desitarget Documentation

Release 0.41.0.dev4129

DESI Collaboration

Aug 05, 2020

Contents

Contents:

CHAPTER 1

desitarget API

1.1 desitarget

Tools for DESI target selection.

1.2 desitarget.brightmask

Module for studying and masking bright sources in the sweeps

desitarget.brightmask.**_rexlike**(*rextype*)
If the object is REX (a round exponential galaxy)

desitarget.brightmask.**generate_safe_locations**(*sourcemask*, *Nperradius=1*)
Given a mask, generate SAFE (BADSKY) locations at its periphery.

Parameters

- **sourcemask** (recarray) – A recarray containing a bright mask as made by, e.g., *desitarget.brightmask.make_bright_star_mask()*
- **Nperradius** (int, optional, defaults to 1.) – Number of safe locations to make per arcsec radius of each mask.

Returns

- **ra** (*array_like.*) – The Right Ascensions of the SAFE (BADSKY) locations.
- **dec** (*array_like.*) – The Declinations of the SAFE (BADSKY) locations.

Notes

- See Tech Note 2346 for details.

desitarget.brightmask.**get_mask_dir**()
Convenience function to grab the MASK_DIR environment variable.

Returns The directory stored in the $MASK_DIR environment variable.

Return type str

desitarget.brightmask.**get_recent_mask_dir**(*input_dir=None*)
Grab the most recent sub-directory of masks in MASK_DIR.

Parameters **input_dir** (str, optional, defaults to None) – If passed and not None, then this is returned as the output.

Returns If *input_dir* is not None, then the most recently created sub-directory (with the appropriate format for a mask directory) in $MASK_DIR is returned.

Return type str

desitarget.brightmask.**get_safe_targets**(*targs*, *sourcemask*)
Get SAFE (BADSKY) locations for targs, set TARGETID/DESI_TARGET.

Parameters

- **targs** (ndarray) – Targets made by, e.g. *desitarget.cuts.select_targets()*.
- **sourcemask** (ndarray) – A bright source mask as made by, e.g. *desitarget.brightmask.make_bright_star_mask()*.

Returns SAFE (BADSKY) locations for *targs* with the same data model as for *targs*.

Return type ndarray

Notes

- Tech Note 2346 details SAFE (BADSKY) locations.
- Tech Note 2348 details setting the SKY bit in TARGETID.
- Hard-coded to create 1 safe location per arcsec of mask radius. The correct number (Nperradius) for DESI is an open question.

desitarget.brightmask.**is_bright_source**(*targs*, *sourcemask*)
Determine whether targets are, themselves, a bright source mask.

Parameters

- **targs** (recarray) – Targets as made by, e.g., *desitarget.cuts.select_targets()*.
- **sourcemask** (recarray) – A recarray containing a bright source mask as made by, e.g., *desitarget.brightmask.make_bright_star_mask()*

Returns **is_mask** – True for *targs* that are, themselves, a mask.

Return type array_like

desitarget.brightmask.**is_in_bright_mask**(*targs*, *sourcemask*, *inonly=False*)
Determine whether a set of targets is in a bright star mask.

Parameters

- **targs** (recarray) – A recarray of targets, skies etc., as made by, e.g., *desitarget.cuts.select_targets()*.

- **sourcemask** (recarray) – A recarray containing a mask as made by, e.g., *desitarget.brightmask.make_bright_star_mask()*
- **inonly** (boolean, optional, defaults to False) – If True, then only calculate the *in_mask* return but not the *near_mask* return, which is about a factor of 2 faster.

Returns

- list – [*in_mask*, *near_mask*] where *in_mask* (*near_mask*) is a boolean array that is True for *targs* that are IN (NEAR) a mask. If *inonly* is True then this is just [*in_mask*].
- class: *list* – [*used_in_mask*, *used_near_mask*] where *used_in_mask* (*used_near_mask*) is a boolean array that is True for masks in *sourcemask* that contain a target at the IN (NEAR) radius. If *inonly* is True then this is just [*used_in_mask*].

desitarget.brightmask.**make_bright_star_mask**(*maglim=12.0*, *matchrad=1.0*, *numproc=32*, *maskepoch=2023.0*, *gaiaepoch=2015.5*, *nside=None*, *pixels=None*)

Make an all-sky bright star mask using Tycho, Gaia and URAT.

Parameters

- **maglim** (float, optional, defaults to 12.) – Faintest magnitude at which to make the mask. This magnitude is interpreted as G-band for Gaia and, in order of preference, VT then HP then BT for Tycho (not every Tycho source has each band).
- **matchrad** (int, optional, defaults to 1.) – Tycho sources that match a Gaia source at this separation in ARCSECONDS are NOT included in the output mask. The matching is performed rigorously, accounting for Gaia proper motions.
- **numproc** (int, optional, defaults to 16.) – Number of processes over which to parallelize
- **maskepoch** (float) – The mask is built at this epoch. Not all sources have proper motions from every survey, so proper motions are used, in order of preference, from Gaia, URAT, then Tycho.
- **gaiaepoch** (float, optional, defaults to Gaia DR2 (2015.5)) – The epoch of the Gaia observations. Should be 2015.5 unless we move beyond Gaia DR2.
- **nside** (int, optional, defaults to None) – If passed, create a mask only in nested HEALPixels in *pixels* at this *nside*. Otherwise, run for the whole sky. If *nside* is passed then *pixels* must be passed too.
- **pixels** (list, optional, defaults to None) – If passed, create a mask only in nested HEALPixels at *nside* for pixel integers in *pixels*. Otherwise, run for the whole sky. If *pixels* is passed then *nside* must be passed too.

Returns

- The bright star mask in the form of *maskdatamodel.dtype*:
- *REF_CAT* is *"T2"* for Tycho and *"G2"* for Gaia.
- *REF_ID* is *Tyc1*'*1,000,000+*'*Tyc2*'*10+*'*Tyc3* for Tycho2; *"sourceid"* for Gaia-DR2 and Gaia-DR2 with URAT.
- *REF_MAG* is, in order of preference, G-band for Gaia, VT then HP then BT for Tycho.
- *URAT_ID* contains the URAT reference number for Gaia objects that use the URAT proper motion, or -1 otherwise.
- The radii are in ARCSECONDS.
- *E1* and *E2* are placeholders for ellipticity components, and are set to 0 for Gaia and Tycho sources.

- *TYPE* is always *PSF* for star-like objects.
- Note that the mask is based on objects in the pixel AT THEIR NATIVE EPOCH *NOT* AT THE INPUT *maskepoch*. It is therefore possible for locations in the output mask to be just beyond the boundaries of the input pixel.

Return type recarray

Notes

- Runs (all-sky) in ~20 minutes for *numproc=32* and *maglim=12*.
- *IN_RADIUS* (*NEAR_RADIUS*) corresponds to *IN_BRIGHT_OBJECT* (*NEAR_BRIGHT_OBJECT*) in *data/targetmask.yaml*. These radii are set in the function *desitarget.brightmask.radius()*.
- The correct mask size for DESI is an open question.
- The *GAIA_DIR*, *URAT_DIR* and *TYCHO_DIR* environment variables must be set.

desitarget.brightmask.**make_bright_star_mask_in_hp**(*nside*, *pixnum*, *verbose=True*, *gaiaepoch=2015.5*, *maglim=12.0*, *matchrad=1.0*, *maskepoch=2023.0*)

Make a bright star mask in a HEALPixel using Tycho, Gaia and URAT.

Parameters

- **nside** (int) – (NESTED) HEALPixel nside.
- **pixnum** (int) – A single HEALPixel number.
- **verbose** (bool) – If True then log informational messages.

Returns The bright star mask in the form of *maskdatamodel.dtype*.

Return type recarray

Notes

- Runs in a a minute or so for a typical nside=4 pixel.
- See *make_bright_star_mask()* for descriptions of the output mask and the other input parameters.

desitarget.brightmask.**mask_targets**(*targs*, *inmaskdir*, *nside=2*, *pixlist=None*)

Add bits for if objects occupy masks, and SAFE (BADSKY) locations.

Parameters

- **targs** (str or *~numpy.ndarray*) – An array of targets/skies etc. created by, e.g., *desitarget.cuts.select_targets()* OR the filename of a file that contains such a set of targets/skies, etc.
- **inmaskdir** (str, optional) – An input bright star mask file or HEALPixel-split directory as made by *desitarget.brightmask.make_bright_star_mask()*
- **nside** (int, optional, defaults to 2) – The nside at which the targets were generated. If the mask is a HEALPixel-split directory, then this helps to perform more efficient masking as only the subset of masks that are in pixels containing *targs* at this *nside* will be considered (together with neighboring pixels to account for edge effects).

- **pixlist** (list or *int*, optional) – A set of HEALPixels corresponding to the *targs*. Only the subset of masks in HEALPixels in *pixlist* at *nside* will be considered (together with neighboring pixels to account for edge effects). If None, then the pixels touched by *targs* is derived from from *targs* itself.

Returns Input targets with the *DESI_TARGET* column updated to reflect the *BRIGHT_OBJECT* bits and SAFE (*BADSKY*) sky locations added around the perimeter of the mask.

Return type ndarray

Notes

- Tech Note 2346 details SAFE (BADSKY) locations.

desitarget.brightmask.**max_objid_bricks**(*targs*)
For a set of targets, return the maximum value of BRICK_OBJID in each BRICK_ID

Parameters **targs** (recarray) – A recarray of targets as made by *desitarget.cuts.select_targets*

Returns **maxobjid** – A dictionary with keys for each unique BRICKID and values of the maximum OBJID in that brick

Return type dictionary

desitarget.brightmask.**plot_mask**(*mask*, *limits=None*, *radius='IN_RADIUS'*, *show=True*)
Plot a mask or masks.

Parameters

- **mask** (recarray) – A mask, as constructed by, e.g. *make_bright_star_mask()*.
- **limits** (list, optional) – RA/Dec plot limits in the form [ramin, ramax, decmin, decmax].
- **radius** – Which mask radius to plot (IN_RADIUS or NEAR_RADIUS).
- **show** (boolean) – If True, then display the plot, Otherwise, just execute the plot commands so it can be added to or saved to file later.

Returns

Return type Nothing

desitarget.brightmask.**radii**(*mag*)
The relation used to set the radius of bright star masks.

Parameters **mag** (flt or recarray) – Magnitude. Typically, in order of preference, G-band for Gaia or VT then HP then BT for Tycho.

Returns

- recarray – The *IN_RADIUS*, corresponding to the *IN_BRIGHT_OBJECT* bit in *data/targetmask.yaml*.
- recarray – The *NEAR_RADIUS*, corresponding to the *NEAR_BRIGHT_OBJECT* bit in data/targetmask.yaml‘.

desitarget.brightmask.**set_target_bits**(*targs*, *sourcemask*, *return_masks=False*)
Apply bright source mask to targets, return desi_target array.

Parameters

- **targs** (recarray) – Targets as made by, e.g., *desitarget.cuts.select_targets()*.
- **sourcemask** (recarray) – A recarray containing a bright source mask as made by, e.g. *desitarget.brightmask.make_bright_star_mask* or desitarget.brightmask.make_bright_source_mask.
- **return_masks** (bool) – If True also return boolean arrays of which of the masks in *sourcemask* contain a target.

Returns

- recarray – *DESI_TARGET* column updates with bright source information bits.
- list, only returned if *return_masks* is True – [*used_in_mask*, *used_near_mask*] where *used_in_mask* (*used_near_mask*) is a boolean array that is True for masks in *sourcemask* that contain a target at the IN (NEAR) radius.

Notes

- Sets IN_BRIGHT_OBJECT and NEAR_BRIGHT_OBJECT via matches to circular and/or elliptical masks.
- Sets BRIGHT_OBJECT via an index match on TARGETID (defined as in *desitarget.targets.encode_targetid()*).

See *desitarget.targetmask* for the definition of each bit.

1.3 desitarget.cmx.cmx_cuts

Target Selection for DESI commissioning (cmx) derived from the cmx wiki.

A collection of helpful (static) methods to check whether an object's flux passes a given selection criterion (*e.g.* STD_TEST).

desitarget.cmx.cmx_cuts.**_get_cmxdir**(*cmxdir=None*)

Retrieve the base cmx directory with appropriate error checking.

cmxdir [str, optional, defaults to CMX_DIR] Directory in which to find commmissioning files to which to match, such as the CALSPEC stars. If not specified, the cmx directory is taken to be the value of the CMX_DIR environment variable.

desitarget.cmx.cmx_cuts.**apply_cuts**(*objects*, *cmxdir=None*, *noqso=False*)

Commissioning (cmx) target selection, return target mask arrays.

Parameters

- **objects** (ndarray) – numpy structured array with UPPERCASE columns needed for target selection, OR a string tractor/sweep filename
- **cmxdir** (str, optional, defaults to CMX_DIR) – Directory to find commmissioning files to which to match, such as the CALSPEC stars. If not specified, the cmx directory is taken to be the value of 2:envvar:*CMX_DIR*.
- **noqso** (boolean, optional, defaults to False) – If passed, do not run the quasar selection. All QSO bits will be set to zero. Intended use is to speed unit tests.

Returns

- ndarray – commissioning target selection bitmask flags for each object.

- array_like – a priority shift of 10*(25-rmag) based on r-band magnitude. (for *STD_DITHER*, *STD_GAIA* sources).
- *See desitarget.cmx.cmx_targetmask.cmx_mask for bit definitions.*

desitarget.cmx.cmx_cuts.**apply_cuts_gaia**(*numproc=4*, *cmxdir=None*, *nside=None*, *pixlist=None*)

Gaia-only-based CMX target selection, return target mask arrays.

Parameters

- **numproc** (int, optional, defaults to 4) – The number of parallel processes to use.
- **cmxdir** (str, optional, defaults to CMX_DIR) – Directory to find commmissioning files to match, such as for the CALSPEC stars. If not specified, taken to be the value of the CMX_DIR environment variable.
- **nside** (int, optional, defaults to *None*) – (NESTED) HEALPix nside used with *pixlist* and *bundlefiles*.
- **pixlist** (list or *int*, optional, defaults to *None*) – Only return targets in a set of (NESTED) HEALpixels at *nside*. Useful for parallelizing, as input files will only be processed if they touch a pixel in the passed list.

Returns

- ndarray – Commissioning target selection bitmask flags for each object.
- ndarray – numpy structured array of Gaia sources that were read in from file for the passed pixel constraints (or no pixel constraints).

Notes

- May take a long time if no pixel constraints are passed.
- Only run on Gaia-only target selections.
- The environment variable $GAIA_DIR must be set.

See desitarget.cmx.cmx_targetmask.cmx_mask for bit definitions.

desitarget.cmx.cmx_cuts.**isBACKUP**(*ra=None*, *dec=None*, *gaiagmag=None*, *primary=None*)

BACKUP targets based on Gaia magnitudes.

Parameters

- **dec** (*ra,*) – Right Ascension and Declination in degrees.
- **gaiagmag** (array_like or None) – Gaia-based g MAGNITUDE (not Galactic-extinction-corrected). (same units as the Gaia data model).
- **primary** (array_like or None) – True for objects that should be passed through the selection.

Returns

- array_like – True if and only if the object is a bright "BACKUP" target.
- array_like – True if and only if the object is a faint "BACKUP" target.

desitarget.cmx.cmx_cuts.**isBGS**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *rfiberflux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *gfracmasked=None*, *rfracmasked=None*, *zfracmasked=None*, *gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *gfracin=None*, *rfracin=None*, *zfracin=None*, *gfluxivar=None*, *rfluxivar=None*, *zfluxivar=None*, *maskbits=None*, *Grr=None*, *w1snr=None*, *gaiagmag=None*, *objtype=None*, *primary=None*, *south=True*, *targtype=None*)

Definition of BGS target classes for SV. Returns a boolean array.

Parameters

- **south** (boolean, defaults to True) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if south=False, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).
- **targtype** (str, optional, defaults to faint) – Pass bright for the BGS_BRIGHT selection or faint for the BGS_FAINT selection or faint_ext for the BGS_FAINT_EXTENDED selection or lowq for the BGS_LOW_QUALITY selection or fibmag for the BGS_FIBER_MAGNITUDE selection.

Returns True if and only if the object is a BGS target of type targtype.

Return type array_like

Notes

- Current version (10/14/19) is version 105 on the SV wiki.
- See *set_target_bits()* for other parameters.

desitarget.cmx.cmx_cuts.**isBGS_colors**(*rflux=None*, *rfiberflux=None*, *south=True*, *targtype=None*, *primary=None*)

Standard set of masking cuts used by all BGS target selection classes (see, e.g., *isBGS()* for parameters).

desitarget.cmx.cmx_cuts.**isELG_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *gfiberflux=None*, *primary=None*, *south=True*)

Color cuts for ELG target selection classes (see, e.g., *desitarget.cuts.set_target_bits()* for parameters).

desitarget.cmx.cmx_cuts.**isFIRSTLIGHT**(*gaiadtype*, *cmxdir=None*, *nside=None*, *pixlist=None*)

First light/Mini-SV targets via reading files from Arjun Dey.

Parameters

- **gaiadtype** (dtype) – Data type (dtype) for Gaia-only CMX targets.
- **cmxdir** (str, optional, defaults to CMX_DIR) – Directory to find commmissioning files. If not specified, taken from the CMX_DIR environment variable.
- **nside** (int, optional, defaults to *None*) – (NESTED) HEALPix nside used with *pixlist* and *bundlefiles*.
- **pixlist** (list or *int*, optional, defaults to *None*) – Only return targets in a set of (NESTED) HEALpixels at *nside*. Useful for parallelizing, as input files will only be processed if they touch a pixel in the passed list.

Returns

- array_like – bit values for each of the first light targets.

- `array_like` – Array of the first light targets munged into Gaia-only format.

`desitarget.cmx.cmx_cuts.`**`isLRG_colors`**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *zfiberflux=None*, *south=True*, *primary=None*)

See *isLRG()* for details.

`desitarget.cmx.cmx_cuts.`**`isQSO_color_high_z`**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *south=True*)

Color cut to select Highz QSO (z>~2.)

`desitarget.cmx.cmx_cuts.`**`isQSO_colors`**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *primary=None*, *south=True*)

Test if sources have quasar-like colors in a color box. (see, e.g., *isQSO_cuts()*).

`desitarget.cmx.cmx_cuts.`**`isQSO_cuts`**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *w1snr=None*, *w2snr=None*, *dchisq=None*, *maskbits=None*, *objtype=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *primary=None*, *south=True*)

Definition of QSO target classes from color cuts. Returns a boolean array.

Parameters **`south`** (`boolean`, defaults to `True`) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if `south=False`, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns `True` for objects that pass the quasar color/morphology/logic cuts.

Return type `array_like`

Notes

- See *set_target_bits()* for other parameters.

`desitarget.cmx.cmx_cuts.`**`isQSO_highz_faint`**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *objtype=None*, *release=None*, *dchisq=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *maskbits=None*, *primary=None*, *south=True*)

Definition of QSO target for highz (z>2.0) faint QSOs. Returns a boolean array.

Parameters **`south`** (`boolean`, defaults to `True`) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if `south=False`, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns `True` for objects that pass the quasar color/morphology/logic cuts.

Return type `array_like`

Notes

- See *set_target_bits()* for other parameters.

`desitarget.cmx.cmx_cuts.`**`isQSO_randomforest`**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *objtype=None*, *release=None*, *dchisq=None*, *maskbits=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *primary=None*, *south=True*)

Definition of QSO target class using random forest. Returns a boolean array.

Parameters **south** (boolean, defaults to True) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if south=False, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns True for objects that pass the quasar color/morphology/logic cuts.

Return type array_like

Notes

- See *set_target_bits()* for other parameters.

desitarget.cmx.cmx_cuts.**isQSOz5_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *w1flux=None*, *w2flux=None*, *primary=None*, *south=True*)

Color cut to select z~5 quasar targets. (See *isQSOz5_cuts()*).

desitarget.cmx.cmx_cuts.**isQSOz5_cuts**(*gflux=None*, *rflux=None*, *zflux=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *w1flux=None*, *w2flux=None*, *w1snr=None*, *w2snr=None*, *dchisq=None*, *maskbits=None*, *objtype=None*, *primary=None*, *south=True*)

Definition of z~5 QSO target classes from color cuts. Returns a boolean array.

Parameters **south** (boolean, defaults to True) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if south=False, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns True for objects that pass the quasar color/morphology/logic cuts.

Return type array_like

Notes

- See *set_target_bits()* for other parameters.

desitarget.cmx.cmx_cuts.**isSTD_calspec**(*ra=None*, *dec=None*, *cmxdir=None*, *matchrad=1.0*, *primary=None*)

Match to CALSPEC stars for commissioning tests.

Parameters

- **dec** (*ra,*) – Right Ascension and Declination in degrees.
- **cmxdir** (str, optional, defaults to CMX_DIR) – Directory to find commmissioning files to match, such as for the CALSPEC stars. If not specified, taken to be the value of the CMX_DIR environment variable.
- **matchrad** (float, optional, defaults to 1 arcsec) – The matching radius in arcseconds.
- **primary** (array_like or None) – True for objects that should be passed through the selection.

Returns True if and only if the object is a "CALSPEC" target.

Return type array_like

desitarget.cmx.cmx_cuts.**isSTD_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *primary=None*)

Select STD stars based on Legacy Surveys color cuts. Returns a boolean array. see *isSTD()* for other details.

desitarget.cmx.cmx_cuts.**isSTD_dither**(*obs_gflux=None*, *obs_rflux=None*, *obs_zflux=None*, *isgood=None*, *primary=None*)

Gaia stars for dithering-and-other tests during commissioning.

Parameters

- **obs_rflux, obs_zflux** (*obs_gflux,*) – The flux in nano-maggies of g, r, z bands WITHOUT any Galactic extinction correction.
- **isgood** (array_like or None) – True for objects that pass the logic cuts in *passesSTD_logic()*.
- **primary** (array_like or None) – True for objects that should be passed through the selection.

Returns

- array_like – True if and only if the object is a Gaia "STD_GAIA" target.
- array_like – A priority shift of 10*(25-rmag) based on r-band magnitude.

Notes

- This version (08/30/18) is version 4 on the cmx wiki.

desitarget.cmx.cmx_cuts.**isSTD_dither_spec**(*gaiagmag=None*, *gaiarmag=None*, *obs_rflux=None*, *isgood=None*, *primary=None*)

Gaia stars for dithering-only tests during commissioning.

Parameters

- **gaiarmag** (*gaiagmag,*) – The Gaia G-band and R-band mean magnitudes.
- **obs_rflux** (array_like or None) – The flux in nano-maggies in Legacy Surveys r-band WITHOUT any Galactic extinction correction. Used for prioritizing.
- **isgood** (array_like or None) – True for objects that pass the logic cuts in *passesSTD_logic()*.
- **primary** (array_like or None) – True for objects that should be passed through the selection.

Returns

- array_like – True if and only if the object is a Gaia "STD_DITHER" target.
- array_like – A priority shift of 10*(25-rmag) based on r-band magnitude.

Notes

- This version (11/02/19) is version 48 on the cmx wiki.

desitarget.cmx.cmx_cuts.**isSTD_gaia**(*primary=None*, *gaia=None*, *astrometricexcessnoise=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *dupsource=None*, *paramssolved=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*)

Gaia quality cuts used to define STD star targets see *isSTD()* for other details.

desitarget.cmx.cmx_cuts.**isSTD_test**(*obs_gflux=None*, *obs_rflux=None*, *obs_zflux=None*, *isgood=None*, *primary=None*)

Very bright Gaia stars for early commissioning tests.

Parameters

- **obs_rflux, obs_zflux** (*obs_gflux,*) – The flux in nano-maggies of g, r, z bands WITHOUT any Galactic extinction correction.
- **isgood** (array_like or None) – True for objects that pass the logic cuts in *passesSTD_logic()*.
- **primary** (array_like or None) – True for objects that should be passed through the selection.

Returns True if and only if the object is a Gaia "test" target.

Return type array_like

Notes

- This version (08/30/18) is version 4 on the cmx wiki.
- See also the Gaia data model.

desitarget.cmx.cmx_cuts.**isSV0_BGS**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *rfiberflux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *gfracmasked=None*, *rfracmasked=None*, *zfracmasked=None*, *gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *gfracin=None*, *rfracin=None*, *zfracin=None*, *gfluxivar=None*, *rfluxivar=None*, *zfluxivar=None*, *maskbits=None*, *Grr=None*, *w1snr=None*, *gaiagmag=None*, *objtype=None*, *primary=None*)

Definition of an SV0-like BGS target. Returns a boolean array.

:param See *set_target_bits()*.:

Returns True if and only if the object is an SV-like BGS target.

Return type array_like

Notes

- Current version (02/19/20) is version 55 on the cmx wiki.
- Hardcoded for south=False.
- Combines bright/faint/faint_ext/fibmag BGS-like SV classes into one bit.
- *desitarget.cmx.cmx_cuts.apply_cuts()* also additionally removes objects from this class that either have Gaia provenance and Gaia G < 16 OR that have Legacy Surveys g < 16.

desitarget.cmx.cmx_cuts.**isSV0_ELG**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *gfiberflux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *maskbits=None*, *primary=None*)

Definition of an SV0-like ELG target. Returns a boolean array.

:param See *set_target_bits()*.:

Returns `True` if and only if the object is an SV-like ELG target.

Return type `array_like`

Notes

- Current version (10/14/19) is version 107 on the SV wiki.
- Hardcoded for south=False.
- Combines all ELG-like SV classes into one bit.

desitarget.cmx.cmx_cuts.**isSV0_LRG**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *rfiberflux=None*, *zfiberflux=None*, *gflux_snr=None*, *rflux_snr=None*, *zflux_snr=None*, *w1flux_snr=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *maskbits=None*, *primary=None*)

Target Definition of an SV0-like LRG. Returns a boolean array.

:param See *set_target_bits()*.:

Returns `True` if and only if the object is an LRG color-selected target. If *floats* are passed, a *float* is returned.

Return type `array_like` or `float`

Notes

- Current version (02/19/20) is version 50 on the cmx wiki.
- Hardcoded for south=False.
- Combines all LRG-like SV classes into one bit.

desitarget.cmx.cmx_cuts.**isSV0_MWS**(*rflux=None*, *obs_rflux=None*, *objtype=None*, *paramssolved=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*, *parallaxerr=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *parallaxovererror=None*, *photbprpexcessfactor=None*, *astrometricsigma5dmax=None*, *gaiaaen=None*, *galb=None*, *gaia=None*, *primary=None*)

Initial SV-like Milky Way Survey selections (MzLS/BASS imaging).

:param - See *set_target_bits()* for parameters.:

Returns

- `array_like` – `True` if and only if the object is a MWS_MAIN or MWS_NEARBY target from early SV/main survey classes.
- `array_like` – `True` if and only if the object is an early-SV/main survey MWS_WD target.

- `array_like` – `True` if and only if the object is an early-SV/main survey SV0_MWS_FAINT target.

Notes

- All Gaia quantities are as in the Gaia data model.
- Returns the equivalent of PRIMARY target classes from version 55 (02/19/20) of the cmx wiki. Ignores target classes that "smell" like secondary targets (as they are outside of the footprint or are based on catalog-matching). Simplifies flag cuts, and simplifies the MWS_MAIN class to not include sub-classes.
- *desitarget.cmx.cmx_cuts.apply_cuts()* also additionally removes objects from this class that either have Gaia provenance and Gaia G < 16 OR that have Legacy Surveys g < 16.

`desitarget.cmx.cmx_cuts.`**`isSV0_QSO`**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *w1snr=None*, *w2snr=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *maskbits=None*, *dchisq=None*, *objtype=None*, *primary=None*)

Target Definition of an SV0-like QSO. Returns a boolean array.

:param See *set_target_bits()*.:

Returns

- `array_like` or `float` –

  `True` if and only if the object is an SV-like QSO target. If *floats* are passed, a *float* is returned.

- `array_like` or `float` – `True` if and only if the object is an SV-like QSO target that passes something like the QSO_Z5 (high-z) selection from SV.

Notes

- Current version (02/19/20) is version 51 on the cmx wiki.
- Current version (03/10/20) for the high-z (QSO_Z5 selection) is version 59 on the cmx wiki.
- Hardcoded for south=False.
- Combines all QSO-like SV classes into one bit.

`desitarget.cmx.cmx_cuts.`**`isSV0_STD`**(*gflux=None*, *rflux=None*, *zflux=None*, *primary=None*, *gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *gfracmasked=None*, *rfracmasked=None*, *zfracmasked=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *gfluxivar=None*, *rfluxivar=None*, *zfluxivar=None*, *objtype=None*, *gaia=None*, *astrometricexcessnoise=None*, *paramssolved=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *dupsource=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*, *bright=False*)

Select STD targets using color cuts and photometric quality cuts.

Parameters **`bright`** (`boolean`, defaults to `False`) – if `True` apply magnitude cuts for "bright" conditions; otherwise, choose "normal" brightness standards. Cut is performed on *gaiagmag*.

Returns `True` if and only if the object is a STD star.

Return type array_like

Notes

- This version (11/05/18) is version 24 on the SV wiki.
- See *set_target_bits()* for other parameters.

desitarget.cmx.cmx_cuts.**notinBGS_mask**(*gflux=None*, *rflux=None*, *zflux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *primary=None*, *gfracmasked=None*, *rfracmasked=None*, *zfracmasked=None*, *gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *gfracin=None*, *rfracin=None*, *zfracin=None*, *w1snr=None*, *gfluxivar=None*, *rfluxivar=None*, *zfluxivar=None*, *Grr=None*, *gaiagmag=None*, *maskbits=None*, *objtype=None*, *targtype=None*)

Standard set of masking cuts used by all BGS target selection classes (see, e.g., isBGS_faint() for parameters).

desitarget.cmx.cmx_cuts.**notinELG_mask**(*maskbits=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *primary=None*)

Standard set of masking cuts used by all ELG target selection classes. (see *set_target_bits()* for parameters).

desitarget.cmx.cmx_cuts.**notinLRG_mask**(*primary=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *zfiberflux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *rflux_snr=None*, *zflux_snr=None*, *w1flux_snr=None*, *maskbits=None*)

See *isLRG()* for details.

Returns True if and only if the object is NOT masked for poor quality.

Return type array_like

desitarget.cmx.cmx_cuts.**passesSTD_logic**(*gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *objtype=None*, *gaia=None*, *pmra=None*, *pmdec=None*, *aen=None*, *dupsource=None*, *paramssolved=None*, *primary=None*)

The default logic/mask cuts for commissioning stars.

Parameters

- **rfracflux, zfracflux** (*gfracflux,*) – Profile-weighted fraction of the flux from other sources divided by the total flux in g, r and z bands.
- **objtype** (array_like or None) – The Legacy Surveys *TYPE* to restrict to point sources.
- **gaia** (boolean array_like or None) – True if there is a match between this object in the Legacy Surveys and in Gaia.
- **pmdec** (*pmra,*) – Gaia-based proper motion in RA and Dec.
- **aen** (array_like or None) – Gaia Astrometric Excess Noise.
- **dupsource** (array_like or None) – Whether the source is a duplicate in Gaia.
- **paramssolved** (array_like or None) – How many parameters were solved for in Gaia.

- **primary** (array_like or None) – True for objects that should be passed through the selection.

Returns

- array_like –

  True if and only if the object passes the logic cuts for cmx stars with fracflux_X < 0.01.

- array_like –

  True if and only if the object passes the logic cuts for cmx stars with fracflux_X < 0.002.

Notes

- This version (08/30/18) is version 4 on the cmx wiki.
- All Gaia quantities are as in the Gaia data model.

desitarget.cmx.cmx_cuts.**select_targets**(*infiles*, *numproc=4*, *cmxdir=None*, *noqso=False*, *nside=None*, *pixlist=None*, *bundlefiles=None*, *extra=None*, *resolvetargs=True*, *backup=True*)

Process input files in parallel to select commissioning (cmx) targets

Parameters

- **infiles** (list or *str*) – List of input filenames (tractor/sweep files) OR one filename.
- **numproc** (int, optional, defaults to 4) – The number of parallel processes to use.
- **cmxdir** (str, optional, defaults to CMX_DIR) – Directory to find commmissioning files to which to match, such as the CALSPEC stars. If not specified, the cmx directory is taken to be the value of CMX_DIR.
- **noqso** (boolean, optional, defaults to False) – If passed, do not run the quasar selection. All QSO bits will be set to zero. Intended use is to speed unit tests.
- **nside** (int, optional, defaults to *None*) – (NESTED) HEALPix nside used with *pixlist* and *bundlefiles*.
- **pixlist** (list or *int*, optional, defaults to *None*) – Only return targets in a set of (NESTED) HEALpixels at *nside*. Useful for parallelizing, as input files will only be processed if they touch a pixel in the passed list.
- **bundlefiles** (int, defaults to *None*) – If not *None*, then, instead of selecting gfas, print the slurm script to run in pixels at *nside*. Is an integer rather than a boolean for historical reasons.
- **extra** (str, optional) – Extra command line flags to be passed to the executable lines in the output slurm script. Used in conjunction with *bundlefiles*.
- **resolvetargs** (boolean, optional, defaults to True) – If True, resolve overlapping north/south Legacy Surveys targets into a set of unique sources based on location.
- **backup** (boolean, optional, defaults to True) – If True, also run the Gaia-only BACKUP_BRIGHT/FAINT targets.

Returns The subset of input targets which pass the cmx cuts, including an extra column for *CMX_TARGET*.

Return type ndarray

Notes

- if numproc==1, use serial code instead of parallel.

1.4 desitarget.cmx.cmx_targetmask

This looks more like a script than an actual module.

1.5 desitarget.cuts

Target Selection for DECALS catalogue data derived from the wiki.

A collection of helpful (static) methods to check whether an object's flux passes a given selection criterion (*e.g.* LRG, ELG or QSO).

`desitarget.cuts.`**`_check_BGS_targtype`**(*targtype*)
Fail if *targtype* is not one of the strings 'bright', 'faint' or 'wise'.

`desitarget.cuts.`**`_check_BGS_targtype_sv`**(*targtype*)
Fail if *targtype* is not one of the strings 'bright', 'faint', 'faint_ext', 'lowq' or 'fibmag'.

`desitarget.cuts.`**`_gal_coords`**(*ra*, *dec*)
Shift RA, Dec to Galactic coordinates.

Parameters **`dec`**(*ra,*) – RA, Dec coordinates (degrees)

Returns

Return type The Galactic longitude and latitude (l, b)

`desitarget.cuts.`**`_get_colnames`**(*objects*)
Simple wrapper to get the column names.

`desitarget.cuts.`**`_is_row`**(*table*)
Return True/False if this is a row of a table instead of a full table.

supports numpy.ndarray, astropy.io.fits.FITS_rec, and astropy.table.Table

`desitarget.cuts.`**`_isonnorthphotsys`**(*photsys*)
If the object is from the northen photometric system

`desitarget.cuts.`**`_prepare_gaia`**(*objects*, *colnames=None*)
Process the various Gaia inputs for target selection.

`desitarget.cuts.`**`_prepare_optical_wise`**(*objects*, *mask=True*)
Process the Legacy Surveys inputs for target selection.

Parameters **`mask`**(`boolean`, optional, defaults to `True`) – Send `False` to turn off any masking cuts based on the *MASKBITS* column. The default behavior is to always mask using *MASKBITS*.

`desitarget.cuts.`**`_psflike`**(*psftype*)
If the object is PSF

`desitarget.cuts.`**`apply_cuts`**(*objects, qso_selection='randomforest', gaiamatch=False, tcnames=['ELG', 'QSO', 'LRG', 'MWS', 'BGS', 'STD'], qso_optical_cuts=False, survey='main', resolvetargs=True, mask=True*)
Perform target selection on objects, returning target mask arrays.

Parameters

- **objects** (ndarray or *str*) – numpy structured array with UPPERCASE columns needed for target selection, OR a string tractor/sweep filename.
- **qso_selection** (str, optional, defaults to 'randomforest') – The algorithm to use for QSO selection; valid options are 'colorcuts' and 'randomforest'
- **gaiamatch** (boolean, optional, defaults to False) – If True, match to Gaia DR2 chunks files and populate Gaia columns to facilitate the MWS and STD selections.
- **tcnames** (list, defaults to running all target classes) – A list of strings, e.g. ['QSO','LRG']. If passed, process targeting only for those specific target classes. A useful speed-up when testing. Options include ["ELG", "QSO", "LRG", "MWS", "BGS", "STD"].
- **qso_optical_cuts** (boolean defaults to False) – Apply just optical color-cuts when selecting QSOs with qso_selection="colorcuts".
- **survey** (str, defaults to 'main') – Specifies which target masks yaml file and target selection cuts to use. Options are 'main' and 'svX' (where X is 1, 2, 3 etc.) for the main survey and different iterations of SV, respectively.
- **resolvetargs** (boolean, optional, defaults to True) – If True, if *objects* consists of all northern (southern) sources then only apply the northern (southern) cuts.
- **mask** (boolean, optional, defaults to True) – Send False to turn off any masking cuts based on the *MASKBITS* column. The default behavior is to always mask using *MASKBITS*.

Returns (desi_target, bgs_target, mws_target) where each element is an ndarray of target selection bitmask flags for each object.

Return type ndarray

Notes

- If objects is an astropy Table with lowercase column names, this converts them to UPPERCASE in-place, thus modifying the input table. To avoid this, pass in objects.copy() instead.
- See *desitarget.targetmask* for the definition of each bit.

desitarget.cuts.**apply_cuts_gaia**(*numproc=4*, *survey='main'*, *nside=None*, *pixlist=None*)

Gaia-only-based target selection, return target mask arrays.

Parameters

- **numproc** (int, optional, defaults to 4) – The number of parallel processes to use.
- **survey** (str, defaults to 'main') – Specifies which target masks yaml file and target selection cuts to use. Options are 'main' and 'svX' (where X is 1, 2, 3 etc.) for the main survey and different iterations of SV, respectively.
- **nside** (int, optional, defaults to *None*) – (NESTED) HEALPix nside used with *pixlist* and *bundlefiles*.
- **pixlist** (list or *int*, optional, defaults to *None*) – Only return targets in a set of (NESTED) HEALpixels at *nside*. Useful for parallelizing, as input files will only be processed if they touch a pixel in the passed list.

Returns

- ndarray – desi_target selection bitmask flags for each object.

- ndarray – bgs_target selection bitmask flags for each object.
- ndarray – mws_target selection bitmask flags for each object.
- ndarray – numpy structured array of Gaia sources that were read in from file for the passed pixel constraints (or no pixel constraints).

Notes

- May take a long time if no pixel constraints are passed.
- Only run on Gaia-only target selections.
- The environment variable $GAIA_DIR must be set.

See desitarget.svX.svX_targetmask.desi_mask or desitarget.targetmask.desi_mask for bit definitions.

desitarget.cuts.**isBACKUP**(*ra=None*, *dec=None*, *gaiagmag=None*, *primary=None*)

BACKUP targets based on Gaia magnitudes.

Parameters

- **dec** (*ra,*) – Right Ascension and Declination in degrees.
- **gaiagmag** (array_like or None) – Gaia-based g MAGNITUDE (not Galactic-extinction-corrected). (same units as the Gaia data model).
- **primary** (array_like or None) – True for objects that should be passed through the selection.

