Balrog

An astronomical imaging simulation d(a)emon[1]

for those who dig too deeply

and too greedily into

their data...

Eric Suchyta and Eric Huff

[1]Technically, the package is not a daemon. Please forgive our attempts at naming humor.

Contents

1. Introduction

BALROG is a package of Python code, intended for use with astronomical imaging data. Strictly speaking, BALROG is a simulation tool. However, its ambition is derived from the aspiration to better characterize and understand real data. By performing a set of simulations, BALROG's intent is to allow observers to infer properties of their images by directly testing on the images themselves.

The core functionality driving BALROG's design is rather straightforward. Galaxies are simulated, trivially writing their simulated properties to a *truth catalog*. Noisy images of the galaxies are then inserted into real data. Source detection software runs over the image, whose measurements for the simulated galaxies can be directly compared to the truth catalog. Accordingly, one is able to answer questions about how measured properties of the image are related to the true properties.

Instead of reinventing the wheel, the BALROG pipeline wraps around existing codes, well known within the Astronomy community. All galaxy simulations are implemented via GalSim[1] (Rowe et al., *in prep.*) and source extraction and measurement occurs using SEXTRACTOR[2] (Bertin & Arnouts, 1996). BALROG facilitates the ease of running these codes en masse over many images, filling in many of the bookkeeping steps in an automated fashion.

Since different users will have different needs, BALROG strives to be as flexible as possible. It includes a well defined framework capable of implementing a wide variety of simulation possibilites. The framework allows users to define their own arguments and functions to plug into BALROG when generating simulated galaxies.

In order to maximize convenience, BALROG has been written making our best attempts at user-friendliness. Example files are packaged with the code so that following installation, the pipeline is able to run out of the box without specifying any arguments. Users are encouraged to use and inspect the default example runs to become more familiar with the BALROG environment. To preserve an intuitive feel, BALROG's simulation framework mimics ordinary Python syntax. Where necessary, files and directories are given understandable names. Numerous errors and warning are handled, printing useful messages about why the exception was raised. Log files are automatically written, useful for follow-up degugging in the cases where exceptions do occur. User configurations are copied into the output directory, owing to the consideration that every run should be reproducible from the output.

In this brief introduction, we have merely scratched the surface explaining BALROG's uses and capabilites. The remainder of the documentation elaborates further. Section 2 enumerates the components of the BALROG pipeline, illustrating its algorithm. Until now, we have been rather vague about practical applications for BALROG. Section 3 addresses just that, offering some examples of what can be done with BALROG outputs. Section 4 discusses what is necessary for installation. Beyond, the remainder of the document concerns how to

[1] https://github.com/GalSim-developers/GalSim
[2] https://www.astromatic.net/software/sextractor

configure and run BALROG. Section 5 presents an approach to hit the ground running and quickly get started with some key features of BALROG. Section 5 is followed by a number of more comprehensive sections, which spell out all the usage details. Section 6 covers command line arguments, and Section 7 establishes how to simulate galaxies within the BALROG framework. The format of BALROG's outputs are explained in Section 8. Section 9 offers some debugging hints.

2. The BALROG Pipeline

Section 1 briefly introduced the workflow through the BALROG pipeline. The purpose of this section is to characterize the algorithm in full. The focus here is methodology, not usage instructions. The text is organized as follows. To begin, the workflow of the pipeline as a whole is described in Section 2.1. This is then subdivided into three sections to be expanded further, with links to the extended sections inside Section 2.1. In Section 2.2, BALROG's required input data is discussed. In Section 2.3, properties of the simulated galaxies are described, with a brief conceptual overview of how the properties can be generated and comments regarding the steps implemented in GalSim. Finally, BALROG's image source detection and measurement is considered in Section 2.4.

2.1 Workflow

To open this section, we preseent Figure 2.1 as a complement to the text, and as a guide for what is to come. The figure is a flowchart visually representing BALROG. Roughly speaking, the text steps throgh this flowchart from left to right. However, as the lines in Figure 2.1 indicate, BALROG runs trace out more than a single horizontal line through the diagram. In order to maintain visually clarity and simplicity, Figure 2.1 does not include every possible detail of the pipeline. Rather, it lays out the structure of how the various steps depend on each other. Effectively, there are two requirements: a set of simulation configurations and some imaging data. The pipeline's central components are GalSim and SEXTRACTOR, which operate on these requirements to produce output catalogs for further analysis. The following paragraphs comment further.

The BALROG pipeline begins by opening a log file and parsing the command line options. The code enforces a number of rules on the user's configurations. If any errors or warnings occur they are directed to the log file. Errors are raised when users make a syntax error and BALROG cannot continue. Generally, warnings occur when something is missing, but the code is able to continue using an internal default. Warnings likely, but not always indicate something is not quite right. The log file remains open, recording messages through the full pipeline.

Next BALROG interprets how the user wants to generate their simulated galaxies. Section 2.3 fully prescribes the attributes of these galaxies. To summarize, each galaxy is composed of one or more Sérsic profiles. The user's directives are executed, and a truth catalog of the galaxy parameters is written.

