path to input FITS filename

Options: indices: indices of input file to sub-select read_dlas: try read DLA HDU from file add_metals: try to read metals HDU and multiply transmission

Returns wave[nwave] transmission[nlya, nwave] metadata[nlya] dlas[ndla] (if read_dlas=True, otherwise None)

Input file must have WAVELENGTH, TRANSMISSION, and METADATA HDUs

2.3.9 desisim.obs

Utility functions related to simulating observations for DESI

desisim.obs.**get_next_expid**(*n=None*)

Return the next exposure ID to use from {proddir}/etc/next_expid.txt and update the exposure ID in that file.

Use file locking to prevent multiple readers from getting the same ID or accidentally clobbering each other while writing.

Parameters **n** (*int, optional*) – number of contiguous expids to return as a list. If None, return a scalar. Note that n=1 returns a list of length 1.

BUGS:

- if etc/next_expid.txt doesn't exist, initial file creation is probably not threadsafe.
- File locking mechanism doesn't work on NERSC Edison, to turned off for now.

desisim.obs.**get_next_tileid**(*program='DARK'*)

Return tileid of next tile to observe

Parameters **program** (*optional*) – dark, gray, or bright

Note: Simultaneous calls will return the same tileid; it does *not* reserve the tileid.

desisim.obs.**get_night**(*t=None*, *utc=None*)

Return YEARMMDD for tonight. The night roles over at local noon. i.e. 1am and 11am is the previous date; 1pm is the current date.

Parameters

- **t** – local time.struct_time tuple of integers (year, month, day, hour, min, sec, weekday, dayofyear, DSTflag) default is time.localtime(), i.e. now
- **utc** – time.struct_time tuple for UTC instead of localtime

---

Note: this only has one second accuracy; good enough for sims but *not* to be used for actual DESI ops.

---

desisim.obs.**new_exposure**(*program*, *nspec=5000*, *night=None*, *expid=None*, *tileid=None*, *nproc=None*, *seed=None*, *obsconditions=None*, *specify_targets={}*, *testslit=False*, *exptime=None*, *arc_lines_filename=None*, *flat_spectrum_filename=None*, *outdir=None*, *overwrite=False*)

Create a new exposure and output input simulation files. Does not generate pixel-level simulations or noisy spectra.

Parameters

- **program** (*str*) – 'arc', 'flat', 'bright', 'dark', 'bgs', 'mws', ...
- **nspec** (*int, optional*) – number of spectra to simulate
- **night** (*str, optional*) – YEARMMDD string
- **expid** (*int, optional*) – positive integer exposure ID
- **tileid** (*int, optional*) – integer tile ID
- **nproc** (*object, optional*) – What does this do?
- **seed** (*int, optional*) – random seed
- **obsconditions** (*str or dict-like, optional*) – see options below
- **specify_targets** (*dict of dicts, optional*) – Define target properties like magnitude and redshift for each target class. Each objtype has its own key,value pair see simspec.templates.specify_galparams_dict() or simsepc.templates.specify_starparams_dict()
- **testslit** (*bool, optional*) – simulate test slit if True, default False; only for arc/flat
- **exptime** (*float, optional*) – exposure time [seconds], overrides obsconditions['EXPTIME']
- **arc_lines_filename** (*str, optional*) – use alternate arc lines filename (used if program="arc")
- **flat_spectrum_filename** (*str, optional*) – use alternate flat spectrum filename (used if program="flat")
- **outdir** (*str, optional*) – output directory
- **overwrite** (*bool, optional*) – optionally clobber existing files

Returns sim, fibermap, meta, obsconditions, objmeta

Return type science

Writes to outdir or $DESI_SPECTRO_SIM/$PIXPROD/{night}/

- fibermap-{expid}.fits
- simspec-{expid}.fits

input obsconditions can be a string 'dark', 'gray', 'bright', or dict-like observation metadata with keys SEEING (arcsec), EXPTIME (sec), AIRMASS, MOONFRAC (0-1), MOONALT (deg), MOONSEP (deg). Output obsconditions is is expanded dict-like structure.

program is used to pick the sky brightness, and is propagated to desisim.targets.sample_objtype() to get the correct distribution of targets for a given program, e.g. ELGs, LRGs, QSOs for program='dark'.

if program is 'arc' or 'flat', then *sim* is truth table with keys FLUX and WAVE; and meta=None and obsconditions=None.

Also see simexp.simarc(), .simflat(), and .simscience(), the last of which simulates a science exposure given surveysim obsconditions input, fiber assignments, and pre-generated mock target spectra.

desisim.obs.**specter_objtype**(*desitype*)
Convert a list of DESI object types into ones that specter knows about

desisim.obs.**update_obslog**(*obstype='science'*, *program='DARK'*, *expid=None*, *dateobs=None*, *tileid=-1*, *ra=None*, *dec=None*)
Update obslog with a new exposure

obstype : 'arc', 'flat', 'bias', 'test', 'science', ... program : 'DARK', 'GRAY', 'BRIGHT', 'CALIB' expid : integer exposure ID, default from get_next_expid() dateobs : time.struct_time tuple; default time.localtime() tileid : integer TileID, default -1, i.e. not a DESI tile ra, dec : float (ra, dec) coordinates, default tile ra,dec or (0,0)

returns tuple (expid, dateobs)

TODO: normalize obstype vs. program; see desisim issue #97

2.3.10 desisim.pixelsplines

Pixel-integrated spline utilities.

Written by A. Bolton, U. of Utah, 2010-2013.

exception desisim.pixelsplines.**PixSplineError**(*value*)

class desisim.pixelsplines.**PixelSpline**(*pixbound*, *flux*)
Pixel Spline object class.

Initialize as follows: PS = PixelSpline(pixbound, flux) where pixbound = array of pixel boundaries in baseline units and flux = array of specific flux values in baseline units.

Assumptions: 'pixbound' should have one more element than 'flux', and units of 'flux' are -per-unit-baseline, for the baseline units in which pixbound is expressed, averaged over the extent of each pixel.

point_evaluate(*xnew*, *missing=0.0*)
Evaluate underlying pixel spline at array of points BUG: input currently needs to be at least 1D array.

resample(*pb_new*)
Method to resample a pixelspline analytically onto a new set of pixel boundaries.

class desisim.pixelsplines.**WeightedRebinCoadder**(*fluxes*, *invvars*, *pixbounds*)
Objet class for weighted rebinning and coaddition of spectra

Initialize as follows: WRC = WeighedRebinCoadder(fluxes, invvars, pixbounds) where fluxes = list of arrays of specific flux values invvars = list of arrays of associated inverse variances pixbounds = list of arrays of pixel boundaries in baseline units

desisim.pixelsplines.**cen2bound**(*pixelcen*)
Convenience function to do the obvious thing to transform pixel centers to pixel boundaries.

desisim.pixelsplines.**compute_duck_slopes**(*pixbound*, *flux*)
Compute the slope of the illuminating quadratic spline at the locations of the 'ducks', i.e., the pixel boundaries, given the integrated flux per unit baseline within the pixels.

ARGUMENTS: pixbound: (npix + 1) ndarray of pixel boundaries, in units of wavelength or log-wavelength or frequency or whatever you like. flux: (npix) ndarray of spectral flux (energy or counts) per abscissa unit, averaged over the extent of the pixel

RETURNS: an (npix+1) ndarray of the slope of the underlying/illuminating flux per unit abscissa spectrum at the position of the pixel boundaries, a.k.a. 'ducks'. The end conditions are taken to be zero slope, so the exterior points of the output are zeros.

desisim.pixelsplines.**gauss_blur_matrix**(*pixbound*, *sig_conv*)

Function to generate a Gaussian blurring matrix for a pixelized spectrum, from specified pixel boundaries and 'sigma' vector. The matrix will be flux-conserving if the spectrum to which it is applied has units of 'counts per unit x', and pixbound and sig_conv both have units of x.

pixbound should have one more element than sig_conv.

Output is a scipy sparse matrix that can implement the blurring as: blurflux = gauss_blur_matrix * flux where 'flux' has the same dimensions as 'sig_conv'.

2.3.11 desisim.pixsim

Tools for DESI pixel level simulations using specter

desisim.pixsim.**_project**(*args*)

Helper function to project photons onto a subimage

Parameters of [psf, wave, phot, specmin] (*tuple/array*) –

Returns (xyrange, subimage) such that xmin, xmax, ymin, ymax = xyrange image[ymin:ymax, xmin:xmax] += subimage

desisim.pixsim.**get_nodes_per_exp**(*nnodes*, *nexposures*, *ncameras*, *user_nodes_per_comm_exp=None*)

Calculate how many nodes to use per exposure

Parameters

- **nnodes** – number of nodes in MPI COMM_WORLD (not number of ranks)
- **nexposures** – number of exposures to process
- **ncameras** – number of cameras per exposure
- **user_nodes_per_comm_exp** (*int, optional*) – user override of number of nodes to use; used to check requirements

Returns number of nodes to include in sub-communicators used to process individual exposures

Notes

- Uses the largest number of nodes per exposure that will still result in efficient node usage
- requires that (nexposures*ncameras) / nnodes = int
- the derived nodes_per_comm_exp * nexposures / nodes = int
- See desisim.test.test_pixsim.test_get_nodes_per_exp() for examples
- if user_nodes_per_comm_exp is given, requires that GreatestCommonDivisor(nnodes, ncameras) / user_nodes_per_comm_exp = int

desisim.pixsim.**mpi_count_nodes**(*comm*)
Return the number of nodes in this communicator

desisim.pixsim.**mpi_split_by_node**(*comm*, *nodes_per_communicator*)
Split an MPI communicator into sub-communicators with integer numbers of nodes per communicator

Parameters

- **comm** – MPI communicator
- **nodes_per_communicator** – number of nodes per sub-communicator

Returns MPI sub-communicator, node_index, total_num_nodes

Notes

- total number of nodes in original communicator must be an integer multiple of nodes_per_communicator
- if comm is split into N sub-communicators, node_index is the index of which of the N is returned for this rank
- total_num_nodes = number of nodes in original communicator

desisim.pixsim.**parallel_project**(*psf*, *wave*, *phot*, *specmin=0*, *ncpu=None*, *comm=None*)
Using psf, project phot[nspec, nw] vs. wave[nw] onto image

Return 2D image

desisim.pixsim.**photpix2raw**(*phot*, *gain=1.0*, *readnoise=3.0*, *offset=None*, *nprescan=7*, *noverscan=50*, *readorder='lr'*, *noisydata=True*)
Add prescan, overscan, noise, and integerization to an image

Parameters

- **phot** – 2D float array of mean input photons per pixel
- **gain** (*float, optional*) – electrons/ADU
- **readnoise** (*float, optional*) – CCD readnoise in electrons
- **offset** (*float, optional*) – bias offset to add
- **nprescan** (*int, optional*) – number of prescan pixels to add
- **noverscan** (*int, optional*) – number of overscan pixels to add
- **readorder** (*str, optional*) – 'lr' or 'rl' to indicate readout order 'lr' : add prescan on left and overscan on right of image 'rl' : add prescan on right and overscan on left of image
- **noisydata** (*boolean, optional*) – if True, don't add noise, e.g. because input signal already had noise from a cosmics image

Returns 2D integer ndarray: image = int((poisson(phot) + offset + gauss(readnoise))/gain)

Integerization happens twice: the mean photons are poisson sampled into integers, but then offets, readnoise, and gain are applied before resampling into ADU integers

This is intended to be used per-amplifier, not for an entire CCD image.

