$\mathcal{L}IRA$ Mini-Tutorial

A. Connors, with help from D. van Dyk and N. M. Stein

1. Introduction

This mini-tutorial describes the theory and practice behind $\mathcal{L}IRA$, the Low-count Image Restoration and Analysis package. We are mindful that potential $\mathcal{L}IRA$ users may bring with them an extremely broad variety of backgrounds and expertise, in astronomy, statistics, image-processing, applied mathematics, physics, and related engineering fields. Conversely, they may also be unaquainted with fields other than their own. Hence we have tried to begin each section at the introductory level. For those who are already advanced in these areas, we have tried to make it modular, so that one can more easily skip familiar material. At the end of each secton, we have put a brief summary. **Make sure this is there.** As well, there are two brief "How To" appendices at the end, showing the steps for downloading and running $\mathcal{L}IRA$ in outline format. We welcome opportunities to clarify the text.

Include Deconvolution, Model-Fitting Picture HERE

In Fgure xxx, we showcase the **Fermi data of a PWN deconvolved with $\mathcal{L}IRA$, compared to the data**. Most astrophysicists or image-processing users wish to begin immediately with this 'Reconstruction', or Poisson image deconvolution, aspect of $\mathcal{L}IRA$. Those used to point source searches want to know: how much will $\mathcal{L}IRA$ 'sharpen' their image data? Those experienced with diffuse sources ask: how will $\mathcal{L}IRA$'s multi-scale structure pick up and quantify broad features without over-smoothing important fine detail? However the strength and usefulness of $\mathcal{L}IRA$ comes from its abaility to *quantify uncertainties* in both cases. This, in turn, comes from having embedded the 'Multi-Scale Reconstruction' in a full probabilistic framework. This gives $\mathcal{L}IRA$ Poisson goodness-of fit capabilities; as well as the ability to quantify multiple scales of unknown diffuse emission — even without 'deconvolving' using a PSF.

For all of these, $\mathcal{L}IRA$ is easier to use if one first understands its basic structure: the underlying likelihood/probability framework; the particular multi-scale representation of the unknown part of the 'true' image; practical MCMC; and using summary statistics in high-dimensonal problems. Therefore, we run through specific examples.

In Section 2, we begin with a brief introduction to the nomenclature for likelihood fitting of Poisson data, and briefly sketch the Poisson-specific multiscale structure. In Section 3, we introduce our simplest example 'sky-truth' and simulated data. Section 4 provides a brief

introduction to multi-scale models in general (including the concepts of levels, smoothing priors, and cycle-spinning), and the Poisson-specific one used by $\mathcal{LIRA}$.

In Section 5, we walk through the basic McMC process with $\mathcal{LIRA}$, using our simplest example data. That is, using no PSF (that is, no smearing, PSF essentially a delta-function), and no (i.e. zero) background, we illustrate the basic MCMC process, outputs, and terms (including: parameter traces, convergence, burn-in, autocorrelation and thinning). We also introduce *summary statistics*.

Finally, in Section 6, we step through a more represenative $\mathcal{LIRA}$ McMC process, containing the more sophisticated statistical tests. These presuppose a best-fit 'Null' or 'Background' model of the expected counts, and demonstrate the process of simulating data based on the 'Null' model, and then running it through $\mathcal{LIRA}$ for comparisons. This also demonstrates the strength of using a *summary statistic* for quantifying uncertainty.

In Section 7, we do the same analysis again, but this time, incoprating smearing (a PSF). We illustrate that the outputs and procedure are (almost) exactly the same as without (i.e. Section 6), except that the input 'Null' or background model should not have been convolved with the isntrument response. Additionally, we discuss the problem of uncertainty in one's PSF (due to calibration, or incompletely known source parameters), and sketch out a very simple (if tedious) procedure for incorporating this via the MCMC sampling. In most of these, we point to sample R-scripts and data-sets in the `docs/examples` directory, so that users can run them for themselves.

We point out that at the end of many of the sections, we have put **Special Note for Users**. These highlight practical tips for those running $\mathcal{LIRA}$, especially for the first time. We have listed them at the end, for handy reference.

Finally, for those ready to get started quickly: in two appendices, we have outlined how to get $\mathcal{LIRA}$, and how to run $\mathcal{LIRA}$.

2. The Likelihood; or, 'Forward-Fitting'

2.1. Fitting Basics.

$\mathcal{L}IRA$ is designed for 'discrete' (statistics jargon), intrinsically Poisson counts (Astrophysics term) data. These data are assumed to be in the form of counts y per pixel ij. Hence, if we have an astrophysics (or other) model of the expected counts λ_{ij} in instrument pixel ij, [1] the probability p of the data y given our model λ has the usual form:

$$p(y|\lambda) = \Pi_{ij} \lambda_{ij}^{y_{ij}} \exp(-\lambda_{ij})/(y_{ij}!) \tag{1}$$

So if ϵ_{kl} represents the effective area (Astro jargon) in each sky-bin; $PSF_{ij,kl}$ is the point-spread-function (Astrophysics term), that is, the probability that a photon from sky-bin kl is recorded in pixel ij (statistics terms); μ_{kl} the physics (or other) model of the sky; and b_{kl} the background; then the λ_{ij} can be written as:

$$\lambda_{ij} = \sum_{kl} (PSF_{ij,kl}(\mu_{kl} + b_{kl})\epsilon_{kl}), \tag{2}$$

per usual.

Astrophysics users may be used to approximating the first equation with a Gauss-Normal distribution. (This is convenient for those accustomed to using χ^2 as a reference distribution.) From doing the Taylor series (in log-space) oneself, one can see that the expected counts, or model λ_{ij}, takes the place of the Gauss-Normal mean; and the standard deviation, or σ_{ij}, for each datum is given by the square root of the model, $\sqrt{\lambda}_{ij}$ (see Primini et al., for example). This in turn is often approximated by the square root of the measured counts: $\sigma_{ij} \approx \sqrt{\lambda}_{ij} \approx \sqrt{y}_{ij}$. There are other careful approximations for σ_{ij} in these Gauss-Normal schemes (e.g. Gehrels et. al.). However, as Siemiginowska et al. 2010 (ADASS proceedings) points out (in an energy-spectrum fitting context), these ignore the skew of the Poisson distribution and produce bias (see Fig 2). Even when the bias is small, since the tails of a Gauss-Normal and Poisson distribution are so different; and since statistical significance and uncertainty calculations use more of the tail of a distribution; uncertainties and significances based on Gauss-Normal distributions (and χ^2) can be over an order of magnitude different than those based on the full Poisson distribution (with low counts per bin) — even when the total counts are quite large. Hence, especially for low-counts-per-bin data, one needs to correct Poisson formulation, rather than the aculturally-accustomed and convenient Gauss-Normal approximation.

---

[1] (Sorry, astronomers - using λ_{ij} for the expected counts — rather than wavelength — is a statistics jargon convention; as is using "y" for the counts data, rather than "n".)

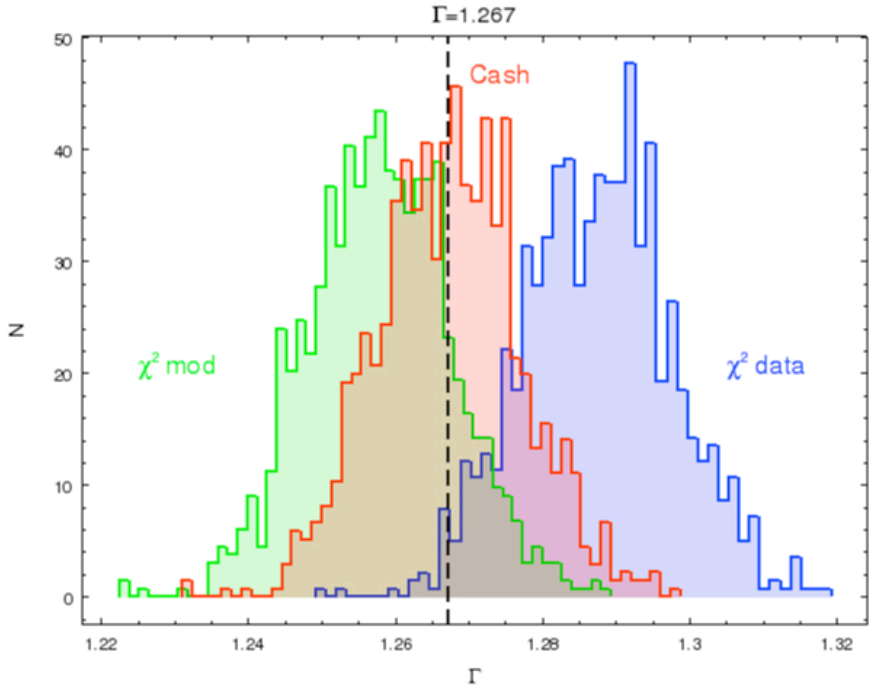


Fig. 1.— From Siemiginowska et al. 2010 (ADASS proceedings).

2.2. Trick: 'Model-Free', or Non-Parametric, Models.

Now, if we do not know a good, astrophysics-based form for μ, we can always replace it with a very flexible model, such as: individual counts per bin (the usual model underlying many astrophysics Richardson-Lucy applications); splines; Fourier components; Markov Random Fields; an orthonormal basis such as wavelets (which were designed for continuous Gauss-Normal data); or an analogous multi-scale representation tailored for (discrete) Poisson data. These are often termed semi-parametric or non-parametric models (i.e almost as many or as many parameters as there are data). The last is what Esch et al. 2004 did, following work by Nowak and Kolaczyk. This class of Poisson-specific multi-scale representations were given the name Multiplicative Multi-scale Innovations (see Willett et al. for nice examples).

There are some complications. First, how does one choose an appropriate flexible model? Second, how does one avoid "over-fitting" and multiple modes, of the sort one expects from ill-constrained 'inverse problems'? Third, now that there are roughly as many parameters, as data, how does one explore the parameter space? We address these in the following two sections.

Special Note for Users: $\mathcal{L}IRA$ **REQUIRES** unprocessed counts, since it uses the Poisson formulation. The data CANNOT be background-subtracted (data-cuts are fine, though); or averaged or extrapolated (as in some CGRO/EGRET all-sky maps), or 'corrected' by an exposure factor (as is often done with TeV telescope data). All such processing belongs in the 'Instrument' model in the forward fitting: effective area; PSF; exposure map; and background. The process will simply NOT WORK (i.e. give you very weird results) if you try running on data that have been 'corrected' in some fashion.

3. Introducing the Simple Example Data: 'Diffuse' E plus 'Point Sources' E

In your $\mathcal{L}IRA$ distribution, in the documents section (that is, in `lira/docs/examples/` or `lira/inits/docs/examples/`), there are: some sample R-scripts; sample data (in `exampledata/`); sample $\mathcal{L}IRA$ outputs (in `outputs/`); and sample post-processing scripts and results (in `postprocessing/`). In this section, we start with a 32x32 bin (simulated) data set, no background specified (i.e. zero background), no PSF[2]. Our simple model shows two renditions of the letter **"E"** on top of a flat background.The "**E**" in the upper right is modeled as being composed of "point-sources"; while the "**E**" in the lower right represents structured diffuse. This will be the "sky-truth" for this example.

1. The original model, or 'sky-truth', can be found in:

   `examples/exampledata/FullTrueModelEEMC2_32x32testE.090120c.fits and`
   `examples/exampledata/FullTrueModelEEMC2_32x32testE.090120c.txt.`

2. Simulated data (i.e Poisson distribution with 'sky-truth' as the mean) are found in:

   `examples/exampledata/PoisDatonsEEMC2_32x32testE.090120c.fits and`
   `examples/exampledata/PoisDatonsEEMC2_32x32testE.090120c.txt.`

3. Sample files containing images of various starting values include:

   `examples/exampledata/start32x32_1.00.fits`

   and

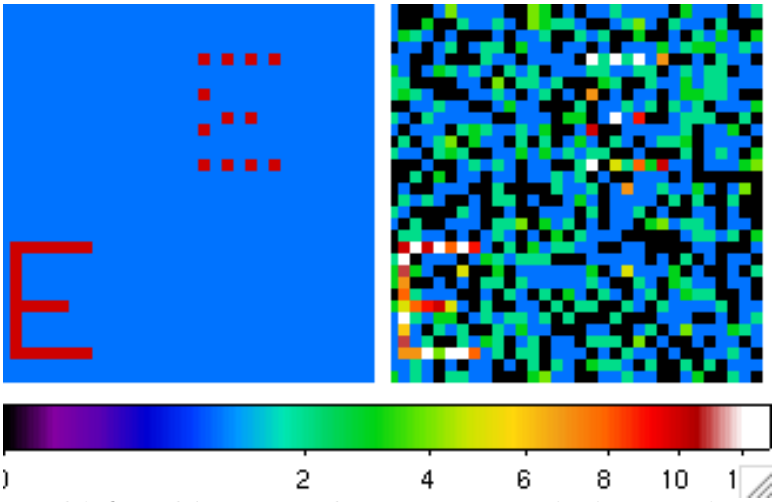
   `examples/exampledata/start32x32_1.00.txt.`

(You may notice that these files have rather long, self-documenting names.)

Sample R-scripts: Here are some sample R-scripts for loading and plotting these kinds of files:

---

[2]Actually, "no PSF" really means no *smearing*, or a Kronecker delta for the PSF in equation 2.



(a) Left: 'Sky-Truth' for this example. We start with this simple model, showing two renditions of the letter "**E**" on top of a flat background. **(b) Right: Simulated Poisson data**, based on the 'Sky-Truth' , and assuming no smearing and flat effective area.

```
##
## Making R-plots of the specified data files:
##

## 1. Text files (no FITSio required):

infileTrueModel = 'examples/data/FullTrueModelEEMC2_32x32testE.090120c.txt'
ImgTrueModelDat <= read.table(infileTrueModel)

## 2. Or, if one has fitsIO:
#infileTrueModel = 'examples/data/FullTrueModelEEMC2_32x32testE.090120c.fits'
#ImgTrueModelDat <= read.array(infileTrueModel)

plot(ImgTrueModelDat$img)
```

4. Flexible Representations for Poisson data: Ways of "Paying Attention"

4.1. Brief History For Fun and Context

> "I really explained in the paper where things came from. Because, well, the mathematicians wouldn't have known. I mean, to them this would have been a question that really came out of nowhere. So, I had to explain it ...
>
> "I was very happy with [the paper]; I had no inkling that it would take off like that... [Of course] the wavelets themselves are used. I mean, more than even that. I explained in the paper how I came to that. I explained both the mathematicians' way of looking at it and then to some extent the applications way of looking at it. And I think engineers who read that had been emphasizing a lot the use of Fourier transforms. And I had been looking at the spatial domain. It generated a different way of considering this type of construction. I think, that was the major impact. Because then other constructions were made as well. But I looked at it differently. A change of paradigm. Well, paradigm, I never know what that means. A change of ... a way of seeing it. A way of paying attention."

Ingrid Daubechies, NAS Oral History

From almost as early as historians have found clear written records, Fourier transforms, or harmonic analyses, have been used in astronomy. For example, Babylonian cuneiform writing from 300 B.C.E. shows harmonic analyses were used to calculate the cyclical but complicated apparent position of the moon on the sky[3], including eclipses and new moons (e,g, Nuegebauer 1975). In the modern era, sines and cosines were used to represent complicated functions as early as the mid-1700's [Gauss on Ceres ref1, ref2]. Euler is generally credited with the first use of the discrete Fourier transform, and of factoring it so that it could be calculated in $\mathcal{O}(N \log N)$ steps rather than N^2 steps, where N is the length of the 1D data. Fourier was the person who pushed the discovery that an appropriate series of sines and cosines could represent virtually any smooth function (Fourier On the understanding of the spread of heat); and therefore could be used to solve difficult differential equations and so forth.

Of course in the twentieth century, with the beginnings of electronic signal processing and computing, use of the Fourier transform took off. It was used for processing everything from audio and seismic signals to image processing and beyond [do I need a ref for this?] The re-discovery, by Cooley and Tukey, of the Fast Fourier Transform algorithm [Cooley and Tukey 1965] brought an additional big boost for these methods.

---

[3]Astronomy term: ephemeris or ephemerides

Let us look more explicitly:

1. Fourier transforms (FT) could represent almost any shape (orthogonal representation) as a simple (linear) combination of high to low frequencies.
2. The original shape could be recovered completely from its Fourier transform (inverse Fourier transform).
3. Hence the FT could be used to help re-express difficult equations (such as convolutions, or differential equations) in a simpler form.
4. If one assumed the signal was stationary (or periodic) and had white noise, then a Fourier transform of a noisy signal could be used to "filter out" the high-frequency noise (Wiener-Kolmogorov filters 1941, 1949). (This is the 'engineering' language. From a statistics perspective, one is obtaining a best estimate of [I think maximum-likelihood and mean are the same...] of the signal, assuming the data have additive Gauss-Normal noise with homoscedastic variance – i.e. constant variance, or white noise.)
5. The Fast Fourier Transform (FFT) algorithm allowed this to be done fast - by recursively factoring the problem into two components, given the previous components, until there were very few calculations of sines and cosines to do (expensive in computer time), and mostly recursive multiplication (cheap).

[From Daubechies Ref "Where do Wavelets Come From" 1996.] However – most signals were not stationary, neither in time for 1D data, nor spatially, in images. By the 1980s, electrical engineers and geophysicists such as Morlet [Other Refs?] tried breaking the data into "windows", then taking its Fourier transform. Specifically, it was generally perceived that large objects in a signal tended to be more slowly varying (lower frequency), and smaller objects had generally finer detail (higher frequency). [4] Hence one fundamental trick used was that *the size of the 'data window' shrank with decreasing wavelength of the Fourier component*, and vice versa. Unfortunately, these intuitive but ad-hoc techniques often led to unacceptable aliasing. Various methods of 'tapering' the window and the frequencies associated with it were tried. There was a fundamental problem, well-articulated from the perspective of Quantum Mechanics (QM) by theoretical physicists such as Marcel Grossman: The Heisenberg QM Uncertainty Principle expresses that a signal cannot be sharply defined in both position and frequency. Mathematicians framed it as: how could a windowed Fourier transform have compact support for both the time (or spatial) domain and the frequency domain, even with different tapering schemes for the windows (and frequencies)? Roots of

[4]Interestingly, this does not hold for data with a lot of 'texture' within a large object or window. For this, other tricks are often used.

deeper understandings from theoretical Quantum mechanics; Quantum Gravity [Aslaksan and Krauder]; the pure mathematics of Harmonic Analysis (originally for dealing with singularities) [Meyer; Calderón]; led Daubechies to create a popular series orthogonal wavelets with compact support. As well, from computer vision [Mallat and co-workers]; there was an explosion of work on the theory and practice of this new 'multi-resolution' analysis. Specifically, a 'multi-resolution' or 'multi-scale' analysis was designed such that: 1) There were successive (integrable) approximation levels V_j on which one's ($\mathcal{L}_2$ smooth) function could be projected; 2) these levels were orthogonal (in the sense of integrating to zero when multiplied times each other); 3) but they could represent an arbitrary function arbitrarily well as the number of levels $j \to \infty$; and, crucially, 4) all levels were *scaled* levels — by $\frac{1}{2^j}$ — of the main shape function V_0 (i.e. *dyadic partitioning* of the space, with 5) V_0 itself being invariant with respect to integer translations (e.g. "Ten lectures on wavelets" Daubechies pg 129-130). The earlier intuitive 'tapered window' work had now been put on a sounder and more robust theoretical basis. Morlet had coined the french words "onde-lets" meaning "little waves"; the English translation, "wavelets", caught on.

Let us make an analogy with Fourier transform methods:

1. Wavelets represent almost any shape (orthogonal representations are possible, depending on the wavelet basis) as a simple (linear) combination of high to low frequencies, in different spatial bins or windows across the data.
2. The original shape could be recovered completely from its wavelet transform (inverse wavelet transform).
3. It could be used to help re-express difficult equations (such as convolutions, or differential equations) in a simpler form.
4. There is no longer a need to assume the signal is stationary. From a probability/statistics viewpoint, one still assumed white noise (that is, Gauss-Normal with mean given by the wavelet representation and variance a constant). Donaho and co-workers demonstrated that, for 'sparse' data, a wavelet decomposition of a 'noisy' signal could be used to "filter' out" noise, by excluding low-probability components when making the reconstruction [Notes from . Kolaczyk – better ref??].
5. Roughly analogous to the FFT algorithm, the multi-scale transform can be thought of as recursively factoring the problem into two components, given the previous components (dyadic partitioning). The smaller components are also associated with successively higher frequency wavelets.

4.2. What does a wavelet look like?

Slezak, E.; Bijaoui, A.; Mars, G. early paper (1990) on wavelets in astronomy gives a nice description of wavelets in action. Another nice reference is at:

`cas.ensmp.fr/~chaplais/wavetour_presentation/Wavetour_presentation_US.html`

Under Construction Wavelets could be redundant (i.e. over-complete) or orthonormal or .... Their shapes could be based on splines; or variations of Gaussians; or ... They can have compact support, or ... Generally, the data are partitioned in two (or four, for images partitioned in X and Y). Each part is compared (via taking the dot product of the data with the wavelet basis for that level of resolution) to the larger part. [IS THIS CLEAR ENOUGH??] Each of these 'children' are then, in turn, partitioned in two; and so on, to the highest level of resolution required for the problem. Analogous to Fourier transforms, the signal could be recovered from the transforms and coefficients. [ ] **Many different waveforms as the bases – explain here Bijaoui's "over-complete" and others more explicitly?** [ ] **Briefly define mother/father/children wavelets.**

Put useful websites here: wavelets.org, Stanford's wavelab, Rice matlab site

Wavelets and Multi-scale methods have grown hugely – for nice overview with astronomy flavor, see book by Starck and Murtagh [REF; mention other recent work?? probably too much to include here].

Special Note for Users: As it is now written, $\mathcal{L}IRA$ REQUIRES the data to be in square maps of dimension 2^n. Any accompanying model, exposure, and background maps must match the data. In theory, Nowak and Kolaczyk have found a way, in one dimension, to lift the 2^n requirement - by implicitly embedding it in a larger 2^n vector. We have not done this work, in 2D, for $\mathcal{L}IRA$.

4.3. Multi-scale methods and Cycle-Spinning

As Eric Kolaczyk, Xiao-Li Meng articulated for 2009/20010 Harvard Stat 310 Astrostatistics discussions, this dyadic partitioning can also be thought of as a binary tree structure.

4.4. Multi-scale methods and Poisson data

Meanwhile, backing up in history: [REF: From E. Kolaczyk lecture] In 1910, Haar had found an ortho-normal basis – very blocky – did the 2^n recursive thing as well. Haar had taken the unit interval, that was one on that interval and zero off of it. Also, one on half, -1 on half, zero elsewhere ... First is the scale function ("father wavelet") and second is the shape function wavelet ("mother wavelet") Now if one starts playing a game of scaling, one can put two of them in each previous scale. Each will be orthogonal ..