The code reads in the imaging data into which the simulated galaxies will be drawn. Refer to Section 2.2 to define what is required of the imaging data in this context. By default, SEXTRACTOR is run over the input image. Please note, at this point no simulated galaxies have been inserted. However, this command is potentially useful for reasons discussed further in Section 2.4. An image of each galaxy, commonly known as a postage stamp, is generated and then added atop the input image. Galaxy postage stamp drawing is treated

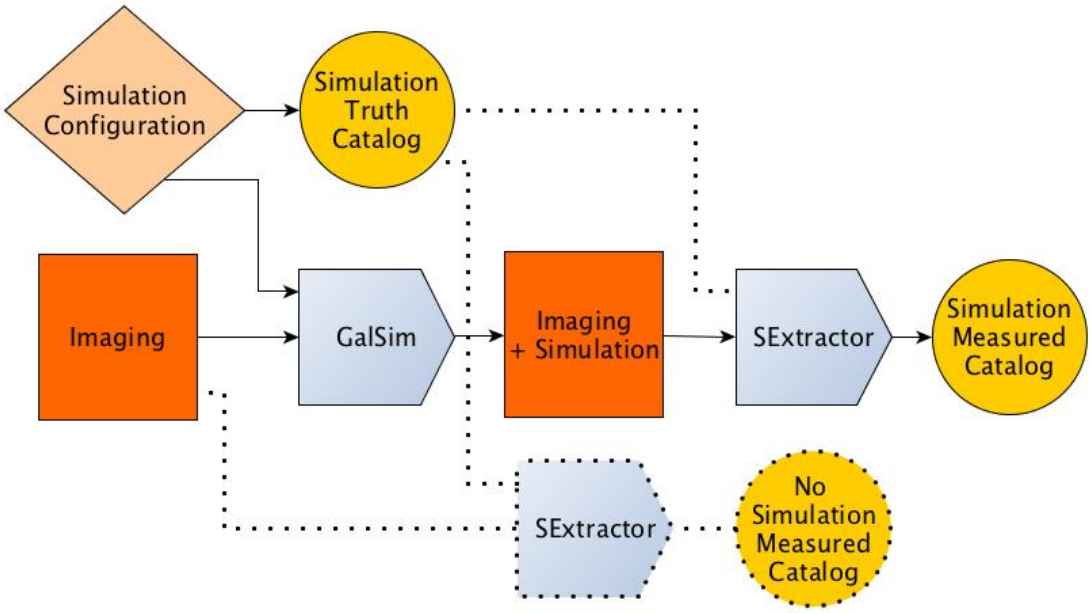

Figure 2.1: Visualization of BALROG's data flow. Optional (but also default) configurations are represented as dotted connections.

in Section 2.3. Once galaxies have been added to the image, SEXTRACTOR is called again. Details of SEXTRACTOR's implementation are deferred to Section 2.4. SEXTRACTOR outputs a catalog of the object measurements, which is now ready for the user to compare with the trutch catalog.

2.2 Input Data

BALROG reads in an astronomical image of pixellated flux values. The data should include a photometric calibration defining what one ADU count means physically. Galaxy simulations are done in apparent magnitude space, and BALROG needs a zeropoint, m_z, to convert this to an image ADU level. m_z is defined in the usual way, where the objects flux, $\mathcal{F}$, and apparent magnitude, m, are related by: $m - m_z = -2.5 \log_{10}(\mathcal{F})$. A default zeropoint of 30 is assumed if one is not otherwise specified. Required with each image is the image's weight map and PSF model. The weight map will be needed when extracting the sources from the image. At a minimum, the PSF model is necessary for convolving the simulated galaxies prior to embedding them in the given image. It is also obligatory if attempting to fit models to deconvolved measurements of galaxy profiles during the source measurement process. The only file type BALROG supports for input PSF models is that generated by PSFEX[1] (Bertin, 2011). Readers should be aware that BALROG itself does not run PSFEX. Thus, generating PSFEX models constitues a prerequisite users must complete prior to running BALROG. BALROG simulations incorporate GalSim's WCS features, and hence the software requires each image contain sa WCS solution. BALROG reads this solution from

[1] www.astromatic.net/software/psfex

the image's header. If no WCS exists in the header, the pipeline enforces a fiducial WCS with a constant pixel scale of 0.263 arcsec/pixel.[2] BALROG supports subsampling the input images if desired, meaning galaxies will only be inserted into a portion of the image. Once galaxies are simulated, the input image's flux values change by adding the galaxies on top of the original images. Algorithmically, this is the only change applied to the input by the entire BALROG pipeline. The weight map and PSF remain unchanged. Conceptually, the bulk of the undertaking goes into simulationg the galaxies.

Is it worth adding some kind of support to generate the PSF model? This introduces some issues doing so, but it might be worthwhile.