desisim.pixsim.**simulate**(*camera*, *simspec*, *psf*, *nspec=None*, *ncpu=None*, *cosmics=None*, *wavemin=None*, *wavemax=None*, *preproc=True*, *comm=None*)
Run pixel-level simulation of input spectra

Parameters

- **camera** (*string*) – b0, r1, .. z9
- **simspec** – desispec.io.SimSpec object from desispec.io.read_simspec()
- **psf** – subclass of specter.psf.psf.PSF, e.g. from desimodel.io.load_psf()

Options: nspec (int): number of spectra to simulate ncpu (int): number of CPU cores to use in parallel cosmics (desispec.image.Image): e.g. from desisim.io.read_cosmics() wavemin (float): minimum wavelength range to simulate wavemax (float): maximum wavelength range to simulate preproc (boolean, optional) : also preprocess raw data (default True)

Returns

(image, rawpix, truepix) tuple, where image is the preproc Image object (only header is meaningful if preproc=False), rawpix is a 2D ndarray of unprocessed raw pixel data, and truepix is a 2D ndarray of truth for image.pix

desisim.pixsim.**simulate_exposure**(*simspecfile*, *rawfile*, *cameras=None*, *ccdshape=None*, *simpixfile=None*, *addcosmics=None*, *comm=None*, ***kwargs*)

Simulate frames from an exposure, including I/O

Parameters

- **simspecfile** – input simspec format file with spectra
- **rawfile** – output raw data file to write

Options: cameras: str or list of str, e.g. b0, r1, .. z9 ccdshape: (npix_y, npix_x) primarily used to limit memory while testing simpixfile: output file for noiseless truth pixels addcosmics: if True (must be specified via command input), add cosmics from real data comm: MPI communicator object

Additional keyword args are passed to pixsim.simulate()

For a lower-level pixel simulation interface that doesn't perform I/O, see pixsim.simulate()

Note: call desi_preproc or desispec.preproc.preproc to pre-process the output desi*.fits file for overscan subtraction, noise estimation, etc.

2.3.12 desisim.qso_template

Stuff dealing with QSO templates.

2.3.13 desisim.qso_template.desi_qso_templ

Module for Fitting PCA to the BOSS QSOs

01-Dec-2014 by JXP

desisim.qso_template.desi_qso_templ.**chk_desi_qso_templates**(*infil=None*, *outfil=None*, *N_perz=100*)

desisim.qso_template.desi_qso_templ.**desi_qso_templates**(*z_wind=0.2*, *zmnx=(0.4, 4.0)*, *outfil=None*, *N_perz=500*, *boss_pca_fil=None*, *wvmnx=(3500.0, 10000.0)*, *rebin_wave=None*, *rstate=None*, *sdss_pca_fil=None*, *no_write=False*, *redshift=None*, *seed=None*, *old_read=False*, *ipad=40*, *cosmo=None*)

Generate QSO templates for DESI

Rebins to input wavelength array (or log10 in wvmnx)

Parameters

- **z_wind** (*float, optional*) – Window for sampling PCAs
- **zmnx** (*tuple, optional*) – Min/max for generation
- **N_perz** (*int, optional*) – Number of draws per redshift window
- **old_read** (*bool, optional*) – Read the files the old way
- **seed** (*int, optional*) – Seed for the random number state
- **rebin_wave** (*ndarray, optional*) – Input wavelengths for rebinning
- **wvmnx** (*tuple, optional*) – Wavelength limits for rebinning (not used with rebin_wave)
- **redshift** (*ndarray, optional*) – Redshifts desired for the templates
- **ipad** (*int, optional*) – Padding for enabling enough models
- **cosmo** (*astropy.cosmology.core, optional*) – Cosmology inistantiation from astropy.cosmology.code

Returns

- **wave** (*ndarray*) – Wavelengths that the spectra were rebinned to
- **flux** (*ndarray (2D; flux vs. model)*)
- **z** (*ndarray*) – Redshifts

desisim.qso_template.desi_qso_templ.**fig_desi_templ_z_i**(*outfil=None*, *boss_fil=None*, *flg=0*)

flg = 0: Redshift flg = 1: imag

desisim.qso_template.desi_qso_templ.**mean_templ_zi**(*zimag*, *debug=False*, *i_wind=0.1*, *z_wind=0.05*, *boss_pca_fil=None*)

Generate 'mean' templates at given z,i

Parameters

- **zimag** (*list of tuples*) – Redshift, imag pairs for the templates
- **i_wind** (*float (0.1 mag)*) – Window for smoothing imag
- **z_wind** (*float (0.05 mag)*) – Window for smoothing redshift

desisim.qso_template.desi_qso_templ.**repackage_coeff**(*boss_pca_fil=None*, *sdss_pca_fil=None*, *outfil='qso_templates_v2.0.fits'*)
Repackage the coefficients and redshifts into a single FITS file

Returns

desisim.qso_template.desi_qso_templ.**tst_random_set**()
Generate a small set of random templates for testing :return:

2.3.14 desisim.qso_template.fit_boss_qsos

Module for Fitting PCA to the BOSS QSOs

01-Dec-2014 by JXP

desisim.qso_template.fit_boss_qsos.**do_boss_lya_parallel**(*istart*, *iend*, *cut_Lya*, *output*, *debug=False*)
Generate PCA coeff for the BOSS Lya DR10 dataset, v2.1

Parameters **cut_Lya** (*boolean (True)*) – Avoid using the Lya forest in the analysis

desisim.qso_template.fit_boss_qsos.**do_sdss_lya_parallel**(*istart*, *iend*, *cut_Lya*, *output*, *debug=False*)
Generate PCA coeff for the SDSS DR7 dataset, 0.5<z<2

Parameters **cut_Lya** (*boolean (True)*) – Avoid using the Lya forest in the analysis

desisim.qso_template.fit_boss_qsos.**failed_parallel**()
Collision with np.dot Might fix with OPENBLAS_NUM_THREADS=1

desisim.qso_template.fit_boss_qsos.**fit_eigen**(*flux*, *ivar*, *eigen_flux*)
Fit the spectrum with the eigenvectors. Pass back the coefficients

desisim.qso_template.fit_boss_qsos.**read_qso_eigen**(*eigen_fil=None*)
Input the QSO Eigenspectra

desisim.qso_template.fit_boss_qsos.**splice_fits**(*flg=0*)
Splices together the various PCA fits for SDSS or BOSS

flg: int (0) 0=BOSS, 1=SDSS

2.3.15 desisim.qso_template.qso_pca

Module for generate QSO PCA templates

24-Nov-2014 by JXP

desisim.qso_template.qso_pca.**do_boss_lya_parallel**(*istart*, *iend*, *output*, *debug=False*, *cut_Lya=True*)
Generate PCA coeff for the BOSS Lya DR10 dataset, v2.1

Parameters **cut_Lya** (*boolean (True)*) – Avoid using the Lya forest in the analysis

desisim.qso_template.qso_pca.**fit_eigen**(*flux*, *ivar*, *eigen_flux*)
Fit the spectrum with the eigenvectors. Pass back the coefficients

desisim.qso_template.qso_pca.**read_qso_eigen**(*eigen_fil=None*)
Input the QSO Eigenspectra

2.3.16 desisim.qso_template.tests

Module for generate QSO PCA templates

24-Nov-2014 by JXP

I don't think the documentation of this module is correct.

2.3.17 desisim.quickcat

Code for quickly generating an output zcatalog given fiber assignment tiles, a truth catalog, and optionally a previous zcatalog.

desisim.quickcat.**get_median_obsconditions**(*tileids*)

Gets the observational conditions for a set of tiles.

Parameters **tileids** – list of tileids that were observed

Returns

Table with the observational conditions for every tile.

It inclues at least the following columns:

```
'TILEID': array of tile IDs
'AIRMASS': array of airmass values on a tile
'EBMV': array of E(B-V) values on a tile
'LINTRANS': array of atmospheric transparency during spectro obs;␣
↪floats [0-1]
'MOONFRAC': array of moonfraction values on a tile.
'SEEING': array of FWHM seeing during spectroscopic observation on a␣
↪tile.
```

desisim.quickcat.**get_observed_redshifts**(*targets*, *truth*, *targets_in_tile*, *obsconditions*, *parameter_filename=None*, *ignore_obscondition=False*)

Returns observed z, zerr, zwarn arrays given true object types and redshifts

Parameters

- **targets** – target catalog table; currently used only for target mask bits
- **truth** – truth table with OIIFLUX, TRUEZ
- **targets_in_tile** – dictionary. Keys correspond to tileids, its values are the arrays of targetids observed in that tile.
- **obsconditions** – table observing conditions with columns 'TILEID': array of tile IDs 'AIRMASS': array of airmass values on a tile 'EBMV': array of E(B-V) values on a tile 'LINTRANS': array of atmospheric transparency during spectro obs; floats [0-1] 'MOONFRAC': array of moonfraction values on a tile. 'SEEING': array of FWHM seeing during spectroscopic observation on a tile.
- **parameter_filename** – yaml file with quickcat parameters
- **ignore_obscondition** – if True, no variation of efficiency with obs. conditions (adjustment of exposure time should correct for mean change of S/N)

Returns tuple of (zout, zerr, zwarn)

desisim.quickcat.**get_redshift_efficiency**(*simtype*, *targets*, *truth*, *targets_in_tile*, *obsconditions*, *params*, *ignore_obscondition=False*)

Simple model to get the redshift effiency from the observational conditions or observed magnitudes+redshuft

Parameters

- **simtype** – ELG, LRG, QSO, MWS, BGS
- **targets** – target catalog table; currently used only for TARGETID
- **truth** – truth table with OIIFLUX, TRUEZ
- **targets_in_tile** – dictionary. Keys correspond to tileids, its values are the arrays of targetids observed in that tile.
- **obsconditions** – table observing conditions with columns 'TILEID': array of tile IDs 'AIRMASS': array of airmass values on a tile 'EBMV': array of E(B-V) values on a tile 'LINTRANS': array of atmospheric transparency during spectro obs; floats [0-1] 'MOONFRAC': array of moonfraction values on a tile. 'SEEING': array of FWHM seeing during spectroscopic observation on a tile.
- **parameter_filename** – yaml file with quickcat parameters
- **ignore_obscondition** – if True, no variation of efficiency with obs. conditions (adjustment of exposure time should correct for mean change of S/N)

Returns

tuple of arrays (observed, p) both with same length as targets

observed: boolean array of whether the target was observed in these tiles

p: probability to get this redshift right

desisim.quickcat.**get_zeff_obs**(*simtype*, *obsconditions*)

desisim.quickcat.**quickcat**(*tilefiles*, *targets*, *truth*, *zcat=None*, *obsconditions=None*, *perfect=False*)

Generates quick output zcatalog

Parameters

- **tilefiles** – list of fiberassign tile files that were observed
- **targets** – astropy Table of targets
- **truth** – astropy Table of input truth with columns TARGETID, TRUEZ, and TRUETYPE
- **zcat** (*optional*) – input zcatalog Table from previous observations
- **obsconditions** (*optional*) – Table or ndarray with observing conditions from surveysim
- **perfect** (*optional*) – if True, treat spectro pipeline as perfect with input=output, otherwise add noise and zwarn!=0 flags

Returns zcatalog astropy Table based upon input truth, plus ZERR, ZWARN, NUMOBS, and TYPE columns

desisim.quickcat.**reverse_dictionary**(*a*)

Inverts a dictionary mapping.

Parameters **a** – input dictionary.

Returns output reversed dictionary.