Returns

- array_like – True if and only if the object is a bright "BACKUP" target.
- array_like – True if and only if the object is a faint "BACKUP" target.
- array_like – True if and only if the object is a very faint "BACKUP" target.

Notes

- Current version (10/24/19) is version 204 on the wiki.

desitarget.cuts.**isBGS**(*rfiberflux=None*, *gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *gfracmasked=None*, *rfracmasked=None*, *zfracmasked=None*, *gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *gfracin=None*, *rfracin=None*, *zfracin=None*, *gfluxivar=None*, *rfluxivar=None*, *zfluxivar=None*, *maskbits=None*, *Grr=None*, *refcat=None*, *w1snr=None*, *gaiagmag=None*, *objtype=None*, *primary=None*, *south=True*, *targtype=None*)

Definition of BGS target classes. Returns a boolean array.

Parameters **targtype** (str, optional, defaults to faint) – Pass bright to use colors appropriate to the BGS_BRIGHT selection or faint to use colors appropriate to the BGS_FAINT selection or wise to use colors appropriate to the BGS_WISE selection.

Returns True if and only if the object is a BGS target of type targtype.

Return type array_like

Notes

- Current version (10/24/18) is version 143 on the wiki.
- See *set_target_bits()* for other parameters.

desitarget.cuts.**isBGS_colors**(*rfiberflux=None*, *gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *south=True*, *targtype=None*, *primary=None*)

Standard set of color-based cuts used by all BGS target selection classes (see, e.g., *isBGS()* for parameters).

desitarget.cuts.**isBGS_lslga**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *refcat=None*, *maskbits=None*, *south=True*, *targtype=None*)

Module to recover the LSLGA objects in all BGS target selection classes (see, e.g., *isBGS()* for parameters).

desitarget.cuts.**isELG**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *maskbits=None*, *south=True*, *primary=None*)

Definition of ELG target classes. Returns a boolean array. (see *set_target_bits()* for parameters).

Notes: - Current version (10/16/19) is version 202 on the wiki.

desitarget.cuts.**isELG_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *south=True*, *primary=None*)

Color cuts for ELG target selection classes (see, e.g., *desitarget.cuts.set_target_bits()* for parameters).

desitarget.cuts.**isLRG**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *zfiberflux=None*, *rflux_snr=None*, *zflux_snr=None*, *w1flux_snr=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *maskbits=None*, *primary=None*, *south=True*)

Parameters **south** (boolean, defaults to `True`) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if `south=False`, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns `True` if and only if the object is an LRG target.

Return type `array_like`

Notes

- Current version (02/18/20) is version 215 on the wiki.
- See *set_target_bits()* for other parameters.

desitarget.cuts.**isLRG_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *zfiberflux=None*, *ggood=None*, *w2flux=None*, *primary=None*, *south=True*)

(see, e.g., *isLRG()*).

Notes

- the *ggood* and *w2flux* inputs are an attempt to maintain backwards-compatibility with the mocks.

desitarget.cuts.**isMWSSTAR_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *primary=None*, *south=True*)

Select a reasonable range of g-r colors for MWS targets. Returns a boolean array.

Parameters

- **rflux, zflux, w1flux, w2flux**(*gflux,*) – array_like The flux in nano-maggies of g, r, z, w1, and w2 bands.
- **primary** – array_like or None Set to `True` for objects to initially consider as possible targets. Defaults to everything being `True`.
- **south** – boolean, defaults to `True` Use color-cuts based on photometry from the "south" (DECaLS) as opposed to the "north" (MzLS+BASS).

Returns boolean array, True if the object has colors like an old stellar population, which is what we expect for the main MWS sample

Return type mask

Notes

The full MWS target selection also includes PSF-like and fracflux cuts and will include Gaia information; this function is only to enforce a reasonable range of color/TEFF when simulating data.

`desitarget.cuts.`**isMWS_WD**(*primary=None*, *gaia=None*, *galb=None*, *astrometricexcessnoise=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *parallaxovererror=None*, *photbprpexcessfactor=None*, *astrometricsigma5dmax=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*, *paramssolved=None*)

Set bits for WHITE DWARF Milky Way Survey targets.

:param see *set_target_bits()* for parameters.:

Returns

array_like. True if and only if the object is a MWS-WD target.

Return type mask

Notes: - Current version (08/01/18) is version 121 on the wiki.

`desitarget.cuts.`**isMWS_main**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *gnobs=None*, *rnobs=None*, *gfracmasked=None*, *rfracmasked=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *parallaxerr=None*, *obs_rflux=None*, *objtype=None*, *gaia=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*, *gaiaaen=None*, *gaiadupsource=None*, *paramssolved=None*, *primary=None*, *south=True*)

Set bits for main `MWS` targets.

:param see *set_target_bits()* for parameters.:

Returns

array_like. `True` if and only if the object is a `MWS_BROAD` target.

mask2 [array_like.] `True` if and only if the object is a `MWS_MAIN_RED` target.

mask3 [array_like.] `True` if and only if the object is a `MWS_MAIN_BLUE` target.

Return type mask1

Notes

- as of 11/2/18, based on version 158 on the wiki.

desitarget.cuts.**isMWS_main_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *parallaxerr=None*, *obs_rflux=None*, *objtype=None*, *paramssolved=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*, *gaiaaen=None*, *primary=None*, *south=True*)

Set of color-based cuts used by MWS target selection classes (see, e.g., *isMWS_main()* for parameters).

desitarget.cuts.**isMWS_nearby**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *objtype=None*, *gaia=None*, *primary=None*, *paramssolved=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *parallaxerr=None*, *obs_rflux=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*)

Set bits for NEARBY Milky Way Survey targets.

:param see *set_target_bits()* for parameters.:

Returns

array_like. True if and only if the object is a MWS-NEARBY target.

Return type mask

Notes: - Current version (09/20/18) is version 129 on the wiki.

desitarget.cuts.**isQSO_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *optical=False*, *south=True*)

Tests if sources have quasar-like colors in a color box. (see, e.g., *isQSO_cuts()*).

desitarget.cuts.**isQSO_cuts**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *w1snr=None*, *w2snr=None*, *deltaChi2=None*, *maskbits=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *release=None*, *objtype=None*, *primary=None*, *optical=False*, *south=True*)

Definition of QSO target classes from color cuts. Returns a boolean array.

Parameters

- **optical** (boolean, defaults to False) – Apply just optical color-cuts.
- **south** (boolean, defaults to True) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if south=False, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns True for objects that pass the quasar color/morphology/logic cuts.

Return type array_like

Notes

- Current version (06/05/19) is version 176 on the wiki.
- See *set_target_bits()* for other parameters.

desitarget.cuts.**isQSO_randomforest**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *objtype=None*, *release=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *deltaChi2=None*, *maskbits=None*, *primary=None*, *south=True*)

Definition of QSO target classes from a Random Forest. Returns a boolean array.

Parameters **south** (boolean, defaults to True) – If False, shift photometry to the Northern (BASS/MzLS) imaging system.

Returns `True` for objects that are Random Forest quasar targets.

Return type `array_like`

Notes

- Current version (04/05/19) is version 173 on the wiki.
- See *`set_target_bits()`* for other parameters.

`desitarget.cuts.`**`isSTD`**(*gflux=None*, *rflux=None*, *zflux=None*, *primary=None*, *gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *gfracmasked=None*, *rfracmasked=None*, *zfracmasked=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *gfluxivar=None*, *rfluxivar=None*, *zfluxivar=None*, *objtype=None*, *gaia=None*, *astrometricexcessnoise=None*, *paramssolved=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *dupsource=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*, *bright=False*, *usegaia=True*, *south=True*)

Select STD targets using color cuts and photometric quality cuts (PSF-like and fracflux). See isSTD_colors() for additional info.

Parameters

- **`rflux, zflux`** (*`gflux,`*) – array_like The flux in nano-maggies of g, r, z bands.
- **`primary`** – array_like or None Set to `True` for objects to initially consider as possible targets. Defaults to everything being `True`.
- **`rfracflux, zfracflux`** (*`gfracflux,`*) – array_like Profile-weighted fraction of the flux from other sources divided by the total flux in g, r and z bands.
- **`rfracmasked, zfracmasked`** (*`gfracmasked,`*) – array_like Fraction of masked pixels in the g, r and z bands.
- **`rnobs, znobs`** (*`gnobs,`*) – array_like The number of observations (in the central pixel) in g, r and z.
- **`rfluxivar, zfluxivar`** (*`gfluxivar,`*) – array_like The flux inverse variances in g, r, and z bands.
- **`objtype`** – array_like or None The TYPE column of the catalogue to restrict to point sources.
- **`gaia`** – boolean array_like or None True if there is a match between this object in the Legacy Surveys and in Gaia.
- **`astrometricexcessnoise`** – array_like or None Excess noise of the source in Gaia.
- **`paramssolved`** – array_like or None How many parameters were solved for in Gaia.
- **`pmdec, parallax`** (*`pmra,`*) – array_like or None Gaia-based proper motion in RA and Dec and parallax
- **`dupsource`** – array_like or None Whether the source is a duplicate in Gaia.
- **`gaiabmag, gaiarmag`** (*`gaiagmag,`*) – array_like or None Gaia-based g-, b- and r-band MAGNITUDES.
- **`bright`** – boolean, defaults to `False` if `True` apply magnitude cuts for "bright" conditions; otherwise, choose "normal" brightness standards. Cut is performed on *gaiagmag*.
- **`usegaia`** – boolean, defaults to `True` if `True` then call *`isSTD_gaia()`* to set the logic cuts. If Gaia is not available (perhaps if you're using mocks) then send `False`, in which

case we use the LS r-band magnitude as a proxy for the Gaia G-band magnitude (ignoring—incorrectly—that we have already corrected for Galactic extinction.)

- **south** – boolean, defaults to `True` Use color-cuts based on photometry from the "south" (DECaLS) as opposed to the "north" (MzLS+BASS).

Returns boolean array, True if the object has colors like a STD star.

Return type mask

Notes

- Gaia-based quantities are as in the Gaia data model.
- Current version (08/01/18) is version 127 on the wiki.

`desitarget.cuts.`**`isSTD_colors`**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *primary=None*, *south=True*)

Select STD stars based on Legacy Surveys color cuts. Returns a boolean array.

Parameters

- **rflux, zflux, w1flux, w2flux** (*gflux,*) – array_like The flux in nano-maggies of g, r, z, w1, and w2 bands.
- **primary** – array_like or None Set to `True` for objects to initially consider as possible targets. Defaults to everything being `True`.
- **south** – boolean, defaults to `True` Use color-cuts based on photometry from the "south" (DECaLS) as opposed to the "north" (MzLS+BASS).

Returns boolean array, True if the object has colors like a STD star target

Return type mask

Notes

- Current version (08/01/18) is version 121 on the wiki.

`desitarget.cuts.`**`isSTD_gaia`**(*primary=None*, *gaia=None*, *astrometricexcessnoise=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *dupsource=None*, *paramssolved=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*)

Gaia quality cuts used to define STD star targets.

Parameters

- **primary** – array_like or None Set to `True` for objects to initially consider as possible targets. Defaults to everything being `True`.
- **gaia** – boolean array_like or None True if there is a match between this object in the Legacy Surveys and in Gaia.
- **astrometricexcessnoise** – array_like or None Excess noise of the source in Gaia (as in the Gaia data model).
- **pmdec, parallax** (*pmra,*) – array_like or None Gaia-based proper motion in RA and Dec and parallax (same units as the Gaia data model).
- **dupsource** – array_like or None Whether the source is a duplicate in Gaia (as in the Gaia data model).

- **paramssolved** – array_like or None How many parameters were solved for in Gaia (as in the Gaia data model).
- **gaiabmag, gaiarmag** (*gaiagmag,*) – array_like or None Gaia-based g, b and r MAGNITUDES (not Galactic-extinction-corrected). (same units as the Gaia data model).

Returns boolean array, True if the object passes Gaia quality cuts.

Return type mask

Notes: - Current version (08/01/18) is version 121 on the wiki.

desitarget.cuts.**notinBGS_mask**(*gnobs=None*, *rnobs=None*, *znobs=None*, *primary=None*, *gfracmasked=None*, *rfracmasked=None*, *zfracmasked=None*, *gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *gfracin=None*, *rfracin=None*, *zfracin=None*, *w1snr=None*, *gfluxivar=None*, *rfluxivar=None*, *zfluxivar=None*, *Grr=None*, *gaiagmag=None*, *maskbits=None*, *targtype=None*)

Standard set of masking cuts used by all BGS target selection classes (see, e.g., *isBGS()* for parameters).

desitarget.cuts.**notinELG_mask**(*maskbits=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *primary=None*)

Standard set of masking cuts used by all ELG target selection classes. (see *set_target_bits()* for parameters).

desitarget.cuts.**notinLRG_mask**(*primary=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *zfiberflux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *rflux_snr=None*, *zflux_snr=None*, *w1flux_snr=None*, *maskbits=None*)

See *isLRG()* for details.

Returns `True` if and only if the object is NOT masked for poor quality.

Return type `array_like`

desitarget.cuts.**notinMWS_main_mask**(*gaia=None*, *gfracmasked=None*, *gnobs=None*, *gflux=None*, *rfracmasked=None*, *rnobs=None*, *rflux=None*, *gaiadupsource=None*, *primary=None*)

Standard set of masking-based cuts used by MWS target selection classes (see, e.g., *isMWS_main()* for parameters).

desitarget.cuts.**select_targets**(*infiles, numproc=4, qso_selection='randomforest', gaiamatch=False, nside=None, pixlist=None, bundlefiles=None, extra=None, radecbox=None, radecrad=None, mask=True, tcnames=['ELG', 'QSO', 'LRG', 'MWS', 'BGS', 'STD'], survey='main', resolvetargs=True, backup=True*)

Process input files in parallel to select targets.

Parameters

- **infiles** (`list` or *str*) – A list of input filenames (tractor or sweep files) OR a single filename.
- **numproc** (`int`, optional, defaults to 4) – The number of parallel processes to use.
- **qso_selection** (`str`, optional, defaults to `'randomforest'`) – The algorithm to use for QSO selection; valid options are `'colorcuts'` and `'randomforest'`.
- **gaiamatch** (`boolean`, optional, defaults to `False`) – If `True`, match to Gaia DR2 chunks files and populate Gaia columns to facilitate the MWS and STD selections.
- **nside** (`int`, optional, defaults to *None*) – The (NESTED) HEALPixel nside to be used with the *pixlist* and *bundlefiles* inputs.

- **pixlist** (list or *int*, optional, defaults to *None*) – Only return targets in a set of (NESTED) HEALpixels at the supplied *nside*. Also useful for parallelizing as input files will only be processed if they touch a pixel in the passed list.
- **bundlefiles** (int, defaults to *None*) – If not *None*, then instead of selecting targets, print, to screen, the slurm script that will approximately balance the input file distribution at *bundlefiles* files per node. So, for instance, if *bundlefiles* is 100 then commands would be returned with the correct *pixlist* values set to pass to the code to pack at about 100 files per node across all of the passed *infiles*.
- **extra** (str, optional) – Extra command line flags to be passed to the executable lines in the output slurm script. Used in conjunction with *bundlefiles*.
- **radecbox** (list, defaults to *None*) – 4-entry list of coordinates [ramin, ramax, decmin, decmax] forming the edges of a box in RA/Dec (degrees). Only targets in this box region will be processed.
- **radecrad** (list, defaults to *None*) – 3-entry list of coordinates [ra, dec, radius] forming a "circle" on the sky. For RA/Dec/radius in degrees. Only targets in this circle region will be processed.
- **mask** (boolean, optional, defaults to True) – Send False to turn off any masking cuts based on the *MASKBITS* column. The default behavior is to always mask using *MASKBITS*.
- **tcnames** (list, defaults to running all target classes) – A list of strings, e.g. ['QSO','LRG']. If passed, process targeting only for those specific target classes. A useful speed-up when testing. Options include ["ELG", "QSO", "LRG", "MWS", "BGS", "STD"].
- **survey** (str, defaults to 'main') – Specifies which target masks yaml file and target selection cuts to use. Options are 'main' and 'svX' (where X is 1, 2, 3 etc.) for the main survey and different iterations of SV, respectively.
- **resolvetargs** (boolean, optional, defaults to True) – If True, resolve targets into northern targets in northern regions and southern targets in southern regions.
- **backup** (boolean, optional, defaults to True) – If True, also run the Gaia-only BACKUP_BRIGHT/FAINT targets.

Returns The subset of input targets which pass the cuts, including extra columns for DESI_TARGET, BGS_TARGET, and MWS_TARGET target selection bitmasks.

Return type ndarray

Notes

- if numproc==1, use serial code instead of parallel.
- only one of pixlist, radecbox, radecrad should be passed. They are all intended to denote regions on the sky, using different formalisms.

desitarget.cuts.**set_target_bits**(*photsys_north*, *photsys_south*, *obs_rflux*, *gflux*, *rflux*, *zflux*, *w1flux*, *w2flux*, *gfiberflux*, *rfiberflux*, *zfiberflux*, *objtype*, *release*, *gfluxivar*, *rfluxivar*, *zfluxivar*, *gnobs*, *rnobs*, *znobs*, *gfracflux*, *rfracflux*, *zfracflux*, *gfracmasked*, *rfracmasked*, *zfracmasked*, *gfracin*, *rfracin*, *zfracin*, *gallmask*, *rallmask*, *zallmask*, *gsnr*, *rsnr*, *zsnr*, *w1snr*, *w2snr*, *deltaChi2*, *dchisq*, *gaia*, *pmra*, *pmdec*, *parallax*, *parallaxovererror*, *parallaxerr*, *gaiagmag*, *gaiabmag*, *gaiarmag*, *gaiaaen*, *gaiadupsource*, *gaiaparamssolved*, *gaiabprpfactor*, *gaiasigma5dmax*, *galb*, *tcnames*, *qso_optical_cuts*, *qso_selection*, *maskbits*, *Grr*, *refcat*, *primary*, *resolvetargs=True*)

Perform target selection on parameters, return target mask arrays.

Parameters

- **photsys_south** (*photsys_north,*) – `True` for objects that were drawn from northern (MzLS/BASS) or southern (DECaLS) imaging, respectively.
- **obs_rflux** (`ndarray`) – *rflux* but WITHOUT any Galactic extinction correction.
- **rflux, zflux, w1flux, w2flux** (*gflux,*) – The flux in nano-maggies of g, r, z, W1 and W2 bands. Corrected for Galactic extinction.
- **rfiberflux, zfiberflux** (*gfiberflux,*) – Predicted fiber flux in 1 arcsecond seeing in g/r/z-band. Corrected for Galactic extinction.
- **release** (*objtype,*) – The Legacy Surveys imaging `TYPE` and `RELEASE` columns.
- **rfluxivar, zfluxivar** (*gfluxivar,*) – The flux inverse variances in g, r, and z bands.
- **rnobs, znobs** (*gnobs,*) – The number of observations (in the central pixel) in g, r and z.
- **rfracflux, zfracflux** (*gfracflux,*) – Profile-weighted fraction of the flux from other sources divided by the total flux in g, r and z bands.
- **rfracmasked, zfracmasked** (*gfracmasked,*) – Fraction of masked pixels in the g, r and z bands.
- **rallmask, zallmask** (*gallmask,*) – Bitwise mask set if the central pixel from all images satisfy each condition in g, r, z.
- **rsnr, zsnr, w1snr, w2snr** (*gsnr,*) – Signal-to-noise in g, r, z, W1 and W2 defined as the flux per band divided by sigma (flux x sqrt of the inverse variance).
- **deltaChi2** (`ndarray`) – chi2 difference between PSF and SIMP, dchisq_PSF - dchisq_SIMP.
- **dchisq** (`ndarray`) – Difference in chi2 between successively more-complex model fits. Columns are model fits, in order, of PSF, REX, EXP, DEV, COMP.
- **gaia** (`ndarray`) – `True` if there is a match between this object in the Legacy Surveys and in Gaia.
- **pmdec, parallax, parallaxovererror** (*pmra,*) – Gaia-based proper motion in RA and Dec, and parallax and error.
- **gaiabmag, gaiarmag** (*gaiagmag,*) – Gaia-based g-, b- and r-band MAGNITUDES.
- **gaiadupsource, gaiaparamssolved** (*gaiaaen,*) – Gaia-based measures of Astrometric Excess Noise, whether the source is a duplicate, and how many parameters were solved for.

- **gaiasigma5dmax** (*gaiabprpfactor,*) – Gaia_based BP/RP excess factor and longest semi-major axis of 5-d error ellipsoid.
- **galb** (ndarray) – Galactic latitude (degrees).
- **tcnames** (list, defaults to running all target classes) – A list of strings, e.g. ['QSO','LRG']. If passed, process only only those specific target classes. A useful speed-up for tests. Options include ["ELG", "QSO", "LRG", "MWS", "BGS", "STD"].
- **qso_optical_cuts** (boolean defaults to False) – Apply just optical color-cuts when selecting QSOs with qso_selection="colorcuts".
- **qso_selection** (str, optional, defaults to *'randomforest'*) – The algorithm to use for QSO selection; valid options are *'colorcuts'* and *'randomforest'*
- **maskbits** (*boolean array_like or None*) – General Legacy Surveys mask bits.
- **Grr** (*array_like or None*) – Gaia G band magnitude minus observational r magnitude.
- **primary** (ndarray) – True for objects that should be considered when setting bits.
- **survey** (str, defaults to 'main') – Specifies which target masks yaml file and target selection cuts to use. Options are 'main' and 'svX' (X is 1, 2, etc.) for the main survey and different iterations of SV, respectively.
- **resolvetargs** (boolean, optional, defaults to True) – If True, if only northern (southern) sources are passed then only apply the northern (southern) cuts to those sources.

Returns (desi_target, bgs_target, mws_target) where each element is an ndarray of target selection bitmask flags for each object.

Return type ndarray

Notes

- Gaia quantities have units as for the Gaia data model.

desitarget.cuts.**shift_photo_north**(*gflux=None*, *rflux=None*, *zflux=None*)

Convert fluxes in the northern (BASS/MzLS) to the southern (DECaLS) system.

Parameters **rflux, zflux** (*gflux,*) – The flux in nano-maggies of g, r, z bands.

Returns

Return type The equivalent fluxes shifted to the southern system.

Notes

- see also https://desi.lbl.gov/DocDB/cgi-bin/private/RetrieveFile?docid=3390;filename=Raichoor_DESI_05Dec2017.pdf;version=1

desitarget.cuts.**shift_photo_north_pure**(*gflux=None*, *rflux=None*, *zflux=None*)

Same as *shift_photo_north_pure()* accounting for zero fluxes.

Parameters **rflux, zflux** (*gflux,*) – The flux in nano-maggies of g, r, z bands.

Returns

Return type The equivalent fluxes shifted to the southern system.

Notes

- see also https://desi.lbl.gov/DocDB/cgi-bin/private/RetrieveFile?docid=3390;filename=Raichoor_DESI_05Dec2017.pdf;version=1

desitarget.cuts.**unextinct_fluxes**(*objects*)
Calculate unextincted DECam and WISE fluxes.

Parameters **objects** – array or Table with columns FLUX_G, FLUX_R, FLUX_Z, MW_TRANSMISSION_G, MW_TRANSMISSION_R, MW_TRANSMISSION_Z, FLUX_W1, FLUX_W2, MW_TRANSMISSION_W1, MW_TRANSMISSION_W2

Returns array or Table with columns GFLUX, RFLUX, ZFLUX, W1FLUX, W2FLUX

Output type is Table if input is Table, otherwise numpy structured array

1.6 desitarget.gaiamatch

Useful Gaia matching and manipulation routines.

desitarget.gaiamatch.**_get_gaia_nside**()
Grab the HEALPixel nside to be used throughout this module.

Returns The HEALPixel nside number for Gaia file creation and retrieval.

Return type int

desitarget.gaiamatch.**find_gaia_files**(*objs*, *neighbors=True*, *radec=False*)
Find full paths to Gaia healpix files for objects by RA/Dec.

Parameters

- **objs** (ndarray) – Array of objects. Must contain at least the columns "RA" and "DEC".
- **neighbors** (bool, optional, defaults to True) – Also return all neighboring pixels that touch the files of interest in order to prevent edge effects (e.g. if a Gaia source is 1 arcsec away from a primary source and so in an adjacent pixel).
- **radec** (bool, optional, defaults to False) – If True then the passed *objs* is an [RA, Dec] list instead of a rec array.

Returns A list of all Gaia files that need to be read in to account for objects at the passed locations.

Return type list

Notes

- The environment variable $GAIA_DIR must be set.

desitarget.gaiamatch.**find_gaia_files_beyond_gal_b**(*mingalb*, *neighbors=True*)
Find full paths to Gaia healpix files beyond a Galactic b.

Parameters

- **mingalb** (float) – Closest latitude to Galactic plane to return HEALPixels (e.g. send 10 to limit to pixels beyond -10o <= b < 10o).
- **neighbors** (bool, optional, defaults to True) – Also return files corresponding to neighboring pixels that touch in order to prevent edge effects (e.g. if a Gaia source might be 1 arcsec beyond mingalb and so in an adjacent pixel).

Returns All Gaia files that need to be read in to account for objects further from the Galactic plane than *mingalb*.

Return type list

Notes

- The environment variable $GAIA_DIR must be set.
- *desitarget.geomask.hp_beyond_gal_b()* is already quite inclusive, so you may retrieve some extra files along the *mingalb* boundary.

desitarget.gaiamatch.**find_gaia_files_box**(*gaiabounds*, *neighbors=True*)

Find full paths to Gaia healpix files in an RA/Dec box.

Parameters

- **gaiabounds** (list) – A region of the sky bounded by RA/Dec. Pass as a 4-entry list to represent an area bounded by [RAmin, RAmax, DECmin, DECmax]
- **neighbors** (bool, optional, defaults to True) – Also return files corresponding to all neighboring pixels that touch the files that touch the box in order to prevent edge effects (e.g. if a Gaia source might be 1 arcsec outside of the box and so in an adjacent pixel)

Returns A list of all Gaia files that need to be read in to account for objects in the passed box.

Return type list

Notes

- Uses the *healpy* routines that rely on *fact*, so the usual warnings about returning different pixel sets at different values of *fact* apply. See: https://healpy.readthedocs.io/en/latest/generated/healpy.query_polygon.html
- The environment variable $GAIA_DIR must be set.

desitarget.gaiamatch.**find_gaia_files_hp**(*nside*, *pixlist*, *neighbors=True*)

Find full paths to Gaia healpix files in a set of HEALPixels.

Parameters

- **nside** (int) – (NESTED) HEALPixel nside.
- **pixlist** (list or *int*) – A set of HEALPixels at *nside*.
- **neighbors** (bool, optional, defaults to True) – Also return files corresponding to all neighbors that touch the pixels in *pixlist* to prevent edge effects (e.g. a Gaia source is 1 arcsec outside of *pixlist* and so in an adjacent pixel).

Returns A list of all Gaia files that need to be read in to account for objects in the passed list of pixels.

Return type list

Notes

- The environment variable $GAIA_DIR must be set.

desitarget.gaiamatch.**find_gaia_files_tiles**(*tiles=None*, *neighbors=True*)

Parameters

- **tiles** (ndarray) – Array of tiles, or None to use all DESI tiles from desimodel.io.load_tiles().
- **neighbors** (bool, optional, defaults to True) – Also return all neighboring pixels that touch the files of interest in order to prevent edge effects (e.g. if a Gaia source is 1 arcsec away from a primary source and so in an adjacent pixel).

Returns A list of all Gaia files that touch the passed tiles.

Return type list

Notes

- The environment variables $GAIA_DIR and $DESIMODEL must be set.

desitarget.gaiamatch.**gaia_csv_to_fits**(*numproc=4*)
Convert files in $GAIA_DIR/csv to files in $GAIA_DIR/fits.

Parameters **numproc** (int, optional, defaults to 4) – The number of parallel processes to use.

Returns But the archived Gaia CSV files in $GAIA_DIR/csv are converted to FITS files in the directory $GAIA_DIR/fits. Also, a look-up table is written to $GAIA_DIR/fits/hpx-to-files.pickle for which each index is an nside=_get_gaia_nside(), nested scheme HEALPixel and each entry is a list of the FITS files that touch that HEAPixel.

Return type Nothing

Notes

- The environment variable $GAIA_DIR must be set.
- if numproc==1, use the serial code instead of the parallel code.
- Runs in 1-3 hours (depending on node) with numproc=32 for 60,000 files.

desitarget.gaiamatch.**gaia_dr_from_ref_cat**(*refcat*)
Determine the Gaia DR from an array of values, check it's unique.

Parameters **ref_cat** (ndarray or *str*) – A *REF_CAT* string or an array of *REF_CAT* strings (e.g. b"G2").

Returns The corresponding Data Release number (e.g. 2)

Return type ndarray

Notes

- In reality, only strips the final integer off strings like "X3". So, can generically be used for that purpose.

desitarget.gaiamatch.**gaia_fits_to_healpix**(*numproc=4*)
Convert files in $GAIA_DIR/fits to files in $GAIA_DIR/healpix.

Parameters **numproc** (int, optional, defaults to 4) – The number of parallel processes to use.

Returns But the archived Gaia FITS files in $GAIA_DIR/fits are rearranged by HEALPixel in the directory $GAIA_DIR/healpix. The HEALPixel sense is nested with nside=_get_gaia_nside(), and each file in $GAIA_DIR/healpix is called healpix-xxxxx.fits, where xxxxx corresponds to the HEALPixel number.

Return type Nothing

Notes

- The environment variable $GAIA_DIR must be set.
- if numproc==1, use the serial code instead of the parallel code.
- Runs in about 1-3 hours with numproc=32 for 60,000 files.

desitarget.gaiamatch.**get_gaia_dir**()
Convenience function to grab the Gaia environment variable.

Returns The directory stored in the $GAIA_DIR environment variable.

Return type str

desitarget.gaiamatch.**is_in_Galaxy**(*objs*, *radec=False*)
An (l, b) cut developed by Boris Gaensicke to avoid the Galaxy.

Parameters

- **objs** (ndarray) – Array of objects. Must contain at least the columns "RA" and "DEC".
- **radec** (bool, optional, defaults to False) – If True then the passed *objs* is an [RA, Dec] list instead of a rec array.

Returns A boolean array that is True for objects that are close to the Galaxy and False for objects that aren't.

Return type ndarray

desitarget.gaiamatch.**make_gaia_files**(*numproc=4*, *download=False*)
Make the HEALPix-split Gaia DR2 files used by desitarget.

Parameters

- **numproc** (int, optional, defaults to 4) – The number of parallel processes to use.
- **download** (bool, optional, defaults to False) – If True then wget the Gaia DR2 csv files from ESA.

Returns

But produces: - Full Gaia DR2 CSV files in $GAIA_DIR/csv. - FITS files with columns from *ingaiadatamodel* in $GAIA_DIR/fits. - FITS files reorganized by HEALPixel in $GAIA_DIR/healpix.

The HEALPixel sense is nested with nside=_get_gaia_nside(), and each file in $GAIA_DIR/healpix is called healpix-xxxxx.fits, where xxxxx corresponds to the HEALPixel number.

Return type Nothing

Notes

- The environment variable $GAIA_DIR must be set.
- if numproc==1, use the serial code instead of the parallel code.
- Runs in about 26 hours if download is `True`.
- Runs in 1-3 hours with numproc=32 if download is `False`.

`desitarget.gaiamatch.`**`match_gaia_to_primary`**(*objs, matchrad=1.0, retaingaia=False, gaiabounds=[0.0, 360.0, -90.0, 90.0]*)

Match a set of objects to Gaia healpix files and return the Gaia information.

Parameters

- **`objs`** (`ndarray`) – Must contain at least "RA" and "DEC".
- **`matchrad`** (`float`, optional, defaults to 1 arcsec) – The matching radius in arcseconds.
- **`retaingaia`** (`float`, optional, defaults to False) – If set, return all of the Gaia information in the "area" occupied by the passed objects (whether a Gaia object matches a passed RA/Dec or not.) THIS ASSUMES THAT THE PASSED OBJECTS ARE FROM A SWEEPS file and that the integer values nearest the maximum and minimum passed RAs and Decs fairly represent the areal "edges" of that file.
- **`gaiabounds`** (`list`, optional, defaults to the whole sky) – Used in conjunction with *retaingaia* to determine over what area to retrieve Gaia objects that don't match a sweeps object. Pass a 4-entry list to represent an area bounded by [RAmin, RAmax, DECmin, DECmax]

Returns The matching Gaia information for each object, where the returned format and columns correspond to *desitarget.gaiamatch.gaiadatamodel*

Return type `ndarray`

Notes

- The first len(objs) objects correspond row-by-row to the passed objects.
- For objects that do NOT have a match in the Gaia files, the "REF_ID" column is set to -1, and all other columns are zero.
- If *retaingaia* is True then objects after the first len(objs) objects are Gaia objects that do not have a sweeps match but that are in the area bounded by *gaiabounds*

`desitarget.gaiamatch.`**`match_gaia_to_primary_single`**(*objs*, *matchrad=1.0*)

Match ONE object to Gaia "chunks" files and return the Gaia information.

Parameters

- **`objs`** (`ndarray`) – Must contain at least "RA" and "DEC". MUST BE A SINGLE ROW.
- **`matchrad`** (`float`, optional, defaults to 1 arcsec) – The matching radius in arcseconds.

Returns The matching Gaia information for the object, where the returned format and columns correspond to *desitarget.secondary.gaiadatamodel*

Return type `ndarray`

Notes

- If the object does NOT have a match in the Gaia files, the "REF_ID" column is set to -1, and all other columns are zero

desitarget.gaiamatch.**pop_gaia_columns**(*inarr*, *popcols*)
Convenience function to pop columns off an input array.

Parameters

- **inarr** (ndarray) – Structured array with various column names.
- **popcols** (list) – List of columns to remove from the input array.

Returns Input array with columns in cols removed.

Return type ndarray

desitarget.gaiamatch.**pop_gaia_coords**(*inarr*)
Convenience function to pop GAIA_RA and GAIA_DEC columns off an array

Parameters **inarr** (ndarray) – Structured array with various column names.

Returns Input array with columns called "GAIA_RA" and/or "GAIA_DEC" removed.

Return type ndarray

desitarget.gaiamatch.**read_gaia_file**(*filename*, *header=False*, *addobjid=False*)
Read in a Gaia healpix file in the appropriate format for desitarget.

Parameters

- **filename** (str) – File name of a single Gaia "healpix-" file.
- **header** (bool, optional, defaults to False) – If True then return (data, header) instead of just data.
- **addobjid** (bool, optional, defaults to False) – Include, in the output, two additional columns. A column "GAIA_OBJID" that is the integer number of each row read from file and a column "GAIA_BRICKID" that is the integer number of the file itself.

Returns Gaia data translated to targeting format (upper-case etc.) with the columns corresponding to *desitarget.gaiamatch.gaiadatamodel*

Return type ndarray

Notes

- A better location for this might be in *desitarget.io*?

desitarget.gaiamatch.**scrape_gaia**(*url='http://cdn.gea.esac.esa.int/Gaia/gdr2/gaia_source/csv/'*, *nfiletest=None*)
Retrieve the bulk CSV files released by the Gaia collaboration.

Parameters

- **url** (str) – The web directory that hosts the archived Gaia CSV files.
- **nfiletest** (int, optional, defaults to None) – If an integer is sent, only retrieve this number of files, for testing.

Returns But the archived Gaia CSV files are written to $GAIA_DIR/csv.

Return type Nothing

Notes

- The environment variable $GAIA_DIR must be set.
- Runs in about 26 hours for 60,000 Gaia files.

desitarget.gaiamatch.**write_gaia_matches**(*infiles*, *numproc=4*, *outdir='.'*)
Match sweeps files to Gaia and rewrite with the Gaia columns added

Parameters

- **infiles** (list or *str*) – A list of input filenames (sweep files) OR a single filename. Arrays in the files must contain at least the columns "RA" and "DEC".
- **numproc** (int, optional, defaults to 4) – The number of parallel processes to use.
- **outdir** (str, optional, default to the current directory) – The directory to write the files.

Returns The original sweeps files with the columns in *gaiadatamodel* added (except for the columns *GAIA_RA* and *GAIA_DEC*) are written to file. The filename is the same as the input filename with the ".fits" replaced by "-gaia$DRmatch.fits" where $DR is extracted from the $GAIA_DIR environment variable.

Return type ndarray

Notes

- if numproc==1, use the serial code instead of the parallel code.
- The environment variable $GAIA_DIR must be set.

1.7 desitarget.geomask

Utility functions for geometry on the sky, masking, etc.

desitarget.geomask.**add_hp_neighbors**(*nside*, *pixnum*)
Add all neighbors in the NESTED scheme to a set of HEALPixels.

Parameters

- **nside** (int) – (NESTED) HEALPixel nside.
- **pixnum** (list or *int* or *~numpy.ndarray*) – HEALPixel numbers (or a single HEALPixel number).

Returns The passed list of pixels with all neighbors added to the list. Only unique pixels are returned, so any duplicate integers in the passed *pixnum* are removed.

Return type list

Notes

- Syntactic sugar around *healpy.pixelfunc.get_all_neighbours()*.

desitarget.geomask.**box_area**(*radecbox*)
Determines the area of an RA, Dec box in square degrees.

Parameters **radecbox** (list) – 4-entry list of coordinates [ramin, ramax, decmin, decmax] forming the edges of a box in RA/Dec (degrees).

Returns The area of the passed box in square degrees.

Return type list

desitarget.geomask.**brick_names_touch_hp**(*nside*, *numproc=1*, *fact=1048576*)

Determine which of a set of brick names touch a set of HEALPixels.

Parameters

- **nside** (int) – (NESTED) HEALPixel nside.
- **numproc** (int, optional, defaults to 1) – The number of parallel processes to use.
- **fact** (int, optional defaults to 2**20) – see documentation for *healpy.query_polygon()*.

Returns A list of lists of input brick names that touch each HEALPixel at *nside*. So, e.g. for *nside=2* the returned list will have 48 entries, and, for example, output[0] will be a list of names of bricks that touch HEALPixel 0.

Return type list

Notes

- Runs in ~65 (10) secs for numproc=1 (32) at nside=2 (fact=4).
- Runs in ~325 (20) secs for numproc=1 (32) at nside=64 (fact=4).
- Takes ~2x as long at the default fact=2**20 compared to fact=4, but fact=2**20 returns far fewer bricks for small *nside*.

desitarget.geomask.**bundle_bricks**(*pixnum*, *maxpernode*, *nside*, *brickspersec=1.0*, *prefix='targets'*, *gather=True*, *surveydirs=None*, *extra=None*, *seed=None*)

Determine the optimal packing for bricks collected by HEALpixel integer.