2.3 Simulated Galaxies

To simulate galaxies, BALROG makes use of GalSim. It calls GalSim's class for implementing Sérsic objects to model the galaxies' light distriubtions as Sérsic profiles. By effectively adding together these Sérsic objects, BALROG allows galaxies to be composed of as many superimposed Sérsic components as desired. Each of these components has its own Sérsic index, half light radius, flux, minor to major axis ratio, and orientation angle. The half light radius is measured along the major axis, and the orienation angle is measured as the major axis' counter-clockwise rotation away from the x-direction. Flux values are generated as magnitudes, then converted to ADU levels using the image's zeropoint. In addtion to its Sérsic components, each galaxy shares five parameters which are identical among each Sérsic component: two centroid coordinate positions (x, y), two components of lensing reduced shear (g_1, g_2), and magnfication. The reduced shear follows the usual lensing notation convention, with positive g_1 along the x-axis, negative g_1 along the y-axis, and positive and negative g_2 rotated 45° from the respective g_1 counterparts. Magnification is the usual $\mu = 1 + \kappa$. To be explictly clear, the shear and magnification are lensing effects applied to a galaxy, and the axis ratio and orientation angle are intrinsic to the galaxy, as they would in the abscence of lensing.

BALROG presents users with a number of different options for how to generate the truth properties of their galaxies. Simple types include a constant to be applied commonly to every galaxy or an array containing an element for each galaxy. Alternatively, values can be sampled from the columns in a catalog file. Multiple columns from the same table are automatically jointly sampled. Last but certainly not least, users can define their own functions which determine the truth paramters of the simulated galaxies. This is perhaps BALROG's most powerful feature. It is this functionality which adds the flexibility for BALROG to support virtually anything users can think of and code up in Python themselves. Conveniently, the functions may operate over the galaxies' truth parameters themselves, allowing one parmeter to be defined in terms of another.

The simulation process runs in a loop with a number of iterations equal to the number of galaxies to be simulated. A subloop then iterates over the number of Sérsic profiles the galaxies are composed of. For each profile, a Sérsic object is obstantiated with the Sérsic index, half light radius, and flux which were generated. Initially these objects are circularly

[2] 0.263 arcsec/pixel is the fiducial pixel scale fore DECam.

symmetric so the shearing method is called to create the galaxy's axis ratio, oriented in the appropriate direction. All these elliptical objects inside the subloop are added together to create the composite galaxy made of superimposed profiles. Lensing is then applied to the combined object, implementing both the lensing reduced shear and magnfication. The PSF model is sampled at the galaxy's centroid position and convolved with the galaxy profile. Finally, the object is convolved with a top-hat pixel response function whose scale is equal to that of the local WCS pixel scale.

The galaxy object is then drawn into a postage stamp image. The pixel scale of the postage stamp is fixed to have the scale of the local WCS at the galaxy's position. The size of the postage stamp is computed automatically by GalSim by dictating that only a certain small enough fraction of the galaxy's flux is permitted to be lost outside the postage stamp's boundaries. Additional arguments are also relevant to the final accuracy of the drawn galaxies, but they go beyond the scope of this section. See Section 7 or the galsim.GSParams documentation for details. Noise is added to the galaxy's postage stamp by GalSim's functions for creating CCD-like noise. The functionality relies on specifying the gain to set the noise level. This gain defaults to unity when nothing else if available. BALROG sets the read noise to zero. The final step in the simulation process is to add the postage stamp to input image, centering the postage stamp onto the galaxy's generated (x, y) centroid coordinates.

2.4 Source Detection and Measurement

BALROG calls SEXTRACTOR at least once, and by default twice throughtout the pipeline. While this manual is not intended to be a comprehensive guide to running SEXTRACTOR, a few comments are in order. For those who have never used SEXTRACTOR, we direct you to the official SEXTRACTOR user manual or alternatively the so-called Source Extractor for Dummies text. We only make remarks about a few of the most pertinent features to be aware of within BALROG.

SEXTRACTOR runs are configured via roughly a handful of files. In this scope, the two most relevant ones are what we will call the *config file*, often denoted with a `.sex` extension, and the *param file*, often denoted with a `.param` extension. In BALROG's default files shipped with the software, the SEXTRACTOR config file is suffixed with a `.config` extension for consistency with our terminology. The config file is allowed to specify any of the arguments which can also be given as command line arguments. These set conditions such as the detection thresholds, the aperture sizes for photometry, the magnitude zeropoint, and the background subtraction strategy. The param file is a list of keywords. Each keyword is a measurment SEXTRACTOR will make for every extracted object.

It is the param file which controls whether or not SEXTRACTOR will perform model fits to the galaxies. How do to this is rather scarely documented, but has been passed down by word of mouth in the Astronomy community. We will make a small aside to elaborate more thoroughly. SEXTRACTOR includes up to two possible types of models to fit. One, denoted by the key `DISK`, is a model with fixed Sérsic index of $n = 1$, i.e. an exponential. The other, denoted by the key `SPHEROID`, is a model with a free Sérsic index. SEXTRACTOR can fit either of these independently or both simultaneously. Each model written into and

uncommented from the param file is fit, meaning if just the `DISK` key is present a disk only model with $n = 1$ is fit; if just the `SPHEROID` key is present a bulge only model with free n is fit; and if both `DISK` and `SPHEROID` keys are present both the disk and bulge are fit simultaneously, which is of course different than fitting them independently. Each model can fit for the flux, magnitude, axis ratio, and orientation angle of the model. The spheroid model also fits the sersic index and half light radius. The disk model fits the scale radius, as opposed to the half light radius. See the `bulge+disk.param` file included with BALROG for the SEXTRACTOR names of all the paramters. The meanings of the names are human readable.