Return type b

2.3.18 desisim.quicksurvey

Code for quickly simulating the survey results given a mock catalog and a list of tile epochs to observe.

Directly depends on the following DESI products:

- desitarget.mtl
- *desisim.quickcat*
- fiberassign

class desisim.quicksurvey.**SimSetup**(*output_path*, *targets_path*, *fiberassign*, *exposures*, *fiberassign_dates*)

Setup to simulate the DESI survey

output_path
Path to write the outputs.x

Type str

targets_path
Path where the files targets.fits can be found

Type str

epochs_path
Path where the epoch files can be found.

Type str

fiberassign
Name of the fiberassign script

Type str

template_fiberassign
Filename of the template input for fiberassign

Type str

n_epochs
number of epochs to be simulated.

Type int

backup_epoch_data(*epoch_id=0*)
Deletes files in the temporary output directory

Parameters **epoch_id** (*int*) – Epoch's ID to backup/copy from the output directory.

cleanup_directories()
Deletes files in the temporary output directory

create_directories()
Creates output directories to store simulation results.

create_surveyfile(*epoch*)
Creates text file of tiles survey_list.txt to be used by fiberassign

Parameters **epoch** (*int*) – epoch of tiles to write

Notes

The file is written to the temporary directory in self.tmp_output_path

epoch_data_exists(*epoch_id=0*)
Check epoch directory for zcat.fits and mtl.fits files.

simulate()
Simulate the DESI setup described by a SimSetup object.

simulate_epoch(*epoch*, *truth*, *targets*, *perfect=False*, *zcat=None*)
Core routine simulating a DESI epoch,

Parameters

- **epoch** (*int*) – epoch to simulate
- **perfect** (*boolean*) – Default: False. Selects whether how redshifts are taken from the truth file. True: redshifts are taken without any error from the truth file. False: redshifts include uncertainties.
- **truth** (*Table*) – Truth data
- **targets** (*Table*) – Targets data
- **zcat** (*Table*) – Redshift Catalog Data

Notes

This routine simulates three steps: * Merged target list creation * Fiber allocation * Redshift catalogue construction

update_observed_tiles(*epoch*)
Creates the list of tilefiles to be gathered to buikd the redshift catalog.

2.3.19 desisim.scripts

Main functions and command-line parsing.

2.3.20 desisim.scripts.brightsims

Generate a canonical set of bright-time simulations.

desisim.scripts.fastframe.**main**(*args=None*)
Converts simspec -> frame files; see fastframe –help for usage options

desisim.scripts.newarc.**main**(*args=None*)
TODO: document

Note: this bypasses specsim since we don't have an arclamp model in surface brightness units; we only have electrons on the CCD

desisim.scripts.newflat.**main**(*args=None*)
Generates a new flat exposure; see newflat –help for usage options

2.3.21 desisim.scripts.pixsim_nights

This is a module.

2.3.22 desisim.scripts.pixsim

This is a module.

desisim.scripts.pixsim.**expand_args**(*args*)
expand camera string into list of cameras

Generate S/N plots as a function of object type for the current production

Read fibermaps and zbest files to generate QA related to redshifts and compare against the 'true' values

2.3.23 desisim.scripts.quickgen

Quickgen quickly simulates pipeline outputs if given input files.

- must provide simspec and fibermap files via newexp script
- Number of spectra to be simulated can be given as an argument for quickgen, but the number of spectra in the simspec file is taken by default
- For this option, airmass and exposure time are keywords given to newexp
- The keywords provided in the examples are all required, additional keywords are provided below
- Collect a set of templates to simulate as a new exposure:

```
newexp --nspec 500 --night 20150915 --expid 0 --flavor dark
```

- newexp keyword arguments:

```
--flavor : arc/flat/dark/gray/bright/bgs/mws/elg/lrg/qso, type=str, default='dark'
--tileid : tile id, type=int
--expid : exposure id, type=int
--exptime : exposure time in seconds, default for arc = 5s, flat = 10s, dark/elg/
↪lrg/qso=1000s, bright/bgs/mws=300s, type=int
--night : YEARMMDD, type=str
--nspec : Number of spectra to simulate, type=int, default=5000
--airmass : type=float, default=1.0
--seed : random number seed, type=int
--testslit : test slit simulation with fewer fibers, action="store_true"
--arc-lines : alternate arc lines filename, type=str, default=None
--flat-spectrum : alternate flat spectrum filename, type=str
```

- Actually do the simulation:

```
simdir=$DESI_SPECTRO_SIM/$PIXPROD/20150915
quickgen --simspec $simdir/simspec-00000000.fits --fibermap $simdir/fibermap-
↪00000000.fits
```

- quickgen output (can also provide frame file only as keyword):

```
1. frame-{camera}-{expid}.fits : raw extracted photons with no calibration at all
2. sky-{camera}-{expid}.fits : the sky model in photons
3. fluxcalib-{camera}-{expid}.fits] : the flux calibration vectors
4. cframe-{camera}-{expid}.fits : flux calibrated spectra

These files are written to $simdir/{expid}
```

- nspec, config, seed, moon-phase, moon-angle, moon-zenith

- simspec, fibermap, nstart, spectrograph, frameonly zrange-qso, zrange-elg, zrange-lrg, zrange-bgs, snerfluxratiorange, add-SNeIa

desisim.scripts.quickgen.**_add_truth**(*hdus*, *header*, *meta*, *trueflux*, *sflux*, *wave*, *channel*)
Utility function for adding truth to an output FITS file.

desisim.scripts.quickspectra.**sim_spectra**(*wave*, *flux*, *program*, *spectra_filename*, *obsconditions=None*, *sourcetype=None*, *targetid=None*, *redshift=None*, *expid=0*, *seed=0*, *skyerr=0.0*, *ra=None*, *dec=None*, *meta=None*, *fibermap_columns=None*, *fullsim=False*, *use_poisson=True*)
Simulate spectra from an input set of wavelength and flux and writes a FITS file in the Spectra format that can be used as input to the redshift fitter.

Parameters

- **wave** – 1D np.array of wavelength in Angstrom (in vacuum) in observer frame (i.e. redshifted)
- **flux** – 1D or 2D np.array. 1D array must have same size as wave, 2D array must have shape[1]=wave.size flux has to be in units of 10^-17 ergs/s/cm2/A
- **spectra_filename** – path to output FITS file in the Spectra format
- **program** – dark, lrg, qso, gray, grey, elg, bright, mws, bgs ignored if obsconditions is not None

Optional: obsconditions : dictionnary of observation conditions with SEEING EXPTIME AIRMASS MOONFRAC MOONALT MOONSEP sourcetype : list of string, allowed values are (sky,elg,lrg,qso,bgs,star), type of sources, used for fiber aperture loss , default is star targetid : list of targetids for each target. default of None has them generated as str(range(nspec)) redshift : list/array with each index being the redshifts for that target expid : this expid number will be saved in the Spectra fibermap seed : random seed skyerr : fractional sky subtraction error ra : numpy array with targets RA (deg) dec : numpy array with targets Dec (deg) meta : dictionnary, saved in primary fits header of the spectra file fibermap_columns : add these columns to the fibermap fullsim : if True, write full simulation data in extra file per camera use_poisson : if False, do not use numpy.random.poisson to simulate the Poisson noise. This is useful to get reproducible random realizations.

desisim.simexp.**_calib_screen_uniformity**(*theta=None*, *radius=None*)
Returns calibration screen relative non-uniformity as a function of theta (degrees) or focal plane radius (mm)

desisim.simexp.**_specsim_config_for_wave**(*wave*, *dwave_out=None*, *specsim_config_file='desi'*)
Generate specsim config object for a given wavelength grid

Parameters **wave** – array of linearly spaced wavelengths in Angstroms

Options:

specsim_config_file: (str) path to DESI instrument config file. default is desi config in specsim package.

Returns specsim Configuration object with wavelength parameters set to match this input wavelength grid

desisim.simexp.**fibermeta2fibermap**(*fiberassign*, *meta*)
Convert a fiberassign + targeting metadata table into a fibermap Table

A future refactor will standardize the column names of fiber assignment, target catalogs, and fibermaps, but in the meantime this is needed.

desisim.simexp.**get_mock_spectra**(*fiberassign*, *mockdir=None*, *nside=64*)

Parameters **fiberassign** – table loaded from fiberassign tile file

Returns (flux, wave, meta) tuple

desisim.simexp.**get_source_types**(*fibermap*)

Return a list of specsim source types based upon fibermap['DESI_TARGET']

Parameters **fibermap** – fibermap Table including DESI_TARGET column

Returns array of source_types 'sky', 'elg', 'lrg', 'qso', 'star'

Unassigned fibers fibermap['TARGETID'] == -1 will be treated as 'sky'

If fibermap.meta['FLAVOR'] = 'arc' or 'flat', returned source types will match that flavor, though specsim doesn't use those as source_types

TODO: specsim/desimodel doesn't have a fiber input loss model for BGS yet, so BGS targets get source_type = 'lrg' (!)

desisim.simexp.**read_fiberassign**(*tilefile_or_id*, *indir=None*)

Returns fiberassignment table for tileid

Parameters **tilefile_or_id** (*int or str*) – tileid (int) or full path to tile file (str)

Returns fiberassignment Table from HDU 1

desisim.simexp.**read_mock_spectra**(*truthfile*, *targetids*, *mockdir=None*)

Reads mock spectra from a truth file

Parameters

- **truthfile** (*str*) – full path to a mocks truth-*.fits file
- **targetids** (*array-like*) – targetids to load from that file
- **mockdir** – ???

Returns (flux, wave, truth) tuples: flux[nspec, nwave]: flux in 1e-17 erg/s/cm2/Angstrom wave[nwave]: wavelengths in Angstroms truth[nspec]: metadata truth table

desisim.simexp.**simarc**(*arcdata*, *nspec=5000*, *nonuniform=False*, *testslit=False*)

Simulates an arc lamp exposure

Parameters

- **arcdata** (*Table*) – Table with columns VACUUM_WAVE and ELECTRONS
- **nspec** (*int, optional*) –
- **nonuniform** (*bool, optional*) – include calibration screen non-uniformity
- **testslit** (*bool, optional*) – this argument is undocumented.

Returns: (wave, phot, fibermap) wave: 1D[nwave] wavelengths in Angstroms phot: 2D[nspec,nwave] photons observed by CCD (i.e. electrons) fibermap: fibermap Table

Note: this bypasses specsim since we don't have an arclamp model in surface brightness units; we only have electrons on the CCD. But it does include the effect of varying fiber sizes.

desisim.simexp.**simflat**(*flatfile*, *nspec=5000*, *nonuniform=False*, *exptime=10*, *testslit=False*, *psfconvolve=True*, *specsim_config_file='desi'*)

Simulates a flat lamp calibration exposure

Parameters

- **flatfile** (*str*) – filename with flat lamp spectrum data
- **nspec** (*int, optional*) – number of spectra to simulate
- **nonuniform** (*bool, optional*) – include calibration screen non-uniformity
- **exptime** (*float, optional*) – exposure time in seconds
- **psfconvolve** (*bool, optional*) – passed to simspec.simulator.Simulator camera_output. if True, convolve with PSF and include per-camera outputs
- **specsim_config_file** (*str, optional*) – path to DESI instrument config file. default is desi config in specsim package.