Parameters

- **pixnum** (np.array) – List of integers, e.g., HEALPixel numbers occupied by a set of bricks (e.g. array([16, 16, 16. . . 12 , 13, 19])).
- **maxpernode** (int) – The maximum number of pixels to bundle together (e.g., if you were trying to pass maxpernode bricks, delineated by the HEALPixels they occupy, parallelized across a set of nodes).
- **nside** (int) – The HEALPixel nside number that was used to generate *pixnum* (NESTED scheme).
- **brickspersec** (float, optional, defaults to 1.) – The rough number of bricks processed per second by the code (parallelized across a chosen number of nodes)
- **prefix** (str, optional, defaults to 'targets') – Corresponds to the executable "X" that is run as select_X for a target type. This could be 'randoms', 'skies', 'targets', 'gfas'. Also, 'supp-skies' can be passed to cover supplemental skies.
- **gather** (bool, optional, defaults to True) – If True then provide a final command for combining all of the HEALPix-split files into one large file. If False, do not provide that command.

- **surveydirs** (list) – Root directories for a Legacy Surveys Data Release. The first element of the list is interpreted as the main directory. IF the list is of length two then the second directory is supplied as "-s2" in the output script. (e.g. ["/global/project/projectdirs/cosmo/data/legacysurvey/dr6"]).
- **extra** (str, optional) – Extra command line flags to be passed to the executable lines in the output slurm script.
- **seed** (int, optional, defaults to 1) – Random seed for file name. Only relevant for *prefix='randoms'*.

Returns

- *Nothing, but prints commands to screen that would facilitate running a*
- *set of bricks by HEALPixel integer with the total number of bricks not*
- *to exceed maxpernode. Also prints how many bricks would be on each node.*

Notes

h/t https://stackoverflow.com/questions/7392143/python-implementations-of-packing-algorithm

desitarget.geomask.**cap_area**(*theta*)

True area of a circle of a given radius drawn on the surface of a sphere

Parameters **theta** (*array_like*) – (angular) radius of a circle drawn on the surface of the unit sphere (in DEGREES)

Returns **area** – surface area on the sphere included within the passed angular radius

Return type array_like

Notes

- The approximate formula pi*theta**2 is only accurate to ~0.0025% at 1o, ~0.25% at 10o, sufficient for bright source mask purposes. But the equation in this function is more general.
- We recast the input array as float64 to circumvent precision issues with np.cos() when radii of only a few arcminutes are passed
- Even for passed radii of 1 (0.1) arcsec, float64 is sufficiently precise to give the correct area to ~0.00043 (~0.043%) using np.cos()

desitarget.geomask.**check_nside**(*nside*)

Flag an error if nside is not OK for NESTED HEALPixels.

Parameters **nside** (int or *~numpy.ndarray*) – The HEALPixel nside number (NESTED scheme) or an array of such numbers.

Returns

Return type Nothing, but raises a ValueRrror for a bad *nside*.

desitarget.geomask.**circle_boundaries**(*RAcens*, *DECcens*, *r*, *nloc*)

Return RAs/Decs of a set of circular boundaries on the sky

Parameters

- **RAcens** (ndarray) – Right Ascension of the centers of the circles (DEGREES)
- **DECcens** (ndarray) – Declination of the centers of the circles (DEGREES)

- **r** (ndarray) – radius of the circles (ARCSECONDS)
- **nloc** (ndarray) – the number of locations to generate, equally spaced around the periphery of *each* circle

Returns

- **ras** (ndarray) –

  The Right Ascensions of nloc equally spaced locations on the periphery of the mask
- **dec** (*array_like.*) –

  The Declinations of nloc equally space locations on the periphery of the mask

desitarget.geomask.**circles**(*x*, *y*, *s*, *c='b'*, *vmin=None*, *vmax=None*, ***kwargs*)

Make a scatter plot of circles. Similar to plt.scatter, but the size of circles are in data scale

Parameters

- **y** (*x,*) – Input data
- **s** (*scalar or array_like, shape (n,)*) – Radius of circles.
- **c** (*color or sequence of color, optional, default : 'b'*) – *c* can be a single color format string, or a sequence of color specifications of length *N*, or a sequence of *N* numbers to be mapped to colors using the *cmap* and *norm* specified via kwargs. Note that *c* should not be a single numeric RGB or RGBA sequence because that is indistinguishable from an array of values to be colormapped. (If you insist, use *color* instead.) *c* can be a 2-D array in which the rows are RGB or RGBA, however.
- **vmax** (*vmin,*) – *vmin* and *vmax* are used in conjunction with *norm* to normalize luminance data. If either are *None*, the min and max of the color array is used.
- **kwargs** (*~matplotlib.collections.Collection* properties) – Eg. alpha, edgecolor(ec), facecolor(fc), linewidth(lw), linestyle(ls), norm, cmap, transform, etc.

Returns paths

Return type *~matplotlib.collections.PathCollection*

Examples

```
>>> a = np.arange(11)
>>> circles(a, a, s=a*0.2, c=a, alpha=0.5, ec='none')
>>> plt.colorbar()
```

References

With thanks to https://gist.github.com/syrte/592a062c562cd2a98a83

desitarget.geomask.**ellipse_boundary**(*RAcen*, *DECcen*, *r*, *e1*, *e2*, *nloc=50*)

Return RA/Dec of an elliptical boundary on the sky

Parameters

- **RAcen** (float) – Right Ascension of the center of the ellipse (DEGREES)
- **DECcen** (float) – Declination of the center of the ellipse (DEGREES)
- **r** (float) – Half-light radius of the ellipse (ARCSECONDS)

- **e1** (float) – First ellipticity component of the ellipse
- **e2** (float) – Second ellipticity component of the ellipse
- **nloc** (int, optional, defaults to 50) – the number of locations to generate, equally spaced around the periphery of the ellipse

Returns

- ndarray – Right Ascensions along the boundary of (each) ellipse
- ndarray – Declinations along the boundary of (each) ellipse

Notes

- **The parametrization is explained at** http://legacysurvey.org/dr4/catalogs/
- **Much of the math is taken from:** https://github.com/dstndstn/tractor/blob/master/tractor/ellipses.py

desitarget.geomask.**ellipse_matrix**(*r*, *e1*, *e2*)

Calculate transformation matrix from half-light-radius to ellipse

Parameters

- **r** (float or ~*numpy.ndarray*) – Half-light radius of the ellipse (ARCSECONDS)
- **e1** (float or ~*numpy.ndarray*) – First ellipticity component of the ellipse
- **e2** (float or ~*numpy.ndarray*) – Second ellipticity component of the ellipse

Returns A 2x2 matrix to transform points measured in coordinates of the effective-half-light-radius to RA/Dec offset coordinates

Return type ndarray

Notes

- **If a float is passed then the output shape is (2,2,1)** otherwise it's (2,2,len(r))
- **The parametrization is explained at** http://legacysurvey.org/dr4/catalogs/
- **Much of the math is taken from:** https://github.com/dstndstn/tractor/blob/master/tractor/ellipses.py

desitarget.geomask.**ellipses**(*x*, *y*, *w*, *h=None*, *rot=0.0*, *c='b'*, *vmin=None*, *vmax=None*, ***kwargs*)

Make a scatter plot of ellipses

Parameters

- **y** (*x,*) – Center of ellipses.
- **h** (*w,*) – Total length (diameter) of horizontal/vertical axis. *h* is set to be equal to *w* by default, ie. circle.
- **rot** (*scalar or array_like, shape (n,)*) – Rotation in degrees (anti-clockwise).
- **c** (*color or sequence of color, optional, default : 'b'*) – *c* can be a single color format string, or a sequence of color specifications of length *N*, or a sequence of *N* numbers to be mapped to colors using the *cmap* and *norm* specified via kwargs.

Note that *c* should not be a single numeric RGB or RGBA sequence because that is indistinguishable from an array of values to be colormapped. (If you insist, use *color* instead.) *c* can be a 2-D array in which the rows are RGB or RGBA, however.

- **vmax** (*vmin,*) – *vmin* and *vmax* are used in conjunction with *norm* to normalize luminance data. If either are *None*, the min and max of the color array is used.
- **kwargs** (*~matplotlib.collections.Collection* properties) – Eg. alpha, edgecolor(ec), facecolor(fc), linewidth(lw), linestyle(ls), norm, cmap, transform, etc.

Returns paths

Return type *~matplotlib.collections.PathCollection*

Examples

```
>>> a = np.arange(11)
>>> ellipses(a, a, w=4, h=a, rot=a*30, c=a, alpha=0.5, ec='none')
>>> plt.colorbar()
```

References

With thanks to https://gist.github.com/syrte/592a062c562cd2a98a83

desitarget.geomask.**hp_beyond_gal_b**(*nside*, *mingalb*, *neighbors=True*)

Find HEALPixels with centers and neighbors beyond a Galactic b.

Parameters

- **nside** (int) – (NESTED) HEALPixel nside.
- **mingalb** (float) – Closest latitude to Galactic plane to return HEALPixels (e.g. send 10 to limit to pixels beyond -10o <= b < 10o).
- **neighbors** (bool, optional, defaults to True) – If False, return all HEALPixels with *centers* beyond the passed *mingalb*. If True also return the neighbors of these pixels (to avoid edge effects).

Returns HEALPixels at the passed *nside* that lie north and south of the passed *mingalb*.

Return type list

Notes

- Pixels are in the NESTED scheme.

desitarget.geomask.**hp_in_box**(*nside*, *radecbox*, *inclusive=True*, *fact=4*)

Determine which HEALPixels touch an RA, Dec box.

Parameters

- **nside** (int) – (NESTED) HEALPixel nside.
- **radecbox** (list) – 4-entry list of coordinates [ramin, ramax, decmin, decmax] forming the edges of a box in RA/Dec (degrees).
- **inclusive** (bool, optional, defaults to True) – see documentation for *healpy.query_polygon()*.

- **fact** (int, optional defaults to 4) – see documentation for *healpy.query_polygon()*.

Returns HEALPixels at the passed *nside* that touch the RA/Dec box.

Return type list

Notes

- Uses *healpy.query_polygon()* to retrieve the RA geodesics and then *hp_in_dec_range()* to limit by Dec.
- When the RA range exceeds 180o, *healpy.query_polygon()* defines the range as that with the smallest area (i.e the box can wrap-around in RA). To avoid any ambiguity, this function will only limit by the passed Decs in such cases.
- Only strictly correct for Decs from -90+1e-3(o) to 90-1e3(o).

desitarget.geomask.**hp_in_cap**(*nside*, *radecrad*, *inclusive=True*, *fact=4*)

Determine which HEALPixels touch an RA, Dec, radius cap.

Parameters

- **nside** (int) – (NESTED) HEALPixel nside.
- **radecrad** (list, defaults to *None*) – 3-entry list of coordinates [ra, dec, radius] forming a cap or "circle" on the sky. ra, dec and radius are all in degrees.
- **inclusive** (bool, optional, defaults to True) – see documentation for *healpy.query_disc()*.
- **fact** (int, optional defaults to 4) – see documentation for *healpy.query_disc()*.

Returns A list of HEALPixels at the passed *nside* that touch the cap.

Return type list

Notes

- Just syntactic sugar around *healpy.query_disc()*.

desitarget.geomask.**hp_in_dec_range**(*nside*, *decmin*, *decmax*, *inclusive=True*)

HEALPixels in a specified range of Declination.

Parameters

- **nside** (int) – (NESTED) HEALPixel nside.
- **decmax** (*decmin,*) – Declination range (degrees).
- **inclusive** (bool, optional, defaults to True) – see documentation for *healpy.query_strip()*.

Returns (Nested) HEALPixels at *nside* in the specified Dec range.

Return type list

Notes

- Just syntactic sugar around *healpy.query_strip()*.
- *healpy.query_strip()* isn't implemented for the NESTED scheme in early healpy versions, so this queries in the RING scheme and then converts to the NESTED scheme.

desitarget.geomask.**is_in_box**(*objs*, *radecbox*)

Determine which of an array of objects are inside an RA, Dec box.

Parameters

- **objs** (ndarray) – An array of objects. Must include at least the columns "RA" and "DEC".
- **radecbox** (list) – 4-entry list of coordinates [ramin, ramax, decmin, decmax] forming the edges of a box in RA/Dec (degrees).

Returns True for objects in the box, False for objects outside of the box.

Return type ndarray

Notes

- Tests the minimum RA/Dec with >= and the maximum with <

desitarget.geomask.**is_in_cap**(*objs*, *radecrad*)

Determine which of an array of objects lie inside an RA, Dec, radius cap.

Parameters

- **objs** (ndarray) – An array of objects. Must include at least the columns "RA" and "DEC".
- **radecrad** (list, defaults to *None*) – 3-entry list of coordinates [ra, dec, radius] forming a cap or "circle" on the sky. ra, dec and radius are all in degrees.

Returns True for objects in the cap, False for objects outside of the cap.

Return type ndarray

Notes

- Tests the separation with <=, so include objects on the cap boundary.
- See also is_in_circle() which handles multiple caps.

desitarget.geomask.**is_in_circle**(*ras*, *decs*, *RAcens*, *DECcens*, *r*)

Whether a set of points is in a set of circular masks on the sky.

Parameters

- **ras** (ndarray) – Array of Right Ascensions to test.
- **decs** (ndarray) – Array of Declinations to test.
- **RAcen** (ndarray) – Right Ascension of the centers of the circles (DEGREES).
- **DECcen** (ndarray) – Declination of the centers of the circles (DEGREES).
- **r** (ndarray) – Radius of the circles (ARCSECONDS).

Returns An array that is the same length as RA/Dec that is `True` for points that are in any of the masks and False for points that are not in any of the masks.

Return type `boolean`

`desitarget.geomask.`**`is_in_ellipse`**(*ras*, *decs*, *RAcen*, *DECcen*, *r*, *e1*, *e2*)

Determine whether points lie within an elliptical mask on the sky

Parameters

- **`ras`** (`ndarray`) – Array of Right Ascensions to test
- **`decs`** (`ndarray`) – Array of Declinations to test
- **`RAcen`** (`float`) – Right Ascension of the center of the ellipse (DEGREES)
- **`DECcen`** (`float`) – Declination of the center of the ellipse (DEGREES)
- **`r`** (`float`) – Half-light radius of the ellipse (ARCSECONDS)
- **`e1`** (`float`) – First ellipticity component of the ellipse
- **`e2`** (`float`) – Second ellipticity component of the ellipse

Returns An array that is the same length as RA/Dec that is True for points that are in the mask and False for points that are not in the mask

Return type `boolean`

Notes

- **The parametrization is explained at** http://legacysurvey.org/dr4/catalogs/
- **Much of the math is taken from:** https://github.com/dstndstn/tractor/blob/master/tractor/ellipses.py

`desitarget.geomask.`**`is_in_ellipse_matrix`**(*ras*, *decs*, *RAcen*, *DECcen*, *G*)

Determine whether points lie within an elliptical mask on the sky

Parameters

- **`ras`** (`ndarray`) – Array of Right Ascensions to test
- **`decs`** (`ndarray`) – Array of Declinations to test
- **`RAcen`** (`float`) – Right Ascension of the center of the ellipse (DEGREES)
- **`DECcen`** (`float`) – Declination of the center of the ellipse (DEGREES)
- **`G`** (`ndarray`) – A 2x2 matrix to transform points measured in coordinates of the effective-half-light-radius to RA/Dec offset coordinates as generated by, e.g., *`desitarget.geomask.ellipse_matrix`*

Returns An array that is the same length as ras/decs that is True for points that are in the mask and False for points that are not in the mask

Return type `boolean`

Notes

- **The parametrization is explained at** http://legacysurvey.org/dr4/catalogs/
- **Much of the math is taken from:** https://github.com/dstndstn/tractor/blob/master/tractor/ellipses.py
- G should have a shape of (2,2) (i.e. np.shape(G) gives (2,2)

desitarget.geomask.**is_in_gal_box**(*objs*, *lbbox*, *radec=False*)
Determine which of an array of objects are in a Galactic l, b box.

Parameters

- **objs** (ndarray or *list*) – An array of objects. Must include at least the columns "RA" and "DEC".
- **radecbox** (list) – 4-entry list of coordinates [lmin, lmax, bmin, bmax] forming the edges of a box in Galactic l, b (degrees).
- **radec** (bool, optional, defaults to False) – If True then the passed *objs* is an [RA, Dec] list instead of a rec array.

Returns True for objects in the box, False for objects outside of the box.

Return type ndarray

Notes

- Tests the minimum l/b with >= and the maximum with <

desitarget.geomask.**is_in_hp**(*objs*, *nside*, *pixlist*, *radec=False*)
Determine which of an array of objects lie inside a set of HEALPixels.

Parameters

- **objs** (ndarray) – Array of objects. Must include at columns "RA" and "DEC".
- **nside** (int) – The HEALPixel nside number (NESTED scheme).
- **pixlist** (list or ~*numpy.ndarray*) – The list of HEALPixels in which to find objects.
- **radec** (bool, optional, defaults to False) – If True *objs* is an [RA, Dec] list instead of a rec array.

Returns True for objects in pixlist, False for other objects.

Return type ndarray

desitarget.geomask.**nside2nside**(*nside*, *nsidenew*, *pixlist*)
Change a list of HEALPixel numbers to a different NSIDE.

Parameters

- **nside** (int) – The current HEALPixel nside number (NESTED scheme).
- **nsidenew** (int) – The new HEALPixel nside number (NESTED scheme).
- **pixlist** (list or ~*numpy.ndarray*) – The list of HEALPixels to be changed.

Returns The altered list of HEALPixels.

Return type ndarray

Notes

- The size of the input list will be altered. For instance, nside=2, pixlist=[0,1] is covered by only pixel [0] at nside=1 but by pixels [0, 1, 2, 3, 4, 5, 6, 7] at nside=4.
- Doesn't check that the passed pixels are valid at *nside*.

desitarget.geomask.**pixarea2nside**(*area*)
Closest HEALPix nside for a given area.

Parameters **area** (float) – area in square degrees.

Returns HEALPix nside that corresponds to passed area.

Return type int

Notes

- Only considers 2**x nside values (1, 2, 4, 8 etc.)

desitarget.geomask.**radec_match_to**(*matchto*, *objs*, *sep=1.0*, *radec=False*, *return_sep=False*)
Match objects to a catalog list on RA/Dec.

Parameters

- **matchto** (ndarray or *list*) – Coordinates to match TO. Must include columns "RA" and "DEC".
- **objs** (ndarray or *list*) – Objects matched to *matchto*. Must include "RA" and "DEC".
- **sep** (float, defaults to 1 arcsecond) – Separation at which to match *objs* to *matchto* in ARCSECONDS.
- **radec** (bool, optional, defaults to False) – If True then *objs* and *matchto* are [RA, Dec] lists instead of rec arrays.
- **return_sep** (bool, optional, defaults to False) – If True then return the separation between each object, not just the indexes of the match.

Returns

- ndarray (of integers) – Indexes in *matchto* where *objs* matches *matchto* at < *sep*.
- ndarray (of integers) – Indexes in *objs* where *objs* matches *matchto* at < *sep*.
- ndarray (of floats) – The distances in ARCSECONDS of the matches. Only returned if *return_sep* is True.

Notes

- Sense is important. Every coordinate pair in *objs* is matched to *matchto*, but NOT every coordinate pair in *matchto* is matched to *objs*. *matchto* is the "parent" catalog being matched to, i.e. we're looking for the instances where *objs* has a match in *matchto*. The array of returned indexes thus can't be longer than *objs*. Consider this example:

```
>>> mainra, maindec = [100], [30]
>>> ras, decs = [100, 100, 100], [30, 30, 30]
>>>
>>> radec_match_to([mainra, maindec], [ras, decs], radec=True)
>>> Out: (array([0, 0, 0]), array([0, 1, 2]))
>>>
>>> radec_match_to([ras, decs], [mainra, maindec], radec=True)
>>> Out: (array([0]), array([0]))
```

- Only returns the CLOSEST match within *sep* arcseconds.

desitarget.geomask.**rewind_coords**(*ranow*, *decnow*, *pmra*, *pmdec*, *epochnow=2015.5*, *epochnowdec=None*, *epochpast=1991.5*, *epochpastdec=None*)

Shift coordinates into the past based on proper motions.

Parameters

- **ranow** (flt or ~*numpy.ndarray*) – Right Ascension (degrees) at "current" epoch.
- **decnow** (flt or ~*numpy.ndarray*) – Declination (degrees) at "current" epoch.
- **pmra** (flt or ~*numpy.ndarray*) – Proper motion in RA (mas/yr).
- **pmdec** (flt or ~*numpy.ndarray*) – Proper motion in Dec (mas/yr).
- **epochnow** (flt or ~*numpy.ndarray*, optional) – The "current" epoch (years). Defaults to Gaia DR2 (2015.5).
- **epochnowdec** (flt or ~*numpy.ndarray*, optional) – If passed and not None then epochnow is interpreted as the epoch of the RA and this is interpreted as the epoch of the Dec.
- **epochpast** (flt or ~*numpy.ndarray*, optional) – Epoch in the past (years). Defaults to Tycho DR2 mean (1991.5).
- **epochpastdec** (flt or ~*numpy.ndarray*, optional) – If passed and not None then epochpast is interpreted as the epoch of the RA and this is interpreted as the epoch of the Dec.

Returns

- ndarray – Right Ascension in the past (degrees).
- ndarray – Declination in the past (degrees).

Notes

- All output RAs will be in the range 0 < RA < 360o.
- Only called "rewind_coords" to correspond to the default *epochnow* > *epochpast*. "fast forwarding" works fine, too, i.e., you can pass *epochpast* > *epochnow* to move coordinates to a future epoch.
- Inaccurate to ~0.1" for motions as high as ~10"/yr (Barnard's Star) after ~200 years because of the simplified cosdec term.

desitarget.geomask.**shares_hp**(*nside*, *objs1*, *objs2*, *radec=False*)

Check if arrays of objects occupy the same HEALPixels.

Parameters

- **nside** (int) – HEALPixel nside integer (NESTED scheme).
- **objs1** (ndarray or *list*) – First set of objects. Must include columns "RA" and "DEC".
- **objs** (ndarray or *list*) – Second set of objects. Must include "RA" and "DEC".
- **radec** (bool, optional, defaults to False) – If True then *objs1* and *objs2* are [RA, Dec] lists instead of rec arrays.

Returns

- ndarray (of booleans) –

  True for objects in *objs1* that share a HEALPixel with objects in objs2 at *nside*. Same length as *objs1*

- ndarray (of booleans) –

  True for objects in *objs2* that share a HEALPixel with objects in objs1 at *nside*. Same length as *objs2*

desitarget.geomask.**sphere_circle_ra_off**(*theta*, *centdec*, *declocs*)

Offsets in RA needed for given declinations in order to draw a (small) circle on the sphere

Parameters

- **theta** (float) – (angular) radius of a circle drawn on the surface of the unit sphere (in DEGREES)
- **centdec** (float) – declination of the center of the circle to be drawn on the sphere (in DEGREES)
- **declocs** (*array_like*) – declinations of positions on the boundary of the circle at which to calculate RA offsets (in DEGREES)

Returns **raoff** – offsets in RA that correspond to the passed dec locations for the given dec circle center (IN DEGREES)

Return type array_like

Notes

- This function is ambivalent to the SIGN of the offset. In other words, it can only draw the semi-circle in theta from -90o->90o, which corresponds to offsets in the POSITIVE RA direction. The user must determine which offsets are to the negative side of the circle, or call this function twice.

desitarget.geomask.**sweep_files_touch_hp**(*nside*, *pixlist*, *infiles*)

Determine which of a set of sweep files touch a set of HEALPixels.

Parameters

- **nside** (int) – (NESTED) HEALPixel nside.
- **pixlist** (list or *int*) – A set of HEALPixels at *nside*.
- **infiles** (list or *str*) – A list of input (sweep filenames) OR a single filename.

Returns

- list – A list of lists of input sweep files that touch each HEALPixel at *nside*. So, e.g. for *nside=2* the returned list will have 48 entries, and, for example, output[0] will be a list of files that touch HEALPixel 0.
- list – The input *pixlist* reduced to just those pixels that touch the area covered by the input *infiles*.
- ndarray – A flattened array of all HEALPixels touched by the input *infiles*. Each HEALPixel will appear multiple times if it's touched by multiple input sweep files.

1.8 desitarget.gfa

Guide/Focus/Alignment targets

desitarget.gfa.**add_urat_pms**(*objs*, *numproc=4*)

Add proper motions from URAT to a set of objects.

Parameters

- **objs** (ndarray) – Array of objects to update. Must include the columns "PMRA", "PMDEC", "REF_ID" (unique per object) "URAT_ID" and "URAT_SEP".
- **numproc** (int, optional, defaults to 4) – The number of parallel processes to use.

Returns The input array with the "PMRA", PMDEC", "URAT_ID" and "URAT_SEP" columns updated to include URAT information.

Return type ndarray

Notes

- Order is retained using "REF_ID": The input and output arrays should have the same order.

desitarget.gfa.**all_gaia_in_tiles**(*maglim=18*, *numproc=4*, *allsky=False*, *tiles=None*, *mindec=-30*, *mingalb=10*, *nside=None*, *pixlist=None*, *addobjid=False*)

An array of all Gaia objects in the DESI tiling footprint

Parameters

- **maglim** (float, optional, defaults to 18) – Magnitude limit for GFAs in Gaia G-band.
- **numproc** (int, optional, defaults to 4) – The number of parallel processes to use.
- **allsky** (bool, defaults to False) – If True, assume that the DESI tiling footprint is the entire sky regardless of the value of *tiles*.
- **tiles** (ndarray, optional, defaults to None) – Array of DESI tiles. If None, then load the entire footprint.
- **mindec** (float, optional, defaults to -30) – Minimum declination (o) to include for output Gaia objects.
- **mingalb** (float, optional, defaults to 10) – Closest latitude to Galactic plane for output Gaia objects (e.g. send 10 to limit to areas beyond -10o <= b < 10o).
- **nside** (int, optional, defaults to *None*) – (NESTED) HEALPix *nside* to use with *pixlist*.
- **pixlist** (list or *int*, optional, defaults to *None*) – Only return sources in a set of (NESTED) HEALpixels at the supplied *nside*.
- **addobjid** (bool, optional, defaults to False) – If True, include, in the output, a column "GAIA_OBJID" that is the integer number of each row read from each Gaia file.

Returns Gaia objects within the passed geometric constraints brighter than *maglim*, formatted like *desitarget.gfa.gfadatamodel*.

Return type ndarray

Notes

- The environment variables $GAIA_DIR and $DESIMODEL must be set.

desitarget.gfa.**gaia_gfas_from_sweep**(*filename*, *maglim=18.0*)

Create a set of GFAs for one sweep file.

Parameters

- **filename** (str) – A string corresponding to the full path to a sweep file name.
- **maglim** (float, optional, defaults to 18) – Magnitude limit for GFAs in Gaia G-band.

Returns GFA objects from Gaia, formatted according to *desitarget.gfa.gfadatamodel*.

Return type ndarray

desitarget.gfa.**gaia_in_file**(*infile*, *maglim=18*, *mindec=-30.0*, *mingalb=10.0*, *nside=None*, *pixlist=None*, *addobjid=False*)

Retrieve the Gaia objects from a HEALPixel-split Gaia file.

Parameters

- **infile** (str) – File name of a single Gaia "healpix" file.
- **maglim** (float, optional, defaults to 18) – Magnitude limit for GFAs in Gaia G-band.
- **mindec** (float, optional, defaults to -30) – Minimum declination (o) to include for output Gaia objects.
- **mingalb** (float, optional, defaults to 10) – Closest latitude to Galactic plane for output Gaia objects (e.g. send 10 to limit to areas beyond -10o <= b < 10o)"
- **nside** (int, optional, defaults to *None*) – (NESTED) HEALPix *nside* to use with *pixlist*.
- **pixlist** (list or *int*, optional, defaults to *None*) – Only return sources in a set of (NESTED) HEALpixels at the supplied *nside*.
- **addobjid** (bool, optional, defaults to False) – If True, include, in the output, a column "GAIA_OBJID" that is the integer number of each row read from file.

Returns Gaia objects in the passed Gaia file brighter than *maglim*, formatted according to *desitarget.gfa.gfadatamodel*.

Return type ndarray

Notes

- A "Gaia healpix file" here is as made by, e.g. *gaia_fits_to_healpix()*

desitarget.gfa.**gaia_morph**(*gaia*)

Retrieve morphological type for Gaia sources.

Parameters **gaia** (ndarray) – Numpy structured array containing at least the columns, *GAIA_PHOT_G_MEAN_MAG* and *GAIA_ASTROMETRIC_EXCESS_NOISE*.

Returns An array of strings that is the same length as the input array and is set to either "GPSF" or "GGAL" based on a morphological cut with Gaia.

Return type array

desitarget.gfa.**select_gfas**(*infiles*, *maglim=18*, *numproc=4*, *nside=None*, *pixlist=None*, *bundlefiles=None*, *extra=None*, *mindec=-30*, *mingalb=10*, *addurat=True*)

Create a set of GFA locations using Gaia and matching to sweeps.

Parameters

- **infiles** (list or *str*) – A list of input filenames (sweep files) OR a single filename.
- **maglim** (float, optional, defaults to 18) – Magnitude limit for GFAs in Gaia G-band.
- **numproc** (int, optional, defaults to 4) – The number of parallel processes to use.
- **nside** (int, optional, defaults to *None*) – (NESTED) HEALPix *nside* to use with *pixlist* and *bundlefiles*.
- **pixlist** (list or *int*, optional, defaults to *None*) – Only return targets in a set of (NESTED) HEALpixels at the supplied *nside*. Useful for parallelizing.

- **bundlefiles** (int, defaults to *None*) – If not *None*, then, instead of selecting gfas, print the slurm script to run in pixels at *nside*. Is an integer rather than a boolean for historical reasons.
- **extra** (str, optional) – Extra command line flags to be passed to the executable lines in the output slurm script. Used in conjunction with *bundlefiles*.
- **mindec** (float, optional, defaults to -30) – Minimum declination (o) for output sources that do NOT match an object in the passed *infiles*.
- **mingalb** (float, optional, defaults to 10) – Closest latitude to Galactic plane for output sources that do NOT match an object in the passed *infiles* (e.g. send 10 to limit to regions beyond -10o <= b < 10o)".
- **addurat** (bool, optional, defaults to True) – If True then substitute proper motions from the URAT catalog where Gaia is missing proper motions. Requires that the URAT_DIR is set and points to data downloaded and formatted by, e.g., *make_urat_files()*.

Returns GFA objects from Gaia with the passed geometric constraints limited to the passed maglim and matched to the passed input files, formatted according to *desitarget.gfa.gfadatamodel*.

Return type ndarray

Notes

- If numproc==1, use the serial code instead of parallel code.
- If numproc > 4, then numproc=4 is enforced for (just those) parts of the code that are I/O limited.

1.9 desitarget.internal

This subpackage contains modules that are for strictly internal use by the desitarget package. End-users should not directly call any functions in this subpackage.

1.10 desitarget.internal.sharedmem

Easier parallel programming on shared memory computers.

The source code is at http://github.com/rainwoodman/sharedmem .

Topics

- *Programming Model*
- *Usage*
- *API References*

1.10.1 Programming Model

MapReduce provides the equivalent to multiprocessing.Pool, with the following differences:

- MapReduce does not require the work function to be picklable.

- MapReduce adds a reduction step that is guaranteed to run on the coordinator process's scope.
- MapReduce allows the use of critical sections and ordered execution in the work function.

Modifications to shared Memory arrays, allocated via

- `sharedmem.empty()`,
- `sharedmem.empty_like()`,
- `sharedmem.copy()`,

are visible by all processes, including the coordinator process.

1.10.2 Usage

The package can be installed via `easy_install sharedmem`. Alternatively, the file `sharedmem.py` can be directly embedded into other projects.

The only external dependency is numpy, since this was designed to work with large shared memory chunks through numpy.ndarray.

Environment variable OMP_NUM_THREADS is used to determine the default number of workers.

Notes

This module depends on the *fork* system call, thus is available only on posix systems (not Windows).

Examples

Sum up a large array

```
>>> input = numpy.arange(1024 * 1024 * 128, dtype='f8')
>>> output = sharedmem.empty(1024 * 1024 * 128, dtype='f8')
>>> with MapReduce() as pool:
>>>    chunksize = 1024 * 1024
>>>    def work(i):
>>>        s = slice (i, i + chunksize)
>>>        output[s] = input[s]
>>>        return i, sum(input[s])
>>>    def reduce(i, r):
>>>        print('chunk', i, 'done')
>>>        return r
>>>    r = pool.map(work, range(0, len(input), chunksize), reduce=reduce)
>>> print numpy.sum(r)
>>>
```

Textual analysis

```
>>> input = file('mytextfile.txt').readlines()
>>> word_count = {'bacon': 0, 'eggs': 0 }
>>> with MapReduce() as pool:
>>>    def work(line):
>>>        words = line.split()
>>>        for word in words:
>>>            word_count[word] += 1
>>>        return word_count
```

(continues on next page)

(continued from previous page)

```
>>>     def reduce(wc):
>>>         for key in word_count:
>>>             word_count[key] += wc[key]
>>> print word_count
>>>
```

pool.ordered can be used to require a block of code to be executed in order

```
>>> with MapReduce() as pool:
>>>     def work(i):
>>>          with pool.ordered:
>>>             print(i)
>>>     pool.map(work, range(10))
```

pool.critical can be used to require a block of code to be executed in a critical section.

```
>>> counter = sharedmem.empty(1)
>>> counter[:] = 0
>>> with MapReduce() as pool:
>>>     def work(i):
>>>          with pool.critical:
>>>              counter[:] += i
>>>     pool.map(work, range(10))
>>> print(counter)
```

1.10.3 API References

desitarget.internal.sharedmem.**set_debug**(*flag*)
Set the debug mode.

In debug mode (flag==True), no workers are spawn. All work are done in serial on the coordinator thread/process. This eases debuggin when the worker throws out an exception.

desitarget.internal.sharedmem.**get_debug**()
Get the debug mode

desitarget.internal.sharedmem.**total_memory**()
Returns the the amount of memory available for use.

This function is not very useful. The memory is obtained from MemTotal entry in /proc/meminfo.

desitarget.internal.sharedmem.**cpu_count**()
Returns the number of worker processes to be spawned.

The default value is the number of physical cpu cores seen by python. `OMP_NUM_THREADS` environment variable overrides it.

On PBS/torque systems if OMP_NUM_THREADS is empty, we try to use the value of `PBS_NUM_PPN` variable.

Notes

On some machines the physical number of cores does not equal the number of cpus shall be used. PSC Blacklight for example.

exception desitarget.internal.sharedmem.**WorkerException**(*reason*, *traceback*)
Represents an exception that has occured during a worker process

reason
The underlining reason of the exception. If the original exception can be pickled, the type of the exception is preserved. Otherwise, a LostExceptionType warning is issued, and reason is of type Exception.

Type Exception, or subclass of Exception.

traceback
The string version of the traceback that can be used to inspect the error.

Type str

exception desitarget.internal.sharedmem.**StopProcessGroup**
StopProcessGroup will terminate the worker process/thread

class desitarget.internal.sharedmem.**background**(*function*, **args*, ***kwargs*)
Asyncrhonized function call via a background process.

Parameters

- **function** (*callable*) – the function to call
- ***args** (*positional arguments*) –
- ****kwargs** (*keyward arguments*) –

Examples

```
>>> def function(*args, **kwargs):
>>>    pass
>>> bg = background(function, *args, **kwargs)
>>> rt = bg.wait()
```

wait()
Wait and join the child process. The return value of the function call is returned. If any exception occurred it is wrapped and raised.

class desitarget.internal.sharedmem.**MapReduce**(*backend=<class 'desitarget.internal.sharedmem.ProcessBackend'>*, *np=None*)
A pool of worker processes for a Map-Reduce operation

Parameters

- **backend** (*ProcessBackend or ThreadBackend*) – ProcessBackend is preferred. ThreadBackend can be used in cases where processes creation is not allowed.
- **np** (*int or None*) – Number of processes to use. Default (None) is from OMP_NUM_THREADS or the number of available cores on the computer. If np is 0, all operations are performed on the coordinator process – no child processes are created.

Notes

Always wrap the call to *map()* in a context manager ('with') block.

map(*func*, *sequence*, *reduce=None*, *star=False*)
Map-reduce with multile processes.

Apply func to each item on the sequence, in parallel. As the results are collected, reduce is called on the result. The reduced result is returned as a list.

Parameters

- **func** (*callable*) – The function to call. It must accept the same number of arguments as the length of an item in the sequence.

  Warning: func is not supposed to use exceptions for flow control. In non-debug mode all exceptions will be wrapped into a *WorkerException*.

- **sequence** (*list or array_like*) – The sequence of arguments to be applied to func.
- **reduce** (*callable, optional*) – Apply an reduction operation on the return values of func. If func returns a tuple, they are treated as positional arguments of reduce.
- **star** (*boolean*) – if True, the items in sequence are treated as positional arguments of reduce.

Returns **results** – The list of reduced results from the map operation, in the order of the arguments of sequence.

Return type list

Raises *WorkerException* – If any of the worker process encounters an exception. Inspect *WorkerException.reason* for the underlying exception.

desitarget.internal.sharedmem.**MapReduceByThread**(*np=None*)
Creates a MapReduce object but with the Thread backend.

The process backend is usually preferred.

desitarget.internal.sharedmem.**empty**(*shape*, *dtype='f8'*)
Create an empty shared memory array.

desitarget.internal.sharedmem.**empty_like**(*array*, *dtype=None*)
Create a shared memory array from the shape of array.

desitarget.internal.sharedmem.**full**(*shape*, *value*, *dtype='f8'*)
Create a shared memory array of given shape and type, filled with *value*.

desitarget.internal.sharedmem.**full_like**(*array*, *value*, *dtype=None*)
Create a shared memory array with the same shape and type as a given array, filled with *value*.

desitarget.internal.sharedmem.**copy**(*a*)
Copy an array to the shared memory.

Notes

copy is not always necessary because the private memory is always copy-on-write.

Use `a = copy(a)` to immediately dereference the old 'a' on private memory

1.11 desitarget.io

Functions for reading, writing and manipulating files related to targeting.

desitarget.io.**_bright_or_dark**(*filename*, *hdr*, *data*, *obscon*, *mockdata=None*)
modify data/file name for BRIGHT or DARK survey OBSCONDITIONS

Parameters

- **filename** (str) – output target selection file.
- **hdr** (class:*str*) – header of the output target selection file.
- **data** (ndarray) – numpy structured array of targets.
- **obscon** (str) – Can be "DARK" or "BRIGHT" to only write targets appropriate for "DARK|GRAY" or "BRIGHT" observing conditions. The relevant *PRIORITY_INIT* and *NUMOBS_INIT* columns will be derived from *PRIORITY_INIT_DARK*, etc. and *filename* will have "bright" or "dark" appended to the lowest DIRECTORY in the input *filename*.
- **mockdata** (dict, optional, defaults to *None*) – Dictionary of mock data to write out (only used in *desitarget.mock.build.targets_truth* via *select_mock_targets*).

Returns

- str – The modified file name.
- data – The modified data.

desitarget.io.**_check_hpx_length**(*hpxlist*, *length=68*, *warning=False*)
Check a list expressed as a csv string won't exceed a length.

desitarget.io.**_get_targ_dir**()
Convenience function to grab the TARGDIR environment variable.

Returns The directory stored in the $GAIA_DIR environment variable.

Return type str

desitarget.io.**add_photsys**(*indata*)
Add the PHOTSYS column to a sweeps-style array.

Parameters **indata** (ndarray) – Numpy structured array to which to add PHOTSYS column.