Returning to BALROG, the default behavior is to run SEXTRACTOR in *association mode*. For those unfamilar with the terminology, association mode means SEXTRACTOR will only make possible measurements at a predefined list of coordinates. Here, the list is given as the simulated galaxies' locations. Hence, SEXTRACTOR is effectively aware of where in the image the simulated galaxies live. If it finds an object at one of the these coordinates, it extracts it; but it does not extract objects anwhere else. This bypasses the need to extract every single object in the image. Namely, it selects against the objects unrelated to the simulated galaxies that existed in the image prior to any simulation, which have no truth information to compare with anyway. The attractiveness of running in association mode is the reduction in execution time it offers. In order to avoid simulated galaxies overlapping each other, simulated galaxies should be inserted into the image at a significantly lower number density than the number density of the image itself. This means association mode would remove the majority of all objects. By default BALROG configures SEXTRACTOR to make a Sérsic fit to each simulated galaxy. Because model fitting is a time expnsive step, association mode offers a significant payoff when using this configuration. For a large dataset, running SEXTRACTOR many times over many images, model fitting each object in every run is likely time infeasible.

In addition to running SEXTRACTOR over the image with the simulated galaxies, by default BALROG is configured to run SEXTRACTOR over the image prior to inserting the simulated galaxies. This is to confirm if the association mode functionality of SEXTRACTOR is genuinely measuring a simulating galaxy. For example, BALROG may happen to place a galaxy into the image a location where a big, bright object already lives. If the simulated galaxy is rather faint, SEXTRACTOR may just interpret its flux as part of the original object, having no idea the simulated galaxy is even there. The SEXTRACTOR run over the pre-simulation image allows users to check for such occurences if they would like. By default, this calls a param file which does not include model fitting. For the case described here, there is no benefit to dedicate the additional time needed for model fitting.

3. What Is Balrog Good For?

What is Balrog good for?

4. Installation

BALROG consists of two depencies: GalSim and SEXTRACTOR. BALROG itself is written entirely in Python and will run out of the box if *sufficiently recent* versions of GalSim and SEXTRACTOR have already been installed. As of the writing of this documentation,[1] BALROG requires the latest version of GalSim, which is version `1.1`. We have not thoroughly tested what constitues a *recent enough* version of SEXTRACTOR. What we are able to say is that the oldest version we have successfully tested against is `2.17.0`. BALROG and GalSim both exist as repositories on GitHub and can be downloaded from the command line.

```
% git clone https://github.com/emhuff/Balrog.git
% git clone https://github.com/GalSim-developers/GalSim.git
```

This should automatically check out the master branch of each. Master is the only supported branch for BALROG. The master branch of GalSim will always be a version $\geq$ `1.1`. SEXTRACTOR is packaged as a tarball, which can be downloaded from its official website. We recommend choosing the most recent version.

The GalSim documentation includes an extensive installation guide, with many helpful hints and links to `FAQ` pages. We refer BALROG users to this guide for how to install GalSim. The GalSim team deserves a big thanks for the utility of their installation notes. The BALROG authors are far from accomplished system adminstrators, but we both were independently successful installing GalSim on our systems by following these notes.

Section 3 of the SEXTRACTOR user manual contains a brief section explaining how to install the software. The code itself also includes a file called `INSTALL`, with slightly more detailed installation instructions. One item not explictly stated in the installation notes is that SEXTRACTOR requires the ATLAS[2] linear algebra package as a dependency. SEXTRACTOR will not install properly unless ATLAS is installed. The ATLAS website includes the requisite source code and an extensive installation guide. Once ATLAS builds and installs, the SEXTRACTOR installation steps should be straightforward to follow and complete.

[1] 04 April 2014

[2] `http://math-atlas.sourceforge.net/`

5. Quick Start

BALROG has been designed with flexibility of use in mind. As a necessary consequence, a number of different configuration possiblities exist, each of which must be adequately explained, which quickly expands the length of the documentation. We realize parsing the entirity of this manual requires some time. Hence, the intent of the this section is to offer a short primer for a few of the most import features BALROG users will want to become familiar with. The comprehensive usage instructions are saved for later sections. The start up pointers below will refer readers to the relevant comprehensive sections for more details.

The fastest way to get started understanding how to configure and use BALROG is to run it using the example files which come packaged with the software, and then examine the input and output of the run. BALROG has been set up such that when the executable Python file is called without any command line arguments, it will run over the example files, filling in defaults as necessary. Thus, this inital call is as simple as:

```
% runbalrog
```

Referring to Table 5.1, `runbalrog` is equivalent to an alias to the file `balrog.py` located within the installation directory, labelled like an environment variable as `$INSTALLDIR`. We will use these conventions throughout the documentation. Section 5.1 briefly addresses the input which was read in for the `runbalrog` command and Section 5.2 introduces the output generated during the run.

5.1 Input

BALROG's input comes in two forms, command line arguments and Python statements. The command line arguments can be printed, along with brief help strings by running:

```
% runbalrog --help
```

Furthermore, a complete description of BALROG's command line arguments can be found in Section 6. In brief, the command line parameters are used to specify input images and their properties, as well as configuration files to use with SEXTRACTOR, and a few other variables every BALROG run will need to define. Each built-in option comes with a default BALROG will assume if the user does not supply one. The default example's image data, weight map, and PSF live in `$INSTALLDIR/default_example/`. The SEXTRACTOR configuration files live in `$INSTALLDIR/astro_config`. All command line argument names and default file names are intended to be transparent.