Returns: (sim, fibermap) sim: specsim Simulator object fibermap: fibermap Table

desisim.simexp.**simscience**(*targets*, *fiberassign*, *obsconditions='DARK'*, *expid=None*, *nspec=None*, *psfconvolve=True*)

Simulates a new DESI exposure from surveysim+fiberassign+mock spectra

Parameters

- **targets** (*tuple*) – tuple of (flux[nspec,nwave], wave[nwave], meta[nspec])
- **fiberassign** (*Table*) – fiber assignments table
- **obsconditions** (*object, optional*) – observation metadata as

  str: DARK (default) or GRAY or BRIGHT

  dict or row of Table with keys:

```
SEEING (arcsec), EXPTIME (sec), AIRMASS,
MOONFRAC (0-1), MOONALT (deg), MOONSEP (deg)
```

  Table including EXPID for subselection of which row to use filename with obsconditions Table; expid must also be set
- **expid** (*int, optional*) – exposure ID
- **nspec** (*int, optional*) – number of spectra to simulate
- **psfconvolve** (*bool, optional*) – passed to simspec.simulator.Simulator camera_output. if True, convolve with PSF and include per-camera outputs

Returns: (sim, fibermap, meta) sim: specsim.simulate.Simulator object fibermap: Table meta: target metadata truth table

See obs.new_exposure() for function to generate new random exposure, independent from surveysim, fiberassignment, and pre-generated mocks.

desisim.simexp.**simulate_spectra**(*wave*, *flux*, *fibermap=None*, *obsconditions=None*, *redshift=None*, *dwave_out=None*, *seed=None*, *psfconvolve=True*, *specsim_config_file='desi'*)

Simulates an exposure without reading/writing data files

Parameters

- **wave** (*array*) – 1D wavelengths in Angstroms
- **flux** (*array*) – 2D[nspec,nwave] flux in 1e-17 erg/s/cm2/Angstrom or astropy Quantity with flux units
- **fibermap** (*Table, optional*) – table from fiberassign or fibermap; uses X/YFOCAL_DESIGN, TARGETID, DESI_TARGET
- **obsconditions** (*dict-like, optional*) – observation metadata including SEEING (arcsec), EXPTIME (sec), AIRMASS, MOONFRAC (0-1), MOONALT (deg), MOONSEP (deg)
- **redshift** (*array-like, optional*) – list/array with each index being the redshifts for that target
- **seed** (*int, optional*) – random seed
- **psfconvolve** (*bool, optional*) – passed to simspec.simulator.Simulator camera_output. if True, convolve with PSF and include per-camera outputs
- **specsim_config_file** (*str, optional*) – path to DESI instrument config file. default is desi config in specsim package.

Returns A specsim.simulator.Simulator object

TODO: galsim support

desisim.simexp.**targets2truthfiles**(*targets*, *basedir*, *nside=64*)
Return list of mock truth files that contain these targets

Parameters

- **targets** – table with TARGETID column, e.g. from fiber assignment
- **basedir** – base directory under which files are found

Returns (truthfiles, targetids): truthfiles: list of truth filenames targetids: list of lists of targetids in each truthfile

i.e. targetids[i] is the list of targetids from targets['TARGETID'] that are in truthfiles[i]

2.3.24 desisim.spec_qa

Tools for DESI QA with simulations It does not cover cosmology simulations.

2.3.25 desisim.spec_qa.high_level

Module to run high_level QA on a given DESI run Written by JXP on 3 Sep 2015

desisim.spec_qa.high_level.**get_meta**()
Get META data on production

desisim.spec_qa.high_level.**main**()
Runs the process

2.3.26 desisim.spec_qa.redshifts

Module to run high_level QA on a given DESI run

Written by JXP on 3 Sep 2015

desisim.spec_qa.redshifts.**calc_dz**(*simz_tab*)
Calcualte deltaz/(1+z) for a given simz_tab

desisim.spec_qa.redshifts.**calc_dzsig**(*simz_tab*)
Calcualte deltaz/sig(z) for a given simz_tab

desisim.spec_qa.redshifts.**calc_obj_stats**(*simz_tab*, *objtype*)
Calculate redshift statistics for a given objtype

Parameters

- **simz_tab** (*Table*) – TODO: document this
- **objtype** (*str*) – Object type, e.g. 'ELG', 'LRG'

Returns **stat_dict** – Survey results for a given object type

Return type dict

desisim.spec_qa.redshifts.**criteria**(*simz_tab*, *objtype=None*, *dvlimit=None*)
Analyze the input table for various criteria

Parameters

- **simz_tab** (*Table*) –
- **objtype** (*str, optional -- Restrict analysis to a specific object type*) –

Returns

- **objtype_mask** (*ndarray*) – Match to input objtype (if any given)
- **z_mask** (*ndarray*) – Analyzed by the redshift analysis software
- **survey_mask** (*ndarray*) – Part of the DESI survey (not filler)
- **dv_mask** (*ndarray*) – Satisfies the dv criterion; Either specific to each objtype or using an input dvlimit
- **zwarn_mask** (*ndarray*) – ZWARN=0

desisim.spec_qa.redshifts.**dz_summ**(*simz_tab*, *outfile=None*, *pdict=None*, *min_count=20*)
Generate a summary figure comparing zfind to ztruth.

Parameters

- **simz_tab** (*Table*) – Table of redshift information.
- **pp** (*PdfPages object*) – This parameter is not documented.
- **pdict** (*dict*) – Guides the plotting parameters
- **min_count** (*int, optional*) – This parameter is not documented.

desisim.spec_qa.redshifts.**load_z**(*fibermap_files*, *zbest_files=None*, *outfil=None*)
Load input and output redshift values for a set of exposures

Parameters

- **fibermap_files** (*list*) – List of fibermap files; None of these should be calibration..

- **zbest_files** (*list, optional*) – List of zbest output files Slurped from fibermap info if not provided
- **outfil** (*str, optional*) – Output file for the table

Returns

- **simz_tab** (*astropy.Table*) – Merged table of simpsec data
- **zb_tab** (*astropy.Table*) – Merged table of zbest output

desisim.spec_qa.redshifts.**match_truth_z**(*simz_tab*, *zb_tab*, *mini_read=False*, *outfil=None*)
Match truth and zbest tables :param simz_tab: astropy.Table; Either generated from load_z() or read from disk via 'truth.fits' :param zb_tab: astropy.Table; Either generated from load_z() or read from disk via 'zcatalog-mini.fits' :param mini_read: bool, optional; Tables were read from the summary tables written to disk :param outfil: str, optional :return: simz_tab: modified in place

desisim.spec_qa.redshifts.**obj_fig**(*simz_tab*, *objtype*, *summ_stats*, *outfile=None*)
Generate QA plot for a given object type

desisim.spec_qa.redshifts.**obj_requirements**(*zstats*, *objtype*)
Assess where a given objtype passes the requirements Requirements from Doc 318 (August 2014)

Parameters

- **zstats** (*Object*) – This parameter is not documented.
- **objtype** (*str*) – Object type, e.g. 'ELG', 'LRG'

Returns Pass/fail dict

Return type dict

desisim.spec_qa.redshifts.**plot_slices**(*x*, *y*, *ok*, *bad*, *x_lo*, *x_hi*, *y_cut*, *num_slices=5*, *min_count=100*, *axis=None*)
Scatter plot with 68, 95 percentiles superimposed in slices.

Requires that the matplotlib package is installed.

Parameters

- **x** (*array of float*) – X-coordinates to scatter plot. Points outside [*x_lo*, *x_hi*] are not displayed.
- **y** (*array of float*) – Y-coordinates to scatter plot. Y values are assumed to be roughly symmetric about zero.
- **ok** (*array of bool*) – Array of booleans that identify which fits are considered good.
- **bad** (*array of bool*) – Array of booleans that identify which fits have failed catastrophically.
- **x_lo** (*float*) – Minimum value of *x* to plot.
- **x_hi** (*float*) – Maximum value of *x* to plot.
- **y_cut** (*float*) – The target maximum value of $|y|$. A dashed line at this value is added to the plot, and the vertical axis is clipped at $|y| = 1.25imesy_{cut}$ (but values outside this range are included in the percentile statistics).
- **num_slices** (*int*) – Number of equally spaced slices to divide the interval [*x_lo*, *x_hi*] into.
- **min_count** (*int*) – Do not use slices with fewer points for superimposed percentile statistics.
- **axis** (*matplotlib axis object or None*) – Uses the current axis if this is None.

desisim.spec_qa.redshifts.**slice_simz**(*simz_tab*, *objtype=None*, *z_analy=False*, *survey=False*, *catastrophic=False*, *goodz=False*, *all_zwarn0=False*, ***kwargs*)

Slice input simz_tab in one of many ways

Parameters

- **z_analy** (*bool, optional*) – redshift analysis required?
- **all_zwarn0** (*bool, optional*) – Ignores catastrophic failures in the slicing to return all sources with ZWARN==0
- **survey** (*bool, optional*) – Only include objects that satisfy the Survey requirements e.g. ELGs with sufficient OII_flux
- **catastrophic** (*bool, optional*) – Restrict to catastropic failures
- **goodz** (*bool, optional*) – Restrict to good redshifts
- **all_zwarn0** – Restrict to ZWARN=0 cases
- ****kwargs** (*passed to criteria*) –

Returns simz_table

Return type Table cut by input parameters

desisim.spec_qa.redshifts.**spectype_confusion**(*simz_tab*, *zb_tab=None*)

Generate a Confusion Matrix for spectral types See the Confusion_matrix_spectypes Notebook in docs/nb for an example

Parameters

- **simz_tab** (*Table*) – Truth table; may be input from truth.fits
- **zb_tab** (*Table (optional)*) – zcatalog/zbest table; may be input from zcatalog-mini.fits If provided, used to match the simz_tab to the zbest quantities

Returns

- **simz_tab** (*astropy.Table*) – Merged table of simpsec data
- **results** (*dict*) – Nested dict. First key is the TRUESPECTYPE Second key is the SPECTYPE e.g. results['QSO']['QSO'] reports the number of True QSO classified as QSO results['QSO']['Galaxy'] reports the number of True QSO classified as Galaxy

desisim.spec_qa.redshifts.**summ_fig**(*simz_tab*, *summ_tab*, *meta*, *outfile=None*)

Generate summary summ_fig :param simz_tab: :param summ_tab: :param meta: :param outfile: :return:

desisim.spec_qa.redshifts.**summ_stats**(*simz_tab*)

Generate summary stats

Parameters **simz_tab** (*Table*) – Table summarizing redshifts

Returns List of summary stat dicts

Return type lis

desisim.spec_qa.redshifts.**zstats**(*simz_tab*, *objtype=None*, *dvlimit=None*, *count=False*, *survey=False*)

Perform statistics on the input truth+z table good = Satisfies dv criteria and ZWARN==0 fail = Fails dv criteria with ZWARN==0 (catastrophic failures) miss = Satisfies dv criteria but ZWARN!=0 (missed opportunities) lost = Fails dv criteria and ZWARN!=0 (lost, but at least we knew it)

Parameters

- **simz_tab** –

- **objtype** –
- **dvlimit** – float, optional – Over-rides object specific dv limits
- **count** – bool, optional
- **survey** – bool, optional – Restrict to targets meeting the Survey criteria (e.g. ELG flux)

Returns just the raw counts of each category :: ngood, nfail, nmiss, nlost else: percentile of each relative to ntot, and ntot

Return type if count=True

2.3.27 desisim.spec_qa.s2n

Module to examine S/N in object spectra

desisim.spec_qa.s2n.**load_all_s2n_values**(*nights*, *channel*, *sub_exposures=None*)
Calculate S/N values for a set of spectra from an input list of nights