Returns Input array with PHOTSYS added (and set using RELEASE).

Return type ndarray

Notes

- The PHOTSYS column is only added if the RELEASE column is available in the passed *indata*.

desitarget.io.**brickname_from_filename**(*filename*)
Parse *filename* to check if this is a tractor brick file.

Parameters **filename** (str) – Name of a tractor brick file.

Returns Name of the brick in the file name.

Return type str

Raises ValueError – If the filename does not appear to be a valid tractor brick file.

desitarget.io.**brickname_from_filename_with_prefix**(*filename*, *prefix=''*)
Parse *filename* to check if this is a brick file with a given prefix.

Parameters

- **filename** (str) – Full name of a brick file.
- **prefix** (str) – Optional part of filename immediately preceding the brickname.

Returns Name of the brick in the file name.

Return type str

Raises `ValueError` – If the filename does not appear to be a valid brick file.

`desitarget.io.`**`check_both_set`**(*hpxlist*, *nside*)
Check that if one of two variables is set, the other is too

`desitarget.io.`**`check_fitsio_version`**(*version='0.9.8'*)
fitsio prior to 0.9.8rc1 has a bug parsing boolean columns.

Parameters **`version`** (`str`, optional) – Default '0.9.8'. Having this parameter allows future-proofing and easier testing.

Raises `ImportError` – If the fitsio version is insufficiently recent.

`desitarget.io.`**`check_hp_target_dir`**(*hpdirname*)
Check fidelity of a directory of HEALPixel-partitioned targets.

Parameters **`hpdirname`** (`str`) – Full path to a directory containing targets that have been split by HEALPixel.

Returns

- `int` – The HEALPixel NSIDE for the files in the passed directory.
- `dict` – A dictionary where the keys are each HEALPixel covered in the passed directory and the values are the file that includes that HEALPixel.

Notes

- Checks that all files are at the same NSIDE.
- Checks that no two files contain the same HEALPixels.
- Checks that HEALPixel numbers are consistent with NSIDE.

`desitarget.io.`**`decode_sweep_name`**(*sweepname*, *nside=None*, *inclusive=True*, *fact=4*)
Retrieve RA/Dec edges from a full directory path to a sweep file

Parameters

- **`sweepname`** (`str`) – Full path to a sweep file, e.g., /a/b/c/sweep-350m005-360p005.fits
- **`nside`** (`int`, optional, defaults to None) – (NESTED) HEALPixel nside
- **`inclusive`** (`book`, optional, defaults to `True`) – see documentation for *healpy.query_polygon()*
- **`fact`** (`int`, optional defaults to 4) – see documentation for *healpy.query_polygon()*

Returns

- `list` (if nside is None) – A 4-entry list of the edges of the region covered by the sweeps file in the form [RAmin, RAmax, DECmin, DECmax] For the above example this would be [350., 360., -5., 5.]
- `list` (if nside is not None) – A list of HEALPixels that touch the files at the passed *nside* For the above example this would be [16, 17, 18, 19]

`desitarget.io.`**`desitarget_nside`**()
Default HEALPix Nside for all target selection algorithms.

`desitarget.io.`**`desitarget_resolve_dec`**()
Default Dec cut to separate targets in BASS/MzLS from DECaLS.

desitarget.io.**find_star_files**(*objs*, *hpxdir*, *nside*, *neighbors=True*, *radec=False*, *strict=False*)
Full paths to HEALPixel-split star files for objects by RA/Dec.

Parameters

- **objs** (ndarray) – Array of objects. Must contain at least columns "RA" and "DEC".
- **hpxdir** (str) – Name of the directory that hosts the HEALPixel-split files. Most likely this directory ends in "/healpix".
- **nside** (int) – The (NESTED) HEALPixel nside integer.
- **neighbors** (bool, optional, defaults to True) – Also return all neighboring pixels that touch the files of interest to prevent edge effects (e.g. if a, say, Gaia source is 1 arcsec away from a primary source and so in an adjacent pixel).
- **radec** (bool, optional, defaults to False) – If True then the passed *objs* is an [RA, Dec] list instead of a rec array.
- **strict** (bool, optional, defaults to False) – Only return files that actually exist. This is useful for, e.g., URAT files, which don't cover the whole sky and so don't have files for every HEALPixel.

Returns A list of all files that need to be read in to account for objects at the passed locations.

Return type list

Notes

- "star" files, here might be Gaia, Tycho or URAT files.

desitarget.io.**find_target_files**(*targdir*, *dr=None*, *flavor='targets'*, *survey='main'*, *obscon=None*, *hp=None*, *nside=None*, *resolve=True*, *supp=False*, *mock=False*, *nohp=False*, *seed=None*, *region=None*, *maglim=None*, *epoch=None*)
Build the name of an output target file (or directory).

Parameters

- **targdir** (str) – Name of a based directory for output target catalogs.
- **dr** (str or int, optional, defaults to "X") – Name of a Legacy Surveys Data Release (e.g. 8)
- **flavor** (str, optional, defaults to *targets*) – Options include "skies", "gfas", "targets", "randoms", "masks".
- **survey** (str, optional, defaults to *main*) – Options include "main", "cmx", "svX" (where X is 1, 2 etc.). Only relevant if *flavor* is "targets".
- **obscon** (str, optional) – Name of the *OBSCONDITIONS* used to make the file (e.g. DARK).
- **hp** (list or int or str, optional) – HEALPixel numbers used to make the file (e.g. 42 or [12, 37] or "42" or "12,37"). Required if mock='True'.
- **nside** (int, optional unless mock='True') – Nside corresponding to healpixel *hp*.
- **resolve** (bool, optional, defaults to True) – If True then find the *resolve* file. Otherwise find the *noresolve* file. Relevant if *flavor* is *targets* or *randoms*.
- **supp** (bool, optional, defaults to False) – If True then find the supplemental targets file. Overrides the *obscon* option.

- **mock** (bool, optional, defaults to False) – If True then construct the file path for mock target catalogs and return (most other inputs are ignored).
- **nohp** (bool, optional, defaults to False) – If True, override the normal behavior for *hp*=*None* and instead construct a filename that omits the *-hpX-* part.
- **seed** (int, optional) – If *seed* is not None, then it is added to the file name just before the ".fits" extension (i.e. "-8.fits" for *seed* of 8). Only relevant if *flavor* is "randoms".
- **region** (int, optional) – If *region* is not None, then it is added to the directory name after *resolve*. Only relevant if *flavor* is "randoms".
- **maglim** (float, optional) – Magnitude limit to which the mask was made. Only relevant if *flavor* is "masks". Must be passed if *flavor* is "masks".
- **epoch** (float) – Epoch at which the mask was made. Only relevant if *flavor* is "masks". Must be passed if *flavor* is "masks".

Returns The name of the output target file (or directory).

Return type str

Notes

- If *hp* is passed, the full file name is returned. If *hp* is None, the directory name where all of the *hp* files are stored is returned. The directory name is the expected input for the *desitarget.io.read** convenience functions (desimodel.io.read_targets_in_hp(), etc.).
- On the other hand, if *hp* is None and *nohp* is True then a filename is returned that just omits the *-hpX-* part.

desitarget.io.**fix_tractor_dr1_dtype**(*objects*)
DR1 tractor files have inconsistent dtype for the TYPE field. Fix this.

Parameters **objects** – numpy structured array from target file.

Returns structured array with TYPE.dtype = 'S4' if needed.

If the type was already correct, returns the original array.

desitarget.io.**gitversion**()
Returns *git describe –tags –dirty –always*, or 'unknown' if not a git repo

desitarget.io.**hpx_filename**(*hpx*)
Return the standard name for HEALPixel-split input files

Parameters **hpx** (str or *int*) – A HEALPixel integer.

Returns Filename in the format used throughout desitarget for HEALPixel-split input databases.

Return type class: *str*

desitarget.io.**is_sky_dir_official**(*skydirname*)
Check a sky file or directory has the correct HEALPixel structure.

Parameters **skydirname** (str) – Full path to either a directory containing skies that have been partitioned by HEALPixel (i.e. as made by *select_skies* with the *bundle_files* option). Or the name of a single file of skies.

Returns True if the passed sky file or (the first sky file in the passed directory) is structured so that the list of healpixels in the file header ("FILEHPX") at the file nside ("FILENSID") in the file nested (or ring) scheme ("FILENEST") is a true reflection of the HEALPixels in the file.

Return type bool

Notes

- A necessary check because although the targets and GFAs are parallelized to run in the exact boundaries of HEALPixels, the skies are parallelized across bricks that have CENTERS in a given HEALPixel.
- If this function returns `False` the remedy is typically to run *bin/repartition_skies*
- If a directory is passed, this isn't an exhaustive check as only the first file is tested. That's enough for just checking the output of *select_skies*, though.

desitarget.io.**iter_files**(*root*, *prefix*, *ext='fits'*)
Iterator over files under in *root* directory with given *prefix* and extension.

desitarget.io.**iter_sweepfiles**(*root*)
Iterator over all sweep files found under root directory.

desitarget.io.**iter_tractorfiles**(*root*)
Iterator over all tractor files found under *root* directory.

Parameters **root** (str) – Path to start looking. Can be a directory or a single file.

Returns An iterator of (brickname, filename).

Return type iterable

Examples

```
>>> for brickname, filename in iter_tractor('./'):
>>>     print(brickname, filename)
```

desitarget.io.**list_sweepfiles**(*root*)
Return a list of sweep files found under *root* directory.

desitarget.io.**list_targetfiles**(*root*)
Return a list of target files found under *root* directory.

desitarget.io.**list_tractorfiles**(*root*)
Return a list of tractor files found under *root* directory.

desitarget.io.**load_pixweight**(*inmapfile*, *nside*, *pixmap=None*)
Loads a pixel map from file and resamples to a different HEALPixel resolution (nside)

Parameters

- **inmapfile** (str) – Name of the file containing the pixel weight map.
- **nside** (int) – After loading, the array will be resampled to this HEALPix nside.
- **pixmap** (*~numpy.array*, optional, defaults to None) – Pass a pixel map instead of loading it from file.

Returns HEALPixel weight map resampled to the requested nside.

Return type array

desitarget.io.**load_pixweight_recarray**(*inmapfile*, *nside*, *pixmap=None*)
Like load_pixweight but for a structured array map with multiple columns

Parameters

- **inmapfile** (str) – Name of the file containing the pixel weight map.
- **nside** (int) – After loading, the array will be resampled to this HEALPix nside.
- **pixmap** (*~numpy.array*, optional, defaults to None) – Pass a pixel map instead of loading it from file.

Returns HEALPixel weight map with all columns resampled to the requested nside.

Return type array

Notes

- Assumes that tha passed map is in the NESTED scheme, and outputs to the NESTED scheme.
- All columns are resampled as the mean of the relevant pixels, except if a column *HPXPIXEL* is passed. That column is reassigned the appropriate pixel number at the new nside.

desitarget.io.**read_external_file**(*filename, header=False, columns=['RA', 'DEC']*)

Read FITS file with loose requirements on upper-case columns and EXTNAME.

Parameters

- **filename** (str) – File name with full directory path included.
- **header** (bool, optional, defaults to False) – If True then return (data, header) instead of just data.
- **columns** (list, optional, defaults to ["RA", "DEC"]) – Specify the desired columns to read.

Returns

- ndarray – The output data array.
- ndarray, optional – The output file header, if input *header* was True.

Notes

- Intended to be used with externally supplied files such as locations to be matched for commissioning or secondary targets.

desitarget.io.**read_target_files**(*filename, columns=None, rows=None, header=False, downsample=None, verbose=False*)

Wrapper to cycle through allowed extensions to read target files.

Parameters

- **filename** (str) – Name of a target file of any type. Target file types include "TARGETS", "GFA_TARGETS" and "SKY_TARGETS".
- **columns** (list, optional) – Only read in these target columns.
- **rows** (list, optional) – Only read in these rows from the target file.
- **header** (bool, optional, defaults to False) – If True then return the header of the file.
- **downsample** (int, optional, defaults to *None*) – If not *None*, downsample the file by this integer value, e.g. for *downsample=10* a file with 900 rows would have 90 random rows read in. Overrode by the *rows* kwarg if it is not *None*.

- **verbose** (bool, optional, defaults to False) – If True then log the file and extension that was read.

desitarget.io.**read_targets_header**(*hpdirname*)
Read in header of a targets file.

Parameters **hpdirname** (str) – Full path to either a directory containing targets that have been partitioned by HEALPixel (i.e. as made by *select_targets* with the *bundle_files* option). Or the name of a single file of targets.

Returns The header of *hpdirname* if it is a file, or the header of the first file encountered in *hpdirname*

Return type FITSHDR

desitarget.io.**read_targets_in_box**(*hpdirname, radecbox=[0.0, 360.0, -90.0, 90.0], columns=None, header=False, downsample=None*)
Read in targets in an RA/Dec box.

Parameters

- **hpdirname** (str) – Full path to either a directory containing targets that have been partitioned by HEALPixel (i.e. as made by *select_targets* with the *bundle_files* option). Or the name of a single file of targets.
- **radecbox** (list, defaults to the entire sky) – 4-entry list of coordinates [ramin, ramax, decmin, decmax] forming the edges of a box in RA/Dec (degrees).
- **columns** (list, optional) – Only read in these target columns.
- **header** (bool, optional, defaults to False) – If True then return the header of either the *hpdirname* file, or the last file read from the *hpdirname* directory.
- **downsample** (int, optional, defaults to *None*) – If not *None*, downsample targets by (roughly) this value, e.g. for *downsample=10* a set of 900 targets would have ~90 random targets returned.

Returns An array of targets in the passed RA/Dec box.

Return type ndarray

Notes

- If *header* is True, then a second output (the file header is returned).

desitarget.io.**read_targets_in_cap**(*hpdirname*, *radecrad*, *columns=None*)
Read in targets in an RA, Dec, radius cap.

Parameters

- **hpdirname** (str) – Full path to either a directory containing targets that have been partitioned by HEALPixel (i.e. as made by *select_targets* with the *bundle_files* option). Or the name of a single file of targets.
- **radecrad** (list) – 3-entry list of coordinates [ra, dec, radius] forming a cap or "circle" on the sky. ra, dec and radius are all in degrees.
- **columns** (list, optional) – Only read in these target columns.

Returns An array of targets in the passed RA/Dec box.

Return type ndarray

desitarget.io.**read_targets_in_hp**(*hpdirname*, *nside*, *pixlist*, *columns=None*, *header=False*, *downsample=None*, *verbose=False*)

Read in targets in a set of HEALPixels.

Parameters

- **hpdirname** (str) – Full path to either a directory containing targets that have been partitioned by HEALPixel (i.e. as made by *select_targets* with the *bundle_files* option). Or the name of a single file of targets.
- **nside** (int) – The (NESTED) HEALPixel nside.
- **pixlist** (list or *int* or *~numpy.ndarray*) – Return targets in these HEALPixels at the passed *nside*.
- **columns** (list, optional) – Only read in these target columns.
- **header** (bool, optional, defaults to False) – If True then return the header of either the *hpdirname* file, or the last file read from the *hpdirname* directory.
- **downsample** (int, optional, defaults to *None*) – If not *None*, downsample targets by (roughly) this value, e.g. for *downsample=10* a set of 900 targets would have ~90 random targets returned.
- **verbose** (bool, optional, defaults to False) – Passed to *read_target_files()*.

Returns An array of targets in the passed pixels.

Return type ndarray

Notes

- If *header* is True, then a second output (the file header is returned).

desitarget.io.**read_targets_in_tiles**(*hpdirname*, *tiles=None*, *columns=None*, *header=False*)

Parameters

- **hpdirname** (str) – Full path to either a directory containing targets that have been partitioned by HEALPixel (i.e. as made by *select_targets* with the *bundle_files* option). Or the name of a single file of targets.
- **tiles** (ndarray, optional) – Array of tiles in the desimodel format, or None to use all DESI tiles from desimodel.io.load_tiles().
- **columns** (list, optional) – Only read in these target columns.
- **header** (bool, optional, defaults to False) – If True then return the header of either the *hpdirname* file, or the last file read from the *hpdirname* directory.

Returns An array of targets in the passed tiles.

Return type ndarray

Notes

- If *header* is True, then a second output (the file header is returned).
- The environment variable $DESIMODEL must be set.

desitarget.io.**read_tractor**(*filename*, *header=False*, *columns=None*)

Read a tractor catalogue or sweeps file.

Parameters

- **filename** (str) – File name of one Tractor or sweeps file.
- **header** (bool, optional) – If True, return (data, header) instead of just data.
- **columns** (list, optional) – Specify the desired Tractor catalog columns to read; defaults to desitarget.io.tsdatamodel.dtype.names + most of the columns in desitarget.gaiamatch.gaiadatamodel.dtype.names, where tsdatamodel is, e.g., basetsdatamodel + dr9addedcols.

Returns Array with the tractor schema, uppercase field names.

Return type ndarray

desitarget.io.**release_to_photsys**(*release*)

Convert RELEASE to PHOTSYS using the releasedict lookup table.

Parameters **objects** (int or ndarray) – RELEASE column from a numpy rec array of targets.

Returns 'N' if the RELEASE corresponds to the northern photometric system (MzLS+BASS) and 'S' if it's the southern system (DECaLS).

Return type str or ndarray

Notes

Flags an error if the system is not recognized.

desitarget.io.**target_columns_from_header**(*hpdirname*)

Grab the _TARGET column names from a TARGETS file or directory.

Parameters **hpdirname** (str) – Full path to either a directory containing targets that have been partitioned by HEALPixel (i.e. as made by *select_targets* with the *bundle_files* option). Or the name of a single file of targets.

Returns The names of the _TARGET columns, notably whether they are SV, main, or cmx _TARGET columns.

Return type list

desitarget.io.**whitespace_fits_read**(*filename*, ***kwargs*)

Use fitsio to read in a file and strip whitespace from all string columns.

Parameters

- **filename** (str) – Name of the file to be read in by fitsio.
- **kwargs** (*arguments that will be passed directly to fitsio.*) –

desitarget.io.**write_gfas**(*targdir*, *data*, *indir=None*, *indir2=None*, *nside=None*, *nsidefile=None*, *hpxlist=None*, *extra=None*)

Write a catalogue of Guide/Focus/Alignment targets.

Parameters

- **targdir** (str) – Path to output target selection directory (the directory structure and file name are built on-the-fly from other inputs).
- **data** (ndarray) – Array of GFAs to write to file.
- **indir2** (*indir,*) – Legacy Survey Data Release directory or directories, write to header of output file if passed (and if not None).

- **nside** (int, defaults to None.) – If passed, add a column to the GFAs array popluated with HEALPixels at resolution *nside*.
- **nsidefile** (int, optional, defaults to *None*) – Passed to indicate in the output file header that the targets have been limited to only certain HEALPixels at a given nside. Used in conjunction with *hpxlist*.
- **hpxlist** (list, optional, defaults to *None*) – Passed to indicate in the output file header that the targets have been limited to only this list of HEALPixels. Used in conjunction with *nsidefile*.
- **extra** (dict, optional) – If passed (and not None), write these extra dictionary keys and values to the output header.

Returns

- int – The number of gfas that were written to file.
- str – The name of the file to which gfas were written.

desitarget.io.**write_in_chunks**(*filename*, *data*, *nchunks*, *extname=None*, *header=None*)

Write a FITS file in chunks to save memory.

Parameters

- **filename** (str) – The output file.
- **data** (ndarray) – The numpy structured array of data to write.
- **nchunks** (:class'int', optional, defaults to *None*) – The number of chunks in which to write the output file.
- **header, clobber, optional** (*extname,*) – Passed through to fitsio.write().

Returns

Return type Nothing, but writes the *data* to the *filename* in chunks.

Notes

- Always OVERWRITES existing files!

desitarget.io.**write_masks**(*maskdir*, *data*, *maglim=None*, *maskepoch=None*, *nside=None*, *extra=None*)

Write a catalogue of masks and associated pixel-level info.

Parameters

- **maskdir** (str) – Path to output mask directory (the file names are built on-the-fly from other inputs).
- **data** (ndarray) – Array of masks to write to file. Must contain at least the columns "RA" and "DEC".
- **maglim** (float, optional, defaults to None) – Magnitude limit to which the mask was made.
- **maskepoch** (float, optional, defaults to None) – Epoch at which the mask was made.
- **nside** (int, defaults to not splitting by HEALPixel.) – The HEALPix nside at which to write the output files.
- **extra** (dict, optional) – If passed (and not None), write these extra dictionary keys and values to the output header.

Returns

- int – The total number of masks that were written.
- str – The name of the directory to which masks were written.

desitarget.io.**write_randoms**(*targdir*, *data*, *indir=None*, *hdr=None*, *nside=None*, *supp=False*, *nsidefile=None*, *hpxlist=None*, *resolve=True*, *north=None*, *extra=None*)

Write a catalogue of randoms and associated pixel-level info.

Parameters

- **targdir** (str) – Path to output target selection directory (the directory structure and file name are built on-the-fly from other inputs).
- **data** (ndarray) – Array of randoms to write to file.
- **indir** (str, optional, defaults to None) – Name of input Legacy Survey Data Release directory, write to header of output file if passed (and if not None).
- **hdr** (str, optional, defaults to None) – If passed, use this header to start the header for *filename*.
- **nside** (int) – If passed, add a column to the randoms array popluated with HEALPixels at resolution *nside*.
- **supp** (bool, optional, defaults to False) – Written to the header of the output file to indicate whether this is a supplemental file (i.e. random locations that are outside the Legacy Surveys footprint).
- **nsidefile** (int, optional, defaults to None) – Passed to indicate in the output file header that the targets have been limited to only certain HEALPixels at a given nside. Used in conjunction with *hpxlist*.
- **hpxlist** (list, optional, defaults to None) – Passed to indicate in the output file header that the targets have been limited to only this list of HEALPixels. Used in conjunction with *nsidefile*.
- **resolve** (bool, optional, defaults to True) – Written to the output file header as *RESOLVE*. If True (False) output directory includes "resolve" ("noresolve").
- **north** (bool, optional) – If passed (and not None), then, if True (False), REGION=north (south) is written to the output header and the output directory name is appended by "north" ("south").
- **extra** (dict, optional) – If passed (and not None), write these extra dictionary keys and values to the output header.

Returns

- int – The number of randoms that were written to file.
- str – The name of the file to which randoms were written.

desitarget.io.**write_secondary**(*targdir*, *data*, *primhdr=None*, *scxdir=None*, *obscon=None*, *drint='X'*)

Write a catalogue of secondary targets.

Parameters

- **targdir** (str) – Path to output target selection directory (the directory structure and file name are built on-the-fly from other inputs).
- **data** (ndarray) – numpy structured array of secondary targets to write.

- **primhdr** (str, optional, defaults to *None*) – If passed, added to the header of the output *filename*.
- **scxdir** (str, optional, defaults to SCND_DIR) – Name of the directory that hosts secondary targets. The secondary targets are written back out to this directory in the subdirectory "outdata" and the *scxdir* is added to the header of the output *filename*.
- **obscon** (str, optional, defaults to *None*) – Can pass one of "DARK" or "BRIGHT". If passed, don't write the full set of secondary target that do not match a primary, rather only write targets appropriate for "DARK|GRAY" or "BRIGHT" observing conditions. The relevant *PRIORITY_INIT* and *NUMOBS_INIT* columns will be derived from *PRIORITY_INIT_DARK*, etc. and *filename* will have "bright" or "dark" appended to the lowest DIRECTORY in the input *filename*.
- **drint** (int, optional, defaults to *X*) – The data release ("dr"*drint*"-") in the output filename.

Returns

- int – The number of secondary targets that do not match a primary target that were written to file.
- str – The name of the file to which such targets were written.

Notes

Two sets of files are written:

- The file of secondary targets that do not match a primary target is written to *targdir*. Such secondary targets are determined from having *RELEASE==0* and *SKY==0* in the *TARGETID*. Only targets with *PRIORITY_INIT* > *-1* are written to this file (this allows duplicates to be resolved in, e.g., finalize()
- Each secondary target that, presumably, was initially drawn from the "indata" subdirectory of *scxdir* is written to an "outdata/targdir" subdirectory of *scxdir*.

desitarget.io.**write_skies**(*targdir*, *data*, *indir=None*, *indir2=None*, *supp=False*, *apertures_arcsec=None*, *nskiespersqdeg=None*, *nside=None*, *nsidefile=None*, *hpxlist=None*, *extra=None*, *mock=False*)

Write a target catalogue of sky locations.

Parameters

- **targdir** (str) – Path to output target selection directory (the directory structure and file name are built on-the-fly from other inputs).
- **data** (ndarray) – Array of skies to write to file.
- **indir2** (*indir,*) – Name of input Legacy Survey Data Release directory/directories, write to header of output file if passed (and if not None).
- **supp** (bool, optional, defaults to False) – Written to the header of the output file to indicate whether this is a file of supplemental skies (sky locations that are outside the Legacy Surveys footprint).
- **apertures_arcsec** (list or *float*, optional) – list of aperture radii in arcseconds to write each aperture as an individual line in the header, if passed (and if not None).
- **nskiespersqdeg** (float, optional) – Number of sky locations generated per sq. deg., write to header of output file if passed (and if not None).

- **nside** (int, optional) – If passed, add a column to the skies array popluated with HEALPixels at resolution *nside*.
- **nsidefile** (int, optional, defaults to *None*) – Passed to indicate in the output file header that the targets have been limited to only certain HEALPixels at a given nside. Used in conjunction with *hpxlist*.
- **hpxlist** (list, optional, defaults to *None*) – Passed to indicate in the output file header that the targets have been limited to only this list of HEALPixels. Used in conjunction with *nsidefile*.
- **extra** (dict, optional) – If passed (and not None), write these extra dictionary keys and values to the output header.
- **mock** (bool, optional, defaults to `False`.) – If `True` then construct the file path for mock sky target catalogs.

Returns

- int – The number of skies that were written to file.
- str – The name of the file to which skies were written.

desitarget.io.**write_targets**(*targdir*, *data*, *indir=None*, *indir2=None*, *nchunks=None*, *qso_selection=None*, *nside=None*, *survey='main'*, *nsidefile=None*, *hpxlist=None*, *scndout=None*, *resolve=True*, *maskbits=True*, *obscon=None*, *mockdata=None*, *supp=False*, *extra=None*)

Write target catalogues.

Parameters

- **targdir** (str) – Path to output target selection directory (the directory structure and file name are built on-the-fly from other inputs).
- **data** (ndarray) – numpy structured array of targets to save.
- **indir2, qso_selection** (*indir,*) – If passed, note these as the input directory, an additional input directory, and the QSO selection method in the output file header.
- **nchunks** (:class'int', optional, defaults to *None*) – The number of chunks in which to write the output file, to save memory. Send *None* to write everything at once.
- **nside** (int, optional, defaults to *None*) – If passed, add a column to the targets array popluated with HEALPixels at resolution *nside*.
- **survey** (str, optional, defaults to "main") – Written to output file header as the keyword *SURVEY*.
- **nsidefile** (int, optional, defaults to *None*) – Passed to indicate in the output file header that the targets have been limited to only certain HEALPixels at a given nside. Used in conjunction with *hpxlist*.
- **hpxlist** (list, optional, defaults to *None*) – Passed to indicate in the output file header that the targets have been limited to only this list of HEALPixels. Used in conjunction with *nsidefile*.
- **maskbits** (*resolve,*) – Written to the output file header as *RESOLVE*, *MASKBITS*.
- **scndout** (str, optional, defaults to *None*) – If passed, add to output header as SCNDOUT.
- **obscon** (str, optional, defaults to *None*) – Can pass one of "DARK" or "BRIGHT". If passed, don't write the full set of data, rather only write targets appropriate for "DARK|GRAY" or "BRIGHT" observing conditions. The relevant *PRIORITY_INIT* and

NUMOBS_INIT columns will be derived from *PRIORITY_INIT_DARK*, etc. and *filename* will have "bright" or "dark" appended to the lowest DIRECTORY in the input *filename*.

- **mockdata** (dict, optional, defaults to *None*) – Dictionary of mock data to write out (only used in *desitarget.mock.build.targets_truth* via *select_mock_targets*).
- **supp** (bool, optional, defaults to False) – Written to the header of the output file to indicate whether this is a file of supplemental targets (targets that are outside the Legacy Surveys footprint).
- **extra** (dict, optional) – If passed (and not None), write these extra dictionary keys and values to the output header.

Returns

- int – The number of targets that were written to file.
- str – The name of the file to which targets were written.

1.12 desitarget.mock

This subpackage contains modules that handle mock data.

1.13 desitarget.mock.build

Build truth and targets catalogs, including spectra, for the mocks.

desitarget.mock.build.**_get_spectra_onepixel**(*specargs*)
Filler function for the multiprocessing.

desitarget.mock.build.**_merge_file_tables**(*fileglob*, *ext*, *outfile=None*, *comm=None*, *addcols=None*, *overwrite=False*)
parallel merge tables from individual files into an output file

Parameters

- **fileglob** (*str*) – glob of files to combine (e.g. '*/blat-*.fits')
- **ext** (*str or int*) – FITS file extension name or number

Options: outfile (str): output file to write comm: MPI communicator object addcols: dict extra columns to add with fill values, e.g. dict(OBSCONDITIONS=1)

Returns merged table as np.ndarray

desitarget.mock.build.**density_fluctuations**(*data*, *log*, *nside*, *nside_chunk*, *seed=None*)
Determine the density of targets to generate, accounting for fluctuations due to reddening, imaging systematics, and large-scale structure.

Parameters

- **data** (dict) – Data on the input mock targets (to be documented).
- **log** (desiutil.logger) – Logger object.
- **nside** (int) – Healpix resolution.
- **nside_chunk** (int) – Healpix resolution for chunking the sample.
- **seed** (int, optional) – Seed for the random number generator. Defaults to None.

Returns

- **indxperchunk** (list) – Indices (in data) of the mock targets to generate per healpixel chunk.
- **ntargperchunk** (numpy.ndarray) – Number of targets to generate per healpixel chunk.
- **areaperpixel** (float) – Area per healpixel (used to construct useful log messages).

desitarget.mock.build.**finish_catalog**(*targets*, *truth*, *objtruth*, *skytargets*, *skytruth*, *healpix*, *nside*, *log*, *seed=None*, *survey='main'*)

Add various mission-critical columns to the target catalog, including hpxpixel, brick_objid, targetid, subpriority, priority, and numobs.

Parameters

- **targets** (astropy.table.Table) – Final set of targets.
- **truth** (astropy.table.Table) – Corresponding truth table for targets.
- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).
- **skytargets** (astropy.table.Table) – Sky targets.
- **skytruth** (astropy.table.Table) – Corresponding truth table for sky targets.
- **healpix** (: int) – Healpixel number.
- **nside** (int) – Nside corresponding to healpix.
- **log** (desiutil.logger) – Logger object.
- **seed** (int, optional) – Seed for the random number generation. Defaults to None.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

Returns

Return type Updated versions of targets, truth, objtruth, skytargets, and skytruth.

desitarget.mock.build.**get_contaminants_onepixel**(*params*, *healpix*, *nside*, *seed*, *nproc*, *log*, *nside_chunk*, *targets*, *truth*, *objtruth*, *trueflux*, *ContamStarsMock=None*, *ContamGalaxiesMock=None*, *no_spectra=False*)

Generate spectra (in parallel) for a set of targets.

Parameters

- **params** (dict) – Dictionary defining the type and number of contaminants.
- **healpix** (: int) – Healpixel number.
- **nside** (int) – Nside corresponding to healpix.
- **seed** (int, optional) – Seed for the random number generation.
- **nproc** (int, optional) – Number of parallel processes to use.
- **log** (desiutil.logger) – Logger object.
- **nside_chunk** (int) – Healpix resolution for chunking the sample to avoid memory problems.
- **targets** (astropy.table.Table) – Target catalog.

- **truth** (astropy.table.Table) – Corresponding truth table.
- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).
- **trueflux** (numpy.ndarray) – Array [npixel, ntarget] of observed-frame spectra. Only computed and returned for non-sky targets and if no_spectra=False.
- **ContamStarsMock** (*desitarget.mock.mockmaker* object) – Maker Class for generating stellar contaminants.
- **ContamGalaxiesMock** (*desitarget.mock.mockmaker* object) – Maker Class for generating extragalactic contaminants.
- **no_spectra** (bool, optional) – Do not generate spectra, e.g., for use with quicksurvey. Defaults to False.

Returns

- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).
- **trueflux** (numpy.ndarray) – Corresponding noiseless spectra.

desitarget.mock.build.**get_spectra**(*data*, *MakeMock*, *log*, *nside*, *nside_chunk*, *seed=None*, *nproc=1*, *sky=False*, *no_spectra=False*, *calib_only=False*, *contaminants=False*)

Generate spectra (in parallel) for a set of targets.

Parameters

- **data** (dict) – Data on the input mock targets (to be documented).
- **MakeMock** (*desitarget.mock.mockmaker* object) – Object to assign spectra to each target class.
- **log** (desiutil.logger) – Logger object.
- **nside** (int) – Healpix resolution corresponding to healpixels.
- **nside_chunk** (int) – Healpix resolution for chunking the sample to avoid memory problems.
- **seed** (int, optional) – Seed for the random number generator. Defaults to None.
- **nproc** (int, optional) – Number of parallel processes to use. Defaults to 1.
- **sky** (bool) – Processing sky targets (which are a special case). Defaults to False.
- **no_spectra** (bool, optional) – Do not generate spectra, e.g., for use with quicksurvey. Defaults to False.
- **calib_only** (bool, optional) – Use targets as calibration (standard star) targets, only. Defaults to False.
- **contaminants** (bool, optional) – Generate spectra for contaminants (mostly affects the log messages). Defaults to False.

Returns

- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.

- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).
- **trueflux** (numpy.ndarray) – Corresponding noiseless spectra.

desitarget.mock.build.**get_spectra_onepixel**(*data*, *indx*, *MakeMock*, *seed*, *log*, *ntarget*, *maxiter=1*, *no_spectra=False*, *calib_only=False*)

Wrapper function to generate spectra for all targets on a single healpixel.

Parameters

- **data** (dict) – Dictionary with all the mock data (candidate mock targets).
- **indx** (int or numpy.ndarray) – Indices of candidate mock targets to consider.
- **MakeMock** (*desitarget.mock.mockmaker* object) – Object to assign spectra to each target class.
- **seed** (int) – Seed for the random number generator.
- **log** (desiutil.logger) – Logger object.
- **ntarget** (int) – Desired number of targets to generate.
- **maxiter** (int) – Maximum number of iterations to generate targets.
- **no_spectra** (bool, optional) – Do not generate spectra, e.g., for use with quicksurvey. Defaults to False.
- **calib_only** (bool, optional) – Use targets as calibration (standard star) targets, only. Defaults to False.

Returns

- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).
- **trueflux** (numpy.ndarray) – Array [npixel, ntarget] of observed-frame spectra. Only computed and returned for non-sky targets and if no_spectra=False.

desitarget.mock.build.**initialize_targets_truth**(*params*, *healpixels=None*, *nside=None*, *output_dir='.'*, *seed=None*, *verbose=False*)

Initialize various objects needed to generate mock targets.

Parameters

- **params** (dict) – Dictionary defining the mock from which to generate targets.
- **healpixels** (numpy.ndarray or int) – Generate mock targets within this set of healpix pixels.
- **nside** (int) – Healpix resolution corresponding to healpixels.
- **output_dir** (str, optional.) – Output directory. Defaults to '.' (current directory).
- **seed** (int, optional) – Seed for the random number generator. Defaults to None.
- **verbose** (bool, optional) – Be verbose. Defaults to False.

Returns

- **log** (desiutil.logger) – Logger object.

- **healpixseeds** (numpy.ndarray or int) – Array of random number seeds (one per healpixels pixel) needed to ensure reproducibility.

Raises ValueError – If params, healpixels, or nside are not defined. A ValueError is also raised if nside > 256, since this exceeds the number of bits that can be accommodated by desitarget.targets.encode_targetid.

desitarget.mock.build.**join_targets_truth**(*mockdir*, *outdir=None*, *overwrite=False*, *comm=None*)

Join individual healpixel targets and truth files into combined tables

Parameters **mockdir** – top level mock target directory

Options: outdir: output directory, default to mockdir overwrite: rewrite outputs even if they already exist comm: MPI communicator; if not None, read data in parallel

desitarget.mock.build.**read_mock**(*params*, *log=None*, *target_name=''*, *seed=None*, *healpixels=None*, *nside=None*, *nside_chunk=128*, *MakeMock=None*, *dndz=None*)

Read a mock catalog.

Parameters

- **params** (dict) – Dictionary summary of the input configuration file, restricted to a particular target (e.g., 'QSO').
- **log** (desiutil.logger) – Logger object.
- **target_name** (str) – Target name; mock.mockmaker.[TARGET_NAME]Maker class to instantiate.
- **seed** (int, optional) – Seed for the random number generator. Defaults to None.
- **healpixels** (numpy.ndarray or *int*) – List of healpixels to read.
- **nside** (int) – Healpix resolution corresponding to healpixels.
- **nside_chunk** (int, optional) – Healpix resolution for chunking the sample to avoid memory problems. (NB: nside_chunk must be <= nside). Defaults to 128.
- **dndz** (dict, optional) – Expected redshift distributions for all target classes. Defaults to None.

Returns **data** – Parsed target data based on the input mock catalog (to be documented).

Return type dict

Raises ValueError – If the mock_density was not returned when expected.

desitarget.mock.build.**targets_truth**(*params*, *healpixels=None*, *nside=None*, *output_dir='.'*, *seed=None*, *nproc=1*, *nside_chunk=128*, *survey='main'*, *verbose=False*, *no_spectra=False*)

Generate truth and targets catalogs, and noiseless spectra.

Parameters

- **params** (dict) – Target parameters.
- **healpixels** (numpy.ndarray or int) – Restrict the sample of mock targets analyzed to those lying inside this set (array) of healpix pixels.
- **nside** (int) – Healpix resolution corresponding to healpixels.
- **output_dir** (str, optional) – Output directory. Defaults to '.' (current directory).
- **seed** (int) – Seed for the random number generation. Defaults to None.

- **nproc** (int, optional) – Number of parallel processes to use. Defaults to 1 (i.e., no multiprocessing).
- **nside_chunk** (int, optional) – Healpix resolution for chunking the sample to avoid memory problems. (NB: nside_chunk must be <= nside). Defaults to 128.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.
- **verbose** (bool, optional) – Be verbose. Defaults to False.
- **no_spectra** (bool, optional) – Do not generate spectra, e.g., for use with quicksurvey.