SO instead of taking it and transforming in into wave-forms—- With Haar's function one is doing coarse local averaging on blocks; AND averaging the differences. Like contrasts at each different scale.

This is easier to handle for Poisson data, in the low-count limit. TIPSH papers by Kolaczyk et al.

Leading to: realization that, for Poisson, if one works on RATIOS instead os SUMS, the likelihood factors nicely. Elegant and Efficient means fast – also could prove it was convex (i.e. only one mode), with their choice of priors.

Definition of MMI's; and handy web-sites.

4.5. Complexity Penalties/Soft Thresholding/Smoothness Priors/Regularization

Story:"We're all Bayesians." attributed to D. Donaho by E. Kolaczyk.

Problem of 'over-fitting':

Astronomer 'standard' for EM is Richardson-Lucy applied pixel by pixel. Three problems, as usually implemented by astronomers (StSCI Richardson-Lucy):: 1) correlations among true values (stat jargon: expected values) of nearby pixels, at varying length scales. ADDRESSED by using multi-scale methods.

Even multi-scale methods can have two problems: 2) OVER-FITTING: How do you tell when to STOP EM de-convolution? A magic number (but see also Gelman's paper on it: run to convergence and go three steps back).

[3) Later we address the problem of error-bars.]

For wavelets, even with EM, moved to 'softer thresholding' *shrinkage* and *complexity penalties*. [REFS?? maybe to those in earlier van Dyk et al??] But structurally, this is

equivalent to Bayesian 'smoothness priors'.

DISCUSSION of 'smoothing prior' (astro jargon) vs multi-scale model for diffuse emission (astro jargon).

Special Note to Users: Note that $\mathcal{L}IRA$ has smoothing intrinsically built into the multi-scale component. The multi-scale portion is NOT engineered to detect point-sources (although it can – but the fluxes will tend to be a little low).

SO let us look at our overall Posterior likelihood:

Now, from eq??, usual Poisson likelihood given lambda.

Now have μ_{kl} given by the Nowak-Kolaczyk MMI:

formulation here

And have priors for smoothness; and hyper-priors for fitting smoothing priors:

And finally hyper-prior parameters set ahead of time via MC tests (Esch et al.)

put eq. here

4.6. Background models and the pre-factor

Special Note; No longer always convex, as we have introduced more competing model components (see next). SO should try multiple starting maps to make sure they all converge to the same final chain. (see next section.)

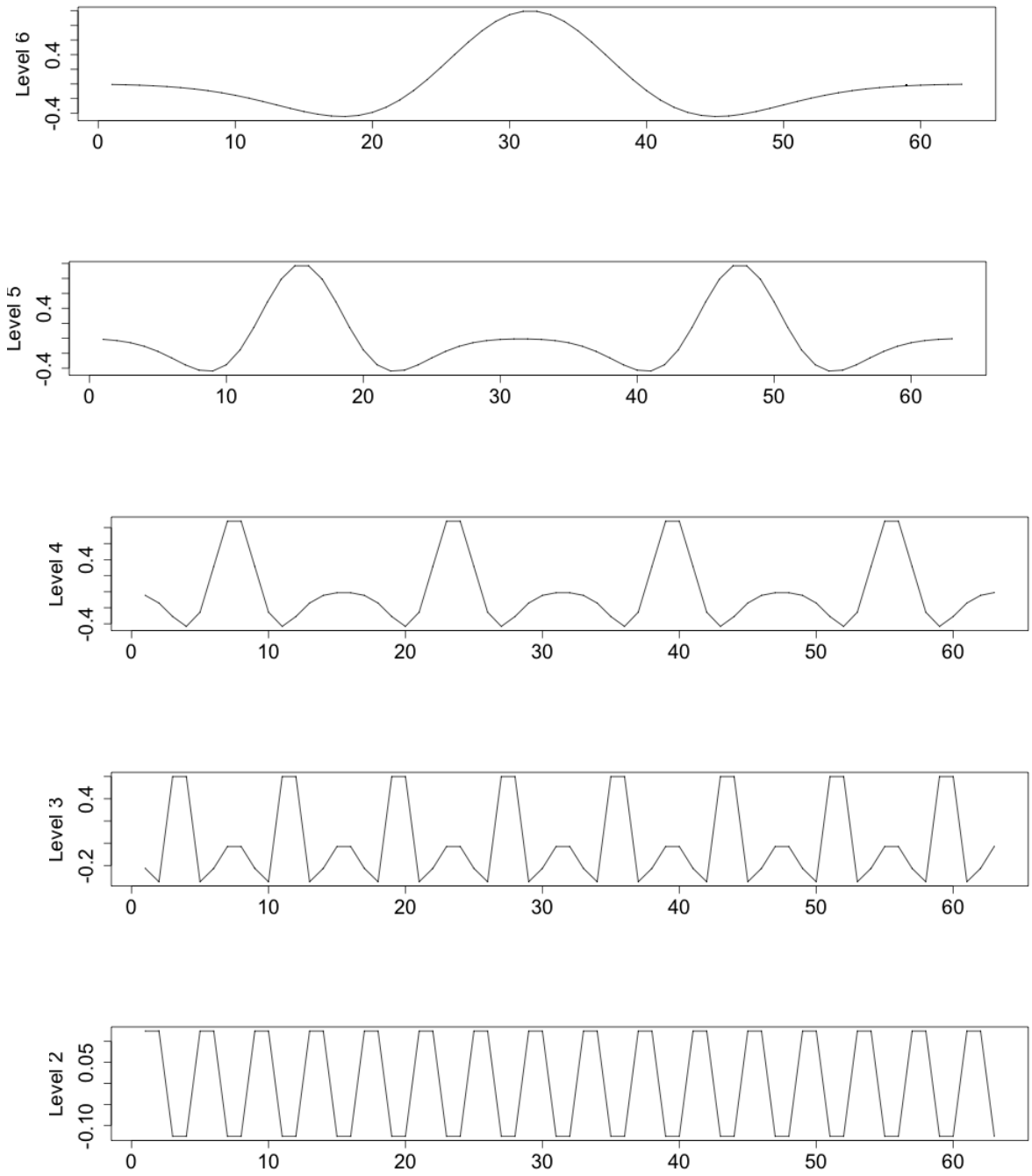


Fig. 2.— Schematic diagram of the popular Mexican Hat wavelet 'atom', at multiple resolutions.

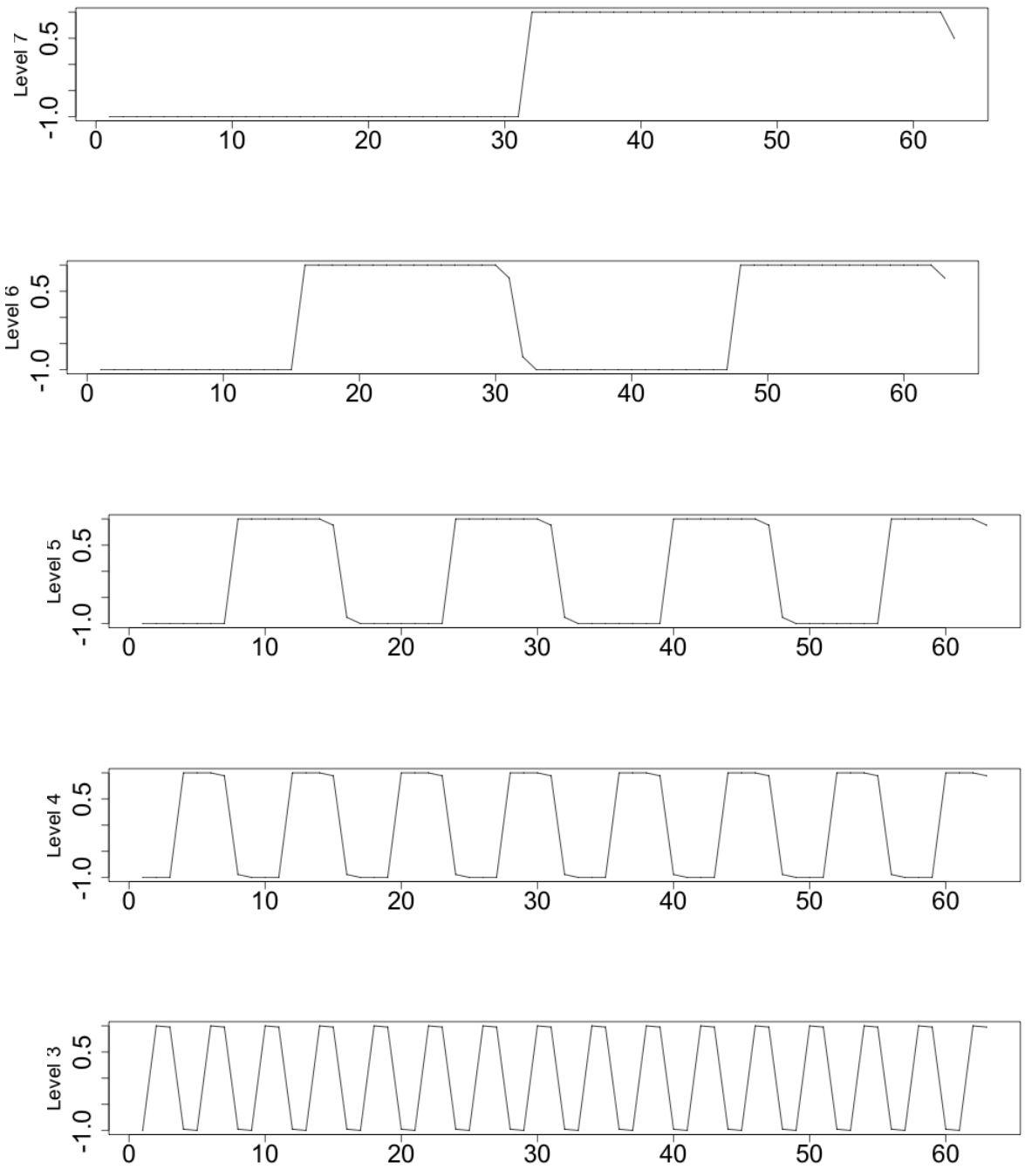


Fig. 3.— Schematic diagram of the Haar wavelet 'atom', at multiple resolutions.

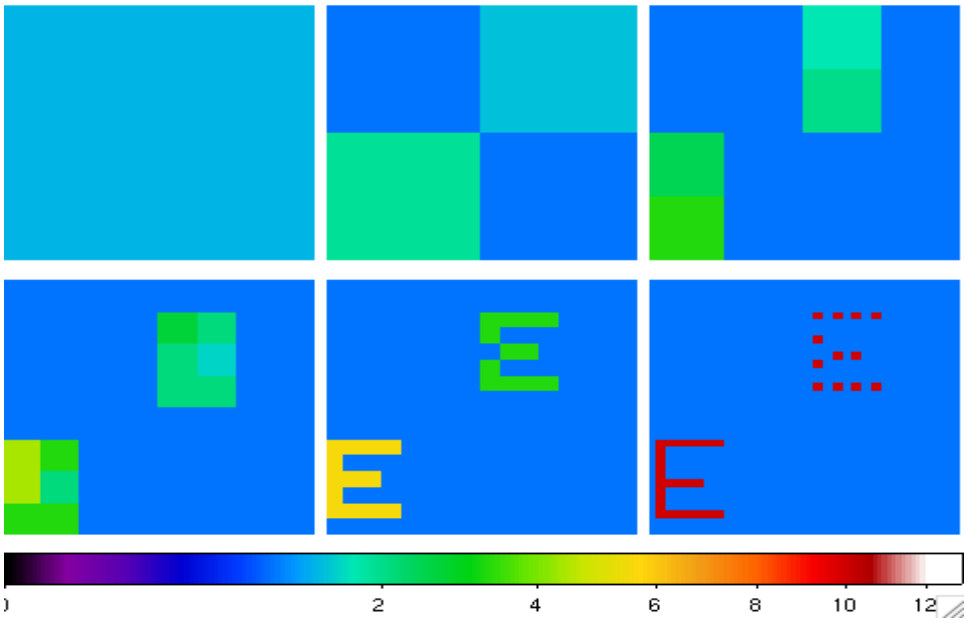


Fig. 4.— **An example of multi-scale methods and partitioning.** Here we illustrate mother-daughter bi-dyadic partitioning traditional in multi-resolution methods. Here, the color of each block simply shows the average counts per pixel in that block. We have used our "true" model as an example. The 1st image shows the average counts over the whole image.

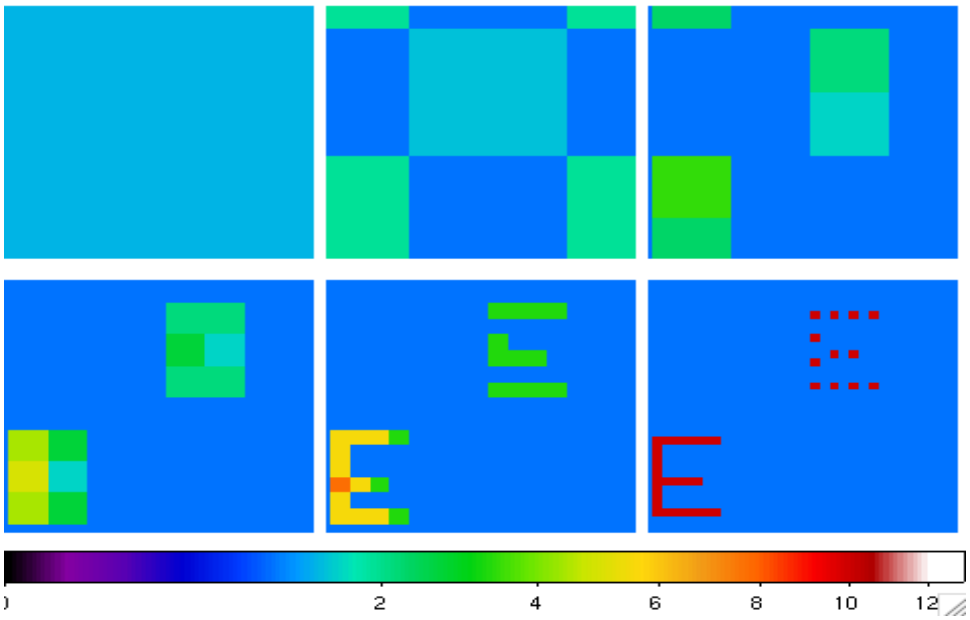


Fig. 5.— Here we illustrate the effect of changing the origin to (9,13) (that is, 'cycle-spinning') on the mother-daughter bi-dyadic partitioning, of our "true" model. The 1st image shows the average counts over the whole image.

5. Basic Markov Chain Monte Carlo: An Overly Simple "E" Example

What it is. **ADD DESCRIPTION HERE OF 'BAYES UMBRELLA'**

$$P(Y_{i+1}|\Theta_i, Inst, Data)$$
$$P(\Theta_{i+1}|Y_{i+1}, Inst, Data)$$

Simple, when not significant missing data.

Add steps for each piece: breaking into bacground and multiscale; prob of from which region of sky does it come, given PSF; etc.

Running it: Simple R-code for "E". In `docs/examples/`, you can find sample scripts for running $\mathcal{L}IRA$. In this section, we are using the 'No Background' script:

```
Simple00_deltaPSF_PoisDatons32x32EEMC2_NoBckgrnd_txt.R
```

(There is an equivalent one for fits rather than text format.) Notice there is no background matrix specified in the call to lira, below. Hence it will be set to zero, and 'the "fit.bkg.scale" parameter should be set to "FALSE". **Dudes, this is a small $\mathcal{L}IRA$ bug: when there is no background, I believe 'fit.bkg.scale' should be forced to be 'False'. Right now one can by accident still have it fit!!!** As well, there is no PSF specified; so the PSF will essentially be a Kronecker-delta (no smearing).

Is it overkill to have the R-scripts reproduced in the tutorial, as follows?

```
require(lira)
require(FITSio)

dataimage <- array(
   scan("exampledata/PoisDatonsEEMC2_32x32testE.090120c.txt"),
   c(32,32))

startimage  <- array(
  scan("exampledata/start32x32_1.00.txt"),
  c(32,32))

img <- lira(
```

```
obs.matrix = dataimage,
start.matrix= startimage,
out.file = "outputs/PoisDatons32x32EEMC2_NoBckgrnd.out",
param.file="outputs/PoisDatons32x32EEMC2_NoBckgrnd.param",
mcmc=TRUE, , fit.bkg.scale = FALSE,
max.iter=1000, thin=1,
burn=0,
alpha.init=c(3,4,5,6,7))
```

For this beginning example, we have set the MCMC parameters to their simplest form. There will be only 1000 iterations; the outputs will have no "burn-in" chopped off the beginning. Since "thin=1", every iteration will be written out. For more serious data analysis, one would often choose "thin" to be roughly the correlation length of the values of the parameters in the successive MCMC samples (often thin= 5 to 10).

Running it. ** PUT EXAMPLE FIGURES HERE OF EACH STEP OF THE MCMC ITRATIONS: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 40, 60, 80, 90, 100

Description of $\mathcal{L}IRA$ outputs: Log; images; params. **Put descriptions and references to the movies here.** "Movies" of the MCMC sample images can be viewed in:

```
docs/examples/outputs/PoisDatons32x32EEMC2_NoBckgrnd.out
```

and

```
docs/examples/outputs/PoisDatons32x32EEMC2_NoBckgrnd_2.out
```

These are the most fun – seeing the letters pop out of an initially uniform image. but how does one tell if they have converged and so forth?

Looking at "traces" of parameters/hyperparamters.

R-plot experts: These were done with an example shell-script that plots each trace individualy. I tried to get the multi-plot environment to work, but failed. Any suggestions? The script used to make these plots is in:

```
docs/examples/postprocessing/sample_Rplot_uni_Param_traces.sh
```

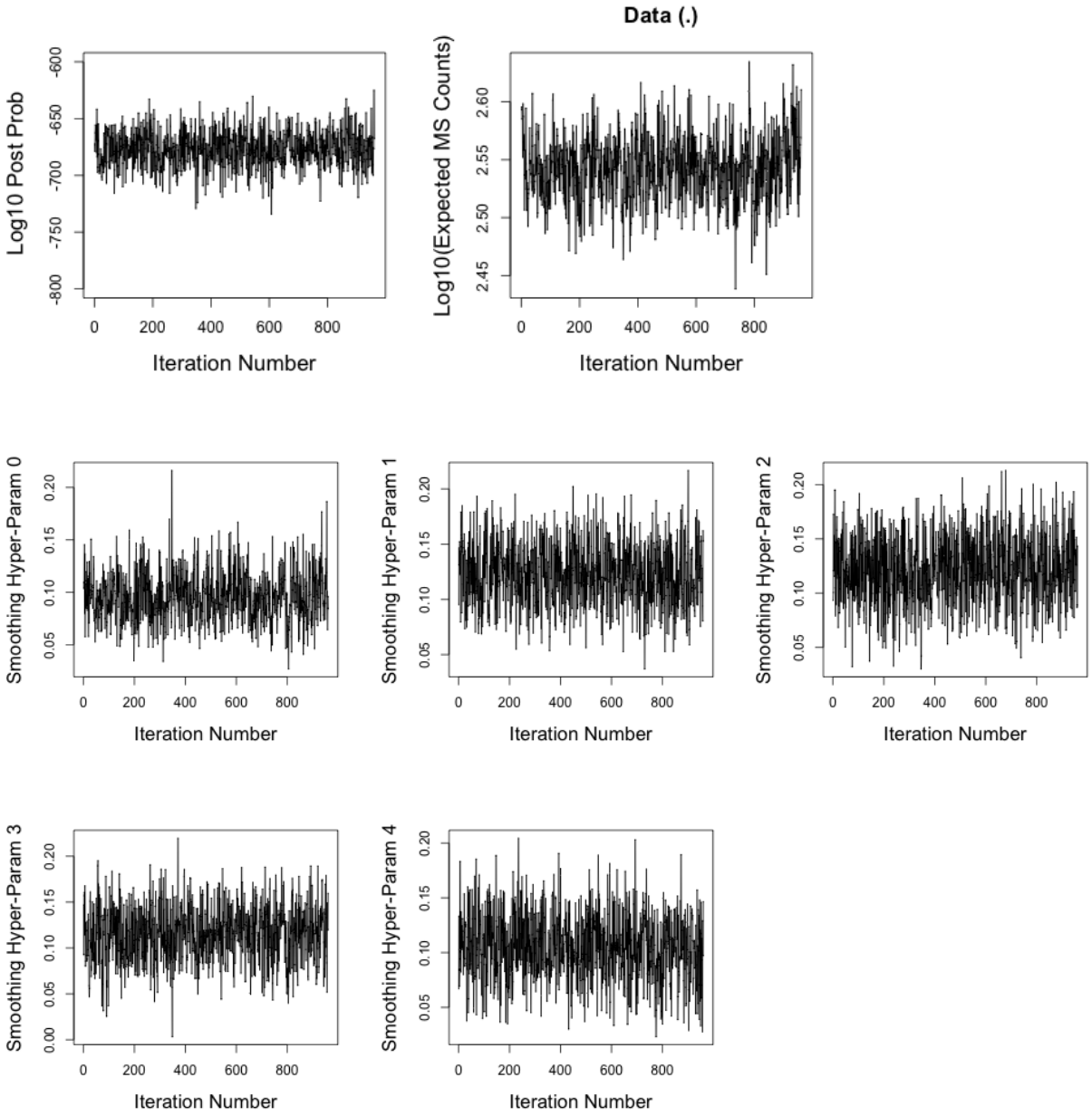


Fig. 6.— Trace, or 'time-series' plots of the values of each MCMC fit-parameter as a function of iteration number.

Making sense of MCMC results: convergence and 'burn-in'. The number of iterations until the MCMC chain converges is called the 'burn-in'. By eye, one can see that the chains converge at somewhat less than 100 iterations. In practice, one wants a minimum of three chains to compare. In this simple case, it is pretty easy to see by eye. For more rigorous

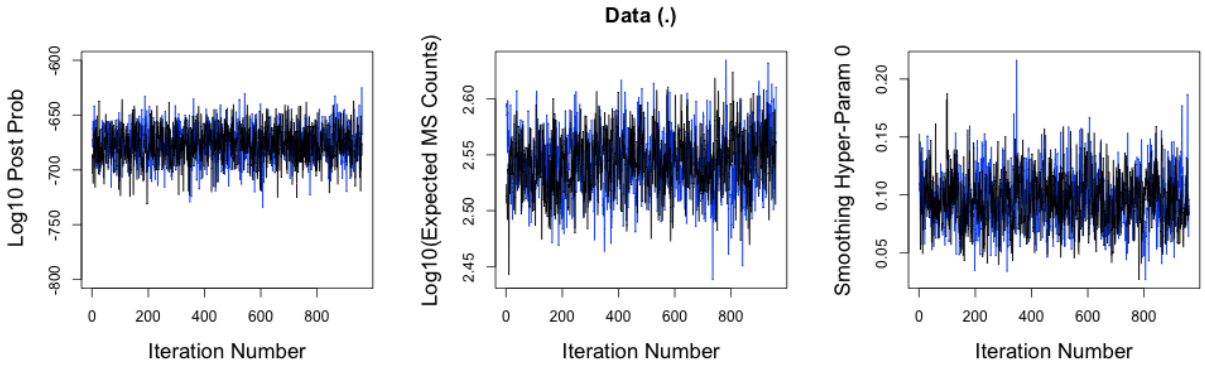


Fig. 7.— Comparison of traces for two different starting values (high=blue; low=red).

comparisons, once can use the R-statistic [**REF? I actualy usually do it by eye!**]

Special Note to Users: Although this is only important when running with a non-zero background model, it is often a good idea to include MCMC runs with: 1) starting values too high (i.e. puts all counts in the MS component, to start); and 2) starting values very low (i.e. puts all counts in the background component, to start). If there is going to be any problem with multiple modes, this can illustrate it rather quickly.

