

# LIRA Tutorial

# Table of Contents

- LIRA inputs
- LIRA helper function
- Example
- Trouble Shooting

# Required R Packages

- lira
  - vkashyap/LIRA on [github.com](https://github.com/vkashyap/LIRA)
- FITSio (R package)

# tutorialLIRA/

- example/
  - Files for a worked example
- LIRA\_main.pdf
  - Original tutorial by A. Connors
- liraOutput.R
  - Simplified LIRA function

The original LIRA inputs and outputs using `lira()` function in `lira` R package

## **LIRA INPUTS**

# Creating Input Files

- Observed Image (obs.matrix):
  - Size:  $2^n \times 2^n$
- Point Spread Function (psf.matrix):
  - Must be small due to wrap around in LIRA
- Baseline Model (bkg.matrix):
  - Size:  $2^n \times 2^n$
  - Models the null distribution
- Starting Matrix (start.matrix):
  - Size:  $2^n \times 2^n$
  - Matrix in which the MCMC will converge from
  - Usually not specified – set as default value (matrix of ones)

# Other Inputs

- `fit.bkg.scale`
  - T = the multiplier on the baseline matrix will fit itself at each iteration
  - F = the multiplier on the baseline matrix does not change (=1)
- `max.iter`
  - The number of LIRA iterations
- `alpha.init`
  - The initial smoothing parameters (these values don't really matter, but try to get something in the same order of magnitude of its convergence)
  - Length must equal n in  $2^n \times 2^n$  image
- `out.file / param.file`
  - The names of the .out and .param files

# LIRA Outputs

- ‘\*.out’
  - An array containing the multiscale count values of each pixel at each iteration
  - A table containing a  $2^n \times 2^n$  image for all iterations
    - ex: a  $2 \times 2$  array for 100 iterations would create a  $2 \times 200$  table
- ‘\*.param’
  - A table of the parameter values at each iteration
  - See A. Connors tutorial for description of parameters



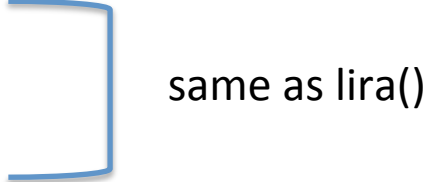
‘liraOutputs.R’ - A simplified version of lira() created by Kathryn McKeough

# **LIRA HELPER FUNCTION**

# My 'helper' function

- Simpler version of LIRA R-function:
  - Uses names of FITS files as input
  - Automatically names output files
  - Ignores unused variables
- 'liraOutput.R' in tutorial directory
- NOTE: you must be in the directory of the input files for this to run

# Input

- `obsFile`
    - Filename for .fits file of the image to be analyzed
  - `psfFile`
    - File name for .fits file of PSF
  - `bkgFile`
    - File name of .fits file of baseline matrix
  - `startFile` (optional)
    - File name of .fits file of the start file
    - If nothing is put matrix will be 1 at every pixel
  - `outDir`
    - Directory of the output file
    - Anything after the final '/' will be a header to the .param and .out files
  - `alpha.init`
  - `maxIter`
  - `Fit.bkg.scale`
- 
- same as `lira()`

An example using real data from quasar 0703

# EXAMPLE

# Example LIRA Run

- Files located in 'liraTutorial' directory

```
| >setwd('.../liraTutorial/example')  
| >source('liraOutput.R')  
| >obsFile<-'img_64x64_0.5.fits'  
| >bkgFile<-'null_q1_c1.fits'  
| >psfFile<-'psf_center_33x33.fits'  
| >startFile<-'null_q1_c1.fits'  
| >  
| >testLira<-liraOutput(obsFile=obsFile, bkgFile=bkgFile, psfFile=psfFile, startFile=startFile,  
|   maxIter=1000, alpha.init<-c(3,4,5,6,7,8), fit.bkg.scale=T,  
|   outDir='outputs/test_')
```

# What it looks like.....

*Code will run in posterior sampling mode.*

*A scale parameter will be fit to the bkg model.*

*The total number of Gibbs draws is 1000, every 1th draws will be saved.*

*The model will be fit using the Multi Scale Prior.*

*The data matrix is 64 by 64.*

*The data file should contain a  $2^6$  by  $2^6$  matrix of counts.*

*Starting Values for the smoothing parameter (alpha):*

*Aggregation Level: 0, alpha: 0.3 (Full Data)*

*Aggregation Level: 1, alpha: 0.4*

*Aggregation Level: 2, alpha: 0.5*

*Aggregation Level: 3, alpha: 0.6*

*Aggregation Level: 4, alpha: 0.7*

*Aggregation Level: 5, alpha: 0.8 (In the 2x2 table)*

*The prior distribution on the total count from the multiscale component is  $\text{Gamma}(1.000000, 0.050000)$ .*

*The hyper-prior smoothing parameter (kappa 2) is 1000.*

Checking your LIRA output and common errors

# **TROUBLESHOOTING**

# Common Errors

- Observed, baseline and starting matrices are not all  $2^n \times 2^n$
- You are using 'liraOutput' outside of the directory with the input files
  - Splicing of input file name effects the output file
- PSF is too large – error in wrap around effect
- PSF is not centered
- If fitted baseline multipliers do not work try `fit.bkg.scale=F`



# Traces

- Plot value at each iteration
- In order to confirm convergence, see if LIRA runs at different starting values converge at the same range of values