Parameters

- **nights** – list
- **channel** – str ('b','r','z')
- **sub_exposures** –

Returns

dict Contains all the S/N info for all nights in the given channel

Return type fdict

desisim.spec_qa.s2n.**load_s2n_values**(*objtype*, *nights*, *channel*, *sub_exposures=None*)
DEPRECATED

Calculate S/N values for a set of spectra

Parameters

- **objtype** – str
- **nights** – list
- **channel** – str
- **sub_exposures** –

Returns

dict Contains S/N info

Return type fdict

desisim.spec_qa.s2n.**obj_s2n_wave**(*s2n_dict*, *wv_bins*, *flux_bins*, *otype*, *outfile=None*, *ax=None*)
Generate QA of S/N for a given object type

desisim.spec_qa.s2n.**obj_s2n_z**(*s2n_dict*, *z_bins*, *flux_bins*, *otype*, *outfile=None*, *ax=None*)
Generate QA of S/N for a given object type vs. z (mainly for ELG)

desisim.spec_qa.s2n.**parse_s2n_values**(*objtype*, *fdict*)
Parse the input set of S/N measurements on objtype

Parameters

- **objtype** – str

- **fdict** – dict Contains all the S/N info for all nights in a given channel

Returns

dict Contains all the S/N info for the given objtype

Return type pdict

2.3.28 desisim.spec_qa.redshifts

Module to run high_level QA on a given DESI run

Written by JXP on 3 Sep 2015

desisim.spec_qa.utils.**catastrophic_dv**(*objtype*)
Pass back catastrophic velocity limit for given objtype From DESI document 318 (August 2014) in docdb

Parameters **objtype** (*str*) – Object type, e.g. 'ELG', 'LRG'

desisim.spec_qa.utils.**elg_flux_lim**(*z*, *oii_flux*)
Assess which objects pass the ELG flux limit Uses DESI document 318 from August 2014

Parameters

- **z** (*ndarray*) – ELG redshifts
- **oii_flux** (*ndarray*) – [OII] fluxes

desisim.spec_qa.utils.**get_sty_otype**()
Styles for plots

desisim.spec_qa.utils.**match_otype**(*tbl*, *objtype*)
Generate a mask for the input objtype :param tbl: :param objtype: str :return: targets: bool mask

2.3.29 desisim.specsim

DESI wrapper functions for external specsim classes.

desisim.specsim.**get_simulator**(*config='desi'*, *num_fibers=1*, *camera_output=True*)
returns new or cached specsim.simulator.Simulator object

Also adds placeholder for BGS fiberloss if that isn't already in the config

2.3.30 desisim.targets

Utility functions for working with simulated targets.

desisim.targets.**_default_wave**(*wavemin=None*, *wavemax=None*, *dw=0.2*)
Construct and return the default wavelength vector.

desisim.targets.**get_simtype**(*spectype*, *desi_target*, *bgs_target*, *mws_target*)
Derive the simulation type from the redshift spectype and target bits

Parameters

- **spectype** – array of 'GALAXY', 'QSO', 'STAR'
- ***_target** – target mask bits

Returns array of simulation types: ELG, LRG, QSO, BGS, ...

TODO: add subtypes of STAR

desisim.targets.**get_targets**(*nspec*, *program*, *tileid=None*, *seed=None*, *specify_targets={}*, *specmin=0*)

Generates a set of targets for the requested program

Parameters

- **nspec** – (int) number of targets to generate
- **program** – (str) program name DARK, BRIGHT, GRAY, MWS, BGS, LRG, ELG, ...

Options:

- tileid: (int) tileid, used for setting RA,dec
- seed: (int) random number seed
- **specify_targets: (dict of dicts) Define target properties like magnitude and redshift** for each target class. Each objtype has its own key,value pair see simspec.templates.specify_galparams_dict() or simsepc.templates.specify_starparams_dict()
- specmin: (int) first spectrum number (0-indexed)

Returns

- fibermap
- targets as tuple of (flux, wave, meta)

desisim.targets.**get_targets_parallel**(*nspec*, *program*, *tileid=None*, *nproc=None*, *seed=None*, *specify_targets={}*)

Parallel wrapper for get_targets()

nproc (int) is number of multiprocessing processes to use.

desisim.targets.**sample_nz**(*objtype*, *n*)

Given *objtype* = 'LRG', 'ELG', 'QSO', 'STAR', 'STD' return array of *n* redshifts that properly sample n(z) from $DESIMODEL/data/targets/nz*.dat

desisim.targets.**sample_objtype**(*nobj*, *program*)

Return a random sampling of object types (dark, bright, MWS, BGS, ELG, LRG, QSO, STD, BAD_QSO)

Parameters **nobj** – number of objects to generate

Returns

(true_objtype, target_objtype) where true_objtype is the array of what type the objects actually are and target_objtype is the array of type they were targeted as

Notes

- Actual fiber assignment will result in higher relative fractions of LRGs and QSOs in early passes and more ELGs in later passes.
- Also ensures at least 2 sky and 1 stdstar, even if nobj is small

2.3.31 desisim.templates

Functions to simulate spectral templates for DESI.

class desisim.templates.**BGS**(*minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *add_SNeIa=False*, *include_mgii=False*, *colorcuts_function=None*, *normfilter_north='BASS-r'*, *normfilter_south='decam2014-r'*, *baseflux=None*, *basewave=None*, *basemeta=None*)

Generate Monte Carlo spectra of bright galaxy survey galaxies (BGSs).

make_templates(*nmodel=100*, *zrange=(0.01*, *0.4)*, *magrange=(15.0*, *20.0)*, *oiiihbrange=(-1.3*, *0.6)*, *logvdisp_meansig=(2.0*, *0.17)*, *minhbetaflux=0.0*, *sne_fluxratiorange=(0.1*, *1.0)*, *sne_filter='decam2014-r'*, *redshift=None*, *mag=None*, *vdisp=None*, *seed=None*, *input_meta=None*, *input_snemeta=None*, *nocolorcuts=False*, *nocontinuum=False*, *agnlike=False*, *novdisp=False*, *south=True*, *restframe=False*, *verbose=False*)

Build Monte Carlo BGS spectra/templates.

See the GALAXY.make_galaxy_templates function for documentation on the arguments and inherited attributes. Here we only document the arguments that are specific to the BGS class.

Parameters

- **oiiihbrange** (*float, optional*) – Minimum and maximum logarithmic [OIII] 5007/H-beta line-ratio. Defaults to a uniform distribution between (-1.3, 0.6).
- **logvdisp_meansig** (*float, optional*) – Logarithmic mean and sigma values for the (Gaussian) stellar velocity dispersion distribution. Defaults to log10-sigma=(2.0+/-0.17) km/s
- **minhbetaflux** (*float, optional*) – Minimum H-beta flux (default 0.0 erg/s/cm2).

Returns (outflux, wave, meta, objmeta) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.
- objmeta (astropy.Table): Additional objtype-specific table data [nmodel] for each spectrum.

In addition, if add_SNeIa=True then a third astropy.Table object, snemeta, is returned with the properties of the simulated SNe.

Raises:

class desisim.templates.**ELG**(*minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *add_SNeIa=False*, *include_mgii=False*, *colorcuts_function=None*, *normfilter_north='BASS-r'*, *normfilter_south='decam2014-r'*, *baseflux=None*, *basewave=None*, *basemeta=None*)

Generate Monte Carlo spectra of emission-line galaxies (ELGs).

make_templates(*nmodel=100*, *zrange=(0.6*, *1.6)*, *magrange=(21.0*, *23.4)*, *oiiihbrange=(-0.5*, *0.2)*, *logvdisp_meansig=(1.9*, *0.15)*, *minoiiflux=0.0*, *sne_fluxratiorange=(0.1*, *1.0)*, *sne_filter='decam2014-r'*, *redshift=None*, *mag=None*, *vdisp=None*, *seed=None*, *input_meta=None*, *input_snemeta=None*, *nocolorcuts=False*, *nocontinuum=False*, *agnlike=False*, *novdisp=False*, *south=True*, *restframe=False*, *verbose=False*)

Build Monte Carlo ELG spectra/templates.

See the GALAXY.make_galaxy_templates function for documentation on the arguments and inherited attributes. Here we only document the arguments that are specific to the ELG class.

Parameters

- **oiiihbrange** (*float, optional*) – Minimum and maximum logarithmic [OIII] 5007/H-beta line-ratio. Defaults to a uniform distribution between (-0.5, 0.2).
- **logvdisp_meansig** (*float, optional*) – Logarithmic mean and sigma values for the (Gaussian) stellar velocity dispersion distribution. Defaults to log10-sigma=(1.9+/-0.15) km/s
- **minoiiflux** (*float, optional*) – Minimum [OII] 3727 flux (default 0.0 erg/s/cm2).

Returns (outflux, wave, meta, objmeta) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.
- objmeta (astropy.Table): Additional objtype-specific table data [nmodel] for each spectrum.

In addition, if add_SNeIa=True then a third astropy.Table object, snemeta, is returned with the properties of the simulated SNe.

Raises:

class desisim.templates.**EMSpectrum**(*minwave=3650.0*, *maxwave=7075.0*, *cdelt_kms=20.0*, *log10wave=None*, *include_mgii=False*)

Construct a complete nebular emission-line spectrum.

Read the requisite external data files and initialize the output wavelength array.

The desired output wavelength array can either by passed directly using LOG10WAVE (note: must be a log-base10, i.e., constant-velocity pixel array!) or via the MINWAVE, MAXWAVE, and CDELT_KMS arguments.

In addition, three data files are required: ${DESISIM}/data/recombination_lines.escv, ${DESISIM}/data/forbidden_lines.esv, and ${DESISIM}/data/forbidden_mog.fits.

TODO (@moustakas): Incorporate AGN-like emission-line ratios. TODO (@moustakas): Think about how to best include dust attenuation in the lines.

Parameters

- **minwave** (*float, optional*) – Minimum value of the output wavelength array [Angstrom, default 3600].
- **maxwave** (*float, optional*) – Minimum value of the output wavelength array [Angstrom, default 10000].
- **cdelt_kms** (*float, optional*) – Spacing of the output wavelength array [km/s, default 20].
- **log10wave** (*numpy.ndarray, optional*) – Input/output wavelength array (log10-Angstrom, default None).
- **include_mgii** (*bool, optional*) – Include Mg II in emission (default False).

log10wave

Wavelength array constructed from the input arguments.

Type numpy.ndarray

line

Table containing the laboratoy (vacuum) wavelengths and nominal line-ratios for several dozen forbidden and recombination nebular emission lines.

Type astropy.Table

forbidmog
Table containing the mixture of Gaussian parameters encoding the forbidden emission-line priors.

Type GaussianMixtureModel

oiiidoublet
Intrinsic [OIII] 5007/4959 doublet ratio (set by atomic physics).

Type float32

niidoublet
Intrinsic [NII] 6584/6548 doublet ratio (set by atomic physics).

Type float32

Raises IOError – If the required data files are not found.

spectrum(*oiiihbeta=None*, *oiihbeta=None*, *niihbeta=None*, *siihbeta=None*, *oiidoublet=0.73*, *siidoublet=1.3*, *linesigma=75.0*, *zshift=0.0*, *oiiflux=None*, *hbetaflux=None*, *seed=None*)
Build the actual emission-line spectrum.

Building the emission-line spectrum involves three main steps. First, the oiiihbeta, oiihbeta, and niihbeta emission-line ratios are either drawn from the empirical mixture of Gaussians (recommended!) or input values are used to construct the line-ratios of the strongest optical forbidden lines relative to H-beta.