In order to flexibly define how galaxies will be simulated, BALROG accepts defined blocks of Python code within the file specified by command line option `--config`. The example's `--config` defaults to `$INSTALLDIR/config.py`. Included within this BALROG configuration file is support for implementing custom user-defined functions and command line arguments.

The core functions defined in the file have a strictly structured syntax which will be fully described in Section 6 and Section 7. The syntax is designed to be as Pythonesque as possible, so many users will likely be able to extrapolate directly from the examples without reading lengthy documentation. The file also includes comment lines as a guide. A slightly more sophisticated configuration file can be found in `$INSTALLDIR/config2.py`. These two examples are designed to demonstrate the range of statements available to BALROG's Python configuration files.

5.2 Output

I might expand this section.

All BALROG output is saved in subdirectories under the directory chosen by command line option `--outdir`. This directory has defaulted to `$INSTALLDIR/default_example/output` for the example run. The complete set of output generated by BALROG is detailed in Section 8; most relevant for getting started are the `balrog_cat` subdirectory and the `balrog_log` subdirectory. `balrog_cat` contains `example.truthcat.sim.fits`, the truth parameters assigned to simulated galaxies and `example.measuredcat.sim.fits`, the simulated galaxies' properties as measured in the image by SEXTRACTOR. `balrog_log` saves log files useful for debugging BALROG runs, including dumps of how the the command line arguments were parsed, SEXTRACTOR's command line output, and any BALROG specific run errors or warning which occured. Section 9 offers some debugging hints.

Table 5.1: Definitions used throughout this manual.

Name	Meaning
`$INSTALLDIR`	Directory where BALROG was installed
`runbalrog`	`$INSTALLDIR/balrog.py`

6. Command Line Options

BALROG runs can be configured via command line options. Two types of options exist. First are the built-in ones, native to BALROG. In addition, BALROG supports a mechanism for users to define their own command line options. To print a list of all BALROG's command line options, both native and user-defined, along with brief help strings, run:

```
% runbalrog --help
```

Section 6.1 further details each of the native options and Section 6.2 explains how to create custom options.

6.1 Built-in Options

BALROG includes a number of built-in optional arguments for each run, defining a variety of parameters such as the input image, the number of galaxies to simulate, the flux calibration, etc. Any options which are not specified assume a default value. The options are intended to be named intuitvely in order to faciltate ease of use. Table 6.1 lists each option, with a description of what it means, including its default. Abbreviations for each option also exist, trading clarity for brevity.

We should decide what the best name for everything is, both the full name and abbreviation.

Table 6.1: Command line arguments natively built-in to BALROG.

Name	**Description**
- Output -	
`--outdir` `-od`	Toplevel directory for BALROG output files. Files will be organized into intuitively named directories under `--outdir`. **DEFAULT**: `$INSTALLDIR/default_example/output/`
`--clean` `-c`	Delete intermediate image files (those in `--outdir/balrog_image`) after catalogs have been written. **DEFAULT**: Unflagged, i.e. effectively false

- Input Images -

`--imagein`
`-ii`
Image to insert simulated galaxies into. Must be in FITS format.
DEFAULT: `$INSTALLDIR/default_example/input/example.fits`

`--imageext`
`-ie`
Index of the FITS extension where the image flux data lives. Indexing begins at 0.
DEFAULT: 0

`--weightin`
`-wi`
File containing the weight map associated with `--imagein`. Must bein FITS format. This can be a separate file from `--imagein` or the same file, where the flux image and weight map live in different extensions.
DEFAULT: `$INSTALLDIR/default_example/input/example.fits`

`--weightext`
`-we`
Index of the FITS extension where the weight map data lives. Indexing begins at 0.
DEFAULT:
if `--imagein` != `--weightin`:
`--weighext` = 0
else:
`--weightext` = `--imageext` + 1

`--psfin`
`-pi`
File containing the PSFEx PSF model for `--imagein`. This is a FITS file, but the convention uses `.psf` as the extension.
DEFAULT: `$INSTALLDIR/default_example/input/example.psf`

- Subsampling -

(Overall, the default behavior uses the entire image.)

`--xmin`
`-xmin`
Pixel x-coordinate for the lower bound of where to place simulated galaxies into `--imagein`; $x \in [1, N_{\text{cols}}]$.
DEFAULT: 1

`--xmax`
`-xmax`
Pixel x-coordinate for the upper bound of where to place simulated galaxies into `--imagein`; $x \in [1, N_{\text{cols}}]$.
DEFAULT: N_{cols}

`--ymin`
`-ymin`
Pixel y-coordinate for the lower bound of where to place simulated galaxies into `--imagein`; $y \in [1, N_{\rm rows}]$.
DEFAULT: 1

`--ymax`
`-ymax`
Pixel y-coordinate for the upper bound of where to place simulated galaxies into `--imagein`; $y \in [1, N_{\rm rows}]$.
DEFAULT: $N_{\rm rows}$