Returns

- *Files 'targets.fits', 'truth.fits', 'sky.fits', 'standards-dark.fits', and*
- *'standards-bright.fits' written to output_dir for the given list of*
- *healpixels.*

1.14 desitarget.mock.io

Code to find the location of the mock data.

desitarget.mock.io.**findfile**(*filetype*, *nside*, *pixnum*, *basedir='.'*, *ext='fits'*, *obscon=None*)

Returns standardized filepath

Parameters

- **filetype** – (str) file prefix, e.g. 'sky' or 'targets'
- **nside** – (int) healpix nside 2**k with 0<k<30
- **pixnum** – (int) healpix NESTED pixel number for this nside

Optional: basedir: (str) base directory ext: (str) file extension obscon: (str) e.g. 'dark', 'bright' to add extra dir grouping

desitarget.mock.io.**get_healpix_dir**(*nside*, *pixnum*, *basedir='.'*)

Returns standardized path

Parameters

- **nside** – (int) healpix nside 2**k with 0<k<30
- **pixnum** – (int) healpix NESTED pixel number for this nside

Optional: basedir: (str) base directory

Note: may standardize with functions in desispec.io, but separate for now

1.15 desitarget.mock.mockmaker

Read mock catalogs and assign spectra.

class desitarget.mock.mockmaker.**BGSMaker**(*seed=None*, *nside_chunk=128*, *survey='main'*, ***kwargs*)

Read BGS mocks, generate spectra, and select targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **nside_chunk** (int, optional) – Healpixel nside for further subdividing the sample when assigning velocity dispersion to targets. Defaults to 128.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

make_spectra(*data=None*, *indx=None*, *seed=None*, *no_spectra=False*)
Generate BGS spectra.

Parameters

- **data** (dict) – Dictionary of source properties.
- **indx** (numpy.ndarray, optional) – Generate spectra for a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not generate spectra. Defaults to False.

Returns

- **flux** (numpy.ndarray) – Target spectra.
- **wave** (numpy.ndarray) – Corresponding wavelength array.
- **meta** (astropy.table.Table) – Spectral metadata table.
- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).

read(*mockfile=None*, *mockformat='durham_mxxl_hdf5'*, *healpixels=None*, *nside=None*, *magcut=None*, *only_coords=False*, *mock_density=False*, ***kwargs*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **mockformat** (str) – Mock catalog format. Defaults to 'durham_mxxl_hdf5'.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **magcut** (float) – Magnitude cut (hard-coded to SDSS r-band) to subselect targets brighter than magcut.
- **only_coords** (bool, optional) – For various applications, only read the target coordinates.
- **mock_density** (bool, optional) – Compute the median target density in the mock. Defaults to False.

Returns Dictionary of target properties with various keys (to be documented).

Return type dict

Raises ValueError – If mockformat is not recognized.

select_targets(*targets*, *truth*, *targetname='BGS'*)
Select BGS targets. Input tables are modified in place.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.

class desitarget.mock.mockmaker.**BuzzardMaker**(*seed=None*, *nside_chunk=128*, *no_spectra=False*, *survey='main'*, ***kwargs*)
Read Buzzard mocks, generate spectra, and select targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not pre-select extragalactic contaminants. Defaults to False.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

extragalactic_contaminants(*seed*, *nmonte=100*)
Pre-select Buzzard/BGS templates that could be photometric contaminants.

make_spectra(*data=None*, *indx=None*, *seed=None*, *no_spectra=False*)
Generate BGS spectra.

Parameters

- **data** (dict) – Dictionary of source properties.
- **indx** (numpy.ndarray, optional) – Generate spectra for a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not generate spectra. Defaults to False.

Returns

- **flux** (numpy.ndarray) – Target spectra.
- **wave** (numpy.ndarray) – Corresponding wavelength array.
- **meta** (astropy.table.Table) – Spectral metadata table.
- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).

read(*mockfile=None*, *mockformat='buzzard'*, *healpixels=None*, *nside=None*, *nside_buzzard=8*, *target_name=''*, *magcut=None*, *only_coords=False*, ***kwargs*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **mockformat** (str) – Mock catalog format. Defaults to 'buzzard'.
- **healpixels** (int) – Healpixel number to read.

- **nside** (int) – Healpixel nside corresponding to healpixels.
- **nside_buzzard** (int) – Healpixel nside indicating how the mock on-disk has been organized. Defaults to 8.
- **target_name** (str) – Name of the target being read. Defaults to ''.
- **magcut** (float) – Magnitude cut (hard-coded to DECam r-band) to subselect targets brighter than magcut.
- **only_coords** (bool, optional) – For various applications, only read the target coordinates.

Returns Dictionary of target properties with various keys (to be documented).

Return type dict

Raises ValueError – If mockformat is not recognized.

select_targets(*targets*, *truth*, *targetname='BGS'*)
Select extragalactic contaminants. Input tables are modified in place.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **targetname** (str) – Target selection cuts to apply.

class desitarget.mock.mockmaker.**ELGMaker**(*seed=None*, *nside_chunk=128*, *survey='main'*, ***kwargs*)
Read ELG mocks, generate spectra, and select targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **nside_chunk** (int, optional) – Healpixel nside for further subdividing the sample when assigning velocity dispersion to targets. Defaults to 128.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

make_spectra(*data=None*, *indx=None*, *seed=None*, *no_spectra=False*)
Generate ELG spectra.

Parameters

- **data** (dict) – Dictionary of source properties.
- **indx** (numpy.ndarray, optional) – Generate spectra for a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not generate spectra. Defaults to False.

Returns

- **flux** (numpy.ndarray) – Target spectra.
- **wave** (numpy.ndarray) – Corresponding wavelength array.
- **meta** (astropy.table.Table) – Spectral metadata table.
- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.

- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).

read(*mockfile=None*, *mockformat='gaussianfield'*, *healpixels=None*, *nside=None*, *only_coords=False*, *mock_density=False*, ***kwargs*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **mockformat** (str) – Mock catalog format. Defaults to 'gaussianfield'.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **only_coords** (bool, optional) – For various applications, only read the target coordinates.
- **mock_density** (bool, optional) – Compute the median target density in the mock. Defaults to False.

Returns Dictionary of target properties with various keys (to be documented).

Return type dict

Raises ValueError – If mockformat is not recognized.

select_targets(*targets*, *truth*, *targetname='ELG'*)
Select ELG targets. Input tables are modified in place.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **targetname** (str) – Target selection cuts to apply.

class desitarget.mock.mockmaker.**LRGMaker**(*seed=None*, *nside_chunk=128*, *survey='main'*, ***kwargs*)
Read LRG mocks, generate spectra, and select targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **nside_chunk** (int, optional) – Healpixel nside for further subdividing the sample when assigning velocity dispersion to targets. Defaults to 128.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

make_spectra(*data=None*, *indx=None*, *seed=None*, *no_spectra=False*)
Generate LRG spectra.

Parameters

- **data** (dict) – Dictionary of source properties.
- **indx** (numpy.ndarray, optional) – Generate spectra for a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not generate spectra. Defaults to False.

Returns

- **flux** (numpy.ndarray) – Target spectra.
- **wave** (numpy.ndarray) – Corresponding wavelength array.
- **meta** (astropy.table.Table) – Spectral metadata table.
- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).

read(*mockfile=None*, *mockformat='gaussianfield'*, *healpixels=None*, *nside=None*, *only_coords=False*, *mock_density=False*, ***kwargs*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **mockformat** (str) – Mock catalog format. Defaults to 'gaussianfield'.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **only_coords** (bool, optional) – For various applications, only read the target coordinates.
- **mock_density** (bool, optional) – Compute the median target density in the mock. Defaults to False.

Returns Dictionary of target properties with various keys (to be documented).

Return type dict

Raises ValueError – If mockformat is not recognized.

select_targets(*targets*, *truth*, *targetname='LRG'*)
Select LRG targets. Input tables are modified in place.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **targetname** (str) – Target selection cuts to apply.

class desitarget.mock.mockmaker.**LYAMaker**(*seed=None*, *use_simqso=True*, *sqmodel='default'*, *balprob=0.0*, *add_dla=False*, *add_metals=False*, *add_lyb=False*, *survey='main'*, ***kwargs*)
Read LYA mocks, generate spectra, and select targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **balprob** (float, optional) – Probability of a including one or more BALs. Defaults to 0.0.
- **add_dla** (bool, optional) – Statistically include DLAs along the line of sight.

- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

make_spectra(*data=None*, *indx=None*, *seed=None*, *no_spectra=False*, *add_dlas=None*, *add_metals=None*, *add_lyb=None*)
Generate QSO spectra with the 3D Lya forest skewers included.

Parameters

- **data** (dict) – Dictionary of source properties.
- **indx** (numpy.ndarray, optional) – Generate spectra for a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not generate spectra. Defaults to False.

Returns

- **flux** (numpy.ndarray) – Target spectra.
- **wave** (numpy.ndarray) – Corresponding wavelength array.
- **meta** (astropy.table.Table) – Spectral metadata table.
- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.

Raises KeyError – If there is a mismatch between MOCKID in the data dictionary and the skewer files on-disk.

read(*mockfile=None*, *mockformat='CoLoRe'*, *healpixels=None*, *nside=None*, *nside_lya=16*, *zmin_lya=None*, *mock_density=False*, *only_coords=False*, ***kwargs*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **mockformat** (str) – Mock catalog format. Defaults to 'CoLoRe'.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **nside_lya** (int) – Healpixel nside indicating how the mock on-disk has been organized. Defaults to 16.
- **zmin_lya** (float) – Minimum redshift of Lya skewers, to ensure no double-counting with QSO mocks. Defaults to None.
- **mock_density** (bool, optional) – Compute the median target density in the mock. Defaults to False.
- **only_coords** (bool, optional) – Only read the target coordinates and some other basic info. Defaults to False.

Returns Dictionary of target properties with various keys (to be documented).

Return type dict

Raises ValueError – If mockformat is not recognized.

select_targets(*targets*, *truth*, *targetname='QSO'*)
Select Lya/QSO targets. Input tables are modified in place.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **targetname** (str) – Target selection cuts to apply.

class desitarget.mock.mockmaker.**MWS_MAINMaker**(*seed=None*, *calib_only=False*, *no_spectra=False*, *survey='main'*, ***kwargs*)

Read MWS_MAIN mocks, generate spectra, and select targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **calib_only** (bool, optional) – Use MWS_MAIN stars as calibration (standard star) targets, only. Defaults to False.
- **no_spectra** (bool, optional) – Initialize and cache template photometry. Defaults to False.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

make_spectra(*data=None*, *indx=None*, *seed=None*, *no_spectra=False*)

Generate MWS_MAIN stellar spectra.

Parameters

- **data** (dict) – Dictionary of source properties.
- **indx** (numpy.ndarray, optional) – Generate spectra for a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not generate spectra. Defaults to False.

Returns

- **flux** (numpy.ndarray) – Target spectra.
- **wave** (numpy.ndarray) – Corresponding wavelength array.
- **meta** (astropy.table.Table) – Spectral metadata table.
- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.

read(*mockfile=None*, *mockformat='galaxia'*, *healpixels=None*, *nside=None*, *nside_galaxia=8*, *target_name='MWS_MAIN'*, *magcut=None*, *faintstar_mockfile=None*, *faintstar_magcut=None*, ***kwargs*)

Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **mockformat** (str) – Mock catalog format. Defaults to 'galaxia'.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.

- **nside_galaxia** (int) – Healpixel nside indicating how the mock on-disk has been organized. Defaults to 8.
- **target_name** (str) – Name of the target being read. Defaults to 'MWS_MAIN'.
- **magcut** (float) – Magnitude cut (hard-coded to SDSS r-band) to subselect targets brighter than magcut.
- **faintstar_mockfile** (str, optional) – Full path to the top-level directory of the Galaxia faint star mock catalog.
- **faintstar_magcut** (float, optional) – Magnitude cut (hard-coded to SDSS r-band) to subselect faint star targets brighter than magcut.

Returns Dictionary of target properties with various keys (to be documented).

Return type dict

Raises ValueError – If mockformat is not recognized.

select_targets(*targets*, *truth*, *targetname='MWS_MAIN'*)
Select various MWS stars and standard stars. Input tables are modified in place.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **targetname** (str) – Target selection cuts to apply.

class desitarget.mock.mockmaker.**MWS_NEARBYMaker**(*seed=None*, *no_spectra=False*, *survey='main'*, ***kwargs*)
Read MWS_NEARBY mocks, generate spectra, and select targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not initialize template photometry. Defaults to False.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

make_spectra(*data=None*, *indx=None*, *seed=None*, *no_spectra=False*)
Generate MWS_NEARBY stellar spectra.

Parameters

- **data** (dict) – Dictionary of source properties.
- **indx** (numpy.ndarray, optional) – Generate spectra for a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not generate spectra. Defaults to False.

Returns

- **flux** (numpy.ndarray) – Target spectra.
- **wave** (numpy.ndarray) – Corresponding wavelength array.
- **meta** (astropy.table.Table) – Spectral metadata table.
- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.

read(*mockfile=None*, *mockformat='mws_100pc'*, *healpixels=None*, *nside=None*, *mock_density=False*, ***kwargs*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **mockformat** (str) – Mock catalog format. Defaults to 'mws_100pc'.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **mock_density** (bool, optional) – Compute the median target density in the mock. Defaults to False.

Returns Dictionary of target properties with various keys (to be documented).

Return type dict

Raises ValueError – If mockformat is not recognized.

select_targets(*targets*, *truth*, *targetname='MWS'*)
Select MWS_NEARBY targets. Input tables are modified in place.

Note: The selection here eventually will be done with Gaia (I think) so for now just do a "perfect" selection.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **targetname** (str) – Target selection cuts to apply.

class desitarget.mock.mockmaker.**QSOMaker**(*seed=None*, *use_simqso=True*, *survey='main'*, ***kwargs*)
Read QSO mocks, generate spectra, and select targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **use_simqso** (bool, optional) – Use desisim.templates.SIMQSO to generated templates rather than desisim.templates.QSO. Defaults to True.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

make_spectra(*data=None*, *indx=None*, *seed=None*, *no_spectra=False*)
Generate tracer QSO spectra.

Parameters

- **data** (dict) – Dictionary of source properties.
- **indx** (numpy.ndarray, optional) – Generate spectra for a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not generate spectra. Defaults to False.

Returns

- **flux** (numpy.ndarray) – Target spectra.
- **wave** (numpy.ndarray) – Corresponding wavelength array.

- **meta** (astropy.table.Table) – Spectral metadata table.
- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).

read(*mockfile=None*, *mockformat='gaussianfield'*, *healpixels=None*, *nside=None*, *zmax_qso=None*, *only_coords=False*, *mock_density=False*, ***kwargs*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **mockformat** (str) – Mock catalog format. Defaults to 'gaussianfield'.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **zmax_qso** (float) – Maximum redshift of tracer QSOs to read, to ensure no double-counting with Lya mocks. Defaults to None.
- **only_coords** (bool, optional) – For various applications, only read the target coordinates.
- **mock_density** (bool, optional) – Compute the median target density in the mock. Defaults to False.

Returns **data** – Dictionary of target properties with various keys (to be documented).

Return type dict

Raises ValueError – If mockformat is not recognized.

select_targets(*targets*, *truth*, *targetname='QSO'*)
Select QSO targets. Input tables are modified in place.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **targetname** (str) – Target selection cuts to apply.

class desitarget.mock.mockmaker.**ReadBuzzard**(***kwargs*)
Read a Buzzard style mock catalog.

readmock(*mockfile=None*, *healpixels=[]*, *nside=[]*, *nside_buzzard=8*, *target_name=''*, *magcut=None*, *only_coords=False*, *seed=None*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **nside_buzzard** (int) – Healpixel nside indicating how the mock on-disk has been organized. Defaults to 8.
- **target_name** (str) – Name of the target being read (e.g., ELG, LRG).

- **magcut** (float) – Magnitude cut (hard-coded to DECam r-band) to subselect targets brighter than magcut.
- **only_coords** (bool, optional) – To get some improvement in speed, only read the target coordinates and some other basic info.
- **seed** (int, optional) – Seed for reproducibility and random number generation.

Returns Dictionary with various keys (to be documented).

Return type dict

Raises

- IOError – If the top-level Galaxia directory is not found.
- ValueError – (1) If either mockfile or nside_galaxia are not defined; (2) if healpixels or nside are not scalar inputs; or (3) if the input target_name is not recognized.

class desitarget.mock.mockmaker.**ReadGAMA**(***kwargs*)

Read a GAMA catalog of BGS targets. This reader will only generally be used for the Survey Validation Data Challenge.

readmock(*mockfile=None*, *healpixels=None*, *nside=None*, *target_name=''*, *magcut=None*, *only_coords=False*)

Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **target_name** (str) – Name of the target being read (e.g., ELG, LRG).
- **magcut** (float) – Magnitude cut (hard-coded to SDSS r-band) to subselect targets brighter than magcut.
- **only_coords** (bool, optional) – To get some improvement in speed, only read the target coordinates and some other basic info.

Returns Dictionary with various keys (to be documented).

Return type dict

Raises

- IOError – If the mock data file is not found.
- ValueError – If mockfile or healpixels are not defined, or if nside is not a scalar.

class desitarget.mock.mockmaker.**ReadGalaxia**(***kwargs*)

Read a Galaxia style mock catalog.

readmock(*mockfile=None*, *healpixels=[]*, *nside=[]*, *nside_galaxia=8*, *target_name='MWS_MAIN'*, *magcut=None*, *faintstar_mockfile=None*, *faintstar_magcut=None*, *seed=None*)

Read the catalog.

Parameters

- **mockfile** (str) – Full path to the top-level directory of the Galaxia mock catalog.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.

- **nside_galaxia** (int) – Healpixel nside indicating how the mock on-disk has been organized. Defaults to 8.
- **target_name** (str) – Name of the target being read (e.g., MWS_MAIN).
- **magcut** (float) – Magnitude cut (hard-coded to SDSS r-band) to subselect targets brighter than magcut.
- **faintstar_mockfile** (str, optional) – Full path to the top-level directory of the Galaxia faint star mock catalog.
- **faintstar_magcut** (float, optional) – Magnitude cut (hard-coded to SDSS r-band) to subselect faint star targets brighter than magcut.
- **seed** (int, optional) – Seed for reproducibility and random number generation.

Returns Dictionary with various keys (to be documented).

Return type dict

Raises

- IOError – If the top-level Galaxia directory is not found.
- ValueError – (1) If either mockfile or nside_galaxia are not defined; (2) if healpixels or nside are not scalar inputs; or (3) if the input target_name is not recognized.

class desitarget.mock.mockmaker.**ReadGaussianField**(***kwargs*)

Read a Gaussian random field style mock catalog.

readmock(*mockfile=None*, *healpixels=None*, *nside=None*, *zmax_qso=None*, *target_name=''*, *mock_density=False*, *only_coords=False*, *seed=None*)

Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **zmax_qso** (float) – Maximum redshift of tracer QSOs to read, to ensure no double-counting with Lya mocks. Defaults to None.
- **target_name** (str) – Name of the target being read (e.g., ELG, LRG).
- **mock_density** (bool, optional) – Compute and return the median target density in the mock. Defaults to False.
- **only_coords** (bool, optional) – To get some improvement in speed, only read the target coordinates and some other basic info.
- **seed** (int, optional) – Seed for reproducibility and random number generation.

Returns Dictionary with various keys (to be documented).

Return type dict

Raises

- IOError – If the mock data file is not found.
- ValueError – If mockfile is not defined or if nside is not a scalar.

class desitarget.mock.mockmaker.**ReadLyaCoLoRe**(***kwargs*)

Read a CoLoRe mock catalog of Lya skewers.

readmock(*mockfile=None*, *healpixels=None*, *nside=None*, *target_name='LYA'*, *nside_lya=16*, *zmin_lya=None*, *mock_density=False*, *sqmodel='default'*, *only_coords=False*, *seed=None*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the top-level directory of the CoLoRe mock catalog.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **target_name** (str) – Name of the target being read (if not LYA).
- **nside_lya** (int) – Healpixel nside indicating how the mock on-disk has been organized. Defaults to 16.
- **zmin_lya** (float) – Minimum redshift of Lya skewers, to ensure no double-counting with QSO mocks. Defaults to None.
- **mock_density** (bool, optional) – Compute and return the median target density in the mock. Defaults to False.
- **only_coords** (bool, optional) – Only read the target coordinates and some other basic info. Defaults to False.
- **seed** (int, optional) – Seed for reproducibility and random number generation.

Returns Dictionary with various keys (to be documented).

Return type dict

Raises

- IOError – If the top-level mock data file is not found.
- ValueError – If mockfile, nside, or nside_lya are not defined.

class desitarget.mock.mockmaker.**ReadMWS_NEARBY**(***kwargs*)
Read a mock catalog of Milky Way Survey nearby targets (MWS_NEARBY).

readmock(*mockfile=None*, *healpixels=None*, *nside=None*, *target_name='MWS_NEARBY'*, *mock_density=False*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **target_name** (str) – Name of the target being read (if not MWS_NEARBY).
- **mock_density** (bool, optional) – Compute and return the median target density in the mock. Defaults to False.

Returns Dictionary with various keys (to be documented).

Return type dict

Raises

- IOError – If the mock data file is not found.
- ValueError – If mockfile is not defined or if nside is not a scalar.

class desitarget.mock.mockmaker.**ReadMWS_WD**(***kwargs*)
Read a mock catalog of Milky Way Survey white dwarf targets (MWS_WD).

readmock(*mockfile=None*, *healpixels=None*, *nside=None*, *target_name='WD'*, *mock_density=False*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **target_name** (str) – Name of the target being read (if not WD).
- **mock_density** (bool, optional) – Compute and return the median target density in the mock. Defaults to False.

Returns Dictionary with various keys (to be documented).

Return type dict

Raises

- IOError – If the mock data file is not found.
- ValueError – If mockfile is not defined or if nside is not a scalar or if the selection index isn't monotonically increasing.

class desitarget.mock.mockmaker.**ReadMXXL**(***kwargs*)
Read a MXXL mock catalog of BGS targets.

readmock(*mockfile=None*, *healpixels=None*, *nside=None*, *target_name='BGS'*, *magcut=None*, *only_coords=False*, *mock_density=False*, *seed=None*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the top-level directory of the CoLoRe mock catalog.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **target_name** (str) – Name of the target being read (if not BGS).
- **magcut** (float) – Magnitude cut (hard-coded to SDSS r-band) to subselect targets brighter than magcut.
- **only_coords** (bool, optional) – To get some improvement in speed, only read the target coordinates and some other basic info.
- **mock_density** (bool, optional) – Compute and return the median target density in the mock. Defaults to False.
- **seed** (int, optional) – Seed for reproducibility and random number generation.

Returns Dictionary with various keys (to be documented).

Return type dict

Raises

- IOError – If the mock data file is not found.
- ValueError – If mockfile is not defined or if nside is not a scalar.

class desitarget.mock.mockmaker.**ReadUniformSky**(***kwargs*)
Read a uniform sky style mock catalog.

readmock(*mockfile=None*, *healpixels=None*, *nside=None*, *target_name=''*, *mock_density=False*, *only_coords=False*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **target_name** (str) – Name of the target being read (e.g., ELG, LRG).
- **mock_density** (bool, optional) – Compute and return the median target density in the mock. Defaults
- **only_coords** (bool, optional) – To get some improvement in speed, only read the target coordinates and some other basic info. Defaults to False.

Returns Dictionary with various keys (to be documented).

Return type dict

Raises

- IOError – If the mock data file is not found.
- ValueError – If mockfile is not defined or if nside is not a scalar.

class desitarget.mock.mockmaker.**SKYMaker**(*seed=None*, *survey='main'*, ***kwargs*)
Read SKY mocks, generate spectra, and select targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

make_spectra(*data=None*, *indx=None*, *seed=None*, *no_spectra=False*)
Generate SKY spectra.

Parameters

- **data** (dict) – Dictionary of source properties.
- **indx** (numpy.ndarray, optional) – Generate spectra for a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not generate spectra. Defaults to False.

Returns

- **flux** (numpy.ndarray) – Target spectra.
- **wave** (numpy.ndarray) – Corresponding wavelength array.
- **meta** (astropy.table.Table) – Spectral metadata table.
- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.

- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).

read(*mockfile=None*, *mockformat='uniformsky'*, *healpixels=None*, *nside=None*, *only_coords=False*, *mock_density=False*, ***kwargs*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **mockformat** (str) – Mock catalog format. Defaults to 'gaussianfield'.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **only_coords** (bool, optional) – For various applications, only read the target coordinates.
- **mock_density** (bool, optional) – Compute the median target density in the mock. Defaults to False.

Returns Dictionary of target properties with various keys (to be documented).

Return type dict

Raises ValueError – If mockformat is not recognized.

select_targets(*targets*, *truth*, *targetname='SKY'*)
Select SKY targets (i.e., everything). Input tables are modified in place.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **targetname** (str) – Target selection cuts to apply.

class desitarget.mock.mockmaker.**STARMaker**(*seed=None*, *no_spectra=False*, *survey='main'*, ***kwargs*)
Lower-level Class for preparing for stellar spectra to be generated, selecting standard stars, and selecting stars as contaminants for extragalactic targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Initialize and cache template photometry. Defaults to False.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

template_photometry(*data=None*, *indx=None*, *rand=None*, *south=True*)
Get stellar photometry from the templates themselves, by-passing the generation of spectra.

class desitarget.mock.mockmaker.**SelectTargets**(*bricksize=0.25*, *survey='main'*, ***kwargs*)
Methods to help select various target types.

Parameters

- **bricksize** (float, optional) – Brick diameter used in the imaging surveys; needed to assign a brickname and brickid to each object. Defaults to 0.25 deg.

- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

KDTree_build(*matrix*, *south=True*, *subtype=''*)
Build a KD-tree.

KDTree_query(*matrix*, *return_dist=False*, *south=True*, *subtype=''*)
Return the nearest template number based on the KD Tree.

KDTree_rescale(*matrix*, *south=False*, *subtype=''*)
Normalize input parameters to [0, 1].

_nospectra_photometry(*meta*, *rand*, *data*, *indx*, *target_name*, *contaminants=False*)
Populate the photometry in meta in no-spectra mode.

_qaplot_scatter_photometry(*targets*, *truth*)
Build a simple QAplot, useful for debugging

_sample_vdisp(*ra*, *dec*, *mean=1.9*, *sigma=0.15*, *fracvdisp=(0.1, 1)*, *seed=None*, *nside=128*)
Assign velocity dispersions to a subset of objects.

get_fiberfraction(*targets*, *south=True*, *ref_seeing=1.0*, *ref_lambda=5500.0*)
Estimate the fraction of the integrated flux that enters the fiber.

Assume a reference seeing value (seeingref) of 1.0 arcsec FWHM at a reference wavelength (lambdaref) of 5500 Angstrom.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **south** (bool) – True for sources with DECaLS photometry and False for sources with BASS+MzLS photometry.
- **ref_seeing** (float) – Reference seeing FWHM in arcsec. Defaults to 1.0.
- **ref_lambda** (float) – Reference wavelength in Angstrom. Defaults to 5500 A.

Returns

- **fiberfraction_g** (numpy.ndarray) – Fraction of the total g-band flux entering the fiber.
- **fiberfraction_r** (numpy.ndarray) – Fraction of the total r-band flux entering the fiber.
- **fiberfraction_z** (numpy.ndarray) – Fraction of the total z-band flux entering the fiber.

Raises ValueError – If fiberfraction is outside the bounds [0-1] (inclusive).

imaging_depth(*data*)
Add the imaging depth to the data dictionary.

Note: In future, this should be a much more sophisticated model based on the actual imaging data releases (e.g., it should depend on healpixel).

Parameters **data** (dict) – Input dictionary of sources with RA, Dec coordinates, modified on output to contain the PSF and galaxy depth in various bands.

is_south(*dec*)
Divide the "north" and "south" photometric systems based on a constant-declination cut.

Parameters **dec** (numpy.ndarray) – Declination of candidate targets (decimal degrees).

mock_density(*mockfile=None*, *nside=64*, *density_per_pixel=False*, *zmax_qso=None*, *zmin_lya=None*)
Compute the median density of targets in the full mock.

Parameters

- **mockfile** (str) – Full path to the mock catalog.
- **nside** (int) – Healpixel nside for the calculation.
- **density_per_pixel** (bool, optional) – Return the density per healpixel rather than just the median density, which may be useful for statistical purposes.
- **zmax_qso** (float) – Maximum redshift of tracer QSOs to read, to ensure no double-counting with Lya mocks. Defaults to None.
- **zmin_lya** (float) – Minimum redshift of Lya skewers, to ensure no double-counting with QSO mocks. Defaults to None.

Returns **mock_density** – Median density of targets per deg2 or target density in all healpixels (if density_per_pixel=True).

Return type int or numpy.ndarray

Raises ValueError – If mockfile is not defined.

mw_dust_extinction(*Rv=3.1*)
Cache the spectroscopic Galactic extinction curve for later use.

Parameters **Rv** (float) – Total-to-selective extinction factor. Defaults to 3.1.

mw_transmission(*data*)
Compute the grzW1W2 Galactic transmission for every object.

Parameters **data** (dict) – Input dictionary of sources with RA, Dec coordinates, modified on output to contain reddening and the MW transmission in various bands.

populate_targets_truth(*flux*, *data*, *meta*, *objmeta*, *indx=None*, *seed=None*, *use_simqso=True*, *truespectype=''*, *templatetype=''*, *templatesubtype=''*)
Initialize and populate the targets and truth tables given a dictionary of source properties and a spectral metadata table.

Parameters

- **data** (dict) – Dictionary of source properties.
- **meta** (astropy.table.Table) – Spectral metadata table.
- **indx** (numpy.ndarray, optional) – Populate the tables of a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **use_simqso** (bool, optional) – Initialize a SIMQSO-style objtruth table. Defaults to True.
- **truespectype** (str or numpy.array, optional) – True spectral type. Defaults to ''.
- **templatetype** (str or numpy.array, optional) – True template type. Defaults to ''.
- **templatesubtype** (str or numpy.array, optional) – True template subtype. Defaults to ''.

Returns

- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **objtruth** (astropy.table.Table) – Corresponding objtype-specific truth table (if applicable).

qamock_sky(*data*, *xlim=(0, 4)*, *nozhist=False*, *png=None*)
Generate a QAplot showing the sky and redshift distribution of the objects in the mock.

Parameters **data** (dict) – Dictionary of source properties.

read_GMM(*target=None*)
Read the GMM for the full range of morphological types of a given target type, as well as the magnitude-dependent morphological fraction.

See desitarget/doc/nb/gmm-dr7.ipynb for details.

remove_north_south_bits(*desi_target*, *bgs_target*, *mws_target*)
Remove all the "north" and "south" targeting bits. See the discussion here for details: https://github.com/desihub/desitarget/pull/426

Parameters

- **desi_target** (int64) – Dark-time targeting bit from targetmask.yaml.
- **bgs_target** (int64) – BGS targeting bit from targetmask.yaml.
- **mws_target** (int64) – MWS targeting bit from targetmask.yaml.

sample_GMM(*nobj*, *isouth=None*, *target=None*, *seed=None*, *morph=None*, *prior_mag=None*, *prior_redshift=None*)
Sample from the GMMs read by self.read_GMM.

See desitarget/doc/nb/gmm-dr7.ipynb for details.

scatter_photometry(*data*, *truth*, *targets*, *indx=None*, *seed=None*, *qaplot=False*)
Add noise to the input (noiseless) photometry based on the depth (as well as the inverse variance fluxes in GRZW1W2).

The input targets table is modified in place.

Parameters

- **data** (dict) – Dictionary of source properties.
- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **indx** (numpy.ndarray, optional) – Scatter the photometry of a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **qaplot** (bool, optional) – Generate a QA plot for debugging.

class desitarget.mock.mockmaker.**WDMaker**(*seed=None*, *calib_only=False*, *no_spectra=False*, *survey='main'*, ***kwargs*)
Read WD mocks, generate spectra, and select targets.

Parameters

- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not initialize template photometry. Defaults to False.
- **calib_only** (bool, optional) – Use WDs as calibration (standard star) targets, only. Defaults to False.
- **survey** (str, optional) – Specify which target masks yaml file to use. The options are *main* (main survey) and *sv1* (first iteration of SV). Defaults to *main*.

make_spectra(*data=None*, *indx=None*, *seed=None*, *no_spectra=False*)
Generate WD spectra, dealing with DA vs DB white dwarfs separately.

Parameters

- **data** (dict) – Dictionary of source properties.
- **indx** (numpy.ndarray, optional) – Generate spectra for a subset of the objects in the data dictionary, as specified using their zero-indexed indices.
- **seed** (int, optional) – Seed for reproducibility and random number generation.
- **no_spectra** (bool, optional) – Do not generate spectra. Defaults to False.

Returns

- **flux** (numpy.ndarray) – Target spectra.
- **wave** (numpy.ndarray) – Corresponding wavelength array.
- **meta** (astropy.table.Table) – Spectral metadata table.
- **targets** (astropy.table.Table) – Target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.

read(*mockfile=None*, *mockformat='mws_wd'*, *healpixels=None*, *nside=None*, *mock_density=False*, ***kwargs*)
Read the catalog.

Parameters

- **mockfile** (str) – Full path to the mock catalog to read.
- **mockformat** (str) – Mock catalog format. Defaults to 'mws_wd'.
- **healpixels** (int) – Healpixel number to read.
- **nside** (int) – Healpixel nside corresponding to healpixels.
- **mock_density** (bool, optional) – Compute the median target density in the mock. Defaults to False.

Returns Dictionary of target properties with various keys (to be documented).

Return type dict

Raises ValueError – If mockformat is not recognized.

select_targets(*targets*, *truth*, *targetname='WD'*)
Select MWS_WD targets and STD_WD standard stars. Input tables are modified in place.

Parameters

- **targets** (astropy.table.Table) – Input target catalog.
- **truth** (astropy.table.Table) – Corresponding truth table.
- **targetname** (str) – Target selection cuts to apply.

wd_template_photometry(*data=None*, *indx=None*, *rand=None*, *subtype='DA'*, *south=True*)
Get stellar photometry from the templates themselves, by-passing the generation of spectra.

desitarget.mock.mockmaker.**_default_wave**(*wavemin=None*, *wavemax=None*, *dw=0.2*)
Generate a default wavelength vector for the output spectra.

desitarget.mock.mockmaker.**empty_targets_table**(*nobj=1*)
Initialize an empty 'targets' table.

Parameters **nobj** (int) – Number of objects.

Returns **targets** – Targets table.

Return type astropy.table.Table

desitarget.mock.mockmaker.**empty_truth_table**(*nobj=1*, *templatetype=''*, *use_simqso=True*)
Initialize an empty 'truth' table.

Parameters

- **nobj** (int) – Number of objects.
- **use_simqso** (bool, optional) – Initialize a SIMQSO-style objtruth table. Defaults to True.

Returns

- **truth** (astropy.table.Table) – Truth table.
- **objtruth** (astropy.table.Table) – Objtype-specific truth table (if applicable).

desitarget.mock.sky.**random_sky**(*nside=2048*, *allsky=True*, *tiles=None*, *maxiter=20*, *outfile=None*)
Returns sky locations within healpixels covering tiles

Options: nside (int): healpixel nside; coverage is uniform at this scale allsky (bool): generate sky positions over the full sky tiles: DESI tiles to cover, for desimodel.footprint.tiles2pix(); only used if allsky=False. maxiter (int): maximum number of iterations to ensure coverage

Generates sky locations that are more uniform than true random, such that every healpixel has a point within it. Note that this should *not* be used for mock randoms.

nside=2048 corresponds to about half of a DESI positioner patrol area and results in ~18M sky locations over the full footprint.

1.16 desitarget.mtl

Merged target lists.

desitarget.mtl.**make_mtl**(*targets*, *obscon*, *zcat=None*, *trim=False*, *scnd=None*)
Adds NUMOBS, PRIORITY, and OBSCONDITIONS columns to a targets table.

Parameters

- **targets** (array or *~astropy.table.Table*) – A numpy rec array or astropy Table with at least the columns TARGETID, DESI_TARGET, NUMOBS_INIT, PRIORITY_INIT. or the corresponding columns for SV or commissioning.
- **obscon** (str) – A combination of strings that are in the desitarget bitmask yaml file (specifically in *desitarget.targetmask.obsconditions*), e.g. "DARK|GRAY". Governs the behavior of how priorities are set based on "obsconditions" in the desitarget bitmask yaml file.
- **zcat** (Table, optional) – Redshift catalog table with columns TARGETID, NUMOBS, Z, ZWARN.
- **trim** (bool, optional) – If True (default), don't include targets that don't need any more observations. If False, include every input target.
- **scnd** (array, *~astropy.table.Table*, optional) – A set of secondary targets associated with the *targets*. As with the *target* must include at least TARGETID, NUMOBS_INIT,

PRIORITY_INIT or the corresponding SV columns. The secondary targets will be padded to have the same columns as the targets, and concatenated with them.

Returns

MTL Table with targets columns plus:

- NUMOBS_MORE - number of additional observations requested
- PRIORITY - target priority (larger number = higher priority)
- OBSCONDITIONS - replaces old GRAYLAYER

Return type Table

1.17 desitarget.myRF

This module computes the Random Forest probability and it stores the RF with our own persistency.

class desitarget.myRF.**myRF**(*data*, *modelDir*, *numberOfTrees=200*, *version=2*)
Class for I/O operations and probability calculation for Random Forest

1.18 desitarget.photo

Implements the photometric transforms between SDSS and DECam using g,r,z documented in DESI-1788v1 https://desi.lbl.gov/DocDB/cgi-bin/private/ShowDocument?docid=1788

desitarget.photo.**cfht2decam**(*g_cfht*, *r_cfht*, *i_cfht*, *z_cfht*)
Converts CFHT magnitudes to DECam magnitudes

Parameters **[griz]_cfht** – CFHT magnitudes (float or arrays of floats)

Returns g_decam, r_decam, z_decam

Note: CFHT griz are inputs, but only grz (no i) are output

desitarget.photo.**decam2cfht**(*g_decam*, *r_decam*, *z_decam*)
Not yet implemented

desitarget.photo.**decam2sdss**(*g_decam*, *r_decam*, *z_decam*)
Not yet implemented

desitarget.photo.**sdss2decam**(*g_sdss*, *r_sdss*, *i_sdss*, *z_sdss*)
Converts SDSS magnitudes to DECam magnitudes

Parameters **[griz]_sdss** – SDSS magnitudes (float or arrays of floats)

Returns g_decam, r_decam, z_decam

Note: SDSS griz are inputs, but only grz (no i) are output

1.19 desitarget.QA

Module dealing with Quality Assurance tests for Target Selection

desitarget.QA.**_in_desi_footprint**(*targs*, *radec=False*)
Convenience function for using is_point_in_desi to find which targets are in the footprint.

Parameters

- **targs** (array or *str*) – Targets in the DESI data model format, or any array that contains RA and DEC columns.
- **radec** (bool, optional, defaults to False) – If True then the passed *objs* is an [RA, Dec] list instead of a rec array.

Returns The INDICES of the input targs that are in the DESI footprint.

Return type integer

desitarget.QA.**_javastring**()
Return a string that embeds a date in a webpage

desitarget.QA.**_load_dndz**(*tcnames=None*)
Load the predicted redshift distributions for each target class.

Parameters **tcnames** (list) – A list of strings, e.g. "['QSO','LRG','ALL'] If passed, return only a dictionary for those specific bits.

Returns A dictionary where the keys are the bit names and the values are the dndz as a function of redshift (as another dictionary with keys 'z', and 'dndz', respectively.

Return type dictionary

desitarget.QA.**_load_systematics**()
Loads information for making systematics plots.