Note that all three of oiiihbeta, oiihbeta, and niihbeta must be specified simultaneously in order for them to be used.

Second, the requested [OII] 3726,29 and [SII] 6716,31 doublet ratios are imposed.

And finally the full emission-line spectrum is self-consistently normalized to *either* an integrated [OII] 3726,29 line-flux *or* an integrated H-beta line-flux. Generally an ELG and LRG spectrum will be normalized using [OII] while the a BGS spectrum will be normalized using H-beta. Note that the H-beta normalization trumps the [OII] normalization (in the case that both are given).

TODO (@moustakas): Add a suitably scaled nebular continuum spectrum. TODO (@moustakas): Add more emission lines (e.g., [NeIII] 3869).

Parameters

- **oiiihbeta** (*float, optional*) – Desired logarithmic [OIII] 5007/H-beta line-ratio (default -0.2). A sensible range is [-0.5,0.2].
- **oiihbeta** (*float, optional*) – Desired logarithmic [OII] 3726,29/H-beta line-ratio (default 0.1). A sensible range is [0.0,0.4].
- **niihbeta** (*float, optional*) – Desired logarithmic [NII] 6584/H-beta line-ratio (default -0.2). A sensible range is [-0.6,0.0].
- **siihbeta** (*float, optional*) – Desired logarithmic [SII] 6716/H-beta line-ratio (default -0.3). A sensible range is [-0.5,0.2].
- **oiidoublet** (*float, optional*) – Desired [OII] 3726/3729 doublet ratio (default 0.73).
- **siidoublet** (*float, optional*) – Desired [SII] 6716/6731 doublet ratio (default 1.3).
- **linesigma** (*float, optional*) – Intrinsic emission-line velocity width/sigma (default 75 km/s). A sensible range is [30-150].

- **zshift** (*float, optional*) – Perturb the emission lines from their laboratory (rest) wavelengths by a factor 1+ZSHIFT (default 0.0). Use with caution!
- **oiiflux** (*float, optional*) – Normalize the emission-line spectrum to this integrated [OII] emission-line flux (default None).
- **hbetaflux** (*float, optional*) – Normalize the emission-line spectrum to this integrated H-beta emission-line flux (default None).
- **seed** (*int, optional*) – input seed for the random numbers.

Returns Tuple of (emspec, wave, line), where emspec is an Array [npix] of flux values [erg/s/cm2/A]; wave is an Array [npix] of vacuum wavelengths corresponding to FLUX [Angstrom, linear spacing]; line is a Table of emission-line parameters used to generate the emission-line spectrum.

class desisim.templates.**GALAXY**(*objtype='ELG'*, *minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *add_SNeIa=False*, *include_mgii=False*, *colorcuts_function=None*, *normfilter_north='BASS-r'*, *normfilter_south='decam2014-r'*, *normline='OII'*, *fracvdisp=(0.1, 40)*, *baseflux=None*, *basewave=None*, *basemeta=None*)

Base class for generating Monte Carlo spectra of the various flavors of galaxies (ELG, BGS, and LRG).

_blurmatrix(*vdisp*, *log=None*)

Pre-compute the blur_matrix as a dictionary keyed by each unique value of vdisp.

lineratios(*nobj*, *oiiihbrange=(-0.5, 0.2)*, *oiidoublet_meansig=(0.73, 0.05)*, *agnlike=False*, *rand=None*)

Get the correct number and distribution of the forbidden and [OII] 3726/3729 doublet emission-line ratios. Note that the agnlike option is not yet supported.

Supporting oiiihbrange needs a different (fast) approach. Suppressing the code below for now until it's needed.

make_galaxy_templates(*nmodel=100*, *zrange=(0.6, 1.6)*, *magrange=(20.0, 22.0)*, *oiiihbrange=(-0.5, 0.2)*, *logvdisp_meansig=(1.9, 0.15)*, *minlineflux=0.0*, *sne_fluxratiorange=(0.01, 0.1)*, *sne_filter='decam2014-r'*, *seed=None*, *redshift=None*, *mag=None*, *vdisp=None*, *input_meta=None*, *input_snemeta=None*, *nocolorcuts=False*, *nocontinuum=False*, *agnlike=False*, *novdisp=False*, *south=True*, *restframe=False*, *verbose=False*)

Build Monte Carlo galaxy spectra/templates.

This function chooses random subsets of the basis continuum spectra (for the given galaxy spectral type), constructs an emission-line spectrum (if desired), redshifts, convolves by the intrinsic velocity dispersion, and then finally normalizes each spectrum to a (generated or input) apparent magnitude.

In detail, each (output) model gets randomly assigned a continuum (basis) template; however, if that template doesn't pass the (spectral) class-specific color cuts (at the specified redshift), then we iterate through the rest of the templates until we find one that *does* pass the color-cuts.

The user also (optionally) has a lot of flexibility over the inputs/outputs and can specify any combination of the redshift, velocity dispersion, and apparent magnitude (in the normalization filter specified in the GALAXY.__init__ method) inputs. Alternatively, the user can pass a complete metadata table, in order to easily regenerate spectra on-the-fly (see the documentation for the input_meta argument, below).

Note: The default inputs are generally set to values which are appropriate for ELGs, so be sure to alter them when generating templates for other spectral classes.

Parameters

- **nmodel** (*int, optional*) – Number of models to generate (default 100).
- **zrange** (*float, optional*) – Minimum and maximum redshift range. Defaults to a uniform distribution between (0.6, 1.6).
- **magrange** (*float, optional*) – Minimum and maximum magnitude in the bandpass specified by self.normfilter_south (if south=True) or self.normfilter_north (if south=False). Defaults to a uniform distribution between (20.0, 22.0).
- **oiiihbrange** (*float, optional*) – Minimum and maximum logarithmic [OIII] 5007/H-beta line-ratio. Defaults to a uniform distribution between (-0.5, 0.2).
- **logvdisp_meansig** (*float, optional*) – Logarithmic mean and sigma values for the (Gaussian) stellar velocity dispersion distribution. Defaults to log10-sigma=1.9+/-0.15 km/s.
- **minlineflux** (*float, optional*) – Minimum emission-line flux in the line specified by self.normline (default 0 erg/s/cm2).
- **sne_fluxratiorange** (*float, optional*) – flux ratio of the SNeIa spectrum with respect to the underlying galaxy in the filter specified in sne_filter. Defaults to a uniform distribution between (0.01, 0.1).
- **sne_filter** (*str*) – filter corresponding to SNE_FLUXRATIORANGE (default 'decam2014-r').
- **seed** (*int, optional*) – Input seed for the random numbers.
- **redshift** (*float, optional*) – Input/output template redshifts. Array size must equal nmodel. Ignores zrange input.
- **mag** (*float, optional*) – Input/output template magnitudes in the bandpass specified by self.normfilter_south (if south=True) or self.normfilter_north (if south=False). Array size must equal nmodel. Ignores magrange input.
- **vdisp** (*float, optional*) – Input/output velocity dispersions in km/s. Array size must equal nmodel.
- **input_meta** (*astropy.Table*) – *Input* metadata table with the following required columns: TEMPLATEID, SEED, REDSHIFT, MAG, and MAGFILTER (see desisim.io.empty_metatable for the expected data types). In addition, in order to faithfully reproduce a previous set of spectra, then VDISP must also be passed (normally returned in the OBJMETA table). If present, then all other optional inputs (nmodel, redshift, mag, zrange, logvdisp_meansig, etc.) are ignored.
- **input_snemeta** (*astropy.Table*) – *Input* table of SN properties with required columns SNE_TEMPLATEID, SNE_EPOCH, SNE_FLUXRATIO, and SNE_FILTER. Only used if add_SNeIa is True. Also, if present then all other optional inputs pertaining to SNe are ignored.
- **nocolorcuts** (*bool, optional*) – Do not apply the color-cuts specified by the self.colorcuts_function function (default False).
- **nocontinuum** (*bool, optional*) – Do not include the stellar continuum in the output spectrum (useful for testing; default False). Note that this option automatically sets nocolorcuts to True and add_SNeIa to False.
- **novdisp** (*bool, optional*) – Do not velocity-blur the spectrum (default False).

- **agnlike** (*bool, optional*) – Adopt AGN-like emission-line ratios (e.g., for the LRGs and some BGS galaxies) (default False, meaning we adopt star-formation-like line-ratios). Option not yet supported.
- **south** (*bool, optional*) – Apply "south" color-cuts using the DECaLS filter system, otherwise apply the "north" (MzLS+BASS) color-cuts. Defaults to True.
- **restframe** (*bool, optional*) – If True, return full resolution restframe templates instead of resampled observer frame.
- **verbose** (*bool, optional*) – Be verbose!

Returns (outflux, wave, meta, objmeta) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.
- objmeta (astropy.Table): Additional objtype-specific table data [nmodel] for each spectrum.

In addition, if add_SNeIa=True then a third astropy.Table object, snemeta, is returned with the properties of the simulated SNe.

Raises ValueError

class desisim.templates.**LRG**(*minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *add_SNeIa=False*, *colorcuts_function=None*, *normfilter_north='MzLS-z'*, *normfilter_south='decam2014-z'*, *baseflux=None*, *basewave=None*, *basemeta=None*)

Generate Monte Carlo spectra of luminous red galaxies (LRGs).

make_templates(*nmodel=100*, *zrange=(0.5, 1.0)*, *magrange=(19.0, 20.2)*, *logvdisp_meansig=(2.3, 0.1)*, *sne_fluxratiorange=(0.1, 1.0)*, *sne_filter='decam2014-r'*, *redshift=None*, *mag=None*, *vdisp=None*, *seed=None*, *input_meta=None*, *input_snemeta=None*, *nocolorcuts=False*, *novdisp=False*, *agnlike=False*, *south=True*, *restframe=False*, *verbose=False*)

Build Monte Carlo BGS spectra/templates.

See the GALAXY.make_galaxy_templates function for documentation on the arguments and inherited attributes. Here we only document the arguments that are specific to the LRG class.

Parameters

- **logvdisp_meansig** (*float, optional*) – Logarithmic mean and sigma values for the (Gaussian) stellar velocity dispersion distribution. Defaults to log10-sigma=(2.3+/-0.1) km/s
- **agnlike** (*bool, optional*) – adopt AGN-like emission-line ratios (not yet supported; defaults False).

Returns (outflux, wave, meta, objmeta) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.
- objmeta (astropy.Table): Additional objtype-specific table data [nmodel] for each spectrum.

In addition, if add_SNeIa=True then a third astropy.Table object, snemeta, is returned with the properties of the simulated SNe.

Raises:

class desisim.templates.**MWS_STAR**(*minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *colorcuts_function=None*, *normfilter_north='BASS-r'*, *normfilter_south='decam2014-r'*, *baseflux=None*, *basewave=None*, *basemeta=None*)

Generate Monte Carlo spectra of Milky Way Survey (magnitude-limited) stars.

make_templates(*nmodel=100*, *vrad_meansig=(0.0*, *200.0)*, *magrange=(16.0*, *20.0)*, *seed=None*, *redshift=None*, *mag=None*, *input_meta=None*, *star_properties=None*, *nocolorcuts=False*, *south=True*, *restframe=False*, *verbose=False*)

Build Monte Carlo spectra/templates for MWS_STAR stars.

See the SUPERSTAR.make_star_templates function for documentation on the arguments and inherited attributes. Here we only document the arguments which are specific to the MWS_STAR class.