- - - - - - - - - - - - - - - - - - - Simulated Galaxy Generation - - - - - - - - - - - - - - - - - - -

`--ngal`
`-ngal`
Number of galaxies to simulate.
DEFAULT: 40

`--zeropoint`
`-zp`
Zeropoint for converting sampled simulation magnitudes to simulated fluxes. SExtractor will also apply this zeropoint to magnitudes. `--zeropoint` can take two types values: a float explicitly defining the zeropoint, or a string referring to a keword written in the header of `--imagein[-imageext]`. If neither of these is successfully found, Balrog uses the default.
DEFAULT:
try:
`--zeropoint` = `--imagein[-imageext].header['SEXMGZPT']`
except:
`--zeropoint` = 30.0

`--gain`
`-gain`
Gain [e^-/ADU] for adding CCD noise to the simulated galaxies. Refer to galsim.CCDNoise documentation for further details. `--gain` can take two types of values: a float explicitly defining the gain, or a string referring to a keword written in the header of `--imagein[-imageext]`. If neither of these is successfully found, Balrog uses the default.
DEFAULT:
try:
`--gain` = `--imagein[-imageext].header['GAIN']`
except:
`--gain` = 1.0

`--seed`
`-s`
Seed to give random number generator for any sampling which requires it, except noise realizations which are always different.
DEFAULT: Current time

- SExtractor -

(Refer to the SExtractor user manual or Source Extractor for Dummies for more help.)

`--sexpath` `-spp`	Full path to SExtractor executable. **DEFAULT**: sex, i.e. system default
`--sexconfig` `-sc`	Config file for running SExtractor (c.f. Section 2.4). **DEFAULT**: `$INSTALLDIR/astro_config/sex.config`
`--sexparam` `-sp`	Param file specifying which measurements SExtractor outputs (c.f. Section 2.4). **DEFAULT**: `$INSTALLDIR/astro_config/bulge.param`. This performs a single Sérsic profile model fit to each galaxy with free Sérsic index.
`--sexnnw` `-sn`	SExtractor neural network file for star-galaxy separation **DEFAULT**: `$INSTALLDIR/astro_config/sex.nnw`
`--sexconv` `-sf`	SExtractor filter convolution file when making detections. **DEFAULT**: `$INSTALLDIR/astro_config/sex.conv`
`--noassoc` `-na`	Do not run SExtractor in association mode (c.f. Section 2.4). **DEFAULT**: Unflagged, i.e. use association mode.
`--noempty` `-ne`	Skip SExtractor run over the original image, prior to any simulation (c.f. Section 2.4). **DEFAULT**: Unflagged, i.e. perform the SExtractor run
`--sexemptyparam` `-sep`	Param file specifying which measurements SExtractor outputs during run over the original image, prior to any simulation (c.f. Section 2.4). **DEFAULT**: `$INSTALLDIR/astro_config/sex.param`. This does not do model fitting.

6.2 User-defined Options

Within the `config.py` file, user's are able to define their own command line options. This occurs within the function `CustomArgs`. Passed to `CustomArgs` as an argument is `parser`, an object made by `python`'s `argparse.ArgumentParser()`. Arguments can be added to parser

according to the usual `argparse` syntax. For those unfamilar with `argparse`, this tutorial contains many useful examples. A simple example of `CustomArgs` is copied below.

```
def CustomArgs(parser):
    parser.add_argument("-cs", "-catalogsample", help="Catalog used to
                        sample simulated galaxy parameter distriubtions
                        from", type=str, default=None)
```

User-defined options are parsed within the function `CustomParseArgs`, also part of `config.py`. Passed as an argument to `CustomParseArgs` is `args`, equivalent to an object returned by `parser.parse_args()`. Each one of the user's command line options becomes an attribute of `args`. A simple version of `CustomParseArgs` has been included below.

```
def CustomParseArgs(args):
    thisdir = os.path.dirname( os.path.realpath(__file__) )
    if args.catalogsample==None:
       args.catalogsample = os.path.join(thisdir, 'cosmos.fits')
```

The ability to define and parse one's own command line arguments is intended to make BALROG flexible to conveniently running a wide variety of different simulation scenarios. These parameters are available to the user when setting up the galaxy simulations. How to define these simulations is described in Section 7 below.

7. Defining How to Simulate Galaxies

Defining how galaxies should be simulated is controlled within `config.py` in the function `SimulationRules`. Passed into the function are three arguments: `args`, `rules`, and `sampled`. `args` refers to the parsed command line arguments, both native BALROG and user-defined. `rules` is an object whose components are overwritten in order to specify how simulated galaxies are sampled. `sampled` gives access to simulated galaxy parameters after they have been sampled. `rules` and `sampled` will become clearer to follow.

BALROG simulates galaxies as N component Sérsic profiles, where N ranges from 1 to as many as desired. Associated with each of these components are five attributes: a Sérsic index, half light radius, magnitude, axis ratio (b/a), and orientation angle (β). In addition, each simulated galaxy has five attributes common among each Sérsic profile: x and y-coordinates, two components of reduced shear (g_1, g_2), and magnification. `SimulationRules`'s argument `rules` is comprised of attributes for these different galaxy characteristics. Users overwrite each of `rules`'s attributes to define their simulations. For example the statement to set each galaxy's magnification to 1 would read:

```
magnification = 1
```

`rules` has 11 attributes in total, whose names are intended to be simple to understand. These are printed and desribed in Table 7.1 below.