Returns A dictionary where the keys are the names of the systematics and the values are arrays of where to clip these systematics in plots

Return type dictionary

desitarget.QA.**_load_targdens**(*tcnames=None*, *bit_mask=None*)
Loads the target info dictionary as in desimodel.io.load_target_info() and extracts the target density information in a format useful for targeting QA plots.

Parameters

- **tcnames** (list) – A list of strings, e.g. "['QSO','LRG','ALL'] If passed, return only a dictionary for those specific bits.
- **bit_mask** (list or *~numpy.array*, optional, defaults to None) – If passed, load the bit names from this mask (with no associated expected densities) rather than loading the main survey bits and densities. Must be a desi mask object, e.g., loaded as *from desitarget.targetmask import desi_mask*. Any bit names that contain "NORTH" or "SOUTH" or calibration bits will be removed. A list of several masks can be passed rather than a single mask.

Returns A dictionary where the keys are the bit names and the values are the densities.

Return type dictionary

Notes

If *bit_mask* happens to correpond to the main survey masks, then the default behavior is triggered (as if *bit_mask=None*).

desitarget.QA.**_parse_tcnames**(*tcstring=None*, *add_all=True*)
Turn a comma-separated string of target class names into a list.

Parameters

- **tcstring** (str, optional, defaults to *"ELG,QSO,LRG,MWS,BGS,STD,(ALL)"*) – Comma-separated names of target classes e.g. QSO,LRG. Options are *ELG*, *QSO*, *LRG*, *MWS*, *BGS*, *STD*.
- **add_all** (boolean, optional, defaults to True) – If True, then include *ALL* in the default names.

Returns The string of names is converted to a list.

Return type list

Notes

- One use of this function is to check for valid target class strings. An IOError is raised if a string is invalid.

desitarget.QA.**_prepare_systematics**(*data*, *colname*)

Functionally convert systematics to more user-friendly numbers.

:param data array: An array of the systematic. :param colname: The column name of the passed systematic, e.g. STARDENS. :type colname: str

Returns The systematics converted by the appropriate function

Return type array

desitarget.QA.**make_qa_page**(*targs*, *mocks=False*, *makeplots=True*, *max_bin_area=1.0*, *qadir='.'*, *clip2foot=False*, *weight=True*, *imaging_map_file=None*, *tcnames=None*, *systematics=True*, *numproc=8*, *downsample=None*)

Make a directory containing a webpage in which to embed QA plots.

Parameters

- **targs** (array or *str*) – An array of targets in the DESI data model format. If a string is passed then the targets are read from the file with the passed name (supply the full directory path). The string can also be a directory of HEALPixel-split target files which will be read in using *desitarget.io.read_targets_in_box()*.
- **mocks** (boolean, optional, default=False) – If True, add plots only relevant to mocks to the webpage.
- **makeplots** (boolean, optional, default=True) – If True, then create the plots as well as the webpage.
- **max_bin_area** (float, optional, defaults to 1 degree) – Bin size in RA/Dec is set as close as possible to this value.
- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **clip2foot** (boolean, optional, defaults to False) – Use desimodel.footprint.is_point_in_desi to restrict *targs* to the DESI spectroscopic footprint.
- **weight** (boolean, optional, defaults to True) – If set, weight pixels to offset underdense pixels at footprint edges. Uses the *imaging_map_file* HEALPix file for real targets and the DESIMODEL HEALPix footprint file for mock targets.
- **imaging_map_file** (str, optional, defaults to no weights) – If *weight* is set, then this is the location of the imaging HEALPix map (e.g. made by desitarget.randoms.pixmap()). Defaults to 1 everywhere (i.e. no weights) for the real targets. If this is not set, then systematics plots cannot be made.

- **tcnames** (list) – String-list, e.g. ['QSO','LRG','ALL'] If passed, return only QA pages for those specific bits. A useful speed-up when testing.
- **systematics** (boolean, optional, defaults to True) – If sent, then add plots of systematics to the front page.
- **numproc** (int, optional, defaults to 8) – The number of parallel processes to use to generate plots.
- **downsample** (int, optional, defaults to *None*) – If not *None*, downsample targets by (roughly) this value, e.g. for *downsample=10* a set of 900 targets would have ~90 random targets returned. A speed-up for experimenting with large files.

Returns But the page *index.html* and associated pages and plots are written to qadir.

Return type Nothing

Notes

If making plots, then the DESIMODEL environment variable must be set to find the file of HEALPixels that overlap the DESI footprint.

desitarget.QA.**make_qa_plots**(*targs*, *qadir='.'*, *targdens=None*, *max_bin_area=1.0*, *weight=True*, *imaging_map_file=None*, *truths=None*, *objtruths=None*, *tcnames=None*, *cmx=False*, *bit_mask=None*, *mocks=False*, *numproc=8*)

Make DESI targeting QA plots given a passed set of targets.

Parameters

- **targs** (array or *str*) – Array of targets in the DESI data model format. If a string is passed then the targets are read from the file with the passed name (supply the full directory path).
- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **targdens** (dictionary, optional) – A dictionary of DESI target classes and the goal density for that class. Used to label the goal density on histogram plots.
- **max_bin_area** (float, optional, defaults to 1 degree) – The bin size for sky maps in RA/Dec correspond to this value.
- **weight** (boolean, optional, defaults to True) – If set, weight pixels using the DESIMODEL HEALPix footprint file to offset under-dense pixels at the footprint edges.
- **imaging_map_file** (str, optional, defaults to no weights) – If *weight* is set, this file is the location of the imaging HEALPixel map (e.g. made by :func:' desitarget.randoms.pixmap()'. If not sent, then weights default to 1 (i.e. no weighting).
- **truths** (array or *str*) – The truth objects from which the targs were derived in the DESI data model format. If a string is passed then read from that file (supply the full directory path).
- **objtruths** (dict) – Object type-specific truth metadata.
- **tcnames** (list, defaults to None) – Strings, e.g. ['QSO','LRG','ALL'] If passed, return only the QA pages for those specific bits. A useful speed-up when testing.
- **cmx** (boolean, defaults to False) – Pass as True to use commissioning bits instead of SV or main survey bits. Commissioning files have no MWS or BGS columns.

- **bit_mask** (list or *~numpy.array*, optional) – Load bit names from this passed mask or list of masks instead of from the (default) main survey mask.
- **mocks** (boolean, optional, default=False) – If True, also make plots that are only relevant to mocks.
- **numproc** (int, optional, defaults to 8) – The number of parallel processes to use to generate plots.

Returns

- float – The total area of the survey used to make the QA plots.
- dict – A nested dictionary of each of the bit-names. Each bit-key has a dictionary of the 10 densest pixels in the DESI tiling. Includes RA, DEC, DENSITY (per sq. deg.) and NSIDE for each HEALpixel.

Notes

- The DESIMODEL environment variable must be set to find the default expected target densities.
- When run, a set of targeting .png plots are written to *qadir*.

desitarget.QA.**mock_qafractype**(*cat*, *objtype*, *qadir='.'*, *fileprefix='mock-fractype'*)

Targeting QA Bar plot of the fraction of each classification type assigned to (mock) targets.

Parameters

- **cat** (array) – An array of targets that contains at least TRUESPECTYPE.
- **objtype** (str) – The name of a DESI target class (e.g., "ELG") that corresponds to the passed cat.
- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **fileprefix** (str, optional, defaults to "mock-fractype" for) – String to be added to the front of the output file name.

Returns But .png plots of target colors are written to qadir. The file is called: {qadir}/{fileprefix}-{objtype}.png.

Return type Nothing

desitarget.QA.**mock_qanz**(*cat*, *objtype*, *qadir='.'*, *area=1.0*, *dndz=None*, *nobjscut=1000*, *fileprefixz='mock-nz'*, *fileprefixzmag='mock-zvmag'*)

Make N(z) and z vs. mag DESI QA plots given a passed set of MOCK TRUTH.

Parameters

- **cat** (array) – An array of targets that contains at least TRUEZ for redshift information and MAG for magnitude information.
- **objtype** (str) – The name of a DESI target class (e.g., "ELG") that corresponds to the passed cat.
- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **area** (float) – Total area in deg2.
- **dndz** (dict) – Dictionary output of *_load_dndz*

- **nobjscut** (int, optional, defaults to 1000) – Make a hexbin plot when the number of objects is greater than nobjscut, otherwise make a scatterplot.
- **fileprefixz** (str, optional, defaults to "color" for) – String to be added to the front of the output N(z) plot file name.
- **fileprefixzmag** (str, optional, defaults to "color" for) – String to be added to the front of the output z vs. mag plot file name.

Returns

But .png plots of target colors are written to qadir. Two plots are made:

The file containing N(z) is called: {qadir}/{fileprefixz}-{objtype}.png.

The file containing z vs. zerr is called: {qadir}/{fileprefixzmag}-{objtype}.png.

Return type Nothing

desitarget.QA.**qacolor**(*cat*, *objtype*, *extinction*, *qadir='.'*, *fileprefix='color'*, *nodustcorr=False*, *mocks=False*, *nobjscut=1000*, *seed=None*)

Make color-based DESI targeting QA plots given a passed set of targets.

Parameters

- **cat** (array) – An array of targets that contains at least FLUX_G, FLUX_R, FLUX_Z and FLUX_W1, FLUX_W2 columns for color information.
- **objtype** (str) – The name of a DESI target class (e.g., "ELG") that corresponds to the passed cat.
- **extinction** (array) – An array containing the extinction in each band of interest, must contain at least the columns MW_TRANSMISSION_G, _R, _Z, _W1, _W2.
- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **fileprefix** (str, optional, defaults to "color") – String to be added to the front of the output file name.
- **nodustcorr** (boolean, optional, defaults to False) – Do not correct for dust extinction.
- **mocks** (boolean, optional, default=False) – If True, input catalog is a "truths" catalog.
- **nobjscut** (int, optional, defaults to 1000) – Make a hexbin plot when the number of objects is greater than nobjscut, otherwise make a scatterplot.
- **seed** (int, optional) – Seed to reproduce random points plotted on hexbin plots.

Returns But .png plots of target colors are written to qadir. The file is called: {qadir}/{fileprefix}-{bands}-{objtype}.png where bands might be, e.g., grz.

Return type Nothing

desitarget.QA.**qagaia**(*cat*, *objtype*, *qadir='.'*, *fileprefix='gaia'*, *nobjscut=1000*, *seed=None*)

Make Gaia-based DESI targeting QA plots given a passed set of targets.

Parameters

- **cat** (array) – An array of targets that contains at least "RA", "PARALLAX", "PMRA" and "PMDEC".
- **objtype** (str) – The name of a DESI target class (e.g., "ELG") that corresponds to the passed cat.

- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **fileprefix** (str, optional, defaults to "gaia") – String to be added to the front of the output file name.
- **nobjscut** (int, optional, defaults to 1000) – Make a hexbin plot when the number of objects is greater than nobjscut, otherwise make a scatterplot.
- **seed** (int, optional) – Seed to reproduce random points plotted on hexbin plots.

Returns

But .png plots of Gaia information are written to qadir. Two plots are made:

The file containing distances from parallax is called: {qadir}/{fileprefix}-{parallax}-{objtype}.png.

The file containing proper motion information is called: {qadir}/{fileprefix}-{pm}-{objtype}.png.

Return type Nothing

desitarget.QA.**qahisto**(*cat*, *objtype*, *qadir='.'*, *targdens=None*, *upclip=None*, *weights=None*, *max_bin_area=1.0*, *fileprefix='histo'*, *catispix=False*)

Visualize the target density with a histogram of densities. First version taken shamelessly from desitarget.mock.QA (which was originally written by *J. Moustakas*).

Parameters

- **cat** (array) – An array of targets that contains at least RA and DEC columns for coordinate information.
- **objtype** (str) – The name of a DESI target class (e.g., "ELG") that corresponds to the passed cat.
- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **targdens** (dictionary, optional, defaults to None) – A dictionary of DESI target classes and the goal density for that class. Used, if passed, to label the goal density on the histogram plot.
- **upclip** (float, optional, defaults to None) – A cutoff at which to clip the targets at the "high density" end to make plots conform to similar density scales.
- **weights** (array, optional, defaults to None) – A weight for each of the passed targets (e.g., to upweight each target in a partial pixel at the edge of the DESI footprint).
- **max_bin_area** (float, optional, defaults to 1 degree) – The bin size in RA/Dec in *targs* is chosen to be as close as possible to this value.
- **fileprefix** (str, optional, defaults to "histo") – String to be added to the front of the output file name.
- **catispix** (boolean, optional, defaults to False) – If this is True, then cat corresponds to the HEALpixel numbers already precomputed using pixels = footprint.radec2pix(nside, cat["RA"], cat["DEC"]) from the RAs and Decs ordered as for weights, rather than the catalog itself. If this is True, then max_bin_area must correspond to the *nside* used to precompute the pixel numbers.

Returns But a .png histogram of target densities is written to qadir. The file is called: {qadir}/{fileprefix}-{objtype}.png.

Return type Nothing

desitarget.QA.**qamag**(*cat*, *objtype*, *qadir='.'*, *fileprefix='nmag'*, *area=1.0*)
Make magnitude-based DESI targeting QA plots given a passed set of targets.

Parameters

- **cat** (array) – An array of targets that contains at least FLUX_G, FLUX_R, FLUX_Z and FLUX_W1, columns for magnitude information.
- **objtype** (str) – The name of a DESI target class (e.g., "ELG") that corresponds to the passed cat.
- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **fileprefix** (str, optional, defaults to "nmag" for) – String to be added to the front of the output file name.
- **area** (float) – Total area in deg2.

Returns But .png plots of target colors are written to qadir. The file is called: {qadir}/{fileprefix}-{filter}-{objtype}.png where filter might be, e.g., g. ASCII versions of those files are also written with columns of magnitude bin and target number density. The file is called {qadir}/{fileprefix}-{filter}-{objtype}.dat.

Return type Nothing

desitarget.QA.**qaskymap**(*cat*, *objtype*, *qadir='.'*, *upclip=None*, *weights=None*, *max_bin_area=1.0*, *fileprefix='skymap'*)
Visualize the target density with a skymap. First version lifted shamelessly from desitarget.mock.QA (which was originally written by *J. Moustakas*).

Parameters

- **cat** (array) – An array of targets that contains at least RA and DEC columns for coordinate information.
- **objtype** (str) – The name of a DESI target class (e.g., "ELG") that corresponds to the passed cat.
- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **upclip** (float, optional, defaults to None) – A cutoff at which to clip the targets at the "high density" end to make plots conform to similar density scales.
- **weights** (array, optional, defaults to None) – A weight for each of the passed targets (e.g., to upweight each target in a partial pixel at the edge of the DESI footprint).
- **max_bin_area** (float, optional, defaults to 1 degree) – The bin size in RA/Dec in *targs* is chosen to be as close as possible to this value.
- **fileprefix** (str, optional, defaults to "radec" for (RA/Dec)) – String to be added to the front of the output file name.

Returns Dictionary of the 10 densest pixels in the DESI tiling. Includes RA, DEC, DENSITY (per sq. deg.) and NSIDE for each HEALpixel.

Return type dict

Notes

In addition to the returned dictionary, a .png sky map is written to *qadir*. The file is called: *{qadir}/{fileprefix}-{objtype}.png*.

desitarget.QA.**qasystematics_scatterplot**(*pixmap*, *syscolname*, *targcolname*, *qadir='.'*, *downclip=None*, *upclip=None*, *nbins=10*, *fileprefix='sysdens'*, *xlabel=None*)

Make a target density vs. systematic scatter plot.

Parameters

- **pixmap** (array) – An array of systematics binned in HEALPixels, made by, e.g. *make_imaging_weight_map*.
- **syscolname** (str) – The name of the passed systematic, e.g. STARDENS.
- **targcolname** (str) – The name of the passed column of target densities, e.g. QSO.
- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **downclip** (float, optional, defaults to None) – A cutoff at which to clip the systematics at the low end.
- **upclip** (float, optional, defaults to None) – A cutoff at which to clip the systematics at the high end.
- **nbins** (int, optional, defaults to 10) – The number of bins to produce in the scatter plot.
- **fileprefix** (str, optional, defaults to "histo") – String to be added to the front of the output file name.
- **xlabel** (str, optional, if None defaults to syscolname) – An informative title for the x-axis of the plot.

Returns But a .png histogram of target densities is written to qadir. The file is called: {qadir}/{fileprefix}-{syscolname}-{targcolname}.png.

Return type Nothing

Notes

The passed pixmap must contain a column FRACAREA which is used to filter out any pixel with less than 90% areal coverage.

desitarget.QA.**qasystematics_skyplot**(*pixmap*, *colname*, *qadir='.'*, *downclip=None*, *upclip=None*, *fileprefix='systematics'*, *plottitle=''*)

Visualize systematics with a sky map.

Parameters

- **pixmap** (array) – An array of systematics binned in HEALPixels, made by, e.g. *make_imaging_weight_map*. Assumed to be in the NESTED scheme and ORDERED BY INCREASING HEALPixel.
- **colname** (str) – The name of the passed systematic, e.g. STARDENS.
- **qadir** (str, optional, defaults to the current directory) – The output directory to which to write produced plots.
- **downclip** (float, optional, defaults to None) – A cutoff at which to clip the systematics at the low end.
- **upclip** (float, optional, defaults to None) – A cutoff at which to clip the systematics at the high end.
- **fileprefix** (str, optional, defaults to "histo") – String to be added to the front of the output file name.

- **plottitle** (str, optional, defaults to empty string) – An informative title for the plot.

Returns But a .png histogram of target densities is written to `qadir`. The file is called: `{qadir}/{fileprefix}-{colname}.png`.

Return type Nothing

desitarget.QA.**read_data**(*targfile*, *mocks=False*, *downsample=None*, *header=False*)
Read in the data, including any mock data (if present).

Parameters

- **targfile** (str) – The full path to a mock target file in the DESI X per cent survey directory structure, e.g., /global/projecta/projectdirs/desi/datachallenge/dc17b/targets/, or to a data file, or to a directory of HEALPixel-split target files which will be read in with *desitarget.io.read_targets_in_box()*.
- **mocks** (boolean, optional, defaults to `False`) – If `True`, read in mock data.
- **downsample** (int, optional, defaults to *None*) – If not *None*, downsample targets by (roughly) this value, e.g. for *downsample=10* a set of 900 targets would have ~90 random targets returned. A speed-up for experimenting with large files.
- **header** (bool, optional, defaults to `False`) – If `True` then return the header of the file as an additional output (targs, truths, objtruths, header) instead of (targs, truths, objtruths).

Returns

- **targs** (array) – A rec array containing the targets catalog.
- **truths** (array) – A rec array containing the truths catalog (if present and *mocks=True*).
- **objtruths** (dict) – Object type-specific truth metadata (if present and *mocks=True*).

1.20 desitarget.skyfibers

Module to assign sky fibers at the pixel-level for target selection

desitarget.skyfibers.**density_of_sky_fibers**(*margin=1.5*)
Use positioner patrol size to determine sky fiber density for DESI.

Parameters **margin** (float, optional, defaults to 1.5) – Factor of extra sky positions to generate. So, for margin=10, 10x as many sky positions as the default requirements will be generated.

Returns The density of sky fibers to generate in per sq. deg.

Return type float

desitarget.skyfibers.**get_brick_info**(*drdirs*, *counts=False*, *allbricks=False*)
Retrieve brick names and coordinates from Legacy Surveys directories.

Parameters

- **drdirs** (list or *str*) – A list of strings, each of which corresponds to a directory pointing to a Data Release from the Legacy Surveys. Can be of length one. e.g. ['/global/project/projectdirs/cosmo/data/legacysurvey/dr7']. or '/global/project/projectdirs/cosmo/data/legacysurvey/dr7' Can be None if *allbricks* is passed.
- **counts** (bool, optional, defaults to `False`) – If `True` also return a count of the number of times each brick appears ([RAcen, DECcen, RAmin, RAmax, DECmin, DECmax, CNT]).

- **allbricks** (bool, optional, defaults to False) – If True ignore *drdirs* and simply return a dictionary of ALL of the bricks.

Returns UNIQUE bricks covered by the Data Release(s). Keys are brick names and values are a list of the brick center and the brick corners ([RAcen, DECcen, RAmin, RAmax, DECmin, DECmax]).

Return type dict

Notes

- Tries a few different ways in case the survey bricks files have not yet been created.

desitarget.skyfibers.**get_supp_skies**(*ras*, *decs*, *radius=2.0*)

Random locations, avoid Gaia, format, return supplemental skies.

Parameters

- **ras** (ndarray) – Right Ascensions of sky locations (degrees).
- **decs** (ndarray) – Declinations of sky locations (degrees).
- **radius** (float, optional, defaults to 2) – Radius at which to avoid (all) Gaia sources (arcseconds).

Returns A structured array of supplemental sky positions in the DESI sky target format that avoid Gaia sources by *radius*.

Return type ndarray

Notes

- Written to be used when *ras* and *decs* are within a single Gaia-file HEALPixel, but should work for all cases.

desitarget.skyfibers.**make_skies_for_a_brick**(*survey, brickname, nskiespersqdeg=None, bands=['g', 'r', 'z'], apertures_arcsec=[0.75], write=False*)

Generate skies for one brick in the typical format for DESI sky targets.

Parameters

- **survey** (object) – *LegacySurveyData* object for a given Data Release of the Legacy Surveys; see LegacySurveyData() for details.
- **brickname** (str) – Name of the brick in which to generate sky locations.
- **nskiespersqdeg** (float, optional) – The minimum DENSITY of sky fibers to generate. Defaults to reading from io() with a margin of 4x.
- **bands** (list, optional, defaults to ['g', 'r', 'z']) – List of bands to be used to define good sky locations.
- **apertures_arcsec** (list, optional, defaults to [0.75]) – Radii in arcsec of apertures for which to derive flux at a sky location.
- **write** (boolean, defaults to False) – If *True*, write the skyfibers object (which is in the format of the output from *sky_fibers_for_brick()*) to file. The file name is derived from the input *survey* object and is in the form:

%(survey.survey_dir)/metrics/%(brick).3s/skies-%(brick)s.fits.gz which is returned by *survey.find_file('skies')*.

Returns a structured array of sky positions in the DESI sky target format for a brick.

Return type ndarray

Notes

- The code generates unique OBJIDs based on an integer counter for the numbers of objects (objs) passed. So, it will fail if the length of objs is longer than the number of bits reserved for OBJID in *desitarget.targetmask*.
- The generated sky fiber locations will cover the pixel-based brick grid, which extends beyond the "true" geometric brick boundaries.

desitarget.skyfibers.**model_density_of_sky_fibers**(*margin=1.5*)

Use desihub products to find required density of sky fibers for DESI.

Parameters **margin** (float, optional, defaults to 1.5) – Factor of extra sky positions to generate. So, for margin=10, 10x as many sky positions as the default requirements will be generated.

Returns The density of sky fibers to generate in per sq. deg.

Return type float

desitarget.skyfibers.**plot_good_bad_skies**(*survey, brickname, skies, outplotdir='.', bands=['g', 'r', 'z']*)

Plot good/bad sky locations against the background of a Legacy Surveys image

Parameters

- **survey** (object) – *LegacySurveyData* object for a given Data Release of the Legacy Surveys; see LegacySurveyData() for details.
- **brickname** (str) – Name of the brick from this DR of the Legacy Surveys to plot as an image.
- **skies** (ndarray) – Array of sky locations and aperture fluxes, as, e.g., returned by *make_skies_for_a_brick()* or *select_skies()*
- **outplotdir** (str, optional, defaults to '.') – Output directory name to which to save the plot, passed to matplotlib's savefig routine. The actual plot is name outplotdir/skies-brickname-bands.png
- **bands** (list, optional, defaults to ['g', 'r', 'z']) – List of bands to plot in the image (i.e. default is to plot a 3-color grz composite). This is particularly useful when the code fails because a Legacy Surveys image-BAND.fits file is not found, in which case that particular band can be redacted from the bands list.

Returns

- *Nothing, but a plot of the Legacy Surveys image for the Data Release corresponding*
- to the *survey* object and the brick corresponding to *brickname* is written to
- *outplotname*. The plot contains the Legacy Surveys imaging with good sky locations
- *plotted in green and bad sky locations in red.*

Notes

- The array *skies* must contain at least the columns 'BRICKNAME', 'RA', 'DEC', and 'DESI_TARGET', but can contain multiple different values of 'BRICKNAME', provided that one of them corresponds to the passed *brickname*.
- If the passed *survey* object doesn't correspond to the Data Release from which the passed *skies* array was derived, then the sky locations could be plotted at slightly incorrect positions. If the *skies* array was read from file, this can be checked by making survey that "DEPVER02" in the file header corresponds to the directory *survey.survey_dir*.

desitarget.skyfibers.**repartition_skies**(*skydirname*, *numproc=1*)

Rewrite a skies directory so each file actually only contains sky locations in the HEALPixels that are listed in the file header.

Parameters

- **skydirname** (str) – Full path to a directory containing files of skies that have been partitioned by HEALPixel (i.e. as made by *select_skies* with the *bundle_files* option).
- **numproc** (int, optional, defaults to 1) – The number of processes over which to parallelize writing files.

Returns

- *Nothing, but rewrites the input directory such that each file only*
- *contains the HEALPixels listed in the file header.*

Notes

- Necessary as although the targets and GFAs are parallelized to run in exact HEALPixel boundaries, skies are parallelized across bricks that have CENTERS in a given HEALPixel.
- The original files, before the rewrite, are retained in the original directory, appended by "-unpartitioned".
- Takes about 25 (6.5, 5, 3.5) minutes for numproc=1 (8, 16, 32).

desitarget.skyfibers.**select_skies**(*survey*, *numproc=16*, *nskiespersqdeg=None*, *bands=['g', 'r', 'z']*, *apertures_arcsec=[0.75]*, *nside=None*, *pixlist=None*, *writebricks=False*)

Generate skies in parallel for bricks in a Legacy Surveys DR.

Parameters

- **survey** (object) – *LegacySurveyData* object for a given Data Release of the Legacy Surveys; see LegacySurveyData() for details.
- **numproc** (int, optional, defaults to 16) – The number of processes over which to parallelize.
- **nskiespersqdeg** (float, optional) – The minimum DENSITY of sky fibers to generate. Defaults to reading from io() with a margin of 4x.
- **bands** (list, optional, defaults to ['g', 'r', 'z']) – List of bands to be used to define good sky locations.
- **apertures_arcsec** (list, optional, defaults to [0.75]) – Radii in arcsec of apertures for which to derive flux at a sky location.

- **nside** (int, optional, defaults to None) – The HEALPixel nside number to be used with the *pixlist* input.
- **pixlist** (list or *int*, optional, defaults to None) – Bricks will only be processed if the CENTER of the brick lies within the bounds of pixels that are in this list of integers, at the supplied HEALPixel *nside*. Uses the HEALPix NESTED scheme. Useful for parallelizing. If pixlist is None then all bricks in the passed *survey* will be processed.
- **writebricks** (boolean, defaults to False) – If *True*, write the skyfibers object for EACH brick (in the format of the output from *sky_fibers_for_brick()*) to file. The file name is derived from the input *survey* object and is in the form: *%(survey.survey_dir)/metrics/%(brick).3s/skies-%(brick)s.fits.gz* which is returned by *survey.find_file('skies')*.

Returns a structured array of sky positions in the DESI sky target format for all bricks in a Legacy Surveys Data Release.

Return type ndarray

Notes

- Some core code in this module was initially written by Dustin Lang (@dstndstn).

desitarget.skyfibers.**sky_fiber_locations**(*skypix*, *gridsize=300*)

The core worker function for *sky_fibers_for_brick*

Parameters

- **skypix** (array) – NxN boolean array of pixels.
- **gridsize** (int, optional, defaults to 300) – Resolution (in pixels) at which to split the *skypix* array in order to find sky locations. For example, if skypix is a 3600x3600 array of pixels, gridsize=300 will return (3600/300) x (3600/300) = 12x12 = 144 locations.

Notes

- Implements the core trick of iteratively eroding the map of good sky locations to produce a distance-from-blobs map, and then return the max values in that map in each cell of a grid.
- Initial version written by Dustin Lang (@dstndstn).

desitarget.skyfibers.**sky_fiber_plots**(*survey, brickname, skyfibers, basefn, bands=['g', 'r', 'z']*)

Make QA plots for sky locations produced by *sky_fibers_for_brick*

Parameters

- **survey** (object) – *LegacySurveyData* object for a given Data Release of the Legacy Surveys; see LegacySurveyData() for details.
- **brickname** (str) – Name of the brick from this DR of the Legacy Surveys to plot as an image.
- **skyfibers** (object) – *skyfibers* object returned by *sky_fibers_for_brick()*
- **basefn** (str) – Base name for the output plot files.
- **bands** (list, optional, defaults to ['g', 'r', 'z']) – List of bands to plot in the image (i.e. default is to plot a 3-color grz composite). This is particularly useful when a Legacy Surveys

image-BAND.fits file is not found, in which case that particular band can be redacted from the bands list.

Returns

- basefn + '-1.png' : Sky Fiber Positions on the full image
- basefn + '-2.png' : Postage stamps around each sky fiber position
- basefn + '-3.png' : Aperture flux at each radius for each sky fiber

Return type Nothing, but plots are written to

Notes

- Initial version written by Dustin Lang (@dstndstn).

desitarget.skyfibers.**sky_fibers_for_brick**(*survey, brickname, nskies=144, bands=['g', 'r', 'z'], apertures_arcsec=[0.5, 0.75, 1.0, 1.5, 2.0, 3.5, 5.0, 7.0]*)

Produce DESI sky fiber locations in a brick, derived at the pixel-level

Parameters

- **survey** (object) – *LegacySurveyData* object for a given Data Release of the Legacy Surveys; see LegacySurveyData() for details.
- **brickname** (str) – Name of the brick in which to generate sky locations.
- **nskies** (float, optional, defaults to 144 (12 x 12)) – The minimum DENSITY of sky fibers to generate
- **bands** (list, optional, defaults to ['g', 'r', 'z']) – List of bands to be used to define good sky locations.
- **apertures_arcsec** (list, optional, defaults to [0.5,0.75,1.,1.5,2.,3.5,5.,7.]) – Radii in arcsec of apertures for which to derive flux at a sky location.

Returns A FITS table that includes: - the brickid - the brickname - the x and y pixel positions of the fiber location from the blobs file - the distance from the nearest blob of this fiber location - the RA and Dec positions of the fiber location - the aperture flux and ivar at the passed *apertures_arcsec*

Return type object

Notes

- Initial version written by Dustin Lang (@dstndstn).
- The generated sky fiber locations will cover the pixel-based brick grid, which extends beyond the "true" geometric brick boundaries.

desitarget.skyfibers.**supplement_skies**(*nskiespersqdeg=None, numproc=16, gaiadir=None, nside=None, pixlist=None, mindec=-30.0, mingalb=10.0, radius=2.0*)

Generate supplemental sky locations using Gaia-G-band avoidance.

Parameters

- **nskiespersqdeg** (float, optional) – The minimum DENSITY of sky fibers to generate. Defaults to reading from io() with a margin of 4x.

- **numproc** (int, optional, defaults to 16) – The number of processes over which to parallelize.
- **gaiadir** (str, optional, defaults to $GAIA_DIR) – The GAIA_DIR environment variable is set to this directory. If None is passed, then it's assumed to already exist.
- **nside** (int, optional, defaults to *None*) – (NESTED) HEALPix *nside* to use with *pixlist*.
- **pixlist** (list or *int*, optional, defaults to *None*) – Only return targets in a set of (NESTED) HEALpixels at the supplied *nside*. Useful for parallelizing across nodes. The first entry sets RELEASE for TARGETIDs, and must be < 1000 (to prevent confusion with DR1 and above).
- **mindec** (float, optional, defaults to -30) – Minimum declination (o) to include for output sky locations.
- **mingalb** (float, optional, defaults to 10) – Closest latitude to Galactic plane for output sky locations (e.g. send 10 to limit to areas beyond -10o <= b < 10o).
- **radius** (float, optional, defaults to 2) – Radius at which to avoid (all) Gaia sources (arcseconds).

Returns a structured array of supplemental sky positions in the DESI sky target format within the passed *mindec* and *mingalb* limits.

Return type ndarray

Notes

- The environment variable $GAIA_DIR must be set, or *gaiadir* must be passed.

1.21 desitarget.sv1.sv1_cuts

Target Selection for DESI Survey Validation derived from the SV wiki.

A collection of helpful (static) methods to check whether an object's flux passes a given selection criterion (*e.g.* LRG, ELG or QSO).

desitarget.sv1.sv1_cuts.**isBACKUP**(*ra=None*, *dec=None*, *gaiagmag=None*, *primary=None*)

BACKUP targets based on Gaia magnitudes.

Parameters

- **dec** (*ra,*) – Right Ascension and Declination in degrees.
- **gaiagmag** (array_like or None) – Gaia-based g MAGNITUDE (not Galactic-extinction-corrected). (same units as the Gaia data model).
- **primary** (array_like or None) – True for objects that should be passed through the selection.

Returns

- array_like – True if and only if the object is a bright "BACKUP" target.
- array_like – True if and only if the object is a faint "BACKUP" target.
- array_like – True if and only if the object is a very faint "BACKUP" target.

Notes

- Current version (10/24/19) is version 114 on the SV wiki.

desitarget.sv1.sv1_cuts.**isBGS**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *rfiberflux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *gfracmasked=None*, *rfracmasked=None*, *zfracmasked=None*, *gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *gfracin=None*, *rfracin=None*, *zfracin=None*, *gfluxivar=None*, *rfluxivar=None*, *zfluxivar=None*, *maskbits=None*, *Grr=None*, *w1snr=None*, *gaiagmag=None*, *objtype=None*, *primary=None*, *south=True*, *targtype=None*)

Definition of BGS target classes. Returns a boolean array.

Parameters

- **south** (boolean, defaults to True) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if south=False, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).
- **targtype** (str, optional, defaults to faint) – Pass bright to use colors appropriate to the BGS_BRIGHT selection or faint to use colors appropriate to the BGS_FAINT selection or faint_ext to use colors appropriate to the BGS_FAINT_EXTENDED selection or lowq to use colors appropriate to the BGS_LOW_QUALITY selection or fibmag to use colors appropriate to the BGS_FIBER_MAGNITUDE selection.

Returns True if and only if the object is a BGS target of type targtype.

Return type array_like

Notes

- Current version (10/14/19) is version 105 on the SV wiki.
- See *set_target_bits()* for other parameters.

desitarget.sv1.sv1_cuts.**isBGS_colors**(*rflux=None*, *rfiberflux=None*, *south=True*, *targtype=None*, *primary=None*)

Standard set of masking cuts used by all BGS target selection classes (see, e.g., *isBGS()* for parameters).

desitarget.sv1.sv1_cuts.**isELG**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *gfiberflux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *maskbits=None*, *south=True*, *primary=None*)

Definition of ELG target classes. Returns a boolean array.

Parameters **south** (boolean, defaults to True) – If False, use cuts for the Northern imaging (BASS/MzLS) otherwise use cuts for the Southern imaging survey (DECaLS).

Returns True if and only if the object is an ELG target.

Return type array_like

Notes

- Current version (10/14/19) is version 107 on the SV wiki.
- See *set_target_bits()* for other parameters.

desitarget.sv1.sv1_cuts.**isELG_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *gfiberflux=None*, *primary=None*, *south=True*)

Color cuts for ELG target selection classes (see, e.g., *desitarget.cuts.set_target_bits()* for parameters).

desitarget.sv1.sv1_cuts.**isLRG**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *zfiberflux=None*, *rflux_snr=None*, *zflux_snr=None*, *w1flux_snr=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *maskbits=None*, *primary=None*, *south=True*)

Target Definition of LRG. Returns a boolean array.

Parameters **south** (boolean, defaults to `True`) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if `south=False`, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns

- `array_like` – `True` if and only if the object is an LRG target.
- `array_like` – `True` for a 4 PASS nominal optical + nominal IR LRG.
- `array_like` – `True` for a 4 PASS object in the LRG SV superset.
- `array_like` – `True` for an 8 PASS nominal optical + nominal IR LRG.
- `array_like` – `True` for an 8 PASS object in the LRG SV superset.

Notes

- Current version (02/18/20) is version 123 on the SV wiki.
- See *set_target_bits()* for other parameters.

desitarget.sv1.sv1_cuts.**isLRG_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *zfiberflux=None*, *south=True*, *primary=None*)

See *isLRG()* for details.

desitarget.sv1.sv1_cuts.**isMWS_WD**(*primary=None*, *gaia=None*, *galb=None*, *astrometricexcessnoise=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *parallaxovererror=None*, *photbprpexcessfactor=None*, *astrometricsigma5dmax=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*)

Set bits for WHITE DWARF Milky Way Survey targets.

:param see *set_target_bits()* for other parameters.:

Returns **mask** – True if and only if the object is a MWS-WD target.

Return type array_like.

Notes

- Current version (08/01/18) is version 121 on the wiki.

desitarget.sv1.sv1_cuts.**isMWS_main_sv**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *gnobs=None*, *rnobs=None*, *gfracmasked=None*, *rfracmasked=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *obs_rflux=None*, *objtype=None*, *gaia=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*, *gaiaaen=None*, *gaiadupsource=None*, *primary=None*, *south=True*)

Set bits for main MWS SV targets.

:param see *set_target_bits()* for parameters.:

Returns

array_like. True if and only if the object is a MWS_MAIN_BROAD target.

mask2 [array_like.] True if and only if the object is a MWS_MAIN_FAINT target.

Return type mask1

Notes

- as of 26/7/19, based on version 79 on the wiki.
- for SV, no astrometric selection or colour separation

desitarget.sv1.sv1_cuts.**isMWS_nearby**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *objtype=None*, *gaia=None*, *primary=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *parallaxerr=None*, *obs_rflux=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*)

Set bits for NEARBY Milky Way Survey targets.

:param see *set_target_bits()* for parameters.:

Returns **mask** – True if and only if the object is a MWS-NEARBY target.

Return type array_like.

Notes

- Current version (09/20/18) is version 129 on the wiki.

desitarget.sv1.sv1_cuts.**isQSO_color_high_z**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *south=True*)

Color cut to select Highz QSO (z>~2.)

desitarget.sv1.sv1_cuts.**isQSO_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *primary=None*, *south=True*)

Test if sources have quasar-like colors in a color box. (see, e.g., *isQSO_cuts()*).

desitarget.sv1.sv1_cuts.**isQSO_cuts**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *w1snr=None*, *w2snr=None*, *dchisq=None*, *maskbits=None*, *objtype=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *primary=None*, *south=True*)

Definition of QSO target classes from color cuts. Returns a boolean array.

Parameters **south** (boolean, defaults to True) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if south=False, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns `True` for objects that pass the quasar color/morphology/logic cuts.

Return type `array_like`

Notes

- Current version (09/25/19) is version 100 on the SV wiki.
- See *set_target_bits()* for other parameters.

desitarget.sv1.sv1_cuts.**isQSO_highz_faint**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *objtype=None*, *release=None*, *dchisq=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *maskbits=None*, *primary=None*, *south=True*)

Definition of QSO target for highz (z>2.0) faint QSOs. Returns a boolean array.

Parameters **south** (`boolean`, defaults to `True`) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if `south=False`, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns `True` for objects that pass the quasar color/morphology/logic cuts.