Making sense of MCMC results: correlation length and 'thinning'. More words about how draws can be corelated, and want independent draws.

Making sense of MCMC results: Quantifying uncertainty, Part 1.

Check the scatter plots: Next, we look at the scatter plots of the various parameters. Scripts for plotting are in:

`Simple00_Rplot_uni_Param_scatterplots.sh`

Any obvious problems? We will (eventually) use one or more of these as a *summary statistic*. Each point on the scatter-plot corresponds to a particular MCMC image. By selecting only certain ranges of our summary statistic (say, $\pm 5\%$), we can set quantitative limits on (say) the brightness, extent, shape of our possible diffuse emission.

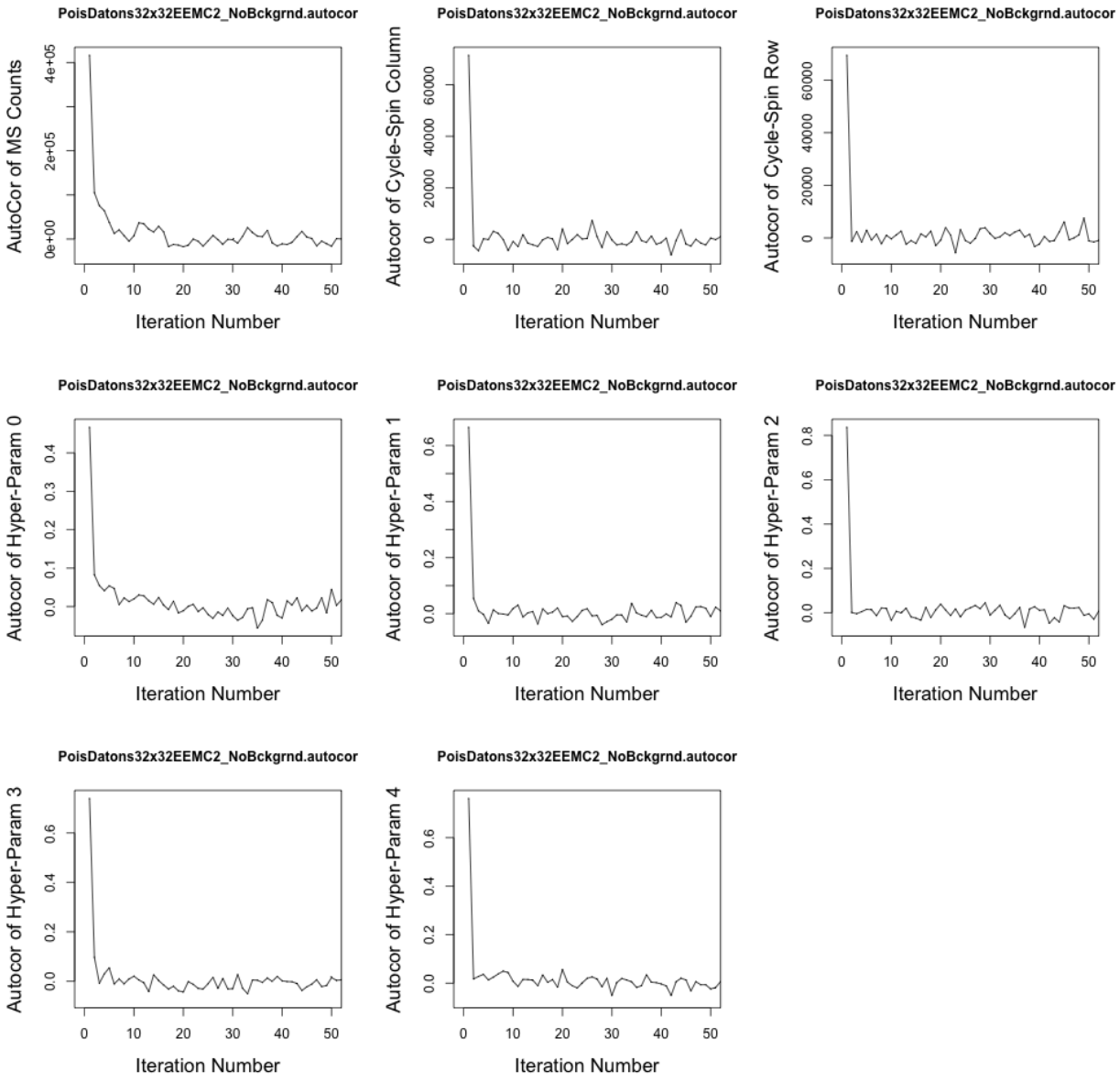


Fig. 8.— Autocorrelation plots for each MCMC fit-parameter as a function of iteration number. After an (abitrary) burn-in of 200 was chopped off, the mean of each parameter was calculated, and the value of its correlation with itself, as a functon of lag, was calculated. Notice that for our 'No Background' case, the correlation length (i.e. the lag at which the value drops to $1/e$ of its starting value) is only a few. **R-experts: These simple autocorrelations were calculated using a couple of Python twiddles. There must be a simple way to do it in R – but I couldn't find it.**

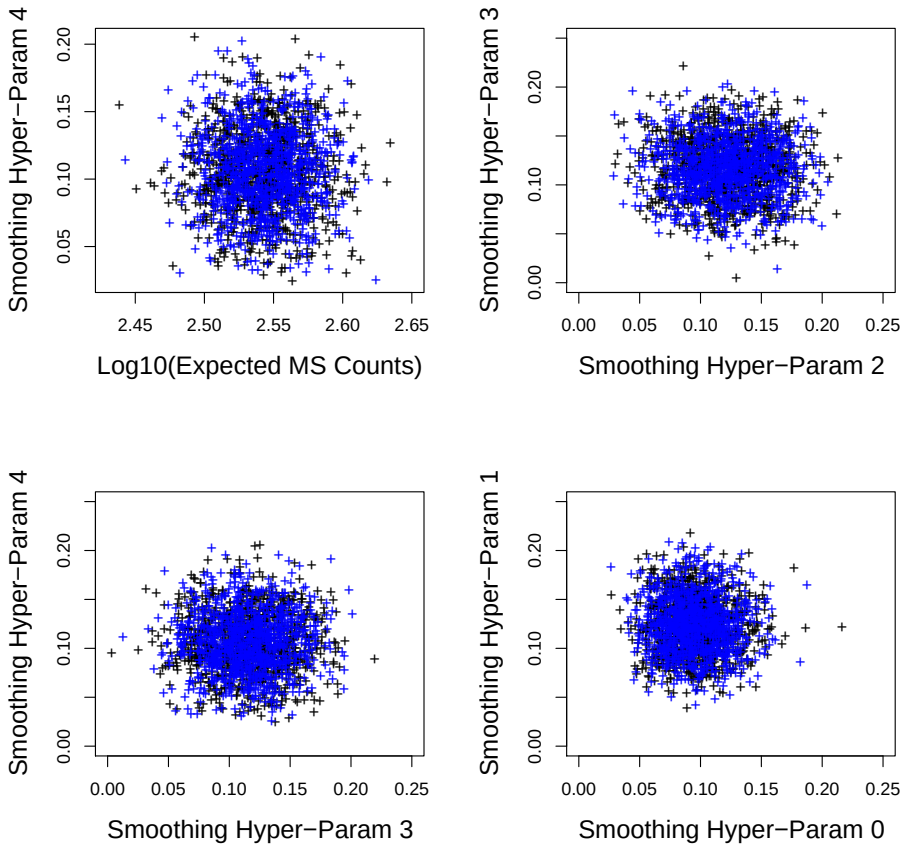


Fig. 9.— Scatter plots of the parameter values for two runs, with different starting values, with a conservative burn-in of 200 chopped off. The blue and black designate runs with low and high starting values, respectively. **What about Hyunsook's nice way of looking at them via convex hull-peeling? Is there an R-script for that?**

However we are actually interested in the images, more than these parameters. By eye, when viewing the movies, one sees both persistent features popping out, and variability around these persistent features. How does one summarize and quantify both?

The next figure displays the mean of each pixel, over all the MCMC sample images, after the burn-in has been chopped off. This is our choice for the 'best-fit' result. For fun and visualization, we also display the standard deviation, skew, and kurtosis. Since by construction, the pixels near each other are *anti-correlated*, pixel-by-pixel representations such as these are not useful quantitative summaries of uncertainty in $\mathcal{L}IRA$ results.

FIGURE OF MEAN HERE!!

XXX STRAIGHEN THIS OUT Instead, we use more global *summary statistics*. In the following figure, we display **(hull-peeling contours?)** contour plots and histograms of the overall fit-parameters. Each value of, say, the Expected MS Counts, corresponds to a small subset of images in our MCMC samples. If we choose the Expected MS Counts as our *summary statistic*, then by selecting, say, the upper and lower 5% of its distribution, we have selected an upper and lower 5% of the sample images as well. (See the histogram in figure ...).

Although the Expected MS Counts is probably the most common summary statistic (corresponding physically to: is there an extra component or source in my data that is not acounted for by my model), it is certainly not the only possible choice. Could use any of the other hyper-parameters; could use the ratios; could us any function of all of them. More examples of this will come in the next section.

6. Markov Chain Monte Carlo: More Realistic

Now we suppose that a clever theorist has come up with a good model for the 'diffuse E' present in the lower left-hand side. This model can be found in:

`exampledata/NullModelEEMC2_32x32testE.090120c.fits, or`
`exampledata/NullModelEEMC2_32x32testE.090120c.txt`

This "Null Model" could be: 1) a purely theoretical model; or 2) the results of a fit to other data OR to these data; or 3) the mean MS component (i.e. output from $\mathcal{L}IRA$) from an earlier time or a different energy band. It could also be: 4) a background model. This is now our best-fit model, or null model, or base-line model. We will use $\mathcal{L}IRA$ to decide if it is a good-enough fit to our data; and if it is not, to quantify the shape of any data-vs-sky-model mis-match.

To run $\mathcal{L}IRA$, we will follow same format as previous section but with null/background incorporated. Note that we are now fitting background pre-factor.

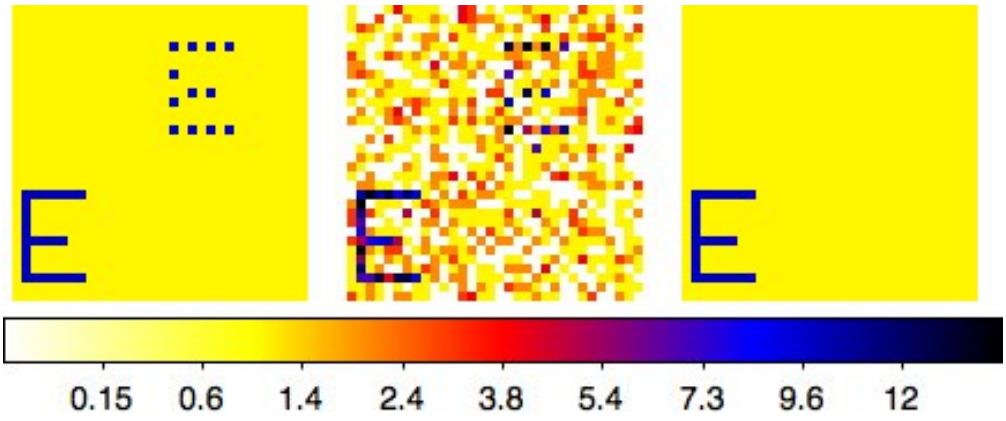


Fig. 10.— **Comparing Data with a Null Model.** Again, we show our Sky-Truth (left-most panel), and our data(center). The Null model we have chosen to use in this section is shown at the right.

6.1. Check Using Short Preliminary Runs:

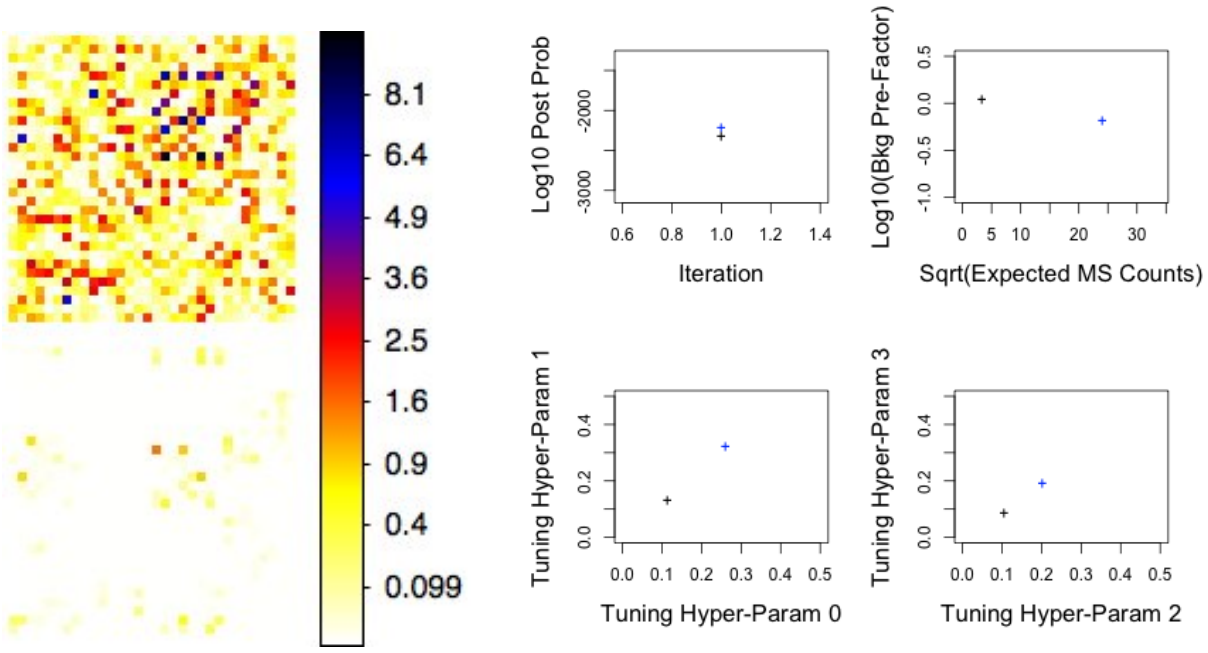
R-script for this sub-section can be found in:

`docs/examples/Simple02_Prelim_deltaPSF_PoisDatons32x32EEMC2vsNullModel_fits.R`

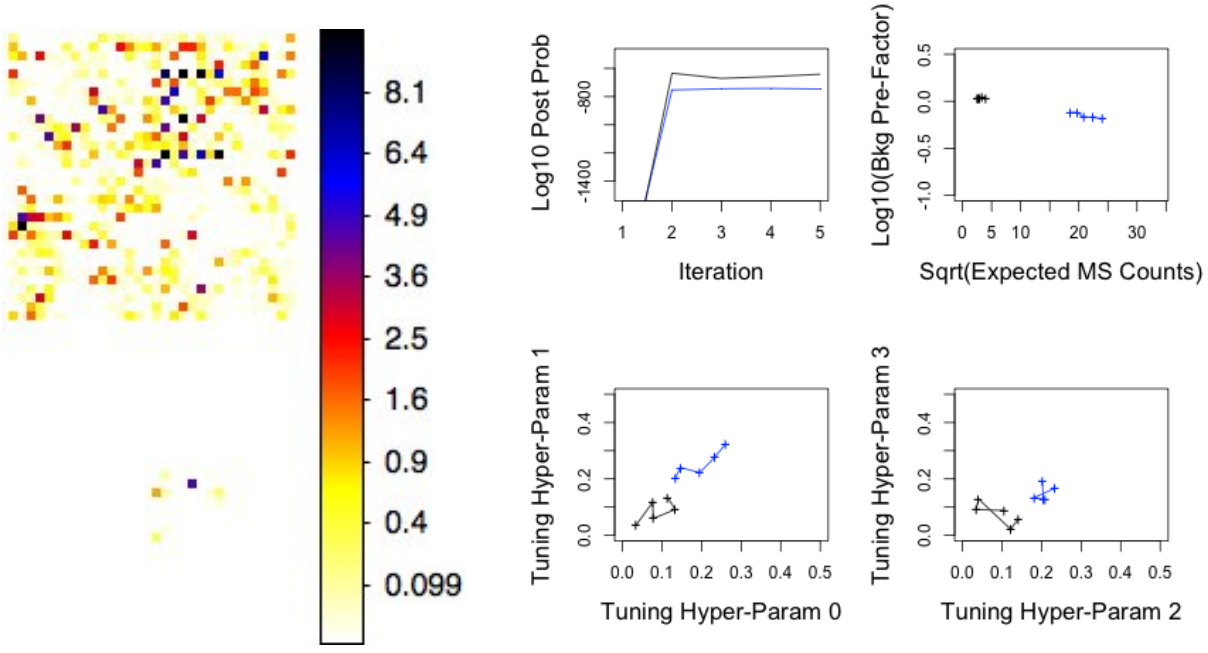
For a serious analysis, one starts with a few short runs, with a bare minimum of three separated starting values. In our examples, we display two runs: one with a starting map with overly high values for the Expected Multi-scale counts component (flat map, a value of 1.00 in each bin); and one with overly low values (flat map, a value of 0.01 in each bin).

Check the image samples: The most interesting way to visualize the convergence is to watch a movie of the MCMC image samples from each run. The "out" (image) files for the MCMC runs that started too high and too low are contained, respectively, in:

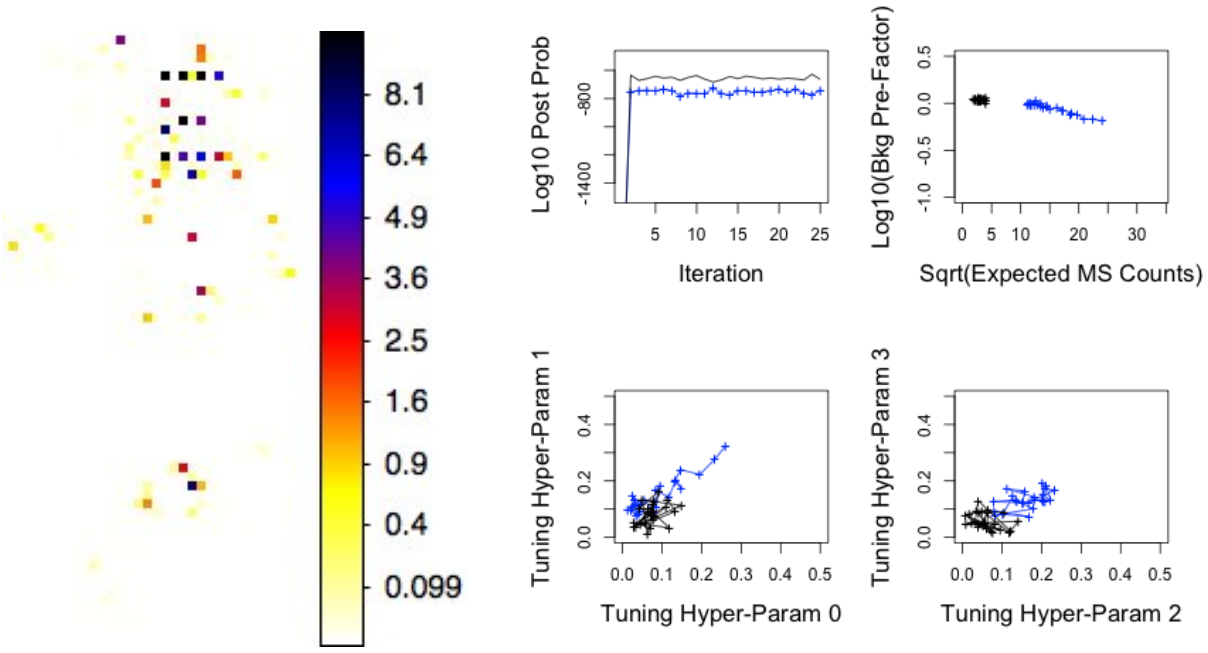
`docs/examples/outputs/Prelim_PoisDatons32x32EEMC2vsNullModel_Strt1.00_viaFits.out`
`docs/examples/outputs/Prelim_PoisDatons32x32EEMC2vsNullModel_Strt0.01_viaFits.out`



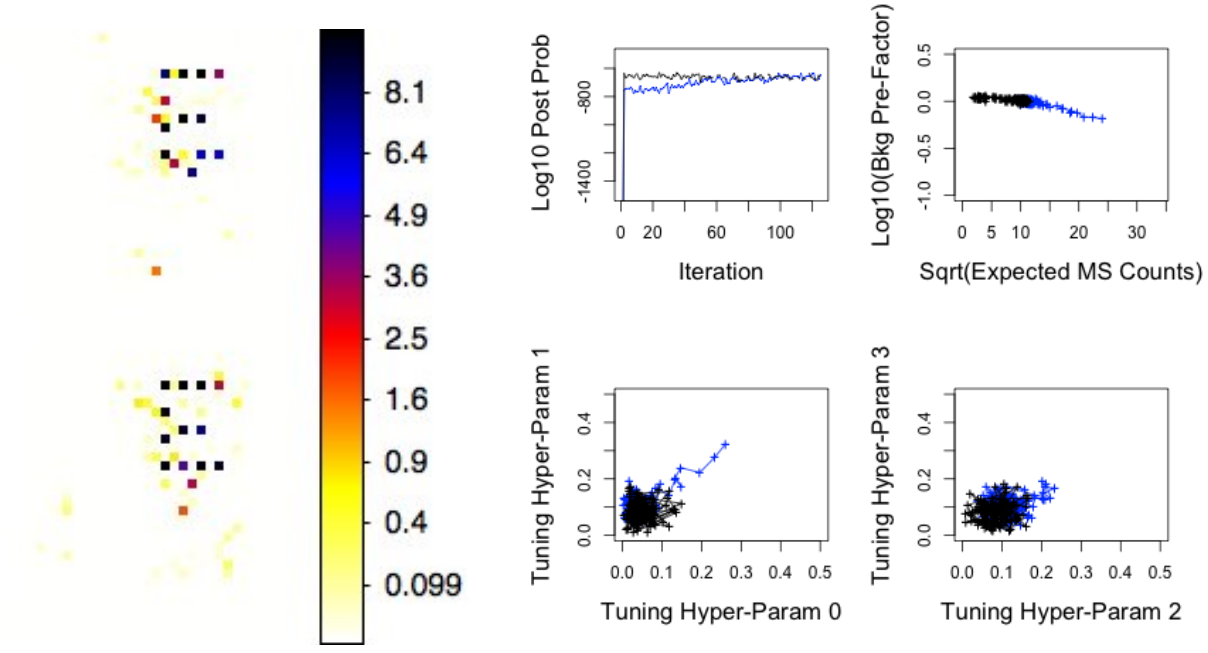
Iteration 001 Two start values: high (top image; blue); low (bottom image; black).