Args:

Returns (outflux, wave, meta, objmeta) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.
- objmeta (astropy.Table): Additional objtype-specific table data [nmodel] for each spectrum.

Raises:

class desisim.templates.**QSO**(*minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *basewave_min=1200*, *basewave_max=25000.0*, *basewave_R=8000*, *normfilter_north='BASS-r'*, *normfilter_south='decam2014-r'*, *colorcuts_function=None*, *balqso=False*, *z_wind=0.2*)

Generate Monte Carlo spectra of quasars (QSOs).

_sample_pcacoeff(*nsample*, *coeff*, *samplerand*)

Draw from the distribution of PCA coefficients.

make_templates(*nmodel=100*, *zrange=(0.5*, *4.0)*, *magrange=(17.0*, *22.7)*, *seed=None*, *redshift=None*, *mag=None*, *input_meta=None*, *N_perz=40*, *maxiter=20*, *uniform=False*, *balprob=0.12*, *lyaforest=True*, *noresample=False*, *nocolorcuts=False*, *south=True*, *verbose=False*)

Build Monte Carlo QSO spectra/templates.

This function generates QSO spectra on-the-fly using PCA decomposition coefficients of SDSS and BOSS QSO spectra. The default is to generate flat, uncorrelated priors on redshift and apparent magnitude (in the bandpass specified by self.normfilter).

However, the user also (optionally) has flexibility over the inputs/outputs and can specify any combination of the redshift and output apparent magnitude. Alternatively, the user can pass a complete metadata table, in order to easily regenerate spectra on-the-fly (see the documentation for the input_meta argument, below).

Note: The templates are only defined in the range 3500-10000 A (observed).

Parameters

- **nmodel** (*int, optional*) – Number of models to generate (default 100).

- **zrange** (*float, optional*) – Minimum and maximum redshift range. Defaults to a uniform distribution between (0.5, 4.0).
- **magrange** (*float, optional*) – Minimum and maximum magnitude in the bandpass specified by self.normfilter_south (if south=True) or self.normfilter_north (if south=False). Defaults to a uniform distribution between (17, 22.7).
- **seed** (*int, optional*) – input seed for the random numbers.
- **redshift** (*float, optional*) – Input/output template redshifts. Array size must equal nmodel. Ignores zrange input.
- **mag** (*float, optional*) – Input/output template magnitudes in the bandpass specified by self.normfilter_south (if south=True) or self.normfilter_north (if south=False). Array size must equal nmodel. Ignores magrange input.
- **input_meta** (*astropy.Table*) – *Input* metadata table with the following required columns: SEED, REDSHIFT, MAG, and MAGFILTER (see desisim.io.empty_metatable for the expected data types). If present, then all other optional inputs (nmodel, redshift, mag, zrange, etc.) are ignored. Note that this argument cannot be used (at this time) to precisely reproduce templates that have had BALs inserted.
- **N_perz** (*int, optional*) – Number of templates per redshift redshift value to generate (default 20).
- **maxiter** (*int*) – maximum number of iterations for findng a non-negative template that also satisfies the color-cuts (default 20).
- **uniform** (*bool, optional*) – Draw uniformly from the PCA coefficients (default False).
- **balprob** (*float, optional*) – Probability that a QSO is a BAL (default 0.12). Only used if QSO(balqso=True) at instantiation.
- **lyaforest** (*bool, optional*) – Include Lyman-alpha forest absorption (default True).
- **noresample** (*bool, optional*) – Do not resample the QSO spectra in wavelength (default False).
- **nocolorcuts** (*bool, optional*) – Do not apply the fiducial rzW1W2 color-cuts cuts (default False).
- **south** (*bool, optional*) – Apply "south" color-cuts using the DECaLS filter system, otherwise apply the "north" (MzLS+BASS) color-cuts. Defaults to True.
- **verbose** (*bool, optional*) – Be verbose!

Returns (outflux, wave, meta, objmeta) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- **wave (numpy.ndarray): Observed-frame wavelength array (Angstrom).** If noresample=True then this is an [nmodel, npix] array (a different observed-frame array for each object), otherwise it's a one-dimensional [npix]-length array.
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.
- objmeta (astropy.Table): Additional objtype-specific table data [nmodel] for each spectrum.

Raises ValueError

class desisim.templates.**SIMQSO**(*minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *nproc=1*, *basewave_min=450.0*, *basewave_max=60000.0*, *basewave_R=8000*, *normfilter_north='BASS-r'*, *normfilter_south='decam2014-r'*, *colorcuts_function=None*, *restframe=False*)

Generate Monte Carlo spectra of quasars (QSOs) using simqso.

_make_simqso_templates(*redshift=None*, *magrange=None*, *seed=None*, *lyaforest=True*, *nocolorcuts=False*, *noresample=False*, *input_qsometa=None*, *south=True*)

Wrapper function for actually generating the templates.

empty_qsometa(*qsometa*, *nmodel*)

Initialize an empty QsoSimPoints object, which contains all the metadata needed to regenerate simqso spectra.

make_templates(*nmodel=100*, *zrange=(0.5, 4.0)*, *magrange=(17.0, 22.7)*, *seed=None*, *redshift=None*, *mag=None*, *maxiter=20*, *input_qsometa=None*, *qsometa_extname='QSOMETA'*, *return_qsometa=False*, *lyaforest=True*, *nocolorcuts=False*, *noresample=False*, *south=True*, *verbose=False*)

Build Monte Carlo QSO spectra/templates.

- This function generates QSO spectra on-the-fly using @imcgreer's simqso. The default is to generate flat, uncorrelated priors on redshift, absolute magnitudes based on the SDSS/DR9 QSOLF, and to compute the corresponding apparent magnitudes using the appropriate per-object K-correction.

  Alternatively, the redshift can be input and the absolute and apparent magnitudes will again be computed self-consistently from the QSOLF.

  Providing apparent magnitudes on *input* is not supported although it could be if there is need. However, one can control the apparent brightness of the resulting QSO spectra by specifying magrange.

- The way the code is currently structured could lead to memory problems if one attempts to generate very large numbers of spectra simultaneously (>10^4, perhaps, depending on the machine). However, it can easily be refactored to generate the appropriate number of templates in chunks at the expense of some computational speed.

  Parameters

  - **nmodel** (*int, optional*) – Number of models to generate (default 100).
  - **zrange** (*float, optional*) – Minimum and maximum redshift range. Defaults to a uniform distribution between (0.5, 4.0).
  - **magrange** (*float, optional*) – Minimum and maximum magnitude in the bandpass specified by self.normfilter_south (if south=True) or self.normfilter_north (if south=False). Defaults to a uniform distribution between (17, 22.7).
  - **seed** (*int, optional*) – input seed for the random numbers.
  - **redshift** (*float, optional*) – Input/output template redshifts. Array size must equal nmodel. Ignores zrange input.
  - **mag** (*float, optional*) – Not currently supported or used, but see magrange. Defaults to None.
  - **maxiter** (*int*) – maximum number of iterations for findng a template that satisfies the color-cuts (default 20).
  - **input_qsometa** (*simqso.sqgrids.QsoSimPoints object or FITS filename*) – Input QsoSimPoints object or FITS filename (with a qsometa_extname HDU) from which to (re)generate the QSO spectra. All other inputs are ignored when this

optional input is present. Please be cautious when using this argument, as it has not been fully tested.

- **qsometa_extname** (*str*) – FITS extension name to read when input_qsometa is a filename. Defaults to 'QSOMETA'.
- **return_qsometa** (*bool, optional*) – Return the simqso.sqgrids.QsoSimPoints object, which contains all the data necessary to regenerate the QSO spectra. In particular, the data attribute is an astropy.Table object which contains lots of useful info. This object can be written to disk with the simqso.sqgrids.QsoSimObjects.write method (default False).
- **lyaforest** (*bool, optional*) – Include Lyman-alpha forest absorption (default True).
- **nocolorcuts** (*bool, optional*) – Do not apply the fiducial rzW1W2 color-cuts cuts (default False).
- **noresample** (*bool, optional*) – Do not resample the QSO spectra in wavelength (default False).
- **south** (*bool, optional*) – Apply "south" color-cuts using the DECaLS filter system, otherwise apply the "north" (MzLS+BASS) color-cuts. Defaults to True.
- **verbose** (*bool, optional*) – Be verbose!

Returns (outflux, wave, meta, qsometa) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.
- objmeta (astropy.Table): Additional objtype-specific table data [nmodel] for each spectrum.

In addition, if return_qsometa=True then a fourth argument, qsometa, is returned (see the return_qsometa documentation, above).

Raises ValueError

class desisim.templates.**STAR**(*minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *baseflux=None*, *basewave=None*, *basemeta=None*)

Generate Monte Carlo spectra of generic stars.

make_templates(*nmodel=100*, *vrad_meansig=(0.0*, *200.0)*, *magrange=(18.0*, *23.5)*, *seed=None*, *redshift=None*, *mag=None*, *input_meta=None*, *star_properties=None*, *south=True*, *restframe=False*, *verbose=False*)

Build Monte Carlo spectra/templates for generic stars.

See the SUPERSTAR.make_star_templates function for documentation on the arguments and inherited attributes. Here we only document the arguments which are specific to the STAR class.

Args:

Returns (outflux, wave, meta, objmeta) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.

- objmeta (astropy.Table): Additional objtype-specific table data [nmodel] for each spectrum.

Raises:

class desisim.templates.**STD**(*minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *colorcuts_function=None*, *normfilter_north='BASS-r'*, *normfilter_south='decam2014-r'*, *baseflux=None*, *basewave=None*, *basemeta=None*)

Generate Monte Carlo spectra of (metal-poor, main sequence turnoff) standard stars (STD).

make_templates(*nmodel=100*, *vrad_meansig=(0.0*, *200.0)*, *magrange=(16.0*, *19.0)*, *seed=None*, *redshift=None*, *mag=None*, *input_meta=None*, *star_properties=None*, *nocolorcuts=False*, *south=True*, *restframe=False*, *verbose=False*)

Build Monte Carlo spectra/templates for STD stars.

See the SUPERSTAR.make_star_templates function for documentation on the arguments and inherited attributes. Here we only document the arguments which are specific to the STD class.

Args:

Returns (outflux, wave, meta, objmeta) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.
- objmeta (astropy.Table): Additional objtype-specific table data [nmodel] for each spectrum.

Raises:

class desisim.templates.**SUPERSTAR**(*objtype='STAR'*, *subtype=''*, *minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *normfilter_north='BASS-r'*, *normfilter_south='decam2014-r'*, *colorcuts_function=None*, *baseflux=None*, *basewave=None*, *basemeta=None*)

Base class for generating Monte Carlo spectra of the various flavors of stars.

make_star_templates(*nmodel=100*, *vrad_meansig=(0.0*, *200.0)*, *magrange=(18.0*, *22.0)*, *seed=None*, *redshift=None*, *mag=None*, *input_meta=None*, *star_properties=None*, *nocolorcuts=False*, *south=True*, *restframe=False*, *verbose=False*)

Build Monte Carlo spectra/templates for various flavors of stars.

This function chooses random subsets of the continuum spectra for the type of star specified by OBJTYPE, adds radial velocity jitter, applies the targeting color-cuts, and then normalizes the spectrum to the magnitude in the given filter.

The user also (optionally) has a lot of flexibility over the inputs/outputs and can specify any combination of the radial velocity and apparent magnitude (in the normalization filter specified in the GALAXY.__init__ method) inputs. Alternatively, the user can pass a complete metadata table, in order to easily regenerate spectra on-the-fly (see the documentation for the input_meta argument, below). Finally, the user can pass a star_properties table in order to interpolate the base templates to non-gridded values of [Fe/H], logg, and Teff.