Return type `array_like`

Notes

- Current version (09/25/19) is version 100 on the SV wiki.
- See *set_target_bits()* for other parameters.

desitarget.sv1.sv1_cuts.**isQSO_randomforest**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *objtype=None*, *release=None*, *dchisq=None*, *maskbits=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *primary=None*, *south=True*)

Definition of QSO target class using random forest. Returns a boolean array.

Parameters **south** (`boolean`, defaults to `True`) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if `south=False`, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns `True` for objects that pass the quasar color/morphology/logic cuts.

Return type `array_like`

Notes

- Current version (09/25/19) is version 100 on the SV wiki.
- See *set_target_bits()* for other parameters.

desitarget.sv1.sv1_cuts.**isQSOz5_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *w1flux=None*, *w2flux=None*, *primary=None*, *south=True*)

Color cut to select z~5 quasar targets. (See *isQSOz5_cuts()*).

desitarget.sv1.sv1_cuts.**isQSOz5_cuts**(*gflux=None*, *rflux=None*, *zflux=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *w1flux=None*, *w2flux=None*, *w1snr=None*, *w2snr=None*, *dchisq=None*, *maskbits=None*, *objtype=None*, *primary=None*, *south=True*)

Definition of z~5 QSO targets from color cuts. Returns a boolean array.

Parameters **south** (boolean, defaults to True) – Use cuts appropriate to the Northern imaging surveys (BASS/MzLS) if south=False, otherwise use cuts appropriate to the Southern imaging survey (DECaLS).

Returns True for objects that pass the quasar color/morphology/logic cuts.

Return type array_like

Notes

- Current version (03/11/20) is version 126 on the SV wiki.
- See *set_target_bits()* for other parameters.

desitarget.sv1.sv1_cuts.**isSTD**(*gflux=None*, *rflux=None*, *zflux=None*, *primary=None*, *gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *gfracmasked=None*, *rfracmasked=None*, *zfracmasked=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *gfluxivar=None*, *rfluxivar=None*, *zfluxivar=None*, *objtype=None*, *gaia=None*, *astrometricexcessnoise=None*, *paramssolved=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *dupsource=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*, *bright=False*)

Select STD targets using color cuts and photometric quality cuts.

Parameters **bright** (boolean, defaults to False) – if True apply magnitude cuts for "bright" conditions; otherwise, choose "normal" brightness standards. Cut is performed on *gaiagmag*.

Returns True if and only if the object is a STD star.

Return type array_like

Notes

- Current version (11/05/18) is version 24 on the SV wiki.
- See *set_target_bits()* for other parameters.

desitarget.sv1.sv1_cuts.**isSTD_colors**(*gflux=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *w2flux=None*, *primary=None*)

Select STD stars based on Legacy Surveys color cuts. Returns a boolean array. see *isSTD()* for other details.

desitarget.sv1.sv1_cuts.**isSTD_gaia**(*primary=None*, *gaia=None*, *astrometricexcessnoise=None*, *pmra=None*, *pmdec=None*, *parallax=None*, *dupsource=None*, *paramssolved=None*, *gaiagmag=None*, *gaiabmag=None*, *gaiarmag=None*)

Gaia quality cuts used to define STD star targets see *isSTD()* for other details.

desitarget.sv1.sv1_cuts.**notinBGS_mask**(*gflux=None*, *rflux=None*, *zflux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *primary=None*, *gfracmasked=None*, *rfracmasked=None*, *zfracmasked=None*, *gfracflux=None*, *rfracflux=None*, *zfracflux=None*, *gfracin=None*, *rfracin=None*, *zfracin=None*, *w1snr=None*, *gfluxivar=None*, *rfluxivar=None*, *zfluxivar=None*, *Grr=None*, *gaiagmag=None*, *maskbits=None*, *objtype=None*, *targtype=None*)

Standard set of masking cuts used by all BGS target selection classes (see, e.g., `isBGS_faint()` for parameters).

desitarget.sv1.sv1_cuts.**notinELG_mask**(*maskbits=None*, *gsnr=None*, *rsnr=None*, *zsnr=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *primary=None*)

Standard set of masking cuts used by all ELG target selection classes. (see *set_target_bits()* for parameters).

desitarget.sv1.sv1_cuts.**notinLRG_mask**(*primary=None*, *rflux=None*, *zflux=None*, *w1flux=None*, *zfiberflux=None*, *gnobs=None*, *rnobs=None*, *znobs=None*, *rflux_snr=None*, *zflux_snr=None*, *w1flux_snr=None*, *maskbits=None*)

See *isLRG()* for details.

Returns `True` if and only if the object is NOT masked for poor quality.

Return type `array_like`

desitarget.sv1.sv1_cuts.**notinMWS_main_sv_mask**(*gaia=None*, *gfracmasked=None*, *gnobs=None*, *gflux=None*, *rfracmasked=None*, *rnobs=None*, *rflux=None*, *gaiadupsource=None*, *primary=None*)

Standard set of masking-based cuts used by MWS target selection classes (see, e.g., *isMWS_main()* for parameters).

desitarget.sv1.sv1_cuts.**set_target_bits**(*photsys_north*, *photsys_south*, *obs_rflux*, *gflux*, *rflux*, *zflux*, *w1flux*, *w2flux*, *gfiberflux*, *rfiberflux*, *zfiberflux*, *objtype*, *release*, *gfluxivar*, *rfluxivar*, *zfluxivar*, *gnobs*, *rnobs*, *znobs*, *gfracflux*, *rfracflux*, *zfracflux*, *gfracmasked*, *rfracmasked*, *zfracmasked*, *gfracin*, *rfracin*, *zfracin*, *gallmask*, *rallmask*, *zallmask*, *gsnr*, *rsnr*, *zsnr*, *w1snr*, *w2snr*, *deltaChi2*, *dchisq*, *gaia*, *pmra*, *pmdec*, *parallax*, *parallaxovererror*, *parallaxerr*, *gaiagmag*, *gaiabmag*, *gaiarmag*, *gaiaaen*, *gaiadupsource*, *gaiaparamssolved*, *gaiabprpfactor*, *gaiasigma5dmax*, *galb*, *tcnames*, *qso_optical_cuts*, *qso_selection*, *maskbits*, *Grr*, *refcat*, *primary*, *resolvetargs=True*)

Perform target selection on parameters, return target mask arrays.

Returns (desi_target, bgs_target, mws_target) where each element is an ndarray of target selection bitmask flags for each object.

Return type `ndarray`

Notes

- Units for Gaia quantities are the same as the Gaia data model.

- See *set_target_bits()* for parameters.

1.22 desitarget.sv1.sv1_targetmask

This looks more like a script than an actual module.

1.23 desitarget.targetmask

This looks more like a script than an actual module.

desitarget.targetmask.**_load_mask_priorities**(*bitdefs*, *handle='priorities'*, *prename=''*)
Priorities and NUMOBS are defined in the yaml file, but they aren't a bitmask and so require some extra processing.

desitarget.targetmask.**load_mask_bits**(*prefix=''*)
Load bit definitions from yaml file.

1.24 desitarget.targets

Presumably this defines targets.

desitarget.targets.**_cmx_calc_priority**(*targets*, *priority*, *obscon*, *unobs*, *done*, *zgood*, *zwarn*, *cmx_mask*, *obsconditions*)
Special-case logic for target priorities in CMX.

Parameters

- **targets** (ndarray) – numpy structured array or astropy Table of targets. Must include the column *CMX_TARGET*.
- **priority** (ndarray) – Initial priority values set, in calc_priorities().
- **obscon** (str) – A combination of strings that are in the desitarget bitmask yaml file (specifically in *desitarget.targetmask.obsconditions*), e.g. "DARK|GRAY". Governs the behavior of how priorities are set based on "obsconditions" in the desitarget bitmask yaml file.
- **unobs** (ndarray) – Boolean flag on targets indicating state UNOBS.
- **done** (ndarray) – Boolean flag on targets indicating state DONE.
- **zgood** (ndarray) – Boolean flag on targets indicating state ZGOOD.
- **zwarn** (ndarray) – Boolean flag on targets indicating state ZWARN.
- **cmx_mask** (BitMask) – The CMX target bitmask.
- **obscondtions** (BitMask) – The CMX obsconditions bitmask.

Returns The updated priority values.

Return type ndarray

Notes

- Intended to be called only from within calc_priority(), where any pre-processing of the target state flags (uobs, done, zgood, zwarn) is handled.

desitarget.targets.**calc_numobs_more**(*targets*, *zcat*, *obscon*)

Calculate target NUMOBS_MORE from masks, observation/redshift status.

Parameters

- **targets** (ndarray) – numpy structured array or astropy Table of targets. Must include the columns *DESI_TARGET*, *BGS_TARGET*, *MWS_TARGET* (or their SV/cmx equivalents) *TARGETID* and *NUMOBS_INIT*.
- **zcat** (ndarray) – numpy structured array or Table of redshift info. Must include *Z*, *ZWARN*, *NUMOBS* and *TARGETID* and BE SORTED ON TARGETID to match *targets* row-by-row. May also contain *NUMOBS_MORE* if this isn't the first time through MTL and *NUMOBS > 0*.
- **obscon** (str) – A combination of strings that are in the desitarget bitmask yaml file (specifically in *desitarget.targetmask.obsconditions*), e.g. "DARK|GRAY". Governs the behavior of how priorities are set based on "obsconditions" in the desitarget bitmask yaml file.

Returns Integer array of number of additional observations (NUMOBS_MORE).

Return type array

Notes

- Will automatically detect if the passed targets are main survey, commissioning or SV and behave accordingly.
- Most targets are updated to NUMOBS_MORE = NUMOBS_INIT-NUMOBS. Special cases include BGS targets which always get NUMOBS_MORE of 1 in bright time and QSO "tracer" targets which always get NUMOBS_MORE=0 in dark time.

desitarget.targets.**calc_priority**(*targets*, *zcat*, *obscon*)

Calculate target priorities from masks, observation/redshift status.

Parameters

- **targets** (ndarray) – numpy structured array or astropy Table of targets. Must include the columns *DESI_TARGET*, *BGS_TARGET*, *MWS_TARGET* (or their SV/cmx equivalents) and *TARGETID*.
- **zcat** (ndarray) – numpy structured array or Table of redshift info. Must include *Z*, *ZWARN*, *NUMOBS* and *TARGETID* and BE SORTED ON TARGETID to match *targets* row-by-row. May also contain *NUMOBS_MORE* if this isn't the first time through MTL and *NUMOBS > 0*.
- **obscon** (str) – A combination of strings that are in the desitarget bitmask yaml file (specifically in *desitarget.targetmask.obsconditions*), e.g. "DARK|GRAY". Governs the behavior of how priorities are set based on "obsconditions" in the desitarget bitmask yaml file.

Returns integer array of priorities.

Return type array

Notes

- If a target passes multiple selections, highest priority wins.
- Will automatically detect if the passed targets are main survey, commissioning or SV and behave accordingly.

desitarget.targets.**decode_targetid**(*targetid*)
break a DESI TARGETID into its constituent parts.

:param int or ndarray: The TARGETID for DESI, encoded according to the bits listed in desitarget.targetid_mask().

Returns

- int or ndarray – The OBJID from Legacy Surveys imaging or the row within a Gaia HEALPixel file in $GAIA_DIR/healpix if *gaia* is not None.
- int or ndarray – The BRICKID from Legacy Surveys imaging. or the Gaia HEALPixel chunk number for files in $GAIA_DIR/healpix if *gaia* is not None.
- int or ndarray – The RELEASE from Legacy Surveys imaging. Or, if < 1000, the secondary target class bit flag number from 'data/targetmask.yaml'. Or, if < 1000 and *sky* is not None, the HEALPixel processing number for SUPP_SKIES.
- int or ndarray – 1 if this object is a mock object (generated from mocks or from a random catalog, not from real survey data), 0 otherwise
- int or ndarray – 1 if this object is a blank sky object, 0 otherwise
- int or ndarray – The Gaia Data Release number (e.g. will be 2 for Gaia DR2). A value of 1 does NOT mean DR1. Rather it has the specific meaning of a DESI first-light commissioning target.

Notes

- if a 1-D array is passed, then an integer is returned. Otherwise an array is returned.
- see also DocDB 2348.

desitarget.targets.**encode_targetid**(*objid=None*, *brickid=None*, *release=None*, *mock=None*, *sky=None*, *gaiadr=None*)
Create the DESI TARGETID from input source and imaging info.

Parameters

- **objid** (int or ndarray, optional) – The OBJID from Legacy Surveys imaging or the row within a Gaia HEALPixel file in $GAIA_DIR/healpix if *gaia* is not None.
- **brickid** (int or ndarray, optional) – The BRICKID from Legacy Surveys imaging. or the Gaia HEALPixel chunk number for files in $GAIA_DIR/healpix if *gaia* is not None.
- **release** (int or ndarray, optional) – The RELEASE from Legacy Surveys imaging. Or, if < 1000, the secondary target class bit flag number from 'data/targetmask.yaml'. Or, if < 1000 and *sky* is not None, the HEALPixel processing number for SUPP_SKIES.
- **mock** (int or ndarray, optional) – 1 if this object is a mock object (generated from mocks or from a random catalog, not from real survey data), 0 otherwise
- **sky** (int or ndarray, optional) – 1 if this object is a blank sky object, 0 otherwise

- **gaiadr** (int or ndarray, optional) – The Gaia Data Release number (e.g. send 2 for Gaia DR2). A value of 1 does NOT mean DR1. Rather it has the specific meaning of a DESI first-light commissioning target.

Returns The TARGETID for DESI, encoded according to the bits listed in `desitarget.targetid_mask()`. If an integer is passed, then an integer is returned, otherwise an array is returned.

Return type int or *~numpy.ndarray*

Notes

- Has maximum flexibility so that mixes of integers and arrays can be passed, in case some value like BRICKID or SKY is the same for a set of objects. Consider, e.g.:

  print(

  targets.decode_targetid(

  targets.encode_targetid(objid=np.array([234,12]), brickid=np.array([234,12]), release=4000, sky=[1,0])))

(array([234,12]), array([234,12]), array([4000,4000]), array([0,0]), array([1,0]))

- See also DocDB 2348.

`desitarget.targets.`**`finalize`**(*targets*, *desi_target*, *bgs_target*, *mws_target*, *sky=False*, *randoms=False*, *survey='main'*, *darkbright=False*, *gaiadr=None*, *targetid=None*)

Return new targets array with added/renamed columns

Parameters

- **targets** (ndarray) – numpy structured array of targets.
- **desi_target** (ndarray) – 1D array of target selection bit flags.
- **bgs_target** (ndarray) – 1D array of target selection bit flags.
- **mws_target** (ndarray) – 1D array of target selection bit flags.
- **sky** (bool, defaults to `False`) – Pass `True` for sky targets, `False` otherwise.
- **randoms** (bool, defaults to `False`) – `True` if *targets* is a random catalog, `False` otherwise.
- **survey** (str, defaults to *main*) – Specifies which target masks yaml file to use. Options are *main*, *cmx* and *svX* (where X = 1, 2, 3 etc.) for the main survey, commissioning and an iteration of SV.
- **darkbright** (bool, optional, defaults to `False`) – If sent, then split *NUMOBS_INIT* and *PRIORITY_INIT* into *NUMOBS_INIT_DARK*, *NUMOBS_INIT_BRIGHT*, *PRIORITY_INIT_DARK* and *PRIORITY_INIT_BRIGHT* and calculate values appropriate to "BRIGHT" and "DARK|GRAY" observing conditions.
- **gaiadr** (int, optional, defaults to `None`) – If passed and not `None`, then build the *TARGETID* from the "GAIA_OBJID" and "GAIA_BRICKID" columns in the passed *targets*, and set the *gaiadr* part of *TARGETID* to whatever is passed.
- **targetid** (int64, optional, defaults to `None`) – In the mocks we compute *TARGETID* outside this function.

Returns

new targets structured array with the following additions:

- renaming OBJID -> BRICK_OBJID (it is only unique within a brick).
- renaming TYPE -> MORPHTYPE (used downstream in other contexts).
- **Adding new columns:**
  - TARGETID: unique ID across all bricks or Gaia files.
  - DESI_TARGET: dark time survey target selection flags.
  - MWS_TARGET: bright time MWS target selection flags.
  - BGS_TARGET: bright time BGS target selection flags.
  - PRIORITY_INIT: initial priority for observing target.
  - SUBPRIORITY: a placeholder column that is set to zero.
  - NUMOBS_INIT: initial number of observations for target.
  - OBSCONDITIONS: bitmask of observation conditions.

Return type ndarray

Notes

- SUBPRIORITY is the only column that isn't populated. This is because it's easier to populate it in a reproducible fashion when collecting targets rather than on a per-brick basis when this function is called. It's set to all zeros.

desitarget.targets.**initial_priority_numobs**(*targets*, *scnd=False*, *obscon='DARK|GRAY|BRIGHT|POOR|TWILIGHT12|TWILIGHT18'*)

highest initial priority and numobs for an array of target bits.

Parameters

- **targets** (ndarray) – An array of targets generated by, e.g., *cuts*. Must include at least (all of) the columns *DESI_TARGET*, *BGS_TARGET*, *MWS_TARGET* or corresponding cmx or SV columns.
- **scnd** (bool, optional, defaults to False) – If True then make all of the comparisons on the *SCND_TARGET* column instead of *DESI_TARGET*, *BGS_TARGET* and *MWS_TARGET*.
- **obscon** (str, optional, defaults to almost all OBSCONDITIONS) – A combination of strings that are in the desitarget bitmask yaml file (specifically in *desitarget.targetmask.obsconditions*).

Returns

- ndarray – An array of integers corresponding to the highest initial priority for each target consistent with the constraints on observational conditions imposed by *obscon*.
- ndarray – An array of integers corresponding to the largest number of observations for each target consistent with the constraints on observational conditions imposed by *obscon*.

Notes

- the initial priority for each target bit is in the file, e.g., data/targetmask.yaml. It can be retrieved using, for example, *desi_mask["ELG"].priorities["UNOBS"]*.
- the input obscon string can be converted to a bitmask using *desitarget.targetmask.obsconditions.mask(blat)*.

desitarget.targets.**main_cmx_or_sv**(*targets*, *rename=False*, *scnd=False*)
determine whether a target array is main survey, commissioning, or SV

Parameters

- **targets** (ndarray) – An array of targets generated by, e.g., *cuts* must include at least (all of) the columns *DESI_TARGET*, *MWS_TARGET* and *BGS_TARGET* or the corresponding commissioning or SV columns.
- **rename** (bool, optional, defaults to False) – If True then also return a copy of *targets* with the input *_TARGET* columns renamed to reflect the main survey format.
- **scnd** (bool, optional, defaults to False) – If True, then add the secondary target information to the output.

Returns

- list – A list of strings corresponding to the target columns names. For the main survey this would be [*DESI_TARGET*, *BGS_TARGET*, *MWS_TARGET*], for commissioning it would just be [*CMX_TARGET*], for SV1 it would be [*SV1_DESI_TARGET*, *SV1_BGS_TARGET*, *SV1_MWS_TARGET*]. Also includes, e.g. *SCND_TARGET*, if *scnd* is passed as True.
- list – A list of the masks that correspond to each column from the relevant main/cmx/sv yaml file. Also includes the relevant SCND_MASK, if *scnd* is passed as True.
- str – The string 'main', 'cmx' or 'svX' (where X = 1, 2, 3 etc.) for the main survey, commissioning and an iteration of SV. Specifies which type of file was sent.
- ndarray, optional, if *rename* is True – A copy of the input targets array with the *_TARGET* columns renamed to *DESI_TARGET*, and (if they exist) *BGS_TARGET*, *MWS_TARGET*.

desitarget.targets.**resolve**(*targets*)
Resolve which targets are primary in imaging overlap regions.

Parameters **targets** (ndarray) – Rec array of targets. Must have columns "RA" and "DEC" and either "RELEASE" or "PHOTSYS".

Returns The original target list trimmed to only objects from the "northern" photometry in the northern imaging area and objects from "southern" photometry in the southern imaging area.

Return type ndarray

desitarget.targets.**set_obsconditions**(*targets*, *scnd=False*)
set the OBSCONDITIONS mask for each target bit.

Parameters

- **targets** (ndarray) – An array of targets generated by, e.g., *cuts*. Must include at least (all of) the columns *DESI_TARGET*, *BGS_TARGET*, *MWS_TARGET* or corresponding cmx or SV columns.

- **scnd** (bool, optional, defaults to False) – If True then make all of the comparisons on the *SCND_TARGET* column instead of *DESI_TARGET*, *BGS_TARGET* and *MWS_TARGET*.

Returns The OBSCONDITIONS bitmask for the passed targets.

Return type ndarray

Notes

- the OBSCONDITIONS for each target bit is in the file, e.g. data/targetmask.yaml. It can be retrieved using, for example, *obsconditions.mask(desi_mask["ELG"].obsconditions)*.

1.25 desitarget.train

Stuff to do with training.

1.26 desitarget.train.train_mva_decals

- Training code developed by E. Burtin
- Update to run with DR3 by Ch. Yeche

This example can be run with the module desitarget/bin/qso_training

Three actions controlled by "step" flag: train - test - extract_myRF

Two examples of random forest (with and without r_mag) Two examples of adaboost (with and without r_mag)

1.26.1 Inputs

The training samples and the test sample are available on nesrc at: /global/project/projectdirs/desi/target/qso_training/

The qso training sample qso_dr3_nora36-42.fits is obtained with QSOs from the fat stripe 82 and bright QSOs (sigma(r)<0.02) of the rest of the footprint. Note that the 36<ra<42 region was exluded of the training sample to allow independant test over this 36<ra<42 region

The star training sample qso_dr3_nora36-42_normalized.fits is obtained with PSF objects of stripe 82, which are not variable (NNVariability<0.3) and not known QSOs. "normalized" means that the r_mag distribution of the stars is exactly the same as that of qsos.

The test sample Stripe82_dr3_decals is the stripe 82. Note this file contains the results of the four algorithm for seed 0. The new probabilities should be _strictly_ identical

1.26.2 Outputs

train Four compressed files are produced. Each file corresponds to one algorithm (*i.e.* adaboost/random forest, with/withour rmag).

test Produce the probabilities for the four algorithms, the results are stored in Stripe82_dr3_decals_newTraining.fits

extract_myRF Use the file rf_model_dr3.pkl.gz produced in step "train" and convert it in a numpy array that can be read by desitarget.myRF class. The results is the compressed numpy array rf_model_dr3.npz

1.27 desitarget.uratmatch

Useful URAT matching and manipulation routines.

desitarget.uratmatch.**_get_urat_nside**()
Grab the HEALPixel nside to be used throughout this module.

Returns The HEALPixel nside number for URAT file creation and retrieval.

Return type int

desitarget.uratmatch.**find_urat_files**(*objs*, *neighbors=True*, *radec=False*)
Find full paths to URAT healpix files for objects by RA/Dec.

Parameters

- **objs** (ndarray) – Array of objects. Must contain the columns "RA" and "DEC".
- **neighbors** (bool, optional, defaults to True) – Also return all pixels that touch the files of interest to prevent edge effects (e.g. if a URAT source is 1 arcsec away from a primary source and so in an adjacent pixel).
- **radec** (bool, optional, defaults to False) – If True then the passed *objs* is an [RA, Dec] list instead of a rec array that contains "RA" and "DEC".

Returns A list of all URAT files to read to account for objects at the passed locations.

Return type list

Notes

- The environment variable $URAT_DIR must be set.

desitarget.uratmatch.**get_urat_dir**()
Convenience function to grab the URAT environment variable.

Returns The directory stored in the $URAT_DIR environment variable.

Return type str

desitarget.uratmatch.**make_urat_files**(*numproc=5*, *download=False*)
Make the HEALPix-split URAT files in one fell swoop.

Parameters

- **numproc** (int, optional, defaults to 5) – The number of parallel processes to use.
- **download** (bool, optional, defaults to False) – If True then wget the URAT binary files from Vizier.

Returns

But produces: - URAT DR1 binary files in $URAT_DIR/binary (if download=True). - URAT CSV files with all URAT columns in $URAT_DIR/csv. - FITS files with columns from *uratdatamodel* in $URAT_DIR/fits. - FITS files reorganized by HEALPixel in $URAT_DIR/healpix.

The HEALPixel sense is nested with nside=_get_urat_nside(), and each file in $URAT_DIR/healpix is called healpix-xxxxx.fits, where xxxxx corresponds to the HEALPixel number.

Return type Nothing

Notes

- The environment variable $URAT_DIR must be set.
- if numproc==1, use the serial, instead of the parallel, code.
- Runs in about 2 hours with numproc=25 if download is `True`.
- Runs in about 1 hour with numproc=25 if download is `False`.

`desitarget.uratmatch.`**`match_to_urat`**(*objs*, *matchrad=1.0*, *radec=False*)

Match objects to URAT healpix files and return URAT information.

Parameters

- **`objs`** (`ndarray`) – Must contain at least "RA" and "DEC".
- **`matchrad`** (`float`, optional, defaults to 1 arcsec) – The radius at which to match in arcseconds.
- **`radec`** (`bool`, optional, defaults to `False`) – If `True` then the passed *objs* is an [RA, Dec] list instead of a rec array.

Returns The matching URAT information for each object. The returned format is as for desitarget.uratmatch.uratdatamodel with an extra column "URAT_SEP" which is the matching distance in ARCSECONDS.

Return type `ndarray`

Notes

- For objects that do NOT have a match in URAT, the "URAT_ID" and "URAT_SEP" columns are -1, and other columns are zero.
- Retrieves the CLOSEST match to URAT for each passed object.
- Because this reads in HEALPixel split files, it's (far) faster for objects that are clumped rather than widely distributed.

`desitarget.uratmatch.`**`scrape_urat`**(*url='http://cdsarc.u-strasbg.fr/ftp/I/329/URAT1/v12/'*, *nfiletest=None*)

Retrieve the binary versions of the URAT files.

Parameters

- **`url`** (`str`) – The web directory that hosts the archived binary URAT files.
- **`nfiletest`** (`int`, optional, defaults to `None`) – If an integer is sent, only retrieve this number of files, for testing.

Returns But the archived URAT files are written to $URAT_DIR/binary.

Return type Nothing

Notes

- The environment variable $URAT_DIR must be set.
- Runs in about 50 minutes for 575 URAT files.

desitarget.uratmatch.**urat_binary_to_csv**()
Convert files in $URAT_DIR/binary to files in $URAT_DIR/csv.

Returns But the archived URAT binary files in $URAT_DIR/binary are converted to CSV files in the $URAT_DIR/csv.

Return type Nothing

Notes

- The environment variable $URAT_DIR must be set.
- Relies on the executable urat/fortran/v1dump, which is only tested at NERSC and might need compiled by the user.
- Runs in about 40 minutes for 575 files.

desitarget.uratmatch.**urat_csv_to_fits**(*numproc=5*)
Convert files in $URAT_DIR/csv to files in $URAT_DIR/fits.

Parameters **numproc** (int, optional, defaults to 5) – The number of parallel processes to use.

Returns But the archived URAT CSV files in $URAT_DIR/csv are converted to FITS files in the directory $URAT_DIR/fits. Also, a look-up table is written to $URAT_DIR/fits/hpx-to-files.pickle for which each index is an nside=_get_urat_nside(), nested scheme HEALPixel and each entry is a list of the FITS files that touch that HEAPixel.

Return type Nothing

Notes

- The environment variable $URAT_DIR must be set.
- if numproc==1, use the serial code instead of the parallel code.
- Runs in about 10 minutes with numproc=25 for 575 files.

desitarget.uratmatch.**urat_fits_to_healpix**(*numproc=5*)
Convert files in $URAT_DIR/fits to files in $URAT_DIR/healpix.

Parameters **numproc** (int, optional, defaults to 5) – The number of parallel processes to use.

Returns But the archived URAT FITS files in $URAT_DIR/fits are rearranged by HEALPixel in the directory $URAT_DIR/healpix. The HEALPixel sense is nested with nside=_get_urat_nside(), and each file in $URAT_DIR/healpix is called healpix-xxxxx.fits, where xxxxx corresponds to the HEALPixel number.

Return type Nothing

Notes

- The environment variable $URAT_DIR must be set.
- if numproc==1, use the serial code instead of the parallel code.
- Runs in about 10 minutes with numproc=25.

CHAPTER 2

desitarget Change Log

2.1 0.41.1 (unreleased)

- No changes yet.

2.2 0.41.0 (2020-08-04)

- Support for python/3.8 and numpy/1.18, including new tests [*PR #631*, *PR #634*]
- **Minor data model fixes, error checks and streamlining [PR #627].**
  - The most important change is that MWS science targets are no longer observed in GRAY or DARK, except for MWS_WDs.
- Cleanup: Avoid absolute path in resource_filename [PR #626].
- **Update masking to be "all-sky" using Gaia/Tycho/URAT [PR #625]:**
  - General desitarget functionality to work with Tycho files.
  - Deprecate using the sweeps to mask bright objects as this is now being done using MASKBITS from the imaging catalogs.
  - Functionality to allow masks to be built at different epochs, via careful treatment of Tycho/Gaia/URAT proper motions.
  - Bright star masks are now explicitly written to a $MASK_DIR.
  - The radius-magnitude relationship is now a single function.
  - Refactoring of unit tests to be simpler and have more coverage.
  - Skies and supplemental skies are now always masked by default.
  - A lack of backward compatibility, which should be OK as the masking formalism wasn't being extensively used.

- **Functionality for iterations of SV beyond sv1 [PR #624]. Includes:**
  - A script to create the necessary files for new iterations of SV.
  - Generalized mask/cuts handling for survey=svX, X being any integer.
  - `targets.main_cmx_or_sv()` also updated to handle survey=svX.
  - **Alter the automated creation of output SV target directory names:**
    - write svX targets to /targets/svX/ instead of just targets/sv/.
  - **Make TARGETID for secondary targets unique for iterations of SVX:**
    - Schema is RELEASE=(X-1)*100 + SCND_BIT for SVX-like surveys...
    - ...and RELEASE=5*100 + SCND_BIT for the Main Survey.
- **Adjust MWS SV1 target classes for new SV schedule [PR #623]:**
  - More generic names for clusters, stream, dwarf targets.
  - Remove ORPHAN, add CV.
  - Lower priority for SEGUE targets.

2.3 0.40.0 (2020-05-26)

- Add RELEASE for dr9i, dr9j (etc.) of the Legacy Surveys [PR #622].
- **Repartition sky files so skies lie in HEALPix boundaries [PR #621]:**
  - Previously, unlike other target classes, skies were written such that the *brick centers* in which they were processed, rather than the sky locations themselves, lay within given HEALPixels.
  - `is_sky_dir_official()` now checks skies are partitioned right.
  - *bin/repartition_skies* now reassigns skies to correct HEALPixels.
  - **In addition, also includes:**
    - Significant (5-10x) speed-ups in `read_targets_in_hp()`.
    - Remove supplemental skies that are near existing sky locations. (which addresses issue #534).
    - A handful of more minor fixes and speed-ups.
- **Various updates to targeting bits and MTL [PR #619]. Includes:**
  - Don't select any BGS_WISE targets in the Main Survey.
  - Always set BGS targets with a ZWARN > 0 to a priority of DONE.
  - Add an informational bit for QSOs selected with the high-z RF (addresses issue #349).
  - MWS targets should drop to a priority of DONE after one observation (but will always be higher priority than BGS for that observation).
  - Update the default priorities for reobserving Lyman-alpha QSOs (as described in issue #486, which this addresses).
- ***NUMOBS_MORE* for tracer QSOs that are also other targets [PR #617]:**
  - Separate the calculation of *NUMOBS_MORE* into its own function.
  - Consistently use *zcut* = 2.1 to define Lyman-Alpha QSOs.

  - Check tracer QSOs that are other targets drop to *NUMOBS_MORE* = 0.
  - New unit test to enforce that check on such tracer QSOs.
  - New unit test to check BGS always gets *NUMOBS_MORE* = 1 in BRIGHT.
  - Enforce maximum seed in `randoms_in_a_brick_from_edges()`.
- Update masks for QSO Random Forest selection for DR8 [PR #615]
- Add a new notebook tutorial about the Merged Target List [PR #614].
- Recognize (and skip) existing (completed) healpixels when running *select_mock_targets* [PR #591].

2.4 0.39.0 (2020-05-01)

- **Help the mocks run on pixel-level imaging data [PR #611]. Includes:**
  - New `geomask.get_brick_info()` function to look up the brick names associated with each HEALPixel.
  - In `randoms.quantities_at_positions_in_a_brick()`, add a *justlist* option to list the (maximal) required input files.
  - Minor bug fixes and documentation updates.
- Update QSO Random Forest selection (and files) for DR8 [PR #610].

2.5 0.38.0 (2020-04-23)

- Minor updates for latest DR9 imaging versions (dr9f/dr9g) [PR #607].
- **Extra columns and features in the random catalogs [PR #606]:**
  - Better error messages and defaults for *bin/supplement_randoms*.
  - Don't calculate APFLUX quantities if aprad=0 is passed.
  - **Pass the randoms through the *finalize* and *make_mtl* functions:**
    - To populate columns needed to run fiberassign on the randoms.
    - Addresses issue #597.
  - Add the *BRICKID* column to the random catalogs.
  - Also add a realistic *TARGETID* (and *RELEASE, BRICK_OBJID*).
  - Recognize failure modes more quickly (and fail more quickly).
  - Write out both "resolve" and "noresolve" (North/South) catalogs.
- Fixes a typo in the priority of MWS_WD_SV targets [PR #601].
- Fixes calc_priority logic for MWS CMX targets [PR #601].
- Separate calc_priority() for CMX into a separate function [PR #601].
- Alter cmx targetmask such that obsconditions can be used to work around MWS/BGS conflicts on MWS CMX tiles [PR #601].
- Update test_priorities() for new MWS CMX targets scheme [PR #601].
- Adds SV0_MWS_FAINT bit [PR #601].

2.6 0.37.3 (2020-04-15)

- **Update QA now basemap dependency is removed [PR #605]:**
  - Also reintroduce unit tests in *test_qa.py*.
  - basemap dependency was removed in desiutil PR #141

2.7 0.37.2 (2020-04-13)

- Fix *select_mock_targets* I/O bug reported in #603 [PR #604].

2.8 0.37.1 (2020-04-07)

- Fix mock QSO density bug reported in #594 [PR #602].
- Fixes a typo in the priority of MWS_WD_SV targets [PR #600].

2.9 0.37.0 (2020-03-12)

- Add *SV0_MWS_CLUSTER_* target classes for commissioning [PR #599].
- Flag the high-z quasar selection in CMX (as *SV0_QSO_Z5*) [PR #598].
- Leak of Bright Stars in BGS Main Survey and BGS SV fixed [PR #596].
- Restrict skies to the geometric boundaries of their brick [PR #595].
- **Changes in CMX after running code for Mini-SV [PR #592]. Includes:**
  - g/G >= 16 for *SV0_BGS*/*SV0_MWS*/*SV0_WD*/*MINI_SV_BGS_BRIGHT*.
  - Remove the LRG *LOWZ_FILLER* class (both in SV and CMX).
  - Mask on *bright* in *MASKBITS* for z~5 QSOs (both in SV and CMX).
  - Remove the 'low quality' (*lowq*) component of *SV0_BGS*.
  - Add optical *MASKBITS* flags for LRGs (in Main Survey, SV and CMX).

2.10 0.36.0 (2020-02-16)

- Add Main Survey LRG/ELG/QSO/BGS cuts to CMX for Mini-SV [PR #590].
- Cut on NOBS > 0 for QSOs and LRGs for Main Survey and SV [PR #589].
- **Fix bug when adding LSLGA galaxies into Main Survey BGS [PR #588]:**
  - Catch cases of bytes/str types as well as zero-length strings.
- Noting (here) that we used the BFG to excise lots of junk [PR #587].
- **Updates and fixes to QA for DR9 [PR #584]. Includes:**
  - Options to pre-process and downsample input files to speed testing.

  - Better labeling of QA output, including cleaning up labeling bugs.
  - Make points in scatter plots black to contrast with blue contours.
  - Smarter clipping of dense pixels in histogram plots and sky maps.
  - Print out densest pixels for each target class, with viewer links.
- **Update BGS Main target selection as stated in [PR #581]. Includes:**
  - Changes in Fibre Magnitude Cut.
  - **LSLGA galaxies manually added to BGS.**
    - Future-proof LSLGA object references changing ('L2' –> 'LX').
  - 'REF_CAT' information passed to throught '_prepare_optical_wise'.
- Tune QSO SV selection for both North and South for dr9d [PR #580].

2.11 0.35.3 (2020-02-03)

- **Further fixes for DR9 [PR #579]. Includes:**
  - Add `SERSIC` columns for the DR9 data model.
  - **Read the bricks file in lower-case in `get_brick_info()`:**
    - As, during DR9 testing, it's been both upper- and lower-case.
  - **Set the default `nside` to `None` for the randoms:**
    - To force the user to choose an `nside`, or fail otherwise.
  - Fix a numpy future/deprecation warning.
- Load yaml config file safely in `mpi_select_mock_targets` [PR #577].
- Fix bugs in updating primary targets with secondary bits set [PR #574].
- Adds more stellar SV targets [PR #574].
- Add LyA features to `select_mock_targets` [PR #565].

2.12 0.35.2 (2019-12-20)

- Fix z~5 QSO bug in CMX/SV0 that was already fixed for SV [PR #576].

2.13 0.35.1 (2019-12-16)

- Fix bugs triggered by empty files or regions of the sky [PR #575].

2.14 0.35.0 (2019-12-15)

- **Preparation for DR9 [PR #573]. Includes:**
  - Update data model, maintaining backwards compatibility with DR8.

  - Don't set the `SKY` bit when setting the `SUPP_SKY` bit.
  - **Users can input a seed (1, 2, 3, etc.) to `bin/select_randoms`:**
    - This user-provided seed is added to the output file name.
    - Facilitates generating a range of numbered random catalogs.
  - Write out final secondaries using `io.find_target_files()`.
- **More clean-up of glitches and minor bugs [PR #570]. Includes:**
  - Remove Python 3.5 unit tests.
  - **Catch AssertionError if NoneType input directory when writing.**
    - Later (correctly) updated to AttributeError directly in master.
  - Assert the data model when reading secondary target files.
  - Use io.find_target_files() to name priminfo file for secondaries.
  - Allow N < 16 nodes when bundling files for slurm.
  - Use the DR14Q file for SV, not the DR16Q file.
- Fix bug where wrong SNRs were passed to z~5 QSO selection [PR #569].
- **General clean-up of glitches and minor bugs [PR #564]. Includes:**
  - Don't include BACKUP targets in the pixweight files.
  - Correctly write all all-sky pixels outside of the Legacy Surveys.
  - Propagate flags like –nosec, –nobackup, –tcnames when bundling.
  - Write –tcnames options to header of output target files.
  - Deprecate the sandbox and file-format-check function.
  - Find LSLGAs using 'L' in *REF_CAT* not 'L2' (to prepare for 'L3').
  - Refactor to guard against future warnings and overflow warnings.
  - Return all HEALpixels at *nside* in `sweep_files_touch_hp()`.
- **Strict `NoneType` checking and testing for fiberfluxes [PR #563]:**
  - Useful to ensure ongoing compatibility with the mocks.
- Bitmasks (1,12,13), rfiberflux cut for BGS Main Survey [PR #562].
- Implement a variety of fixes to *select_mock_targets* [PR #561].
- **Fixes and updates to `secondary.py` [PR #530]:**
  - Fix a bug that led to incorrect `OBSCONDITIONS` for secondary-only targets.
  - Secondary target properties can override matched primary properties, but only for restricted combinations of DESI_TARGET bits (MWS and STD).
- **Add stellar SV targets [PR #530]:**
  - Add MWS SV target definitions in `sv1_targetmask` and `cuts`.
  - Science WDs are now a secondary target class.
  - Adds a bright limit to the `MWS-NEARBY` sample.
  - Add stellar SV secondary targets in `sv1_targetmask`.