Table 7.1: Attributes of the rules defining the simulated galxies.

Attribute	**Meaning**
`rules.x`	Galaxy centroid x-coordinate [pixels], first pixel = 1
`rules.y`	Galaxy centroid y-coordinate [pixels], first pixel = 1
`rules.g1`	Reduced shear, g_1 component
`rules.g2`	Reduded shear, g_2 component
`rules.magnification`	Magnification, $1 + \kappa$
`rules.nProfiles`	Number of superimposed Sérsic profiles
`rules.sersicindex`	Sérsic index
`rules.halflightradus`	Sérsic half light radius
`rules.magnitude`	Galaxy brightness in magnitudes
`rules.axisratio`	Minor to major axis ratio, b/a
`rules.beta`	Orientaiton angle of major axis (measured from x-axis)

`rules.sersicindex`, `rules.halflightradius`, `rules.magnitude`, `rules.axisratio`, and `rules.beta` must be arrays, whose length is equal to `rules.nProfiles`. For example,

simulating galaxies with both an expontential and a de Vaucouleurs component would read:

```
rules.nProfiles = 2
rules.sersicindex = [1,4]
```

Simulation rules can assume four types. The first is a constant, meaning each of the galaxies in the simulated galaxy sample will have the same value for the selected parameter. Rules can also be assigned as an array, equal in length to the number of simulated galaxies. Simulated galaxy i for the chosen parameter then assumes the value in element i of the array. Additionally, sampling can be drawn from a catlog. Multiple parameters selected from the same data table are automatically jointly sampled. Finally a function can be used. Users write their own `python` function, then feed this function and its necessary arguments as the arguments to BALROG's `Function` command. The defined function must return an array equal in length to the number of simulated galaxies. Like the array sampling type, galaxy i will use element i of the returned array. Currently only positional arguments are supported within the user-defined functions, but support for keyword arguments will be added. `Function` affords both flexiblity and convenience when deciding how to sample the simulated galaxies. See `config.py` for examples using `Function` as well as other sampling types. One simple example of how to implement each of the four different types is shown in Table 7.2.

Table 7.2: Syntax examples for each of the simulation types BALROG understands.

Type	**Example**
Constant	`rules.g1 = 0`
Array	`rules.axisratio = [np.ones(args.ngal)/np.arange(1,args.ngal+1), sampled.axisratio[0]]`
Catalog	`rules.magnitude = [Catalog('cosmos.fits', 0, 'IMAG'), Catalog('cosmos.fits', 0, 'IMAG')]`
Function	`rules.g2 = Function(function=NFW2, args=(sampled.x,sampled.y))`

The second and fourth examples in Table 7.2 makes use of `sampled`, the third argument passed to `SimulationRules`. `sampled` is an object allowing users to refer to the properties of the simulated galaxies after they have been sampled. Referring to the second example above, the first element of `rules.axisratio` is an array decreasing incrementally from 1 to 0. The second elemment of `rules.axisratio` then says to use whatever the sampled values of the first element turn out to be. Here, this equates to setting the second element of `rules.axisratio` to the same array as the first element of `rules.axisratio`, so the use of `sampled` is not really necessary. However, return to the fourth example in Table 7.2. Image `NFW2` as a function which takes two arguments, `x` and `y`, which returns the g_2 component of shear at position (x,y) from an NFW halo with mass $10^{15} M_{\odot}$ whose center lies at (0,0). Now image `sampled.x` and `sampled.y` were sampled randomly:

```
def Random(minimum, maximum, size):
    return np.random.uniform(minimum, maximum, size)
```

```
rules.x = Random(args.xmin, args.xmax, args.ngal)
rules.y = Random(args.ymin, args.ymax, args.ngal)
```

`sampled.x` and `sampled.y` now represent the values for x and y after the random sampling has occurred. Along these lines, BALROG can build-up simulations with fairly little recoding between completely different types of simulations. BALROG makes sure that all the simulation parameters are sampled in the proper order and will throw an error if something ambiguous was defined by the user.

8. Output

Each BALROG run generates a number of output files. These are organized into a fixed directory structure. Users indicate the `--outdir` command line option, and the remainder of the naming scheme occurs automatically, placing files in subdirectories under `--outdir`. Four subdirectories are written, labelled accoring to what type of files they contain. Table 8.1 lists the contents of each of these subdirectories, giving a brief desription of each file. Depending on how BALROG was configured, not necessarily every file in Table 8.1 will be present in every run. The `*` symbol in Table 8.1 will be replaced with the base name of the input image file. For example, if the input image is named `example.fits`, `*` will be replaced with `example`. If the input file name does not end with the `.fits` extension, the file name itself is used as the base name. This does not include any diretories preceeding the file name. For example, if the input image was given as `/Users/somebody/home/image.f`, the base name would be `image`.