Note:

- The default inputs are generally set to values which are appropriate for generic stars, so be sure to alter them when generating templates for other spectral classes.
- If both input_meta and star_properties are passed, then star_properties is ignored.

Parameters

- **nmodel** (*int, optional*) – Number of models to generate (default 100).
- **vrad_meansig** (*float, optional*) – Mean and sigma (standard deviation) of the radial velocity "jitter" (in km/s) that should be included in each spectrum. Defaults to a normal distribution with a mean of zero and sigma of 200 km/s.
- **magrange** (*float, optional*) – Minimum and maximum magnitude in the bandpass specified by self.normfilter_south (if south=True) or self.normfilter_north (if south=False). Defaults to a uniform distribution between (18, 22).
- **seed** (*int, optional*) – input seed for the random numbers.
- **redshift** (*float, optional*) – Input/output (dimensionless) radial velocity. Array size must equal nmodel. Ignores vrad_meansig input.
- **mag** (*float, optional*) – Input/output template magnitudes in the bandpass specified by self.normfilter_south (if south=True) or self.normfilter_north (if south=False). Array size must equal nmodel. Ignores magrange input.
- **input_meta** (*astropy.Table*) – *Input* metadata table with the following required columns: TEMPLATEID, SEED, REDSHIFT, MAG, and MAGFILTER (see desisim.io.empty_metatable for the expected data types). If present, then all other optional inputs (nmodel, redshift, mag, zrange, vrad_meansig, etc.) are ignored.
- **star_properties** (*astropy.Table*) – *Input* table with the following required columns: REDSHIFT, MAG, MAGFILTER, TEFF, LOGG, and FEH (except for WDs, which don't need to have an FEH column). Optionally, SEED can also be included in the table. When this table is passed, the basis templates are interpolated to the desired physical values provided, enabling large numbers of mock stellar spectra to be generated with physically consistent properties. However, be warned that the interpolation scheme is very rudimentary.
- **nocolorcuts** (*bool, optional*) – Do not apply the color-cuts specified by the self.colorcuts_function function (default False).
- **south** (*bool, optional*) – Apply "south" color-cuts using the DECaLS filter system, otherwise apply the "north" (MzLS+BASS) color-cuts. Defaults to True.
- **restframe** (*bool, optional*) – If True, return full resolution restframe templates instead of resampled observer frame.
- **verbose** (*bool, optional*) – Be verbose!

Returns (outflux, wave, meta) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.

Raises ValueError

class desisim.templates.**WD**(*minwave=3600.0*, *maxwave=10000.0*, *cdelt=0.2*, *wave=None*, *subtype='DA'*, *colorcuts_function=None*, *normfilter_north='BASS-g'*, *normfilter_south='decam2014-g'*, *baseflux=None*, *basewave=None*, *basemeta=None*)

Generate Monte Carlo spectra of white dwarfs.

make_templates(*nmodel=100*, *vrad_meansig=(0.0*, *200.0)*, *magrange=(16.0*, *19.0)*, *seed=None*, *redshift=None*, *mag=None*, *input_meta=None*, *star_properties=None*, *nocolorcuts=False*, *south=True*, *restframe=False*, *verbose=False*)

Build Monte Carlo spectra/templates for WD stars.

See the SUPERSTAR.make_star_templates function for documentation on the arguments and inherited attributes. Here we only document the arguments which are specific to the WD class.

Args:

Returns (outflux, wave, meta, objmeta) tuple where:

- outflux (numpy.ndarray): Array [nmodel, npix] of observed-frame spectra (1e-17 erg/s/cm2/A).
- wave (numpy.ndarray): Observed-frame [npix] wavelength array (Angstrom).
- meta (astropy.Table): Table of meta-data [nmodel] for each output spectrum.
- objmeta (astropy.Table): Additional objtype-specific table data [nmodel] for each spectrum.

Raises `ValueError` – If the INPUT_META or STAR_PROPERTIES table contains different values of SUBTYPE.

desisim.templates.**specify_galparams_dict**(*templatetype*, *zrange=None*, *magrange=None*, *oiiihbrange=None*, *logvdisp_meansig=None*, *minlineflux=None*, *sne_rfluxratiorange=None*, *redshift=None*, *mag=None*, *vdisp=None*, *nocolorcuts=None*, *nocontinuum=None*, *agnlike=None*, *novdisp=None*, *restframe=None*)

Creates a dictionary of keyword variables to be passed to GALAXY.make_templates (or one of GALAXY's child classes). Allows the user to fully define the templated spectra, via defining individual targets or ranges in values. Values already specified in get_targets are not included here. Anything not define or set to None will not be assigned and CLASS.make_templates will assume the following as defaults:

- nmodel=100, zrange=(0.6, 1.6), magrange=(21.0, 23.5),
- oiiihbrange=(-0.5, 0.2), logvdisp_meansig=(1.9, 0.15),
- minlineflux=0.0, sne_rfluxratiorange=(0.01, 0.1),
- seed=None, redshift=None, mag=None, vdisp=None,
- input_meta=None, nocolorcuts=False, nocontinuum=False,
- agnlike=False, novdisp=False, restframe=False, verbose=False

Parameters

- **nmodel** (*) – Number of models to generate (default 100).
- **zrange** (*) – Minimum and maximum redshift range. Defaults to a uniform distribution between (0.6, 1.6).
- **magrange** (*) – Minimum and maximum magnitude in the bandpass specified by self.normfilter. Defaults to a uniform distribution between (21, 23.4) in the r-band.
- **oiiihbrange** (*) – Minimum and maximum logarithmic [OIII] 5007/H-beta line-ratio. Defaults to a uniform distribution between (-0.5, 0.2).
- **logvdisp_meansig** (*) – Logarithmic mean and sigma values for the (Gaussian) stellar velocity dispersion distribution. Defaults to log10-sigma=1.9+/-0.15 km/s.
- **minlineflux** (*) – Minimum emission-line flux in the line specified by self.normline (default 0 erg/s/cm2).

- **sne_rfluxratiorange** (*) – r-band flux ratio of the SNeIa spectrum with respect to the underlying galaxy. Defaults to a uniform distribution between (0.01, 0.1).
- **seed** (*) – Input seed for the random numbers.
- **redshift** (*) – Input/output template redshifts. Array size must equal nmodel. Ignores zrange input.
- **mag** (*) – Input/output template magnitudes in the band specified by self.normfilter. Array size must equal nmodel. Ignores magrange input.
- **vdisp** (*) – Input/output velocity dispersions. Array size must equal nmodel. Ignores magrange input.
- **input_meta** (*) – *Input* metadata table with the following required columns: TEMPLATEID, SEED, REDSHIFT, VDISP, MAG (where mag is specified by self.normfilter). In addition, if add_SNeIa is True then the table must also contain SNE_TEMPLATEID, SNE_EPOCH, and SNE_RFLUXRATIO columns. See desisim.io.empty_metatable for the required data type for each column. If this table is passed then all other optional inputs (nmodel, redshift, vdisp, mag, zrange, logvdisp_meansig, etc.) are ignored.
- **nocolorcuts** (*) – Do not apply the color-cuts specified by the self.colorcuts_function function (default False).
- **nocontinuum** (*) – Do not include the stellar continuum in the output spectrum (useful for testing; default False). Note that this option automatically sets nocolorcuts to True and add_SNeIa to False.
- **novdisp** (*) – Do not velocity-blur the spectrum (default False).
- **agnlike** (*) – Adopt AGN-like emission-line ratios (e.g., for the LRGs and some BGS galaxies) (default False, meaning we adopt star-formation-like line-ratios). Option not yet supported.
- **restframe** (*) – If True, return full resolution restframe templates instead of resampled observer frame.
- **verbose** (*) – Be verbose!

Returns

dictionary containing all of the values passed defined with variable names as the corresonding key. These are intentionally identical to those passed to the make_templates classes above

Return type

- fulldef_dict (dict)

desisim.templates.**specify_starparams_dict**(*templatetype*, *vrad_meansig=None*, *magrange=None*, *redshift=None*, *mag=None*, *input_meta=None*, *star_properties=None*, *nocolorcuts=None*, *restframe=None*)

Creates a dictionary of keyword variables to be passed to SUPERSTAR.make_templates (or one of SUPERSTAR's child classes). Allows the user to fully define the templated spectra, via defining individual targets or ranges in values. Values already specified in get_targets are not included here. Anything not define or set to None will not be assigned and CLASS.make_templates will assume the following as defaults:

- nmodel=100, vrad_meansig=(0.0, 200.0),
- magrange=(18.0, 23.5), seed=None, redshift=None,
- mag=None, input_meta=None, star_properties=None,

- nocolorcuts=False, restframe=False, verbose=False

Parameters

- **nmodel** (*) – Number of models to generate (default 100).
- **vrad_meansig** (*) – Mean and sigma (standard deviation) of the radial velocity "jitter" (in km/s) that should be included in each spectrum. Defaults to a normal distribution with a mean of zero and sigma of 200 km/s.
- **magrange** (*) – Minimum and maximum magnitude in the bandpass specified by self.normfilter. Defaults to a uniform distribution between (18, 23.5) in the r-band.
- **seed** (*) – input seed for the random numbers.
- **redshift** (*) – Input/output (dimensionless) radial velocity. Array size must equal nmodel. Ignores vrad_meansig input.
- **mag** (*) – Input/output template magnitudes in the band specified by self.normfilter. Array size must equal nmodel. Ignores magrange input.
- **input_meta** (*) – *Input* metadata table with the following required columns: TEMPLATEID, SEED, REDSHIFT, and MAG (where mag is specified by self.normfilter). See desisim.io.empty_metatable for the required data type for each column. If this table is passed then all other optional inputs (nmodel, redshift, mag, vrad_meansig etc.) are ignored.
- **star_properties** (*) – *Input* table with the following required columns: REDSHIFT, MAG, TEFF, LOGG, and FEH (except for WDs, which don't need to have an FEH column). Optionally, SEED can also be included in the table. When this table is passed, the basis templates are interpolated to the desired physical values provided, enabling large numbers of mock stellar spectra to be generated with physically consistent properties.
- **nocolorcuts** (*) – Do not apply the color-cuts specified by the self.colorcuts_function function (default False).
- **restframe** (*) – If True, return full resolution restframe templates instead of resampled observer frame.
- **verbose** (*) – Be verbose!

Returns

dictionary containing all of the values passed defined with variable names as the corresonding key. These are intentionally identical to those passed to the make_templates classes above

Return type

- fulldef_dict (dict)

2.3.32 desisim.util

Utility functions for desisim. These may belong elsewhere?

desisim.util.**dateobs2night**(*dateobs*)

Convert UTC dateobs to KPNO YEARMMDD night string

Parameters **dateobs** – float -> interpret as MJD str -> interpret as ISO 8601 YEAR-MM-DDThh:mm:ss.s string astropy.time.Time -> UTC python datetime.datetime -> UTC

TODO: consider adding format option to pass to astropy.time.Time without otherwise questioning the dateobs format

desisim.util.**medxbin**(*x*, *y*, *binsize*, *minpts=20*, *xmin=None*, *xmax=None*)
Compute the median (and other statistics) in fixed bins along the x-axis.

desisim.util.**spline_medfilt2d**(*image*, *kernel_size=201*)
Returns a 2D spline interpolation of a median filtered input image

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

Symbols

A

B

C

D

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

V

W

Z