Iteration 005 Two start values: high (top image; blue); low (bottom image; black).



Iteration 025 Two start values: high (top image; blue); low (bottom image; '+').



Iteration 125 Two start values: high (top image; blue); low (bottom image; '+').

Check the traces: For quantitative comparisons, one starts with the parameter values of the MCMC samples. These are contained in:

`docs/examples/outputs/Prelim_PoisDatons32x32EEMC2vsNullModel_Strt1.00_viaFits.param`
`docs/examples/outputs/Prelim_PoisDatons32x32EEMC2vsNullModel_Strt0.01_viaFits.param`

From comparing the traces (plots of parameter values versus iteration number), one gets the burn-in (number of iterations until convergence). In this example, in figure **?** one sees a burn-in of 200 would be conservative. That is what we shall use.

Check correlation length: Next, we examine the autocorrelation of the parameters. How 'sticky' are our MCMC chains? What is the correlation length? From figure **?**, one sees that the correlation length of the multi-scale component and the background pre-factor (the 'stickiest' parameters) are just a few. (Scripts for plotting are in:

`Simple02_Prelim_Rplot_uni_autocor.sh`

)**Should add how-to-do autocor in "R" rather than python...** Therefore, if we 'thin' by accepting only every 5th iteration, we can be sure they are independent draws. This is what we will use for our longer, more serious, analysis runs.

Check the scatter plots: It is also a good idea to look briefly at the scatter plots of the various parameters. Scripts for plotting are in:

`Simple02_Prelim_Rplot_uni_Param_scatterplots.sh`

Any obvious problems?

As well, to see that the results make sense, we display the resulting preliminary image mean and higher moments, as in figure **?**. We emphasize that these are for visualization purposes only. Because of the anti-correlation structure of our multi-scale model, we cannot use pixel-by-pixel means or moments to get serious quantitative limits on what is 'real' in an image. Instead, later, we will use summary statistics (tagged to each sample image). The sample images are sorted on this summary statistic (or statistics). Hence the upper and lower limits of the summary statistic(s) will correspond to sets of images – which gives one the upper and lower limits on the shape and intensity of any multi-scale component.

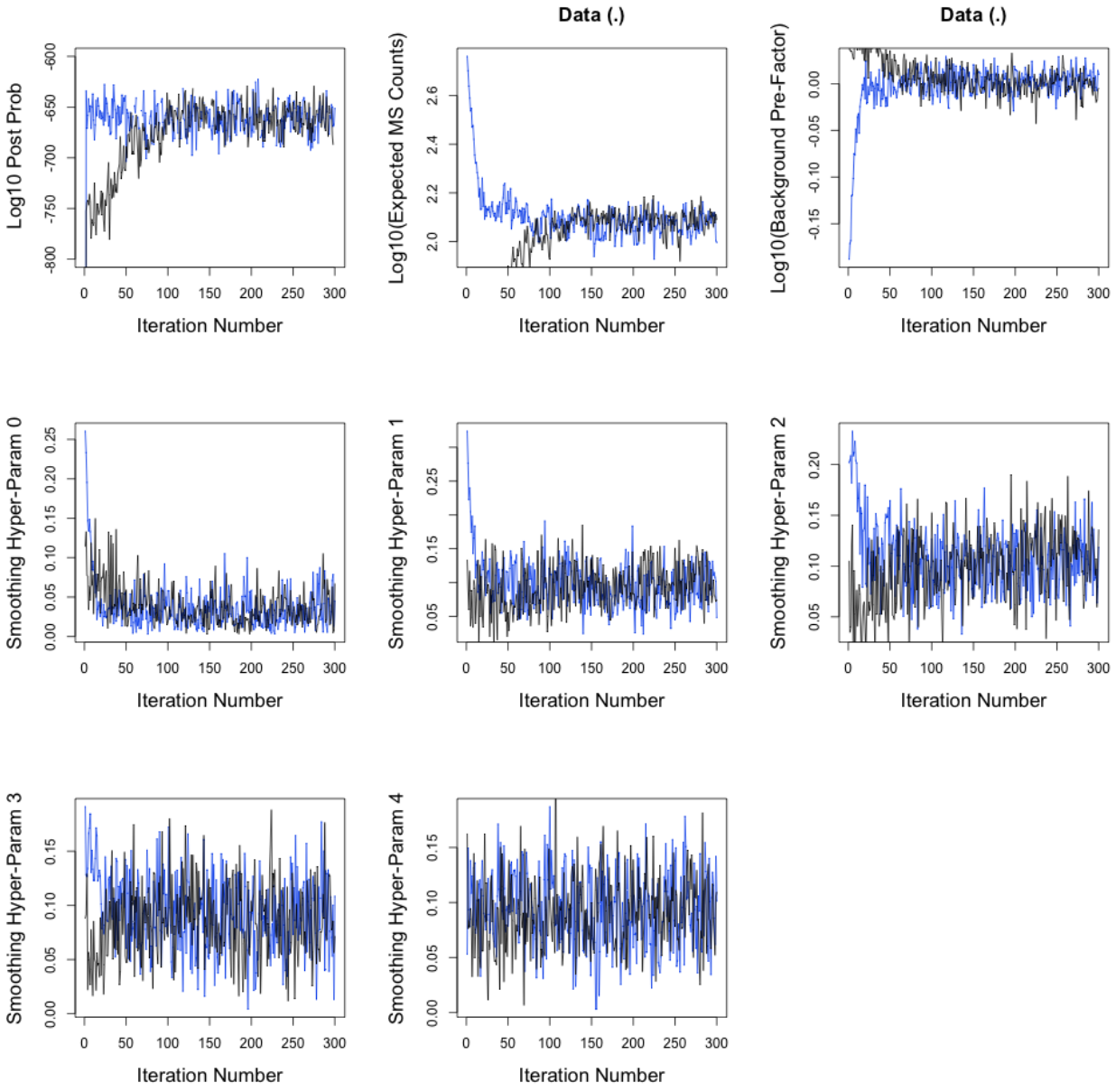


Fig. 11.— Plots of the traces for two short, preliminary runs, with different starting values, showing the values of each MCMC fit-parameter as a function of iteration number. Black indicates high starting values, while blue indicates low. These will give us the burn-in (about 150 to 200) and autocorrelation (next figure).

6.2. Longer, More Serious Analysis Runs:

We see that it is very quick to run these. (The CPU time is recorded at the end of the '*.Rout' files.) So we feel free to do much longer runs, with 2000 iterations each. Notice

these are run with thin=5, and with burnin (used in post-MCMC analysis) of 200. **Q for Nathan: is "burnin" in the R $\mathcal{L}IRA$ call used only to make the figures in the R-run?**

R-script for this sub-section can be found in:

`docs/examples/Simple02_deltaPSF_PoisDatons32x32EEMC2vsNullModel_fits.R`

For this part of the serious analysis, we use longer runs, again with a minimum of three separated starting values. In our examples, we display two runs: one with a starting map with overly high values for the Expected Multi-scale counts component (flat map, a value of 1.00 in each bin); and one with overly low values (flat map, a value of 0.01 in each bin).

Check the image samples: The most interesting way to visualize the convergence is to watch a movie of the MCMC image samples from each run. Recall that these images are now 'thinned', so one sees only every 5th MCMC draw. The "out" (image) files for the MCMC runs that started too high and too low are contained, respectively, in:

`docs/examples/outputs/PoisDatons32x32EEMC2vsNullModel_Strt1.00_viaFits.out`
`docs/examples/outputs/PoisDatons32x32EEMC2vsNullModel_Strt0.01_viaFits.out`

INCLUDE FUN MOVIES OF MCMC RUNS CONVERGING AND CYCLE-SPINNING?

Check the traces: For quantitative comparisons, one starts with the parameter values of the MCMC samples. These are contained in:

`docs/examples/outputs/PoisDatons32x32EEMC2vsNullModel_Strt1.00_viaFits.param`
`docs/examples/outputs/PoisDatons32x32EEMC2vsNullModel_Strt0.01_viaFits.param`

We also display the traces of our runs on simulated null data. (More on these later.) **??** one sees a burn-in of 200 would be conservative. That is what we shall use.

Check correlation length: Next, we re-examine the autocorrelation of the parameters. How 'sticky' are our MCMC chains? What is the correlation length? from figure **?**, one sees that the correlation length of the multi-scale component and the background pre-factor (the 'stickiest' parameters) are just a few. (Scripts for plotting are in:

`Simple02_Prelim_Rplot_uni_autocor.sh`

Note that we have thinned by accepting only every 5th iteration, we can be sure they are independent draws.

Check the scatter plots: It is also a good idea to look again at the scatter plots of the various parameters. Scripts for plotting are in:

`Simple02_Rplot_uni_Param_scatterplots.sh`

Any obvious problems?

As well, to see that the results make sense, we display the resulting preliminary image mean and higher moments, n figure **?**. We emphasize that these are for visualization purposes only. Because of the anti-correlation structure of our multi-scale model, we cannot use pixel-by-pixel means or moments to get serious quantitative limits on what is 'real' in an image. Instead, later, we will use summary statistics (tagged to each sample image). The sample images are sorted on this summary statistic (or statistics). Hence the upper and lower limits of the summary statistic(s) will correspond to sets of images – which gives one the upper and lower limits on the shape and intensity of any multi-scale component.

6.2.1. Comparing data with null model for goodness of fit

Now, we create a number of MC Poisson realizations of our null, or baseline (i.e our best-fit model). These can be found in:

```
PoisNulDat0EEMC2_32x32testE.090120c.fits
PoisNulDat1EEMC2_32x32testE.090120c.fits
PoisNulDat2EEMC2_32x32testE.090120c.fits
PoisNulDat3EEMC2_32x32testE.090120c.fits
PoisNulDat4EEMC2_32x32testE.090120c.fits
PoisNulDat5EEMC2_32x32testE.090120c.fits
```

and the similar files in '*.txt' format. Here, we will display results on six Poisson realizations of our null, **?**. We will run $\mathcal{L}IRA$ on them in exactly the same way that we did on our data. Then we will compare the scatter plots. What summary statistic gives the best separation?

In figure **?**, one sees that, for the null model, there is a tendency for the smoothing hyper-parameters to cluster near zero, compared to those for the real data. However there is not clear separation between the two. The only parameter showing good separation is the total multi-scale counts. This is not surprising, as our data-vs-model mis-match is due to the existence of an extra component that is well-modeled by the multi-scale component.

6.3. Using Summary Statistics

Now we are in a position to ask: how much overlap is there between the Null and the Data distributions? This is what gives us our quantitative limits. **figures here.**

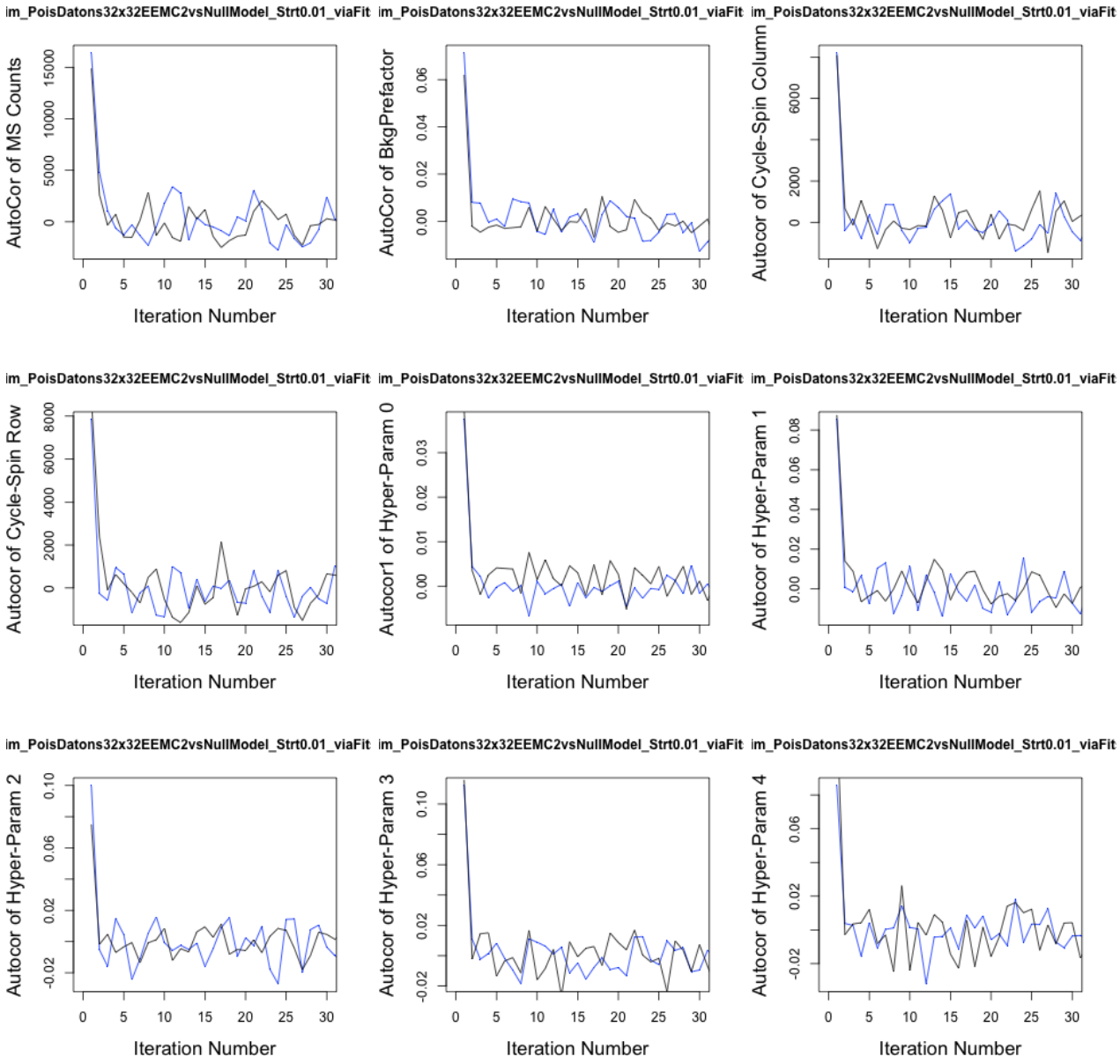


Fig. 12.— Autocorrelation parameter as a function of iteration number. After a conservative burn-in of 200 was chopped off, the mean of each parameter was calculated, and the value of its correlation with itself, as a functon of lag, was calculated. Even for expected Multi-Scale Counts, the correlation length (i.e. the lag at which the value drops to $1/e$ of its starting value) is only a few. Recall that the cycle-spin row and column are essentially completely random and have no correlation. It is interesting to note that the correlation lengths for all the smoothing hyper-parameters are only a few.

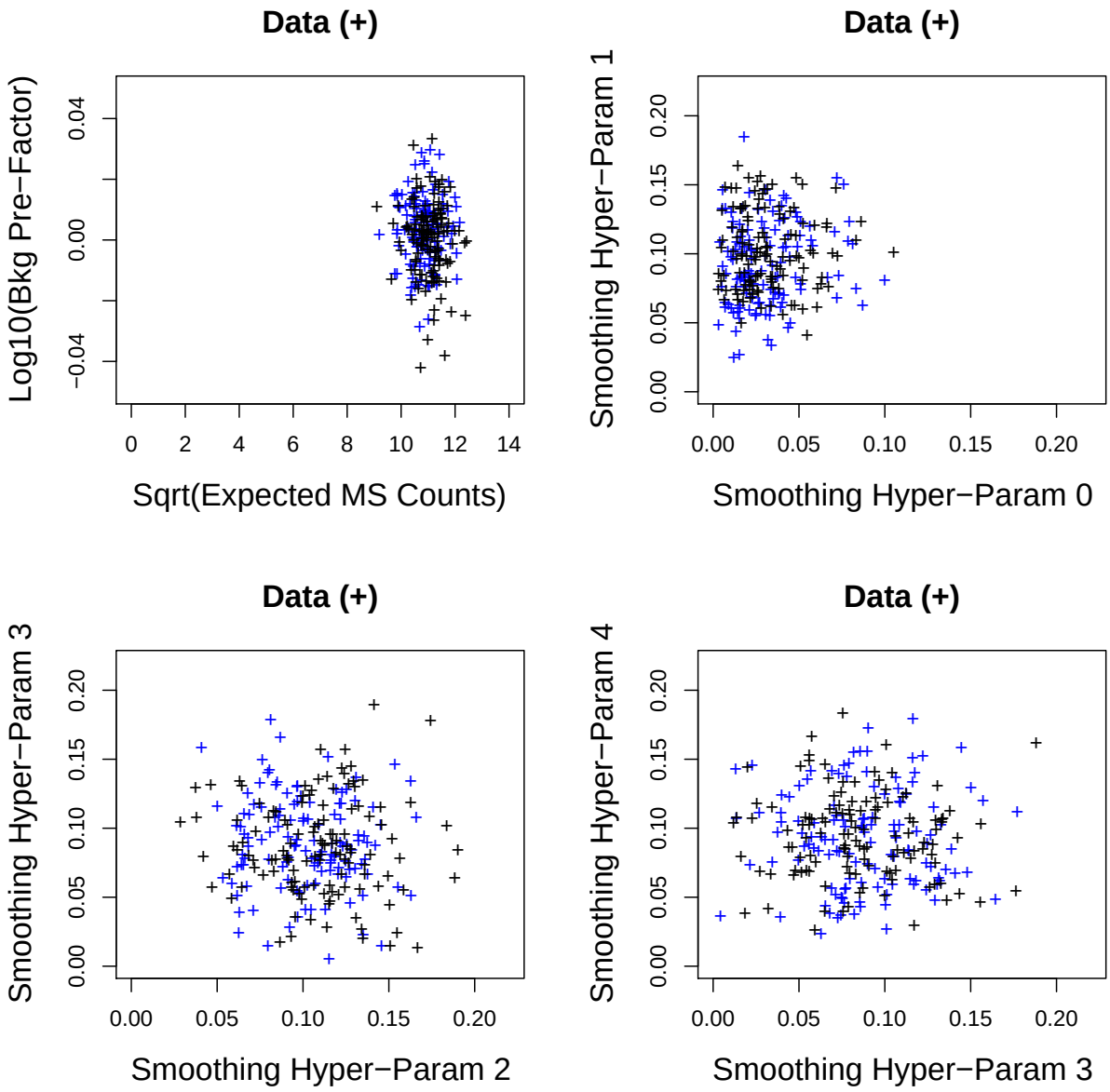


Fig. 13.— Scatter plots of the parameter values for two short, preliminary runs, with different starting values, with a conservative burn-in of 200 chopped off. The blue and black designate runs with low and high starting values, respectively.

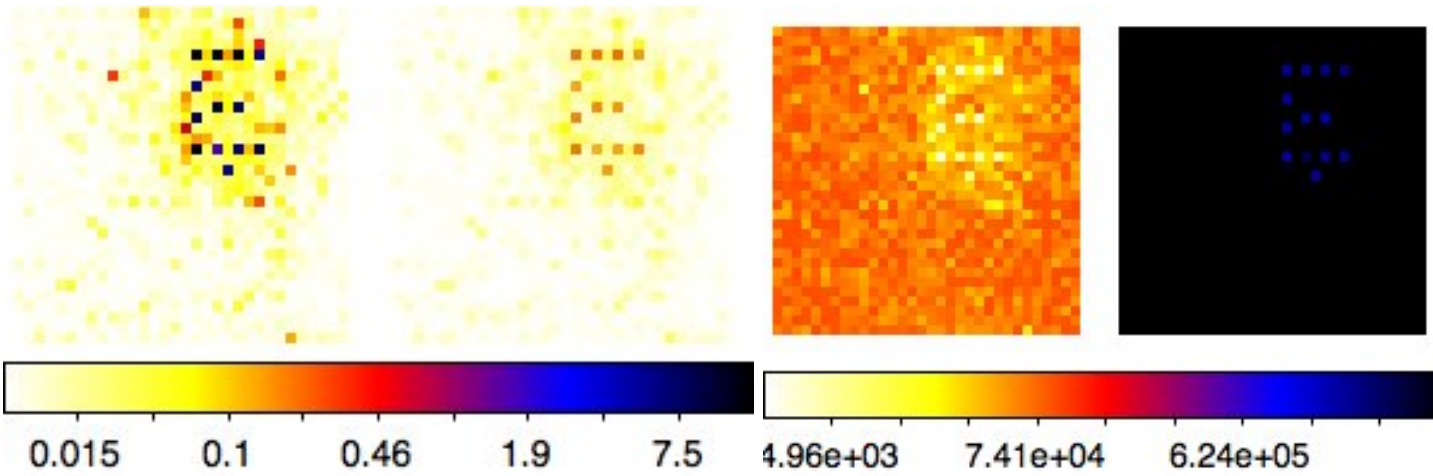


Fig. 14.— **Image Moments of Preliminary Runs for visualization.** Here, we display the mean, sigma, skew and kurtosis of our two preliminary runs (combined). The results look reasonable. (Recall that since, by construction, values in neighboring blocks are *anti-correlated*, these cannot be used for statistical significance.

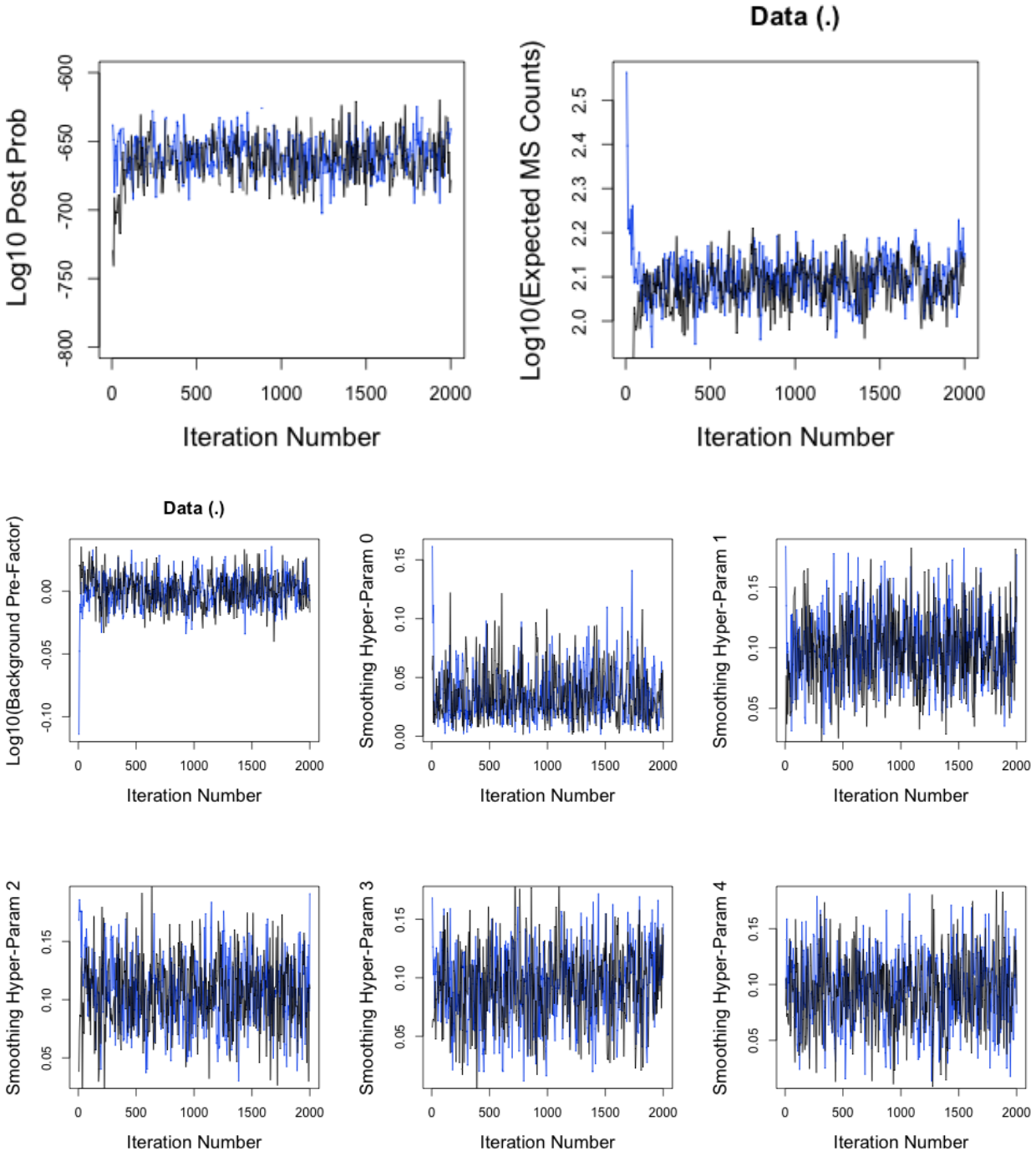


Fig. 15.— Plots of the traces for two of the longer analysis runs, with different starting values, showing the values of each MCMC fit-parameter as a function of iteration number. Black indicates high starting values, while blue indicates low. These again show a burn-in of about 150 to 200 and autocorrelation of a few draws (next figure).

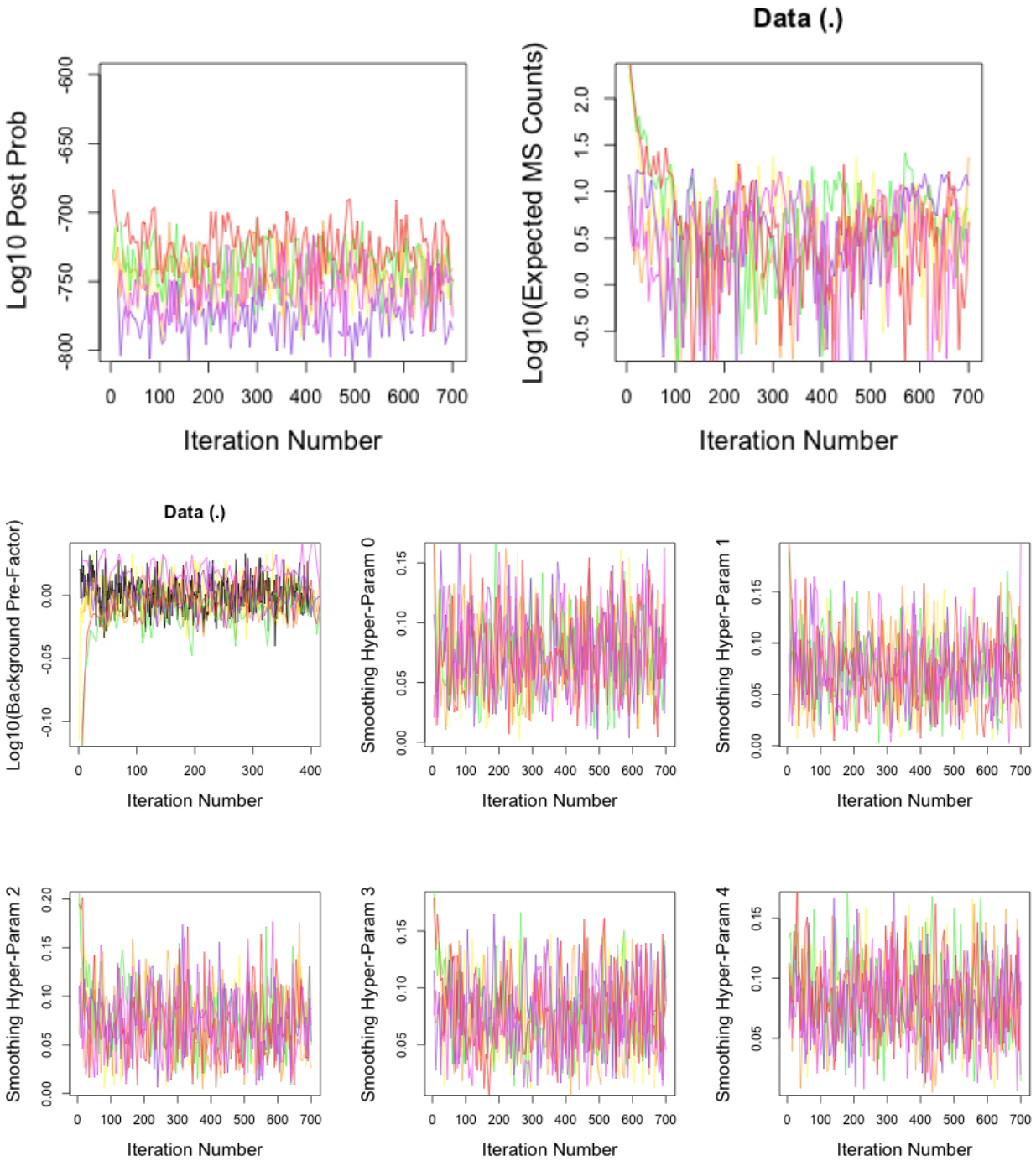


Fig. 16.— Plots of the traces for 6 runs on 6 null data-sets, with different starting values, showing the values of each MCMC fit-parameter as a function of iteration number. We us these to quantitative summaries to compare with the real data runs.

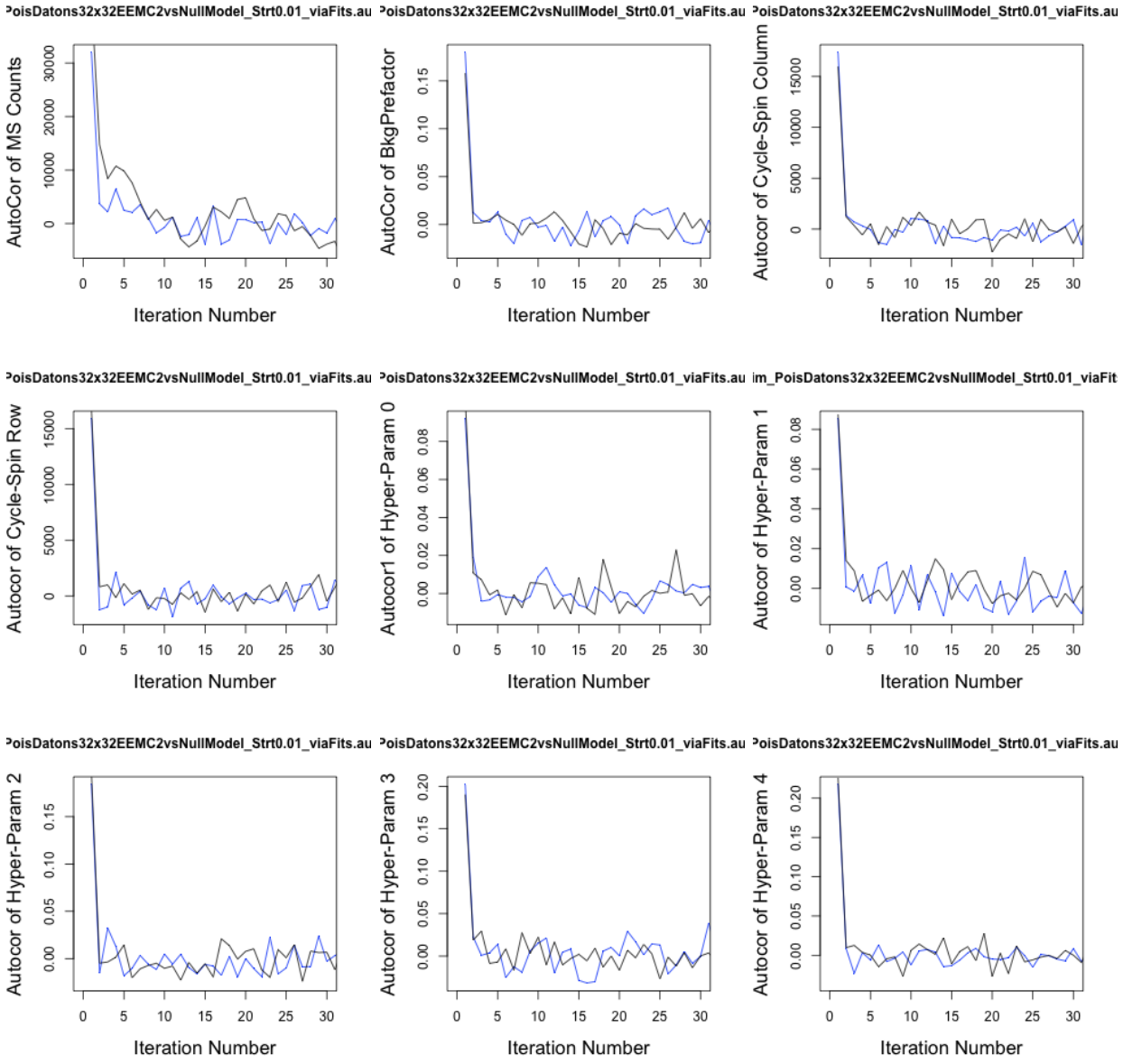


Fig. 17.— Autocorrelation parameter as a function of iteration number. After a conservative burn-in of 200 was chopped off, the mean of each parameter was calculated, and the value of its correlation with itself, as a functon of lag, was calculated. Even for expected Multi-Scale Counts, the correlation length (i.e. the lag at which the value drops to $1/e$ of its starting value) is only a few. Recall that this is only every 5th iteration. Also, recall that the cycle-spin row and column are essentially completely random and have no correlation.

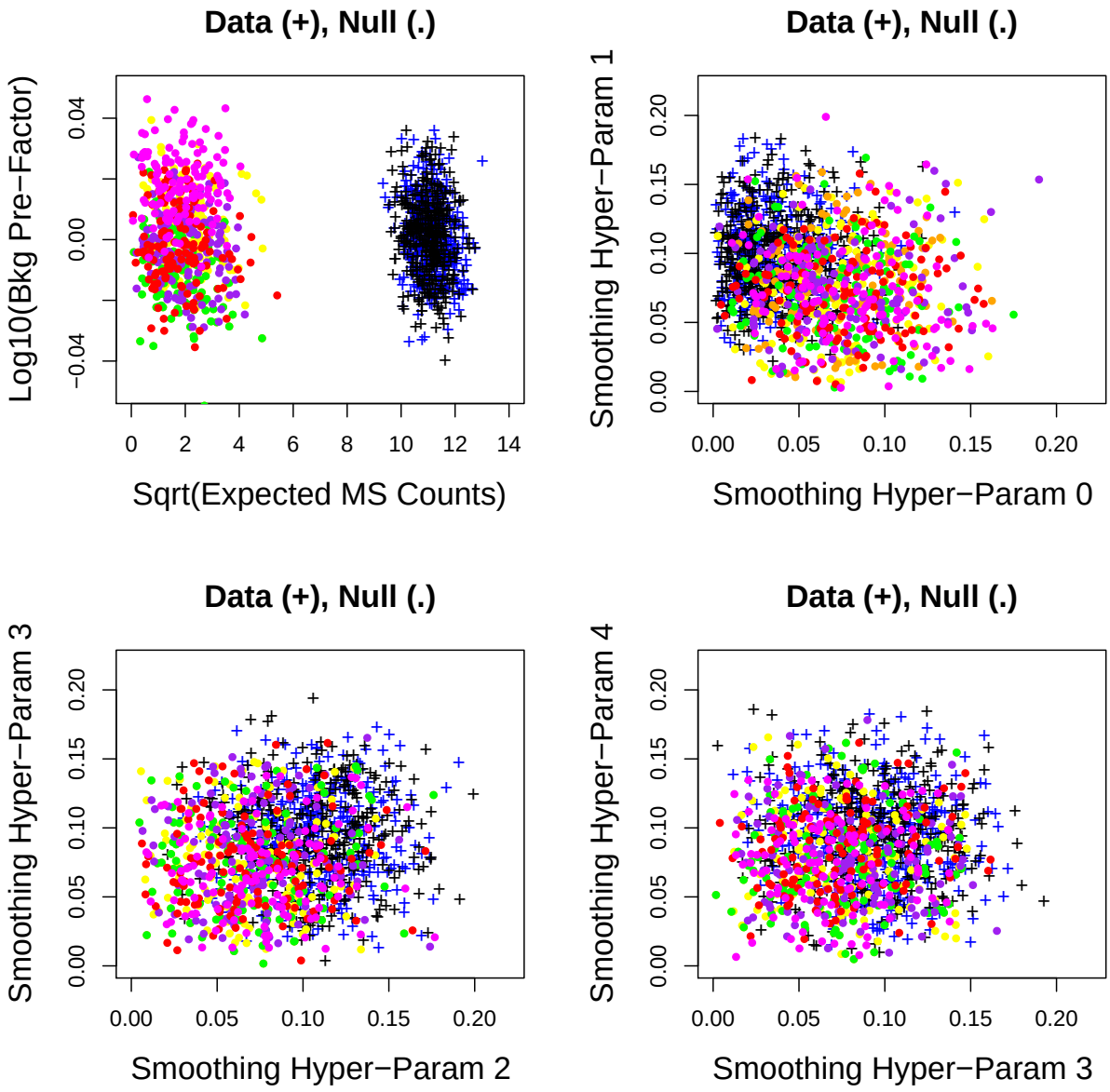


Fig. 18.— **Comparing Results on 'Interesting' and 'Null' datasets:** Scatter plots of the parameter values for two runs on the real data, with different starting values, with a conservative burn-in of 200 chopped off. The blue and black designate runs with low and high starting values, respectively. The results for null data sets are plotted in many colors: red, orange, yellow, purple, magenta.

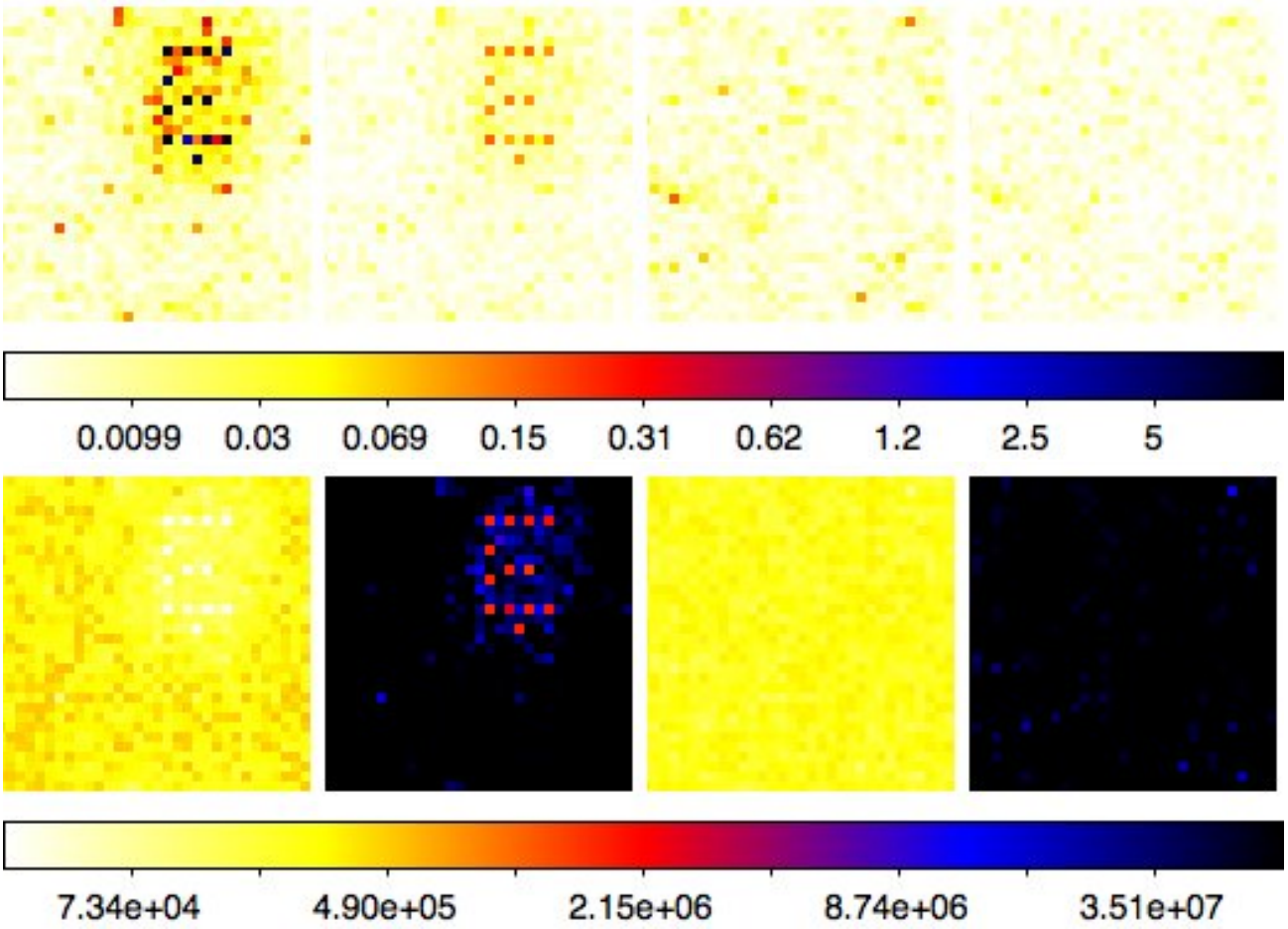


Fig. 19.— **Image Moments of Full Data And Null runs for visualization.** In the top panel, we display the mean and square root of the variance, respectively; for runs on the data (left two panels) and the six null simulations (right panels). In the bottom panel, we display the skew and kurtosis, respectively; again for runs on the data (left two panels) and the six null simulations (right panels).

7. McMC and Quantitative Limits: with PSF

Now we suppose that the previous 32×32 bin examples had actually been re-binned do that one did not need to take into account the instrument smearing, or point-spread-function (PSF). In this section, we simulate finer binning (128×128), and a simple Gaussian, spherical PSF with $\sigma = 1.5$bins.

We will run the same procedure as in the previous section, but now with the PSF specified.

Our new 'high resolution' data sets can be found in:

`exampledata/`

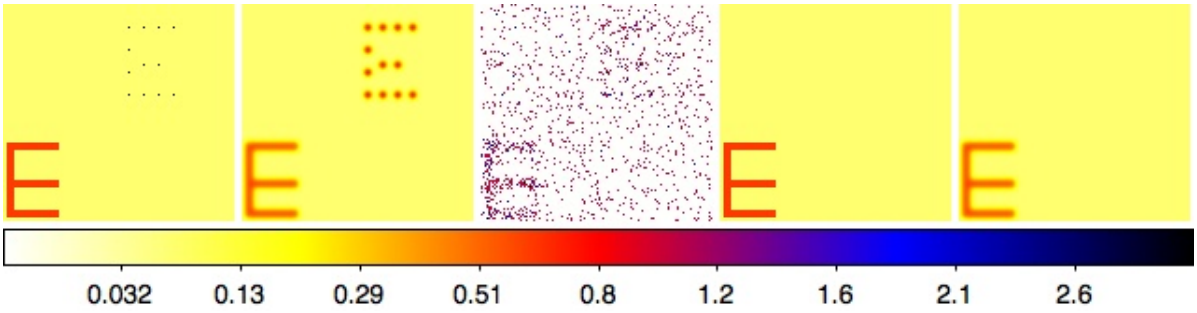


Fig. 20.— **Comparing Data with a Null Model.** As before, we show our Sky-Truth in the two left-most panels (1st uncolvolved, 2nd convolved with PSF); and our data(center). The Null model we have chosen to use in this section is shown in the two panels at right (1st uncolvolved, 2nd convolved with PSF).

7.1. Check Using Short Preliminary Runs:

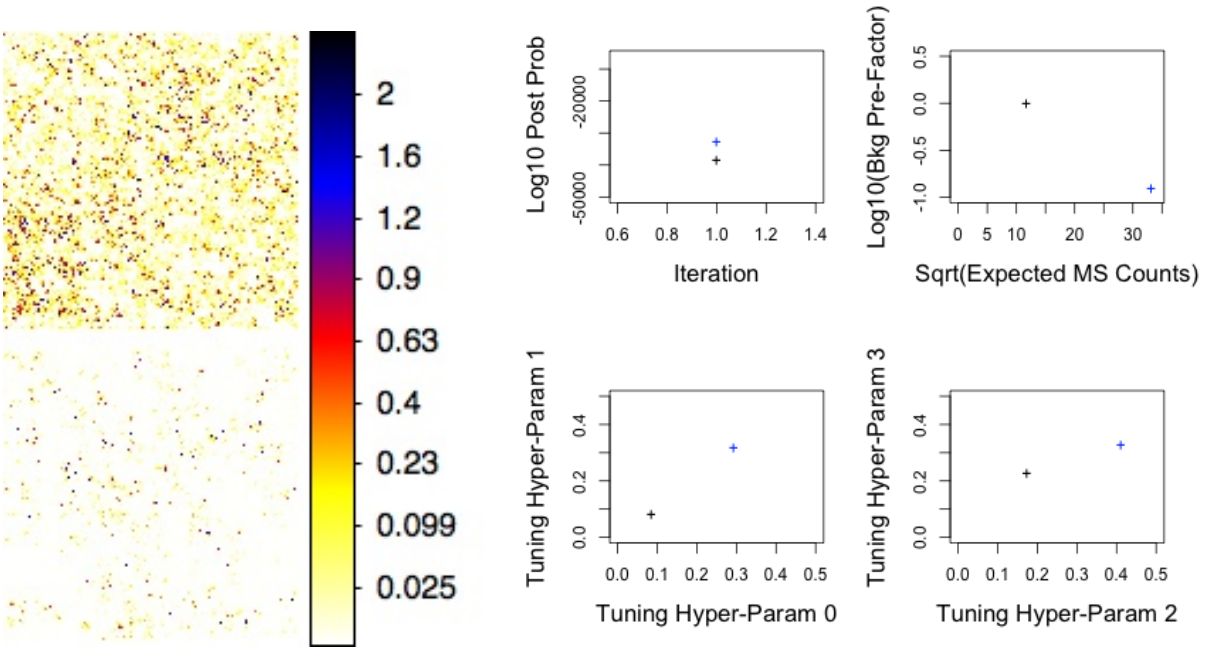
The script for the preliminary run can be found in:

`Simple04etcetc.R`

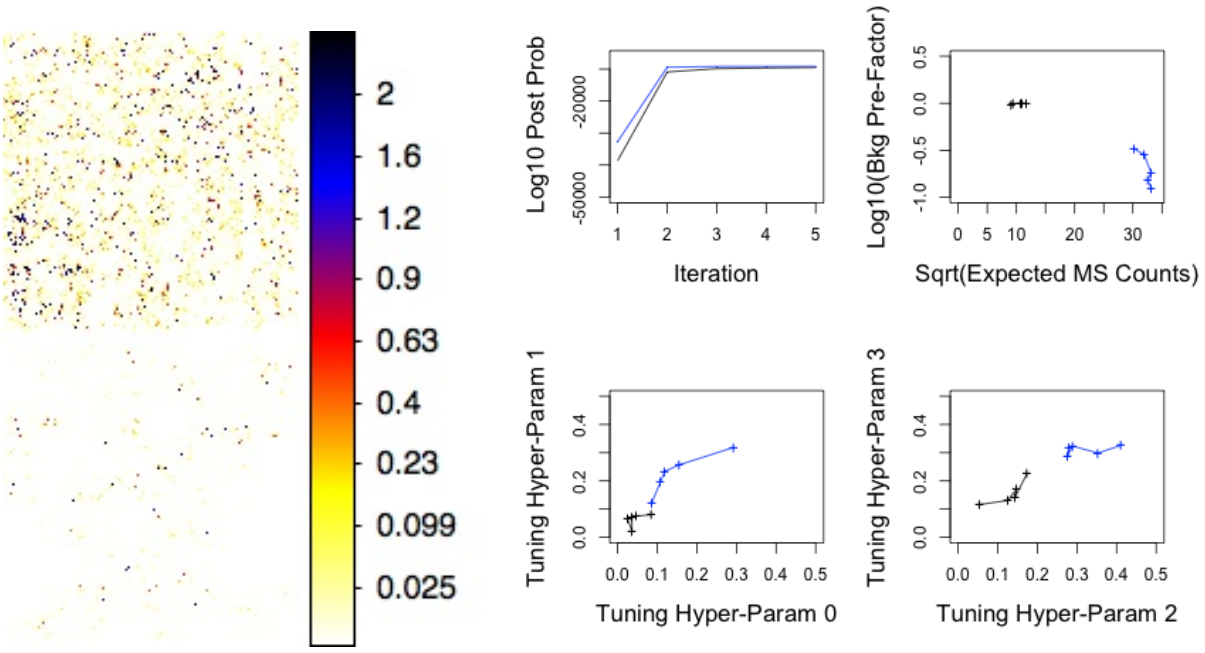
As before, the output files for these preliminary runs can be found in:

`Prelim04etc`

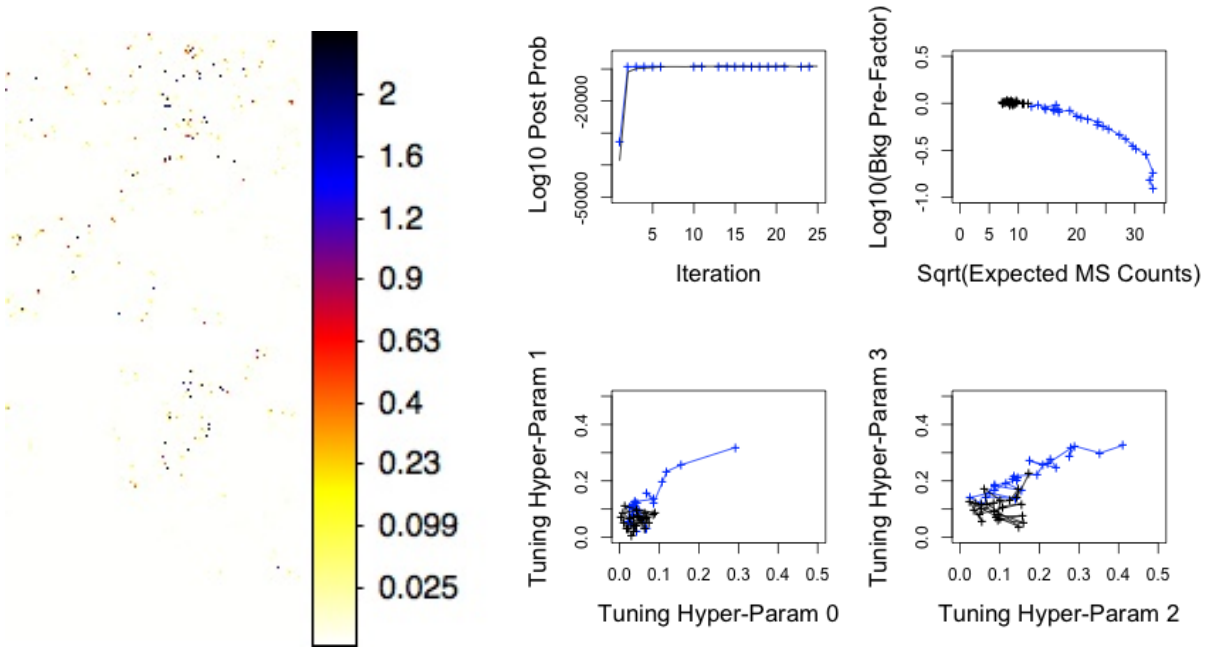
- Check the image samples:



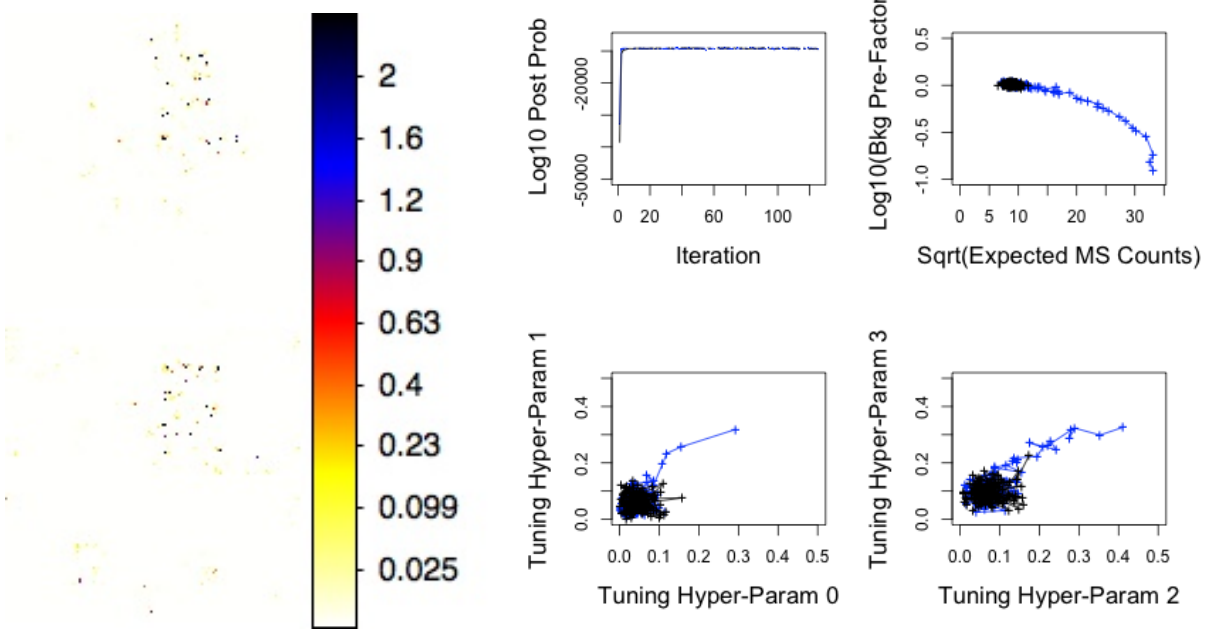
Iteration 001 Two start values: high (top image; blue); low (bottom image; black).



Iteration 005 Two start values: high (top image; blue); low (bottom image; black).



Iteration 025 Two start values: high (top image; blue); low (bottom image; '+').



Iteration 125 Two start values: high (top image; blue); low (bottom image; '+').

- Check the traces:
- Check correlation length:
- Check the scatter plots: It is also a good idea to look briefly at the scatter plots of the various parameters. Scripts for plotting are in:

  `Simple04_Prelim_Rplot_uni_Param_scatterplots.sh`

  Any obvious problems?
- Mean and Moments

7.2. Longer, More Serious Analysis Runs:

We see that it is not quite so quick to run these.

Nevertheless:

R-script for this sub-section can be found in:

7.3. Comparing Results on 'Interesting' and 'Null' datasets:

7.3.1. Comparing data with null model for goodness of fit

Now, we create a number of MC Poisson realizations of our null, or baseline (i.e our best-fit model). These can be found in:

```
PoisNulDat0EEMC2_32x32testE.090120c.fits
PoisNulDat1EEMC2_32x32testE.090120c.fits
PoisNulDat2EEMC2_32x32testE.090120c.fits
PoisNulDat3EEMC2_32x32testE.090120c.fits
PoisNulDat4EEMC2_32x32testE.090120c.fits
PoisNulDat5EEMC2_32x32testE.090120c.fits
```

and the similar files in '*.txt' format. Here, we will display results on six Poisson realizations of our null, **?**. We will run $\mathcal{L}IRA$ on them in exactly the same way that we did on our data. Then we will compare the scatter plots. What summary statistic gives the best separation?

8. Incorporating Calibration Uncertainties

9. Anything Else?? More real-life examples??

Maybe needs some text here with explicit examples, using datafiles/PSFs/etc from Fermi and/or Chandra

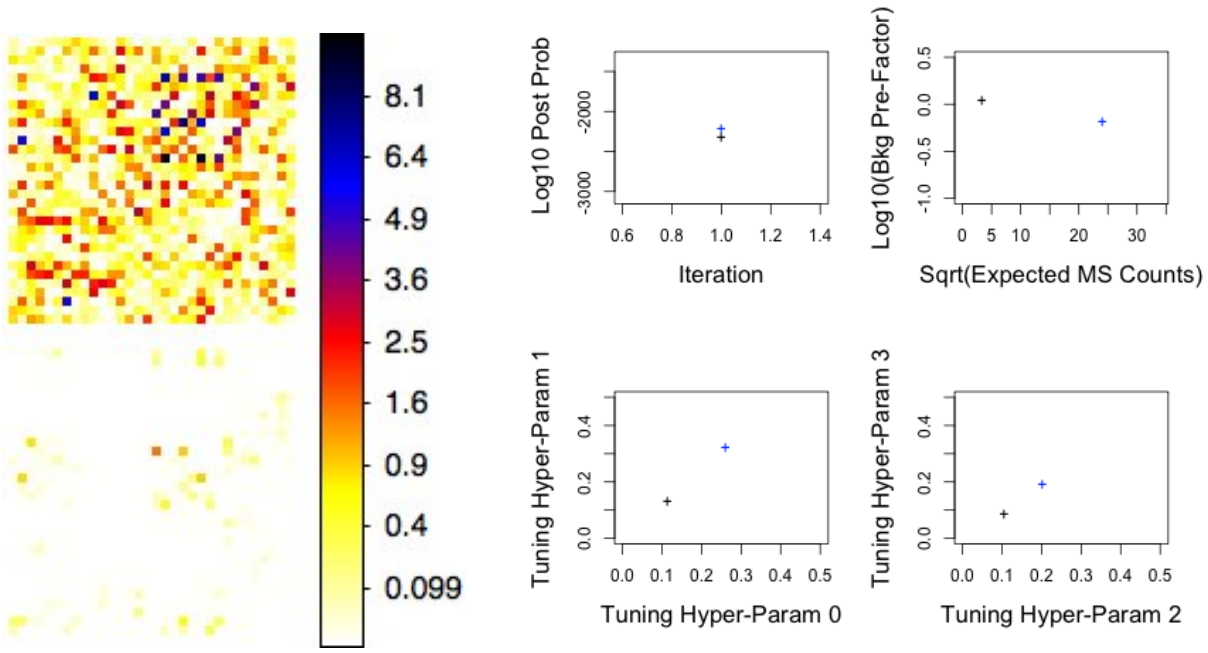
10. References

* James W. Cooley and John W. Tukey * An Algorithm for the Machine Calculation of Complex Fourier Series * Mathematics of Computation, Vol. 19, No. 90 (Apr., 1965), pp. 297-301 (article consists of 5 pages) * Published by: American Mathematical Society * Stable URL: http://www.jstor.org/stable/2003354

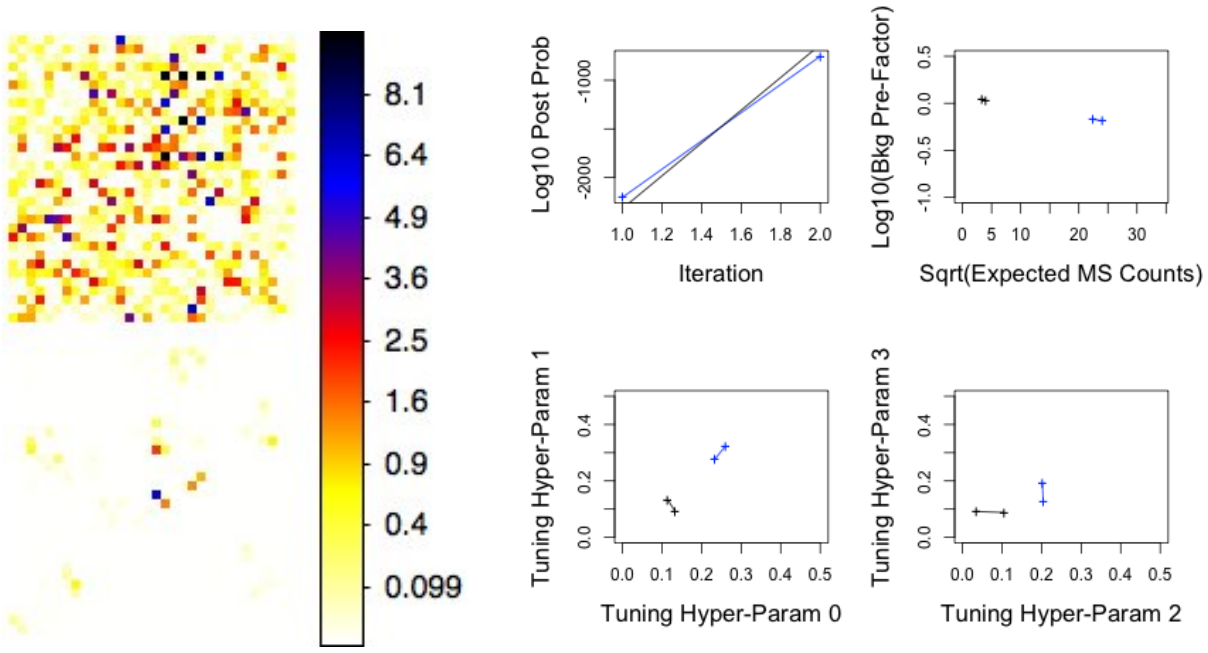
APPENDIX A: Pseudo-Movies: Illustrative Samples from the McMC Process

A. Simple Quanitative $\mathcal{L}IRA$ McMC: An Illustration

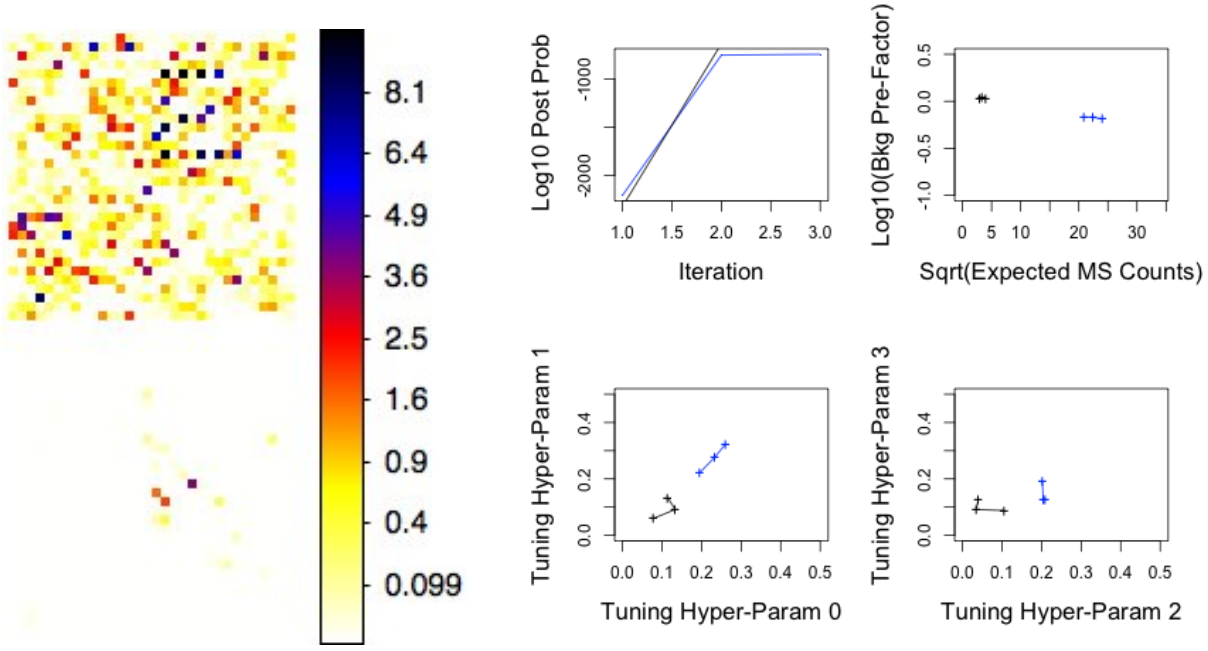
Here we approximae a 'movie' of the McMC chains, by showing both the image at a ew selected iterations, and the parameters up unitl that iteration.



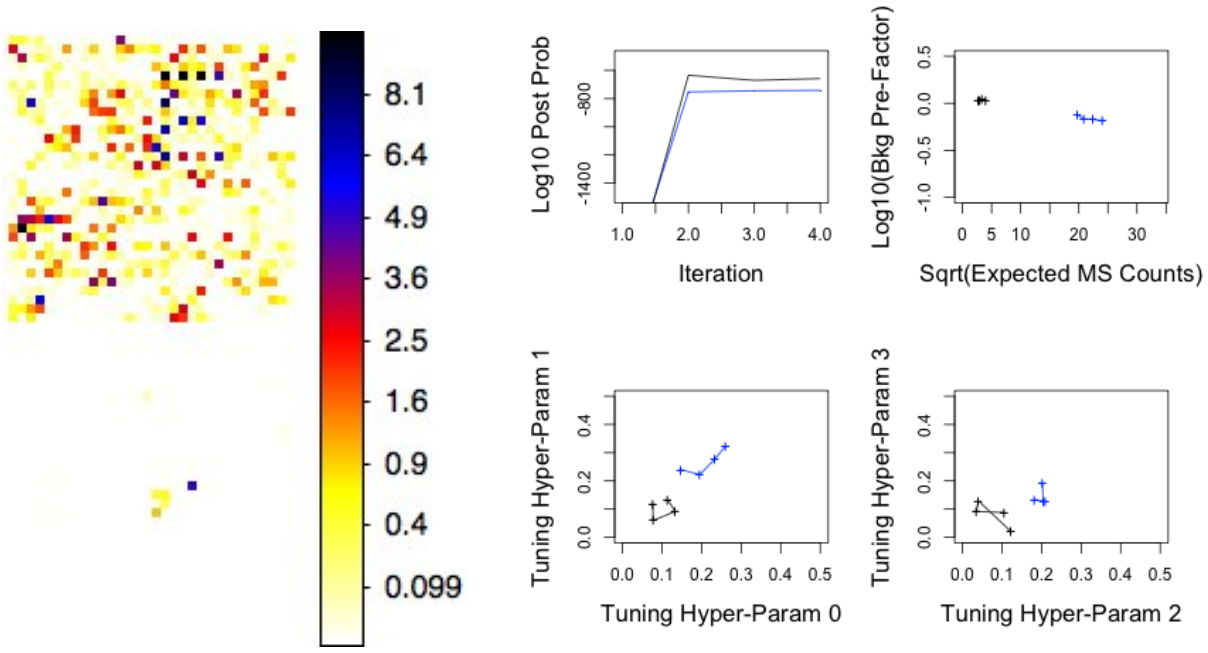
Iteration 001 Two start values: high (top image; blue); low (bottom image; black).



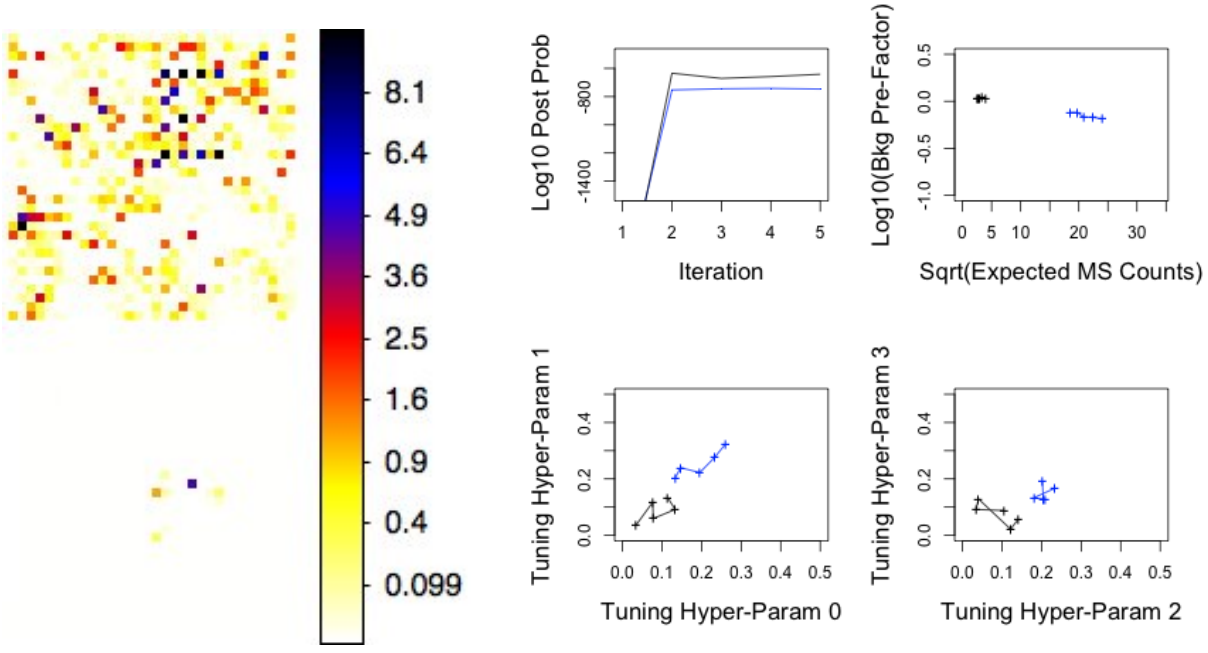
Iteration 002 Two start values: high (top image; blue); low (bottom image; black).



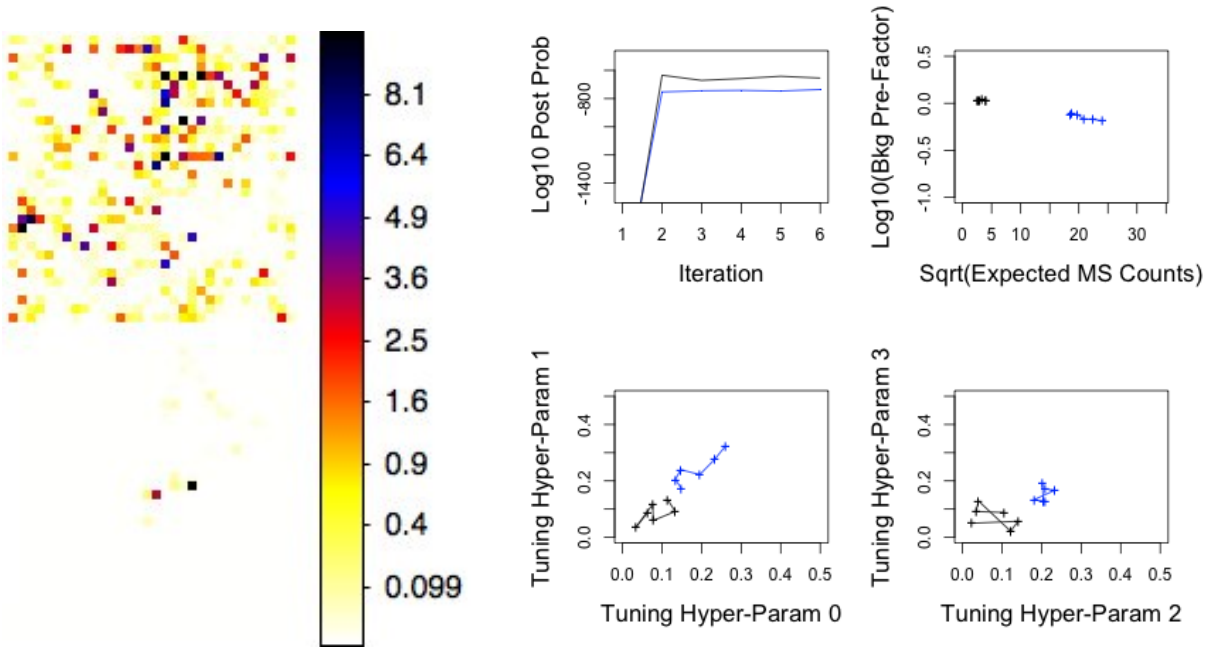
Iteration 003 Two start values: high (top image; blue); low (bottom image; black).



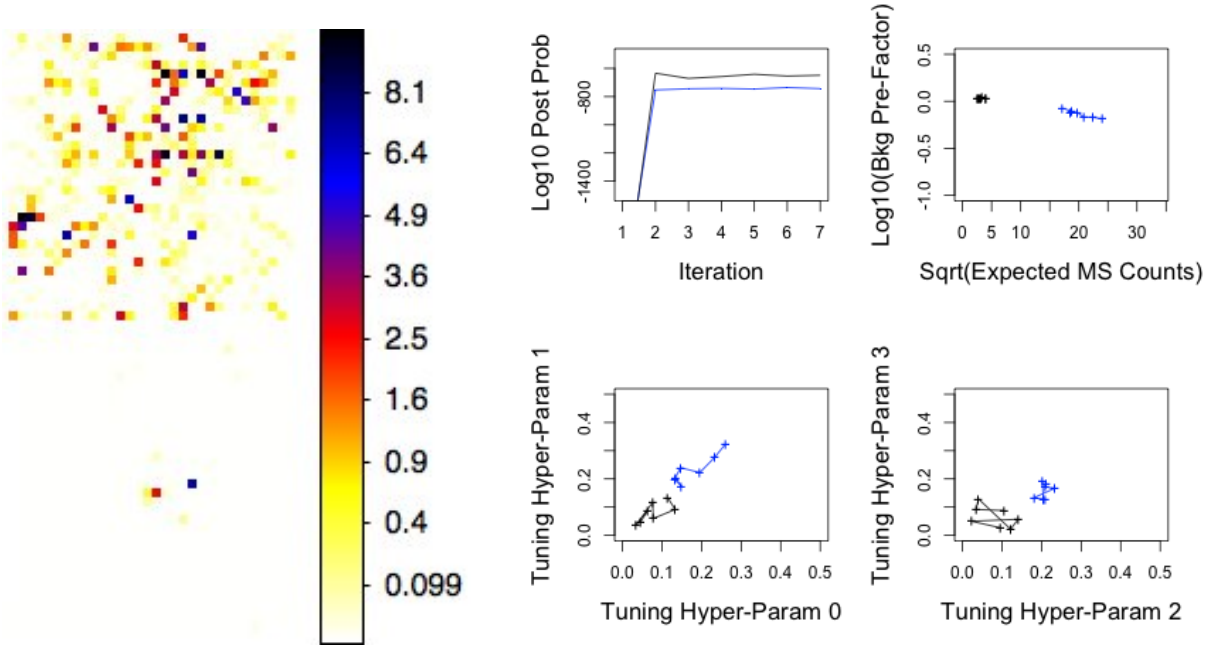
Iteration 004 Two start values: high (top image; blue); low (bottom image; black).



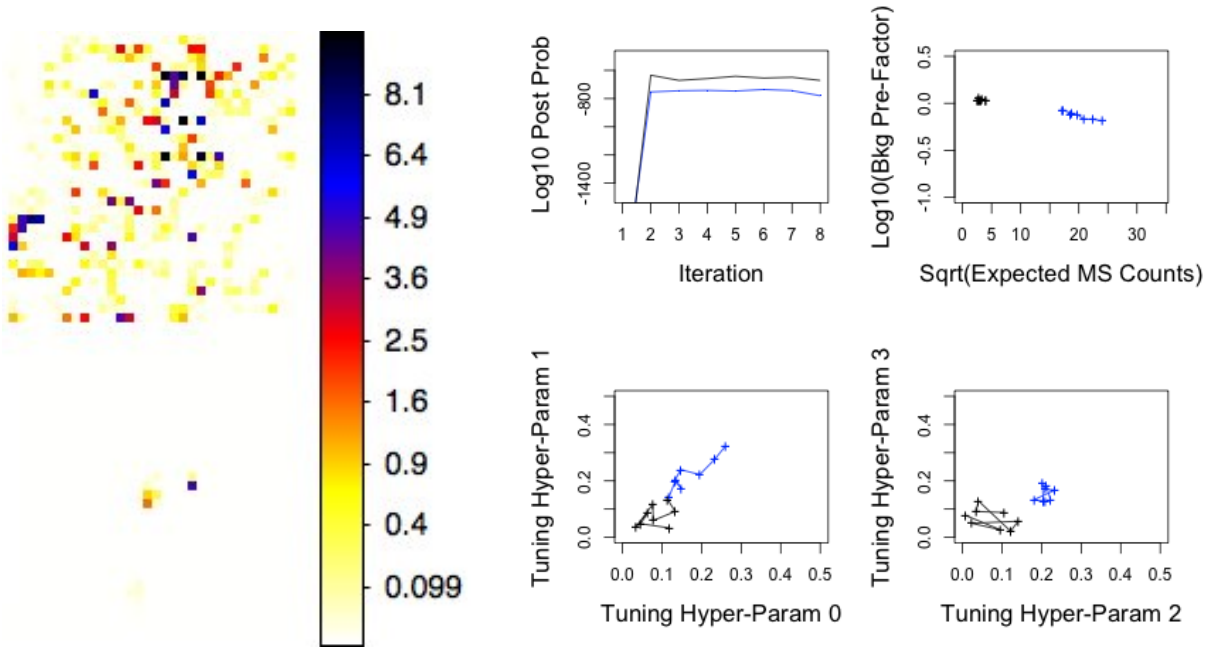
Iteration 005 Two start values: high (top image; blue); low (bottom image; black).



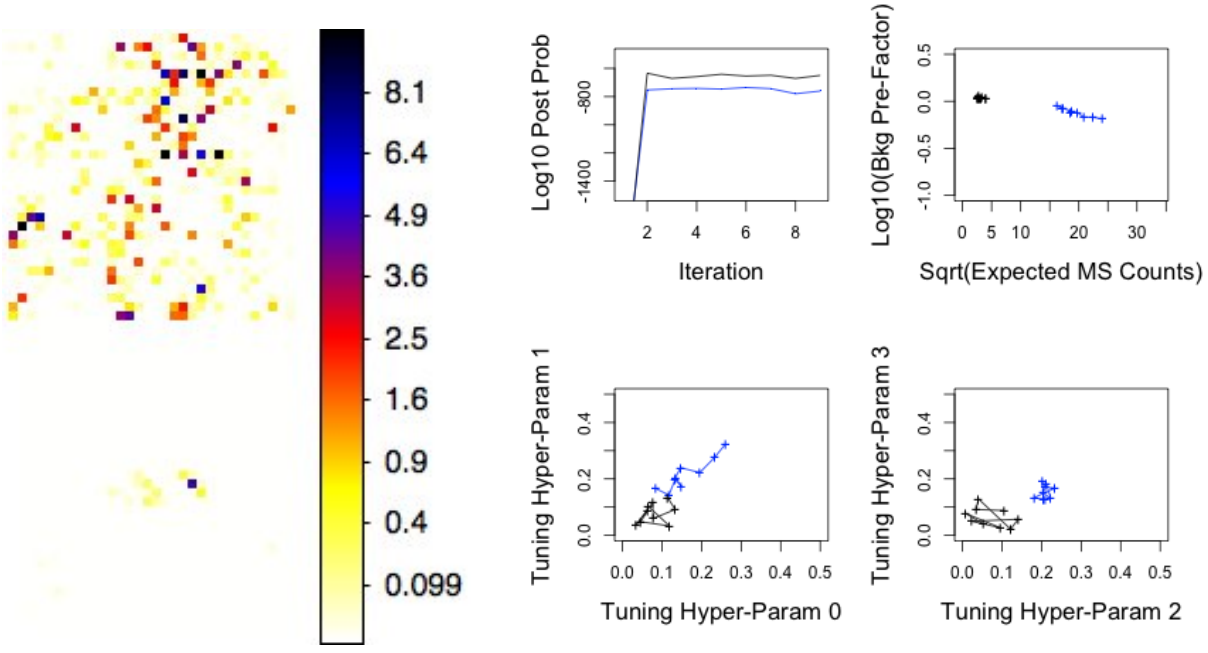
Iteration 006 Two start values: high (top image; blue); low (bottom image; black).



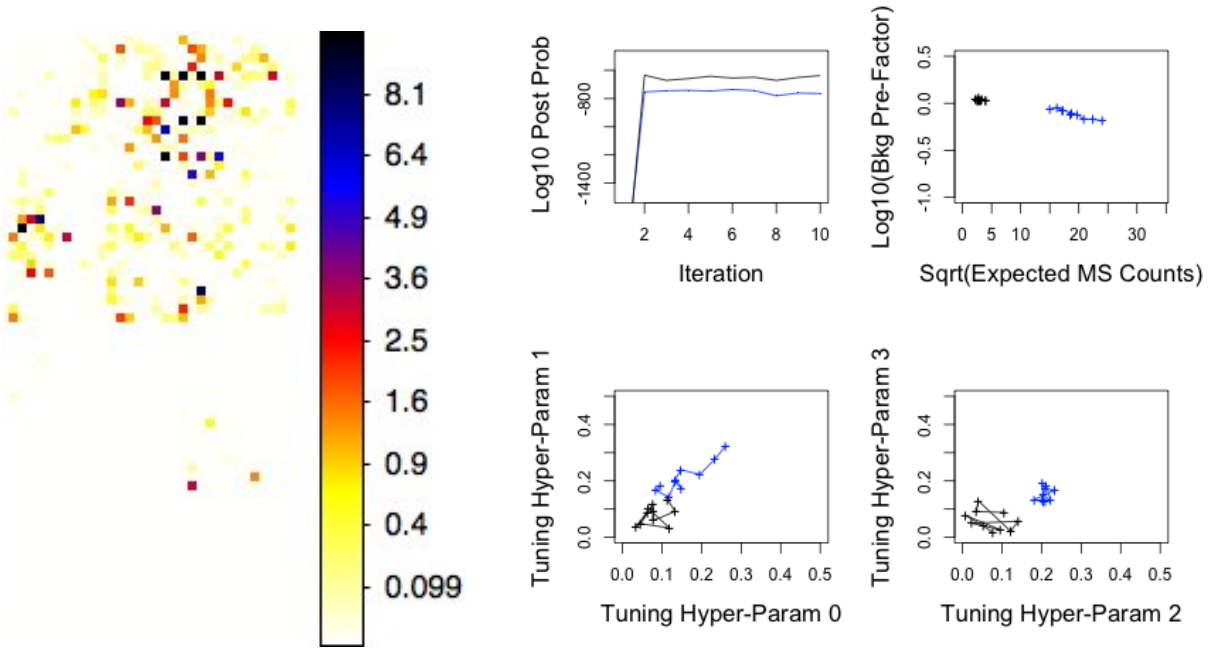
Iteration 007 Two start values: high (top image; blue); low (bottom image; black).



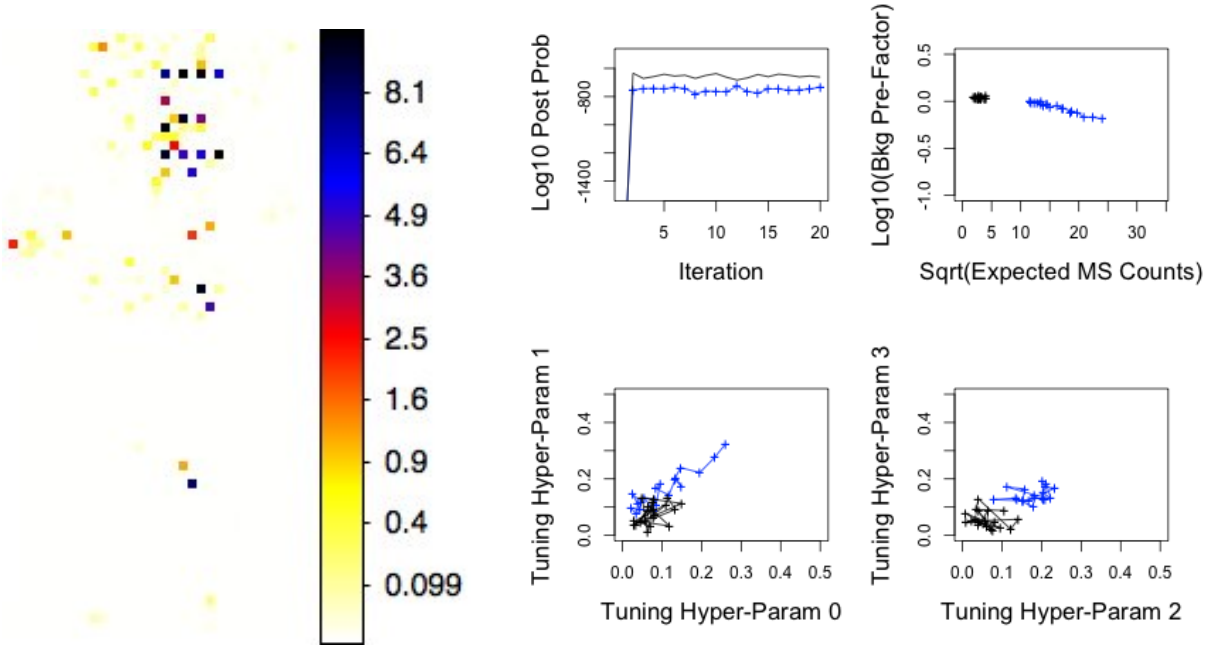
Iteration 008 Two start values: high (top image; blue); low (bottom image; black).



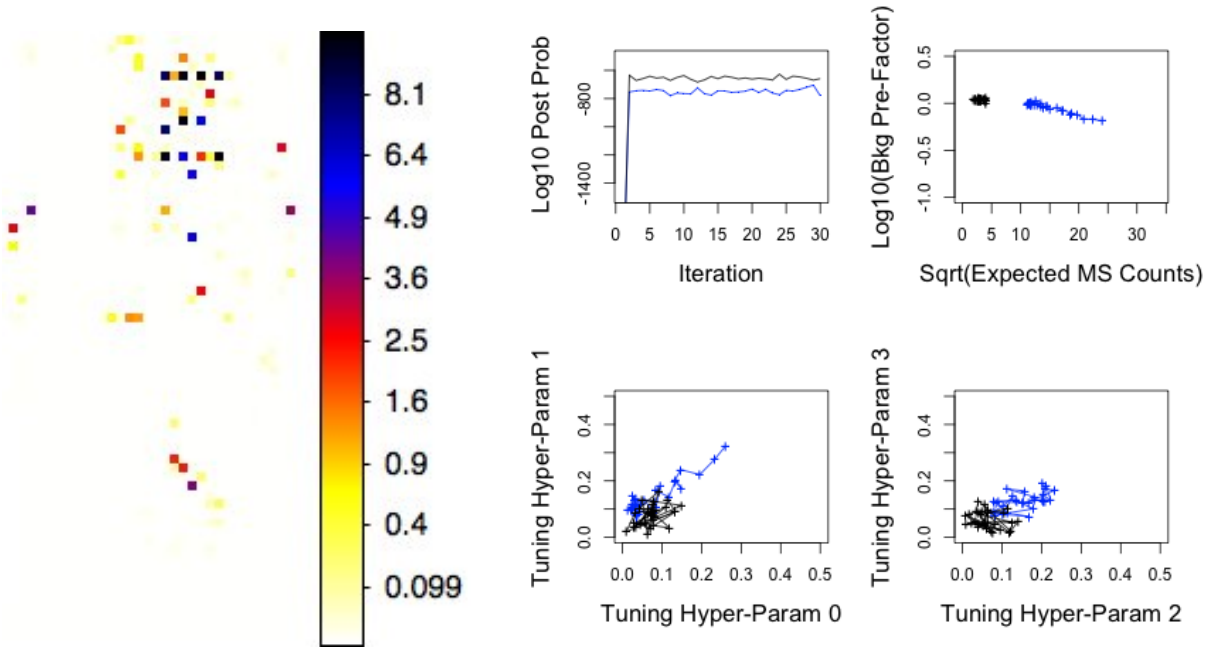
Iteration 009 Two start values: high (top image; blue); low (bottom image; black).



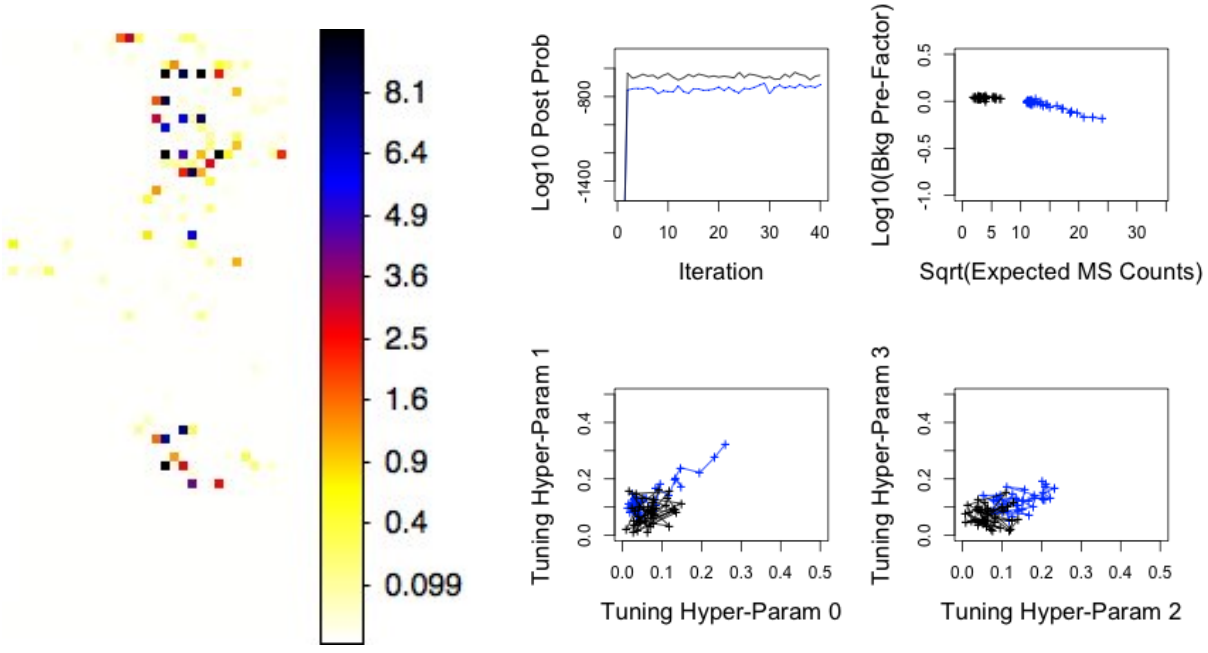
Iteration 010 Two start values: high (top image; blue); low (bottom image; black).



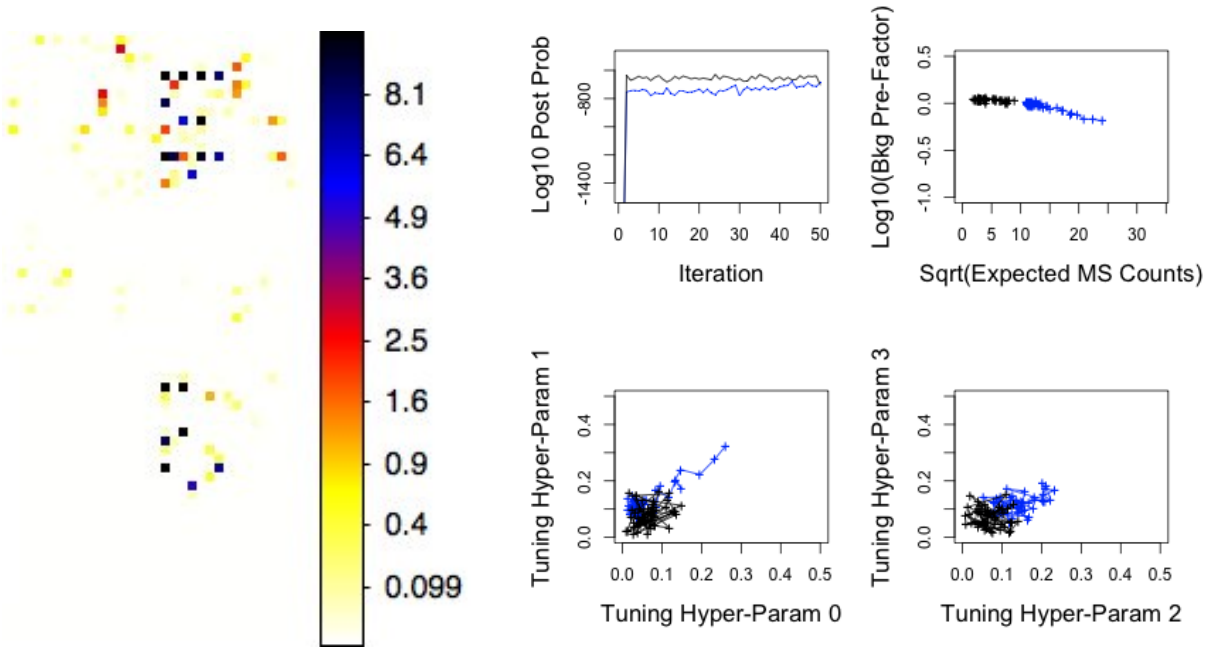
Iteration 020 Two start values: high (top image; blue); low (bottom image; black).



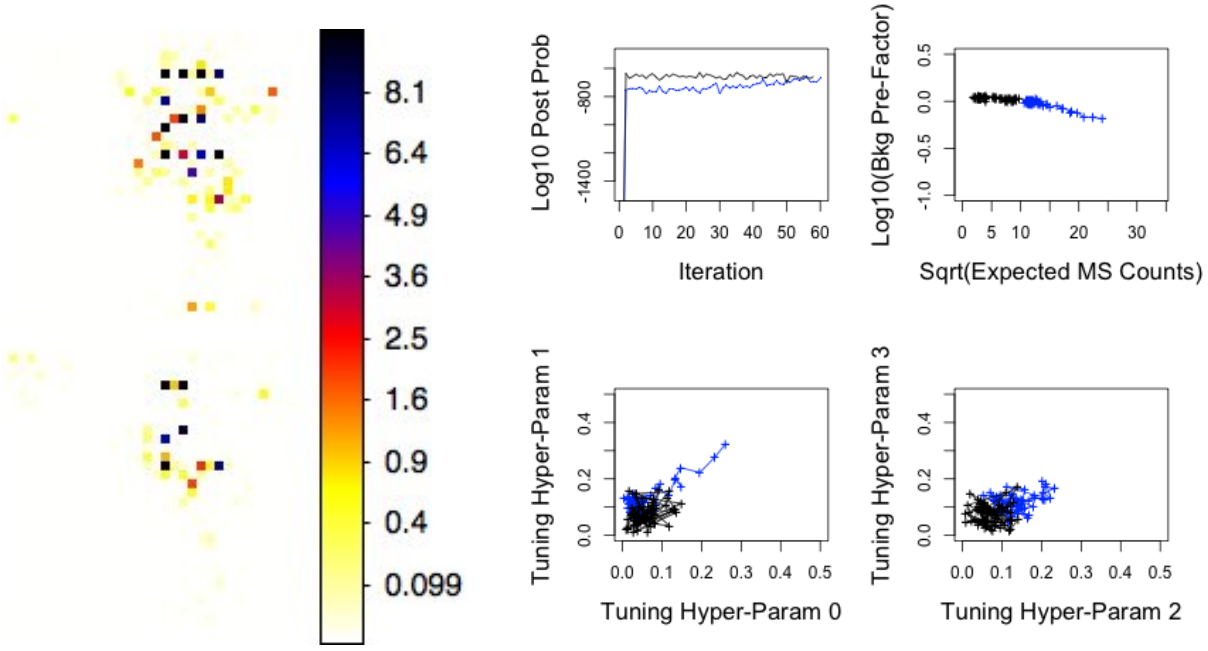
Iteration 030 Two start values: high (top image; blue); low (bottom image; black).



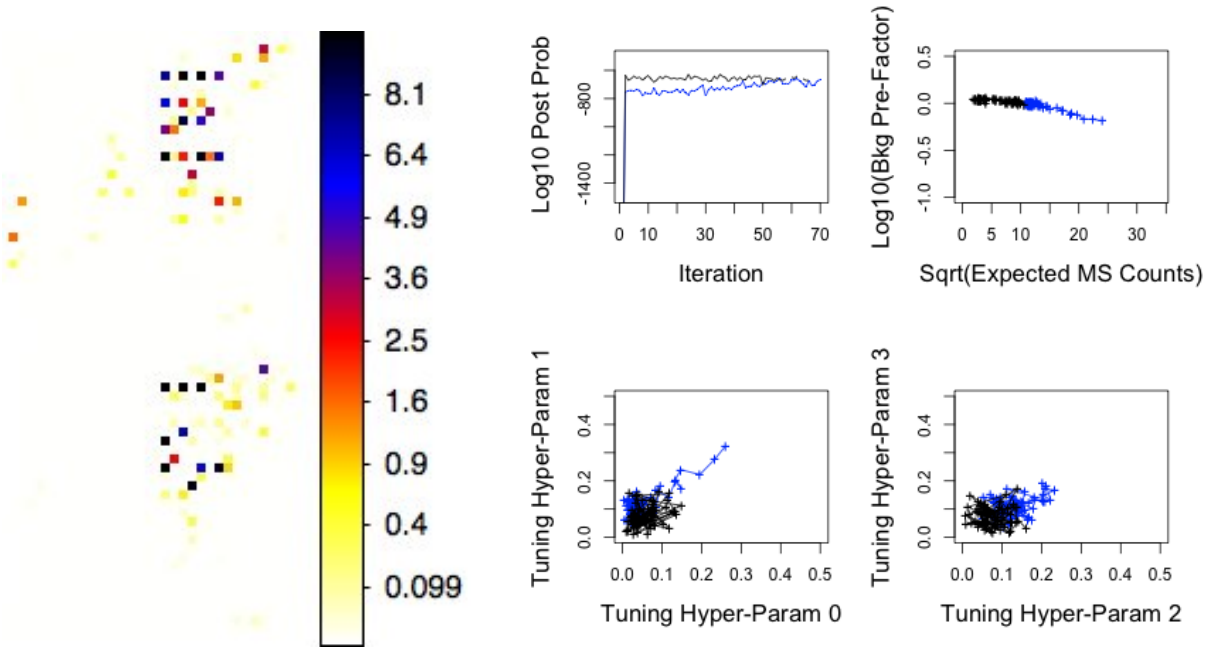
Iteration 040 Two start values: high (top image; blue); low (bottom image; black).



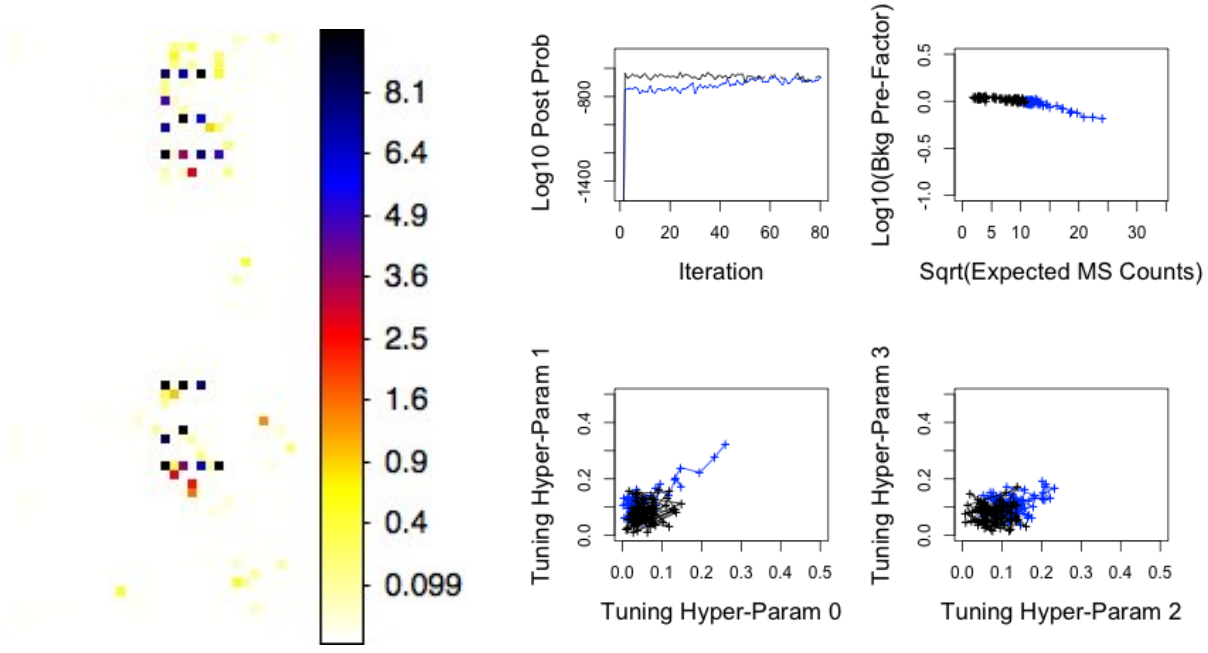
Iteration 050 Two start values: high (top image; blue); low (bottom image; black).



Iteration 060 Two start values: high (top image; blue); low (bottom image; black).



Iteration 070 Two start values: high (top image; blue); low (bottom image; black).



Iteration 080 Two start values: high (top image; blue); low (bottom image; black).

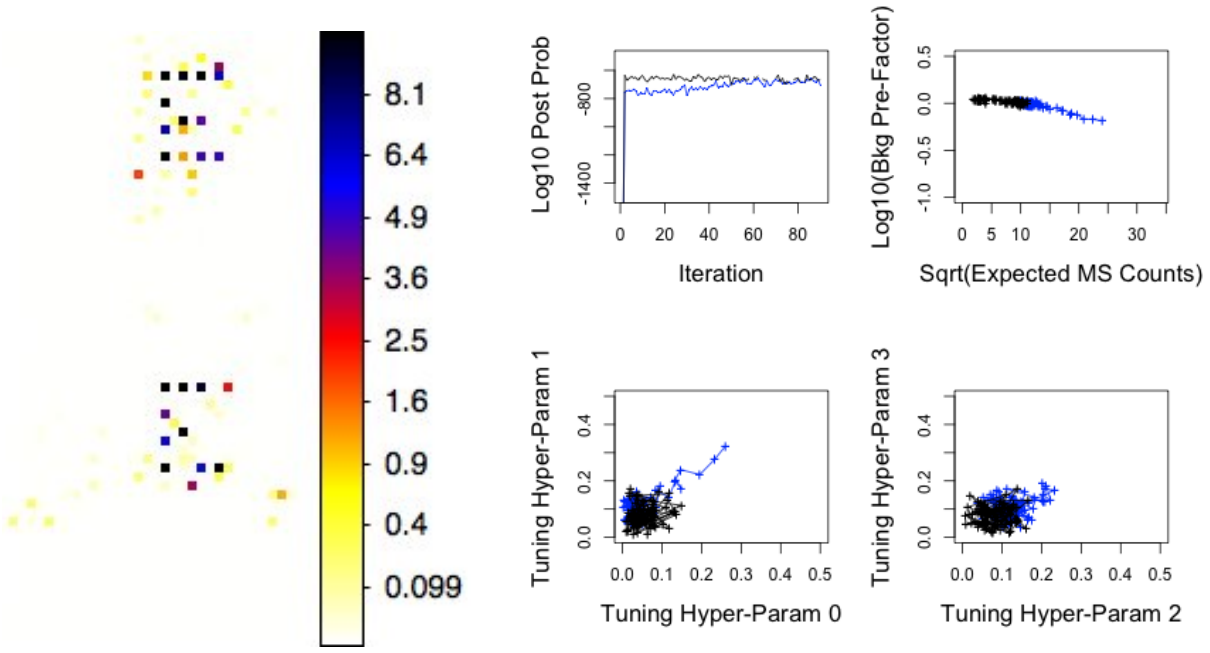


Fig. 21.— **Iteration 090** Two start values: high (top image; blue); low (bottom image; black).

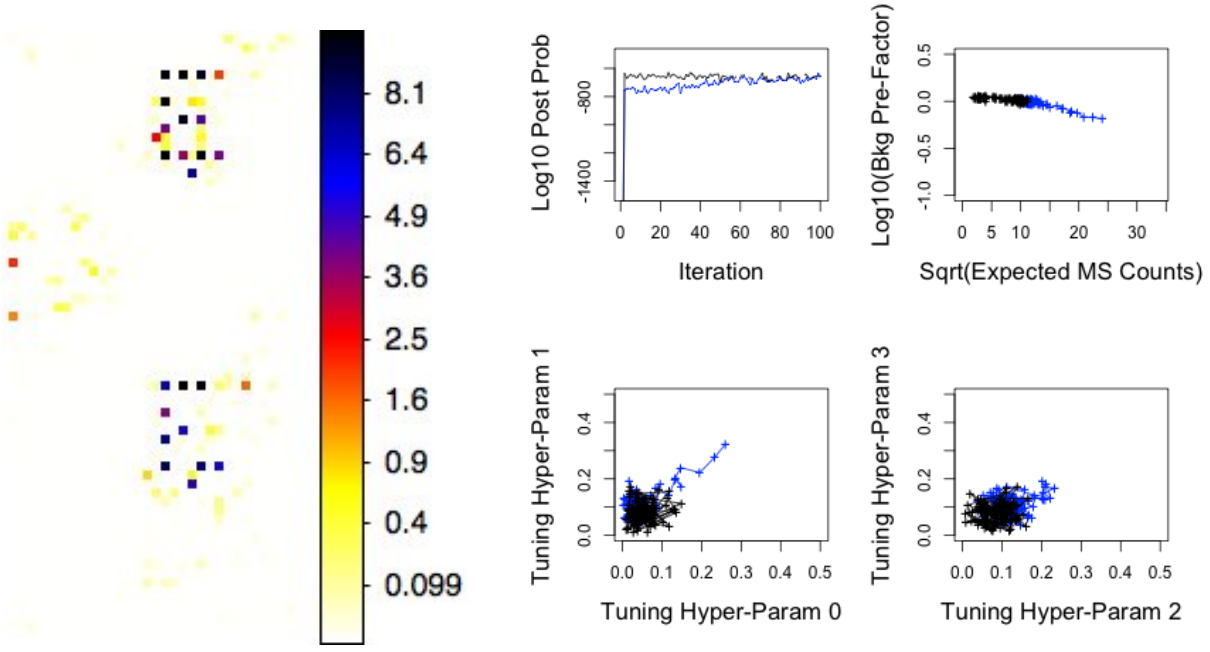
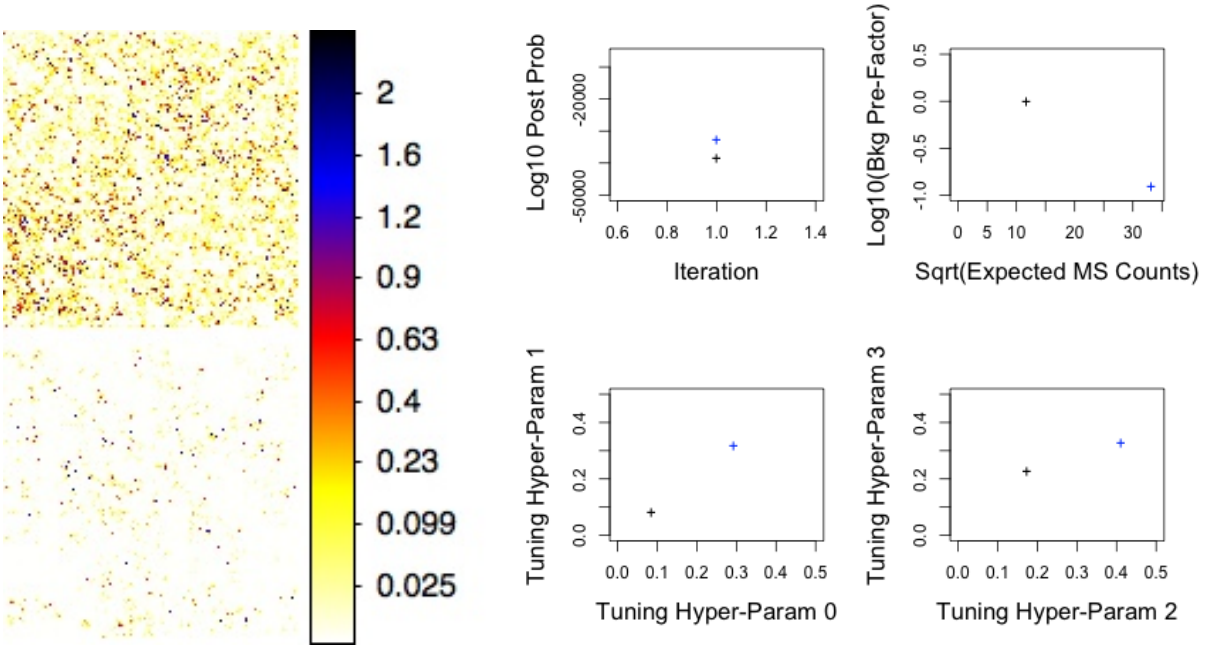


Fig. 22.— **Iteration 100** Two start values: high (top image; blue); low (bottom image; black).

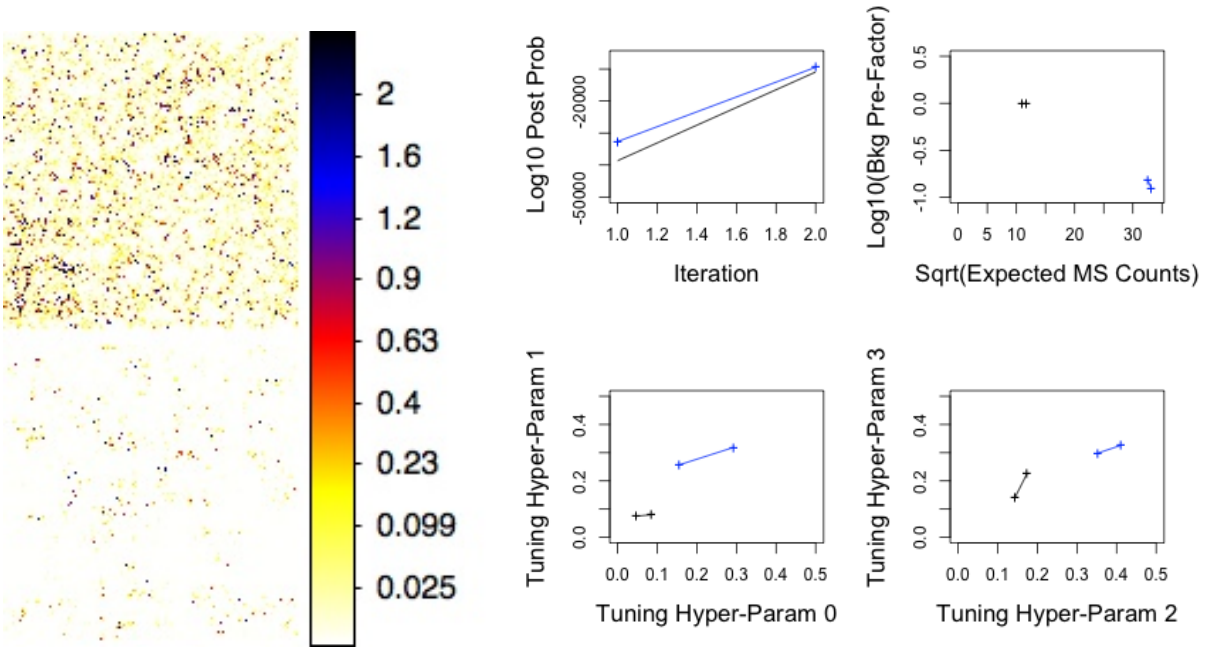
B. Quantitative $\mathcal{L}IRA$ McMC with PSF: An Illustration

And ths smae here, but with a PSF!

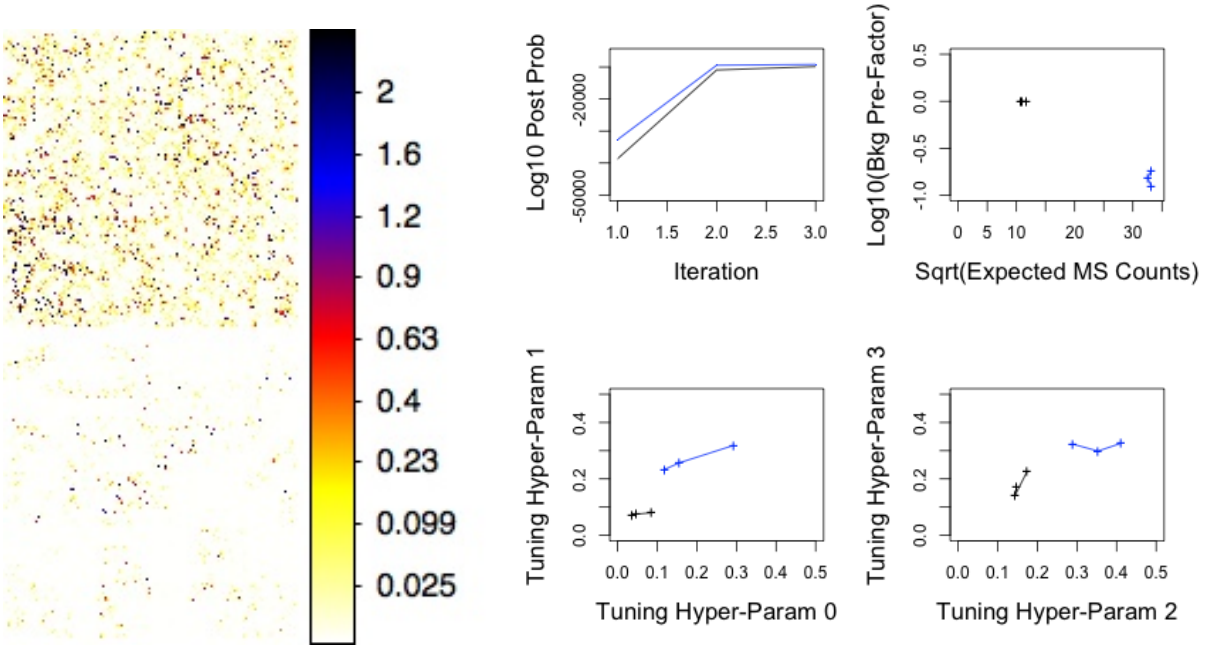
Here we approximate a 'movie' of the McMC chains, by showing both the image at a ew selected iterations, and the parameters up unitl that iteration.