  - Remove the `BACKSTOP` secondary bit.

2.15 0.34.0 (2019-11-03)

- **Update SV0 (BGS, ELG, LRG, QSO) classes for commissioning [PR #560].**
  - Also add new `STD_DITHER` target class for commissioning.
- **All-sky/backup targets, new output data model [PR #558]. Includes:**
  - Add all-sky/backup/supplemental targets for SV.
  - Add all-sky/backup/supplemental targets for the Main survey.
  - Write dark/bright using, e.g. *targets/dark/targets-*.fits* format.
  - New *targets/targets-supp/targets-*.fits* format for output.
  - Add `io.find_target_files()` to parse output data model.
  - **File names now generated automatically in *io.write_** functions:**
    - File-name-generation used by randoms, skies, targets and gfas.
    - *select_** binaries for these classes use this functionality.
  - Change CMX `BACKUP_FAINT` limit to G < 19.

2.16 0.33.3 (2019-10-31)

- Add cuts for z = 4.3-4.8 quasar into the z5QSO selection [PR #559].

2.17 0.33.2 (2019-10-17)

- Add FIBERFLUX_IVAR_G/R/Z to mock skies when merging [PR #556].
- Fix minor bugs in *select_mock_targets* [PR #555].
- **Update the ELG selections for SV [PR #553]. Includes:**
  - **Four new bit names:**
    - `ELG_SV_GTOT`, `ELG_SV_GFIB`.
    - `ELG_FDR_GTOT`, `ELG_FDR_GFIB`.
  - Associated new ELG selections with north/south differences.
  - Propagate `FIBERFLUX_G` from the sweeps for SV ELG cuts.
  - Increase the default sky densities by a factor of 4x.
  - Relax CMX `BACKUP_FAINT` limit to G < 21 to test fiber assign.
- Bright-end `FIBERFLUX_R` cut on `BGS_FAINT_EXT` in SV [PR #552].
- **Update LRG selections for SV [PR #550]. Includes:**
  - The zfibermag faint limit is changed from 21.6 to 21.9.
  - IR-selected objects with r-W1>3.1 not subjected to the sliding cut.

2.18 0.33.1 (2019-10-13)

- **Enhancements and on-sky clean-up for SV and CMX [PR #551]. Includes:**
  - Add areas contingent on `MASKBITS` to the `pixweight-` files.
  - Change `APFLUX` to `FIBERFLUX` for skies and supp-skies.
  - Add new M33 First Light program.
  - Change priorities for the First Light programs.
  - Retain Tycho, and sources with no measured proper motion, in GFAs.
  - Add the `REF_EPOCH` column to all target files.

2.19 0.33.0 (2019-10-06)

- **Update skies, GFAs and CMX targets for all-sky observing [PR #548]:**
  - Process and output GFAs, skies and CMX targets split by HEALPixel.
  - "bundling" scripts to parallelize GFAs, skies, CMX by HEALPixel.
  - Bundle across all HEALPixels (not just those in the footprint).
  - Add pixel information to file headers for GFAs, skies and CMX.
  - Write all-sky CMX targets separately from in-footprint targets.
  - Add back-up and first light targets for commissioning.
  - New TARGETID encoding scheme for Gaia-only and first light targets.
  - Resolve commissioning targets from the Legacy Surveys.
  - io.read functions can now process SKY and GFA target files.
  - New function to read in targets restricted to a set of DESI tiles.
  - Implement Boris Gaensicke's geographical cuts for Gaia.
  - Update unit tests to use DR8 files.
- **Further updates to changes in PR #531, [PR #544]. Includes:**
  - A *–writeall* option to *select_secondary* writes a unified target file without the BRIGHT/DARK split, as for *select_targets*
  - Removes duplicate secondaries that arise from multiple matches to one primary and secondary targets appearing in more than one input file. The duplciate with highest *PRIORTIY_INIT* is retained.
- Update mocks to match latest data-based targets catalogs [PR #543].
- **Add new redshift 5 (`QSO_Z5`) SV QSO selection [PR #539]. Also:**
  - Remove all Tycho and LSLGA sources from the GFA catalog.
  - Minor improvements to documentation for secondary targets.
  - Use N/S bricks for skies when S/N bricks aren't available.
- Tune, high-z, faint (`QSO_HZ_F`) SV QSO selection [PR #538]
- **Use `SPECTYPE` from `zcat` to set `NUMOBS_MORE` [PR #537]:**

  - Updates behavior for tracer QSOs vs. LyA QSOs in MTL.
- **Update LRG selections for DR8 [PR #532]. Includes:**
  - New LRG selection for SV with fewer bits.
  - New `LOWZ_FILLER` class for SV.
  - Add LRG 4PASS and 8PASS bits/classes using cuts on `FLUX_Z`.
  - New and simplified LRG selection for the Main Survey.
  - Deprecate Main Survey 1PASS/2PASS LRGs, all LRGs now have one pass.
  - Deprecate some very old code in *desitarget.targets*.
- **Finalize secondaries, add BRIGHT/DARK split [PR #531]. Includes:**
  - Updated data model for secondaries.
  - New secondary output columns (`OBSCONDITIONS`, proper motions).
  - Add a cached file of primary TARGETIDs to prevent duplicates.
  - Create a more reproducible TARGETID for secondaries.
  - Automatically write secondaries split by BRIGHT/DARK.
  - Add option to pass secondary file in MTL.
  - **Insist on observing layer/conditions for MTL:**
    - **Ensures correct behavior for dark targets in bright time...**
      - ...and bright-time targets observed in dark-time.
  - Minor update to the `MWS_BROAD` class.
- Add info on versioning, main_cmx_or_sv to bitmask notebook [PR #527]

2.20 0.32.0 (2019-08-07)

- **Add URAT catalog information [PR #526]. Includes:**
  - New module to retrieve URAT data from Vizier and reformat it.
  - Code to match RAs/Decs to URAT, as part of that new URAT module.
  - Substitute URAT PMs for GFAs where Gaia has not yet measured PMs.
- **Update CMX and Main Survey target classes [PR #525]. Includes:**
  - New `SV0_WD`, `SV0_STD_FAINT` target classes for commissioning.
  - Mild updates to `SV0_BGS` and `SV0_MWS` for commissioning.
  - New `BGS_FAINT_HIP` (high-priority BGS) Main Survey class.
  - Explicit checking on `ASTROMETRIC_PARAMS_SOLVED` for MWS targets.
  - Add 3-sigma parallax slop in `MWS_MAIN` survey target class.
- **Add `OBSCONDITIONS` to target files [PR #523] Also includes:**
  - Split target files explicitly into bright and "graydark" surveys.
  - Default to such a file-spilt for SV and Main (not for cmx).

    - Adds an informational bit for supplemental sky locations.
- **Use `MASKBITS` instead of `BRIGHTSTARINBLOB` [PR #521]. Also:**
    - Extra options and checks when making and vetting bundling scripts.
    - Option to turn off commissioning QSO cuts to speed unit tests.
- Add ELG/LRG/QSO/STD selection cuts for commissioning [PR #519].
- Add full set of columns to supplemental skies file [PR #518].
- Fix some corner cases when reading HEALPixel-split files [PR #518].

2.21 0.31.1 (2019-07-05)

- Pass Gaia astrometric excess noise in cmx MWS SV0 [PR #516].

2.22 0.31.0 (2019-06-30)

- `MASKBITS` of `BAILOUT` for randoms when no file is found [PR #515].
- Near-trivial fix for an unintended change to the isELG API introduced in PR #513 [PR #514].
- **Preliminary ELG cuts for DR8 imaging for main and SV [PR #513].**
    - Don't deprecate wider SV bits, yet, ELGs may still need them.
- **Further updates to generating randoms for DR8. [PR #512]. Includes:**
    - Add WISE depth maps to random catalogs and pixweight files.
    - **Code to generate additional supplemental randoms catalogs.**
        - Supplemental, here, means (all-sky) outside of the footprint.
    - Executable to split a random catalog into N smaller catalogs.
    - **Fixes a bug in `targets.main_cmx_or_sv()`.**
        - Secondary columns now aren't the default if rename is `True`.
    - **Better aligns data model with expected DR8 directory structure.**
        - Also fixes directory-not-found bugs when generating skies.
- **Add "supplemental" (outside-of-footprint) skies [PR #510]:**
    - Randomly populates sky area beyond some minimum Dec and Galactic b.
    - Then avoids all Gaia sources at some specified radius.
    - **Fixes a bug where `geomask.hp_in_box()` used geodesics for Dec.**
        - Dec cuts should be small circles, not geodesics.
- **First implementation for secondary targets [PR #507]. Includes:**
    - Framework and design for secondary targeting process.
    - Works automatically for both Main Survey and SV files.
    - **New bitmasks for secondaries that populate `SCND_TARGET` column.**
        - can have any `PRIORITY_INIT` and `NUMOBS_INIT`.

  - A reserved "veto" bit to categorically reject targets.
  - **Rigorous checking of file formats...**
    * ...and that files correspond to secondary bits.
  - **Example files and file structure (at NERSC) in `SCND_DIR`.**
    * /project/projectdirs/desi/target/secondary.
  - **Secondary targets are matched to primary targets on RA/Dec.**
    * unless a (per-source) `OVERRIDE` column is set to `True`.
  - Secondary-primary matches share the primary `TARGETID`.
  - **Non-matches and overrides have their own `TARGETID`.**
    * with `RELEASE == 0`.
  - **Non-override secondary targets are also matched to themselves.**
    * `TARGETID` and `SCND_TARGET` correspond for matches.

2.23 0.30.1 (2019-06-18)

- Fix normalization bug in QSO tracer/Lya mock target densities [PR #509].
- Tune "Northern" QSO selection and color shifts for Main and SV [PR #506]
- **Follow-up PR to PR #496 with two changes and bug fixes [PR #505]:**
  - Select QSO targets using random forest by default.
  - Bug fix: Correctly populate `REF_CAT` column (needed to correctly set MWS targeting bits).

2.24 0.30.0 (2019-05-30)

- Drop Gaia fields with np.rfn to fix Python 3.6/macOS bug [PR #502].
- Apply the same declination cut to the mocks as to the data [PR #501].
- **Add information to GFA files [PR #498]. Includes:**
  - Add columns `PARALLAX`, `PARALLAX_IVAR`, `REF_EPOCH`.
  - Remove `REF_EPOCH` from GFA file header, as it's now a column.
  - Sensible defaults for Gaia-only `REF_EPOCH`, `RA/DEC_IVAR`.
  - **Use fitsio.read() instead of *`desitarget.io.read_tractor()`*.**
    * It's faster and special handling of input files isn't needed.
- **General clean-up of target selection code [PR #497]. Includes:**
  - Deprecate old functions in *`desitarget.gfa`*.
  - **Greatly simplify `io.read_tractor()`.**
    * Backwards-compatability is now only guaranteed for DR6-8.
  - Guard against warnings (e.g. divide-by-zero) in cuts and SV cuts.
  - **Default to only passing North (S) sources through North (S) cuts.**

        * Retain previous behavior if `--noresolve` flag is passed.
- Add SV support to select_mock_targets [PR #496]
- **A few more updates and enhancements for DR8 [PR #494]. Includes:**
    - Add `WISEMASK_W1` and `WISEMASK_W2` to random catalogs.
    - Deprecate `BRIGHTBLOB` in favor of `MASKBITS` for targets.
    - **Add `qso_selection==colorcuts` in `set_target_bits.sv1_cuts()`**
        * This should facilitate QSO selection for SV mocks.
- Add `REF_CAT` and Gaia BP and RP mags and errors to GFAs [PR #493].
- Minor bug fix in how *select_mock_targets* handles Lya targets [PR #444].
- **Further updates and enhancements for DR8 [PR #490]. Includes:**
    - Resolve sky locations and SV targets in North/South regions.
    - Update sky and SV slurming for DR8-style input (two directories).
    - Write both of two input directories to output file headers.
    - Parallelize plot production to speed-up QA by factors of 8.
    - Add `PSFSIZE` to randoms, pixweight maps and QA plots.
    - QA and pixweight maps work fully for SV-style files and bits.
    - Pixweight code can now take HEALpixel-split targets as input.
    - Add aperture-photometered background flux to randoms catalogs.
    - Additional unit test module (`test.test_geomask()`).
    - Deprecate *make_hpx_density_file*; use *make_imaging_weight_map*.
    - `io.read_targets_in_a_box()` can now read headers.
    - Update unit test data for new DR8 columns and functionality.
- **Update QSO targeting algorithms for DR8 [PR #489]. Includes:**
    - Update baseline quasar selection for the main survey.
    - Update QSO bits and selection algorithms for SV.
- Remove GFA/Gaia duplicates on `REF_ID` not `BRICKID` [PR #488].
- **Various bug and feature fixes [PR #484]. Includes:**
    - Fix crash when using sv_select_targets with *–tcnames*.
    - Only import matplotlib where explicitly needed.
- Update *select_mock_targets* to (current) DR8 data model [PR #480].

2.25 0.29.1 (2019-03-26)

- Add `REF_CAT`, `WISEMASK_W1/W2` to DR8 data model [PR #479].
- Use speed of light from scipy [PR #478].

2.26 0.29.0 (2019-03-22)

- **Update SV selection for DR8 [PR #477]. Includes:**
  - New SV targeting bits for QSOs and LRGs.
  - New SV selection algorithms for QSOs, ELGs and LRGs.
  - MTL fixes to handle SV LRGs (which are now not 1PASS/2PASS).
  - QA can now interpret HEALPixel-split targeting files.
  - Updated test files for the quasi-DR8 imaging data model.
  - SKY and BAD_SKY added to commissioning bits yaml file.
  - Randoms in overlap regions, and for DR8 dual directory structure.
  - Write overlap regions in addition to resolve for targets/randoms.
- Change instances of *yaml.load* to *yaml.safe_load* [PR #475].
- Fix Gaia files format in doc string (healpix not healpy) [PR #474].
- Write Gaia morphologies and allow custom tilings for GFAs [PR #467].
- **Initial updates for DR8 [PR #466]. Includes:**
  - DR8 data model updates (e.g BRIGHTSTARBLOB -> bitmask BRIGHTBLOB).
  - Apply resolve capability to targets and randoms.
  - Handle BASS/MzLS and DECaLS existing in the same input directory.
- **New resolve capability for post-DR7 imaging [PR #462]. Includes:**
  - Add `RELEASE` to GFA data model to help resolve duplicates.
  - Resolve N/S duplicates by combining `RELEASE` and areal cuts.
  - Apply the new resolve code (`targets.resolve()`) to GFAs.
  - Deprecate Gaia-matching code for GFAs, as we no longer need it.
- Add code to select GFAs for cmx across wider sky areas [PR #461].

2.27 0.28.0 (2019-02-28)

- *desitarget.mock.build.targets_truth* fixes for new priority calcs [PR #460].
- **Updates to GFAs and skies for some cmx issues [PR #459]. Includes:**
  - Assign `BADSKY` using `BLOBDIST` rather than aperture fluxes.
  - Increase default density at which sky locations are generated.
  - Store only aperture fluxes that match the DESI fiber radius.
  - Ensure GFAs exist throughout the spectroscopic footprint.
- **Refactor SV/main targeting for spatial queries [PR #458]. Includes:**
  - Many new spatial query capabilities in *desitarget.geomask*.
  - Parallelize target selection by splitting across HEALPixels.
  - **Wrappers to read in HEALPix-split target files split by:**

        * HEALPixels, RA/Dec boxes, RA/Dec/radius caps, column names.
    - **Only process subsets of targets in regions of space, again including:**
        * HEALPixels, RA/Dec boxes, RA/Dec/radius caps.
    - New unit tests to check these spatial queries.
    - Updated notebook including tutorials on spatial queries.
- Update the SV selections for BGS [PR #457].
- **Update MTL to work for SV0-like cmx and SV1 tables [PR #456]. Includes:**
    - Make SUBPRIORITY a random number (0->1) in skies output.
    - New `targets.main_cmx_or_sv()` to parse flavor of survey.
    - Update `targets.calc_priority()` for SV0-like cmx and SV1 inputs.
    - `mtl.make_mtl()` can now process SV0-like cmx and SV1 inputs.
    - New unit tests for SV0-like cmx and SV1 inputs to MTL.
- Deprecate `targets.calc_priority()` that had table copy [PR #452].
- Update SV QSO selections, add seed and DUST_DIR for randoms [PR #449].
- Style changes to conform to PEP 8 [PR #446], [PR #447], [PR #448].

2.28 0.27.0 (2018-12-14)

- **Remove reliance on Legacy Surveys for Gaia data [PR #438]. Includes:**
    - Use `$GAIA_DIR` environment variable instead of passing a directory.
    - Functions to wget Gaia DR2 CSV files and convert them to FITS.
    - Function to reorganize Gaia FITS files into (NESTED) HEALPixels.
    - Use the NESTED HEALPix scheme for Gaia files throughout desitarget.
    - Change output column `TYPE` to `MORPHTYPE` for GFAs.
- Move *select-mock-targets.yaml* configuration file to an installable location for use by *desitest* [PR #436].
- Significant enhancement and refactor of *select_mock_targets* to include stellar and extragalactic contaminants [PR #427].

2.29 0.26.0 (2018-12-11)

- Refactor QSO color cuts and add hard r > 17.5 limit [PR #433].
- **Refactor of MTL and MTL-related enhancements [PR #429]. Includes:**
    - Use targets file *NUMOBS_INIT* not `targets.calc_numobs()`.
    - Use targets file *PRIORITY_INIT* not `targets.calc_priority()`.
    - Remove table copies from *desitarget.mtl* to use less memory.
    - New function `targets.calc_priority_no_table()` to use less memory.
    - Set informational (*NORTH/SOUTH*) bits to 0 *PRIORITY* and *NUMOBS*.

  - Set priorities using *LRG_1PASS/2PASS* bits rather than on *LRG*.
- **Minor updates to *select_mock_targets* [PR #425].**
  - Use pre-computed template photometry (requires *v3.1* basis templates).
  - Include MW dust extinction in the spectra.
  - Randomly assign a radial velocity to superfaint mock targets.
- Update default mock catalogs used by *select_mock_targets* [PR #424]
- Update Random Forests for DR7 quasar selection [PR #423]
- Fix bugs in main MWS selections [PR #422].
- Fix *python setup.py install* for cmx and sv1 directories [PR #421].
- **More updates to target classes, mainly for SV [PR #418]. Includes:**
  - First full implementations of *QSO*, *LRG*, *ELG*, and *STD* for SV.
  - Update and refactor of *MWS* and *BGS* classes for the main survey.
  - Change name of main survey *MWS_MAIN* class to *MWS_BROAD*.
  - Augment QA code to handle SV sub-classes such as *ELG_FDR_FAINT*.

2.30 0.25.0 (2018-11-07)

- Randomize mock ordering for Dark Sky mocks which aren't random [PR #416].
- **Updates to several target classes [PR #408]. Includes:**
  - Refactor of the *ELG* and *MWS_MAIN* selection algorithms.
  - Update of the *ELG* and *MWS_MAIN* selection cuts.
  - Change *MWS_WD* priority to be higher than that of *BGS* target classes.
  - Set skies to *BAD* only if both g-band and r-band are missing.
- Refactor of BGS selections to separate masking and color cuts [PR #407].
- Quicksurvey MTL fix [PR #405].
- Mocks use QSO color cuts instead of random forest [PR #403].
- **Updates to Bright Galaxy Survey and QSO selections [PR #402]. Includes:**
  - Updates to *BGS_FAINT* and *BGS_BRIGHT* target selections.
  - New *BGS_WISE* selection and implementation.
  - New data model columns *BRIGHTSTARINBLOB* and *FRACIN_*.
  - Add cut on *BRIGHTSTARINBLOB* to QSO selection.
  - Modify I/O to retain (some) backwards-compatibility between DR6 and DR7.
  - Updated unit test example files with appropriate columns.
  - Speed-up of *cuts* unit tests without loss of coverage.
- Updated mock sky catalog with positions over a larger footprint [PR #398].
- Major update to *select_mock_targets* to use the latest (v3.0) basis templates [PR #395].

- Propagate per-class truth HDUs into final merged truth file [PR #393].
- Incorporate simple WISE depth model in *select_mock_targets* which depends on ecliptic latitude [PR #391].

2.31 0.24.0 (2018-09-26)

- Fix bug in code that produces data for unit tests [PR #387].
- Rescale spectral parameters when generating and querying kd-trees in *select_mock_targets* [PR #386].
- **Bug fixes: [PR #383].**
  - Use *parallax_err* when selecting *MWS_NEARBY* targets.
  - In *select_mock_targets* do not use Galaxia to select WDs and 100pc targets.
- Refactor QA to work with commissioning and SV files and add (first) unit tests for QA. [PR #382].
- Estimate FIBERFLUX_[G,R,Z] for mock targets. [PR #381].
- **First fully working version of SV code [PR #380]. Includes:**
  - (Almost) the only evolving part of the code for SV is now the cuts.
  - Unit tests for SV that should be easy to maintain.
  - Bit and column setting for SV that should be maintainable.
  - SV0 (commissioning) MWS cuts.
  - Updated STD cuts to fix a *fracmasked* typo.
  - Alterations to Travis coverage to exclude some external code.
- Fix a bug which resulted in far too few standard stars being selected in the mocks [PR #378].
- Fix a bug in how the *objtruth* tables are written out to by *select_mock_targets* [PR #374].
- Remove Python 2.7 from Travis, add an allowed-to-fail PEP 8 check [PR #373].
- Function to read `RA`, `DEC` from non-standard external files [PR #372].
- **Update the data model for output target files [PR #372]:**
  - Change `TYPE` to `MORPHTYPE`.
  - Add `EBV`, `FIBERFLUX_G,R,Z` and `FIBERTOTFLUX_G,R,Z`.
- **Additional commissioning (cmx) classes and priorities [PR #370]. Includes:**
  - New functions to define several more commissioning classes.
  - A `$CMX_DIR` to contain files of cmx sources to which to match.
  - An example `$CMX_DIR` is `/project/projectdirs/desi/target/cmx_files`.
  - Functionality to reset initial priorities for commissioning targets.
  - Downloading fitsio using pip/astropy to fix Travis.
- Significant enhancement of *select_mock_targets* (see PR for details) [PR #368].
- Include per-band number counts for targets on the QA pages [PR #367].
- Use new `desiutil.dust.SFDMap()` module [PR #366].
- Set the `STD_WD` bit (it's identical to the `MWS_WD` bit) [PR #364].

- Add notebook for generating Gaussian mixture models from DR7 photometry and morphologies of ELG, LRG, and BGS targets [PR #363 and PR #365].
- **Make commissioning (cmx) target selection fully functional [PR #359]. Includes:**
  - Initial target selection algorithms.
  - First unit tests for cmx (> 90% coverage).
  - `SV_TARGET` and `CMX_TARGET` as output columns instead of as a bit.
- **Remove "legacy" code in QA [PR #359].**
  - Weight maps can now be made with `desitarget.randoms.pixmap()`.
- Add isELG_colors functions [PR #357].
- Adapt cuts.isSTD_colors to deal with different north/south color-cuts [PR #355].
- **Refactor to allow separate commissioning and SV target selections [PR #346]:**
  - Added `sv` and `commissioning` directories.
  - **New infrastructure to have different cuts for SV and commissioning:**
    - separate target masks (e.g. `sv/data/sv_targetmask.yaml`).
    - separate cuts modules (e.g. `sv_cuts.py`).
  - Added executables for SV/commissioning (e.g. `select_sv_targets`).
  - Initial `NUMOBS` and `PRIORITY` added as columns in `targets-` files.
  - Initial `NUMOBS` is now hardcoded in target masks, instead of being set by MTL.
  - `SV` bits added to target masks to track if targets are from SV/comm/main.
  - sv/comm/main can now be written to the header of the `targets-` files.
  - `SUBPRIORITY` is set when writing targets to facilitate reproducibility.
- Set `NUMOBS` for LRGs in MTL using target bits instead of magnitude [PR #345].
- **Update GFA targets [PR #342]:**
  - Handle reading Gaia from sweeps as well as matching. Default to *not* matching.
  - Makes Gaia matching radius stricter to return only the best Gaia objects.
  - Retains Gaia RA/Dec when matching, instead of RA/Dec from sweeps.
  - Fixes a bug where Gaia objects in some HEALPixels weren't being read.
  - Add Gaia epoch to the GFA file header (still needs passed from the sweeps).

2.32 0.23.0 (2018-08-09)

Includes non-backwards compatible changes to standard star bit names.

- STD/STD_FSTAR -> STD_FAINT, with corresponding fixes for mocks [PR #341].
- **Match sweeps to Gaia and write new sweeps with Gaia columns [PR #340]:**
  - Also add `BRIGHTSTARINBLOB` (if available) to target output files.
  - And include a flag to call STD star cuts function without Gaia columns.

2.33 0.22.0 (2018-08-03)

Includes non-backwards compatible changes to standard star target mask bit names and selection function names.

- **Produce current sets of target bits for DR7 [PR #338]:**
  - Update the `LRG`, `QSO`, `STD` and `MWS` algorithms to align with the wiki.
  - In particular, major updates to the `STD` and `MWS` selections.
  - Don't match to Gaia by default, only if requested.
  - Maintain capability to match to Gaia if needed for earlier Data Releases.
  - Run subsets of target classes by passing, e.g.. `--tcnames STD,QSO`.
  - Update unit test files to not rely on Gaia.
  - Bring Data Model into agreement with Legacy Surveys sweeps files.
  - Rename `FSTD` to be `STD` throughout.
  - QA fails gracefully if weight maps for systematics aren't passed.

2.34 0.21.1 (2018-07-26)

- **Update the schema for target selection QA [PR #334]:**
  - Sample imaging pixels from the Legacy Surveys to make random catalogs.
  - Add E(B-V) from SFD maps and stellar densities from Gaia to the randoms.
  - Sample randoms to make HEALpixel maps of systematics and target densities.
  - Sample randoms in HEALPixels to precisely estimate imaging footprint areas.
  - Make several new systematics plots.
  - Make new plots of parallax and proper motion information from Gaia.

2.35 0.21.0 (2018-07-18)

- Fix bug when generating targeting QA for mock catalogs [PR #332].
- Add support for GAMA/BGS mocks and new calib_only option in *mock.targets_truth* [PR #331].
- Add `RA_IVAR` and `DEC_IVAR` to the GFA Data Model [PR #329].
- **Update the Gaia Data Model [PR #327]:**
  - Output columns formatted as expected downstream for GFA assignment.
  - Align Gaia Data Model in matching and I/O with the Legacy Surveys.
- Allow environment variables in select_mock_targets config file [PR #325].
- **First version of Milky Way Survey selection [PR #324]:**
  - Catalog-matches to Gaia using `desitarget.gaimatch`.
  - Sets MWS_MAIN, MWS_WD and MWS_NEARBY bits.
  - Makes individual QA pages for MWS (and other) bits.

- **Change GFA selection to be Gaia-based [PR #322]:**
  - Update the `select_gfas` binary to draw from Gaia DR2.
  - Parallelize across sweeps files to add fluxes from the Legacy Surveys.
  - Gather all Gaia objects to some magnitude limit in the sweeps areas.
- **Add `desitarget.gaimatch` for matching to Gaia [PR #322]:**
  - Can perform object-to-object matching between Gaia and the sweeps.
  - Can, in addition, retain all Gaia objects in an RA/Dec box.
- Mock targets bug fixes [PR #318].
- Add missing GMM files to installations [PR #316].
- **Introduction of pixel-level creation of sky locations [PR #313]:**
  - Significant update of *desitarget.skyfibers*
  - `desitarget.skyutilities.astrometry` to remove `astrometry.net` dependency.
  - `desitarget.skyutilities.legacypipe` to remove `legacypipe` dependency.
  - Grids sky locations by applying a binary erosion to imaging blob maps.
  - Sinks apertures at the resulting sky locations to derive flux estimates.
  - Sets the `BAD_SKY` bit using high flux levels in those apertures.
  - `desitarget.skyfibers.bundle_bricks()` to write a slurm script.
  - Parallelizes via HEALPixels to run in a few hours on interactive nodes.
  - Adds the `select_skies` binary to run from the command line.
  - Includes `gather_skies` binary to collect results from parallelization.
  - Adds functionality to plot good/bad skies against Legacy Survey images.
- select_mock_targets full footprint updates [PR #312].
- QA fix for testing without healpix weight map [PR #311].
- New QSO random forest [PR #309].
- Restore the no-spectra option of select_mock_targets, for use with quicksurvey [PR #307].
- **Better handling of imaging survey areas for QA [PR #304]:**
  - `desitarget.imagefootprint` to build HEALPix weight maps of imaging.
  - Executable (bin) interface to make weight maps from the command line.
  - *desitarget.io* loader to resample maps to any HEALPix *nside*.
  - Update *desitarget.QA* to handle new imaging area weight maps.
- Improve north/south split functions for LRG and QSO color cuts [PR #302].
- **Minor QA and selection cuts updates [PR #297]:**
  - QA matrix of target densities selected in multiple classes.
  - Functions to allow different north/south selections for LRGs.

2.36 0.20.1 (2018-03-29)

- Add a bright (g>21) flux cut for ELGs. [PR #296].

2.37 0.20.0 (2018-03-24)

- Added compare_target_qa script [PR #289].
- Astropy 2.x compatibility [PR #291].
- **Update of sky selection code [PR #290]. Includes:**
  - Use the *desitarget.brightmask* formalism to speed calculations.
  - Pass around a magnitude limit on masks from the sweeps (to better avoid only objects that are genuinely detected in the sweeps).
  - Reduce the default margin to produce ~1700 sky positions per sq. deg.
- **Retuning of DR6 target densities [PR #294]. Includes:**
  - Tweaking the QSO random forest probability.
  - Adding a new ELG selection for the northern (MzLS/BASS) imaging.
  - Slight flux shifts to reconcile the northern and southern (DECaLS) imaging.
  - Initial functionality for different North/South selections.
- **Some reformatting of output target files and bits [PR #294]:**
  - Introducing a `NO_TARGET` bit.
  - Renaming the `BADSKY` bit `BAD_SKY` for consistency with other bits.
  - Including `FRACDEV` and `FRACDEV_IVAR` as outputs.

2.38 0.19.1 (2018-03-01)

- Fix bug whereby FLUX and WAVE weren't being written to truth.fits files [PR #287].
- Include OBSCONDITIONS in mock sky/stdstar files for fiberassign [PR #288].

2.39 0.19.0 (2018-02-27)

This release includes significant non-backwards compatible changes to importing target mask bits and how mock spectra are generated.

- Major refactor of select_mock_targets code infrastructure [PR #264].
- Restructure desi_mask, bgs_mask, etc. imports to fix readthedocs build [PR #282].
- Update RELEASE dictionary with 6000 (northern) for DR6 [PR #281].

2.40 0.18.1 (2018-02-23)

- Open BGS hdf5 mocks read-only to fix parallelism bug [PR #278].

2.41 0.18.0 (2018-02-23)

- New target density fluctuations model based on DR5 healpixel info [PR #254].
- Include (initial) mock QA plots on targeting QA page [PR #262]
- Added *select_gfa* script [PR #275]
- Update masking for ellipses ("galaxies") in addition to circles ("stars") [PR #277].

2.42 0.17.1 (2017-12-20)

- HPXNSIDE and HPXPIXEL as header keywords for mocks too [PR #246].

2.43 0.17.0 (2017-12-20)

- Support LyA skewers v2.x format [PR #244].
- Split LRGs into PASS1/PASS2 separate bits [PR #245].
- Sky locations infrastructure [PR #248].
- Mock targets densities fixes [PR #241 and PR #242].

2.44 0.16.2 (2017-11-16)

- Allows different star-galaxy separations for quasar targets for different release numbers [PR #239].

2.45 0.16.1 (2017-11-09)

- fixes to allow QA to work with mock data [PR #235].
- cleanup of mpi_select_mock_targets [PR #235].
- adds BGS properties notebook documentation [PR #236].

2.46 0.16.0 (2017-11-01)

- **General clean-up prior to running DR5 targets [PR #229].**
  - Use `desiutil.log` instead of verbose (everywhere except mocks)
  - Change `HEALPix` references to header keywords instead of dependencies
  - Include `SUBPRIORITY` and shape parameter `IVARs` in target outputs

- Include GMM model data for mocks when installing [PR #222].
- Initial simplistic code for generating sky positions [PR #220]

2.47 0.15.0 (2017-09-29)

- Refactored *desitarget.QA* to calculate density fluctuations in HEALPixels instead of in bricks [PR #217]:
- Updated *desitarget.io* for the DR5 RELEASE number [PR #214]:
- **Updated *desitarget.QA* to produce QA plots [PR #210]:**
  - Has a simple binary that runs the plot-making software in full
  - Creates (weighted) 1-D and 2-D density plots
  - Makes color-color plots
  - Produces a simple .html page that wraps the plots, e.g. http://portal.nersc.gov/project/desi/users/adamyers/desitargetQA/
- **Changes for mocks [PR #200]:**
  - Fix isLRG vs. isLRG_colors
  - Correct random seeds when processing pix in parallel
  - Misc other small bug fixes
- Added `mpi_select_mock_targets`
- **Changes for mocks [*PR #228*]:**
  - Refactor of `targets_truth_no_spectra`
  - Solves bug of healpix patterns present in target mocks.
  - Removes current implementation for target fluctuations.
- Added `desitarget.mock.sky.random_sky` [PR #219]

2.48 0.14.0 (2017-07-10)

- **Significant update to handle transition from pre-DR4 to post-DR4 data model [PR #189]:**
  - *desitarget.io* can now read old DR3-style and new DR4-style tractor and sweeps files
  - *desitarget.cuts* now always uses DR4-style column names and formats
  - new 60-bit `TARGETID` schema that incorporates `RELEASE` column from imaging surveys
  - `desitarget.brightstar` builds masks on DR4-style data, uses `RELEASE` to set DR
  - HEALPix pixel number (current nside=64) added to output target files
  - `select_targets` passes around information related to `HEALPix`
  - column `PHOTSYS` added to output files, recording North or South for the photometric system
  - unit tests that explicitly used columns and formats from the data model have been updated

2.49 0.13.0 (2017-06-15)

- Fix bug when no Lya QSOs are on a brick [PR #191].
- Additional QA plots for mock target catalogs [PR #190]
- Additional debugging and support for healpix input to `select_mock_targets` [PR #186].
- Set specific DONE, OBS, and DONOTOBSERVE priorities [PR #184].

2.50 0.12.0 (2017-06-05)

- Changed refs to `desispec.brick` to its new location at `desiutil.brick` [PR #182].
- Fix ELG and stdstar mock densities; add mock QA [PR #181].
- Updated LRG cuts significantly to match proposed shift in LRG target density [PR #179].
- **Major expansion of bright object masking functionality (for circular masks) [PR #176]:**
  - Generate SAFE/BADSKY locations around mask perimeters
  - Set the target bits (including TARGETID) for SAFE/BADSKY sky locations
  - Set a NEAR_RADIUS warning for objects close to (but not in) a mask
  - Plot more realistic mask shapes by using ellipses
- **Significant expansion of the mocks-to-targets code [PR #173 and PR #177]:**
  - Better and more graceful error handling.
  - Now includes contaminants.
  - Much better memory usage.
  - Updated QA notebook.
- Add Random Forest selection for ELG in the sandbox [PR #174].
- Fix ELG and stdstar mock densities; add mock QA [PR #181].

2.51 0.11.0 (2017-04-14)

- New cuts for standards [PR #167]
- Ensured objtype was being passed to `isFSTD()`.
- Added mock -> targets+spectra infrastructure

2.52 0.10.0 (2017-03-27)

- Update Travis configuration to catch documentation errors.
- WIP: refactor of mock.build
- added mock.spectra module to connect mock targets with spectra
- fix overflow in LRG sandbox cuts [PR #160]

- fixed many documentation syntax errors

2.53 0.9.0 (2017-03-03)

- Include mapping from MOCKID -> TARGETID.
- Added shapes to gaussian mixture model of target params [PR #150].
- Added basic bright star masking.
- Updates for mock targets.
- Added `desitarget.sandbox.cuts` area for experimental work.
- Add ELG XD and new LRG to sandbox.

2.54 0.8.2 (2016-12-03)

- Updates for mocks integrated with quicksurvey.

2.55 0.8.1 (2016-11-23)

- Fix *`select_targets()`* and `gitversion()` for Python 3.

2.56 0.8.0 (2016-11-23)

- Adds DESI_TARGET bits for bright object masking.
- MTL sets priority=-1 for any target with IN_BRIGHT_OBJECT set.
- Many updates for reading and manipulating mock targets.
- Adds BGS_FAINT target selection.

2.57 0.7.0 (2016-10-12)

- Added functionality for Random Forest into quasar selection.
- Updates to be compatible with Python 3.5.
- Refactor of merged target list (mtl) code.
- Update template module file to DESI+Anaconda standard.

2.58 0.6.1 (2016-08-18

- PR #59: fix LRG selection (z < 20.46 not 22.46).

2.59 0.6.0 (2016-08-17)

- Big upgrade for how Tractor Catalogues are loaded to DB. Only the mapping between Catalogue and DB naming is hardcoded. Compatible DR2.
- Python parallelism. Can choose mulprocessing OR mpi4py.
- Unit test script that compares random rows from random Catalogues with what is in the DB.

2.60 0.5.0 (2016-08-16)

- Added obscondition and truesubtype to mocks (PR #55; JFR).
- refactored cut functions to take all fluxes so that they have same call signature (PR #56; JM).
- Move data into Python package to aid pip installs (PR #47; BAW).
- Support for Travis, Coveralls and ReadTheDocs (BAW).

2.61 0.4.0 (2016-07-12)

- Updated code from DECaLS DR1 to load DR2 tractor catalaogues to psql db.
- Basic unit test script for checking that db rows match tractor catalogues.

2.62 0.3.3 (2016-03-08)

- Added *isMWSSTAR_colors()*.
- Allow user to specify columns when reading tractor files.
- New code for generating merged target list (MTL).
- Removed unused npyquery code.

2.63 0.3.2 (2016-02-15)

- Add this changes.rst; fix _version.py.

2.64 0.3.1 (2016-02-14)

- PR #30: isolated *desitarget.io* imports in *desitarget.cuts*.
- _version.py is wrong in this tag.

2.65 0.3 (2016-02-14)

- PR #29 and PR #27 refactor `desitarget.cuts` to include per-class functions.
- Other changes in git log before (this changes.rst didn't exist yet).
- _version.py is wrong in this tag.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

P

Q

R

S

T

U

W