Table 8.1: File structure of the output written by BALROG. * is replaced with the base name of the input image

Path	**Contents**
`--outdir/balrog_cat/`	Output catalog files
`*.measuredcat.nosim.fits`	SEXTRACTOR catalog over original (subsampled) image, prior to simulation.
`*.measuredcat.sim.fits`	SEXTRACTOR catalog over image which includes simulated galaxies.
`*.truthcat.sim.fits`	Truth catalog of the simulated galaxies' properties.
`--outdir/balrog_image/`	Output images
`*.nosim.fits`	Copy of (subsampled) input image.
`*.sim.fits`	Image containing simulated galaxies.
`*.example.psf`	Copy of (subsampled) PSFEX PSF image.
`--outdir/balrog_log/`	Log files
`--outdir/balrog_sexconfig/`	Files used to configure SEXTRACTOR
`*.assoc.nosim.txt`	SEXTRACTOR's association mode matching list for run without simulated galaxies.
`*.assoc.sim.txt`	SEXTRACTOR's association mode matching list for run with simulated galaxies.
`*.measuredcat.nosim.sex.params`	Copy of input SEXTRACTOR parameter file used for run without simulated galaxies, which properly factors in association mode.

`*.measuredcat.sim.sex.params`	Copy of input SExtractor parameter file used for run with simulated galaxies, which properly factors in association mode.

9. Debugging

Each galaxy is initally simulated within its own image, commonly known as a postage stamp. The postage stamp pixel values are then added to the appropriate pixel values in the original input image. To avoid losing a significant portion of a galaxies outside the postage stamp boundaries, an appropriate postage stamp size must be chosen. Bigger postage stamps exclude less flux. However, the dimensions should not become too large because larger postage stamps take longer to simulate. To designate postage stamp sizes, BALROG operates using an adjustable threshold, I_t. The postage stamp will be chosen such that the galaxy's surface brightness profile must fall below I_t before drawing the postage stamp boundaries.

BALROG determines its postage stamp sizes conservatively, airing on the side of making size estimages larger than perhaps strictly required. To begin, let us assume the simulated galaxies are composed of a single Sérsic profile. We will generalize to multiple superimposed profiles to follow. First a GalSim `Sersic` object is obstantiated. Such objects within GalSim are all initally circularly symmetric. The galaxy is then stretched according to its minor to major axis ratio, β. β's major axis direction is taken to be along the x-axis. Next, the galaxy's lensing shear is applied, expressing the magnitude of its reduced shear $g = \sqrt{g_1^2 + g_2^2}$ as an axis ratio $q = (1-g)/(1+g)$. q's major axis is also taken to be along the x-direction. Finishing the lensing effects, the galaxy is magnified, changing its area and flux. Effectively, the galaxy now exists as an elliptical Sérsic profile. Further refferences to the half light radius will refer to that along the major axis. Please note, the intrinsic axis ratio and lensing shear are applied in the same direction. Although the galaxy's simulation specifications need not orient the two identically, by adopting the same direction, this step has made the galaxy as stretched as possible. Thus the x-direction is necessarily the direction of greatest distance for the galaxy's profile to fall below I_t, and in the abscence of PSF distortions, a large enough postage stamp size along the x-axis is certainly sufficient in any other direction. Given a Sérsic surface brightness profile, the distance r_s at which the profile falls to I_t is easily computed:

$$I_t = I_e \exp\left(-b_n \left[(r_s/r_e)^{1/n} - 1\right]\right) \tag{9.1}$$

$$\implies r_s = r_e \left[1 - \frac{1}{b_n} \ln\left(\frac{I_t}{I_e}\right)\right]^n. \tag{9.2}$$

r_e is the half light radius along the major axis, $I_e = I(r_e)$, and b_n is a constant determined by the Sérsic index, n.

Including PSF distortions, Equation 9.2 is no longer the end of the story. When determining postage stamp sizes, BALROG approximates the PSF as a Gaussian. The width of the distribution is fixed to a fiducial seeing value of $\sigma = 1.5$ arcsec. This PSF generates an additional r_p to be added to r_s. Precisely, r_p is computed by scaling the PSF to the surface brightness of the galaxy's centroid, I_0, and asking for the distance r_p along the major axis

where the distribution equals I_t.

$$I_t = \frac{I_0}{\sigma\sqrt{2\pi}} \exp\left(\frac{r_p^2}{2\sigma^2}\right) \tag{9.3}$$

$$\implies r_p = \sigma\sqrt{2\ln\left(\frac{I_0}{I_t}\right)}. \tag{9.4}$$

Effectively, what we calculate is the size of the region over which the modelled galaxy's centroid is redistributed because of the PSF assumed, drawing the boundaries of the region where the surface brightntess drops below I_t. Because every position in the modelled galaxy's profile other than the centroid has a surface brightness less than I_0, the PSF will redistribute the coordinate's light over a smaller region compared to the centroid's. Therefore, the Gaussian PSF will not move the threshold originally located at r_s by a distance of more than r_p.

The fiducial seeing should probably be made an adjustable parameter. I didn't do that thus far just not to make another command line argument.

Bibliography

Bertin E., 2011, in Evans I. N., Accomazzi A., Mink D. J., Rots A. H., eds, Astronomical Society of the Pacific Conference Series Vol. 442, p. 435

Bertin E., Arnouts S., 1996, A&AS, 117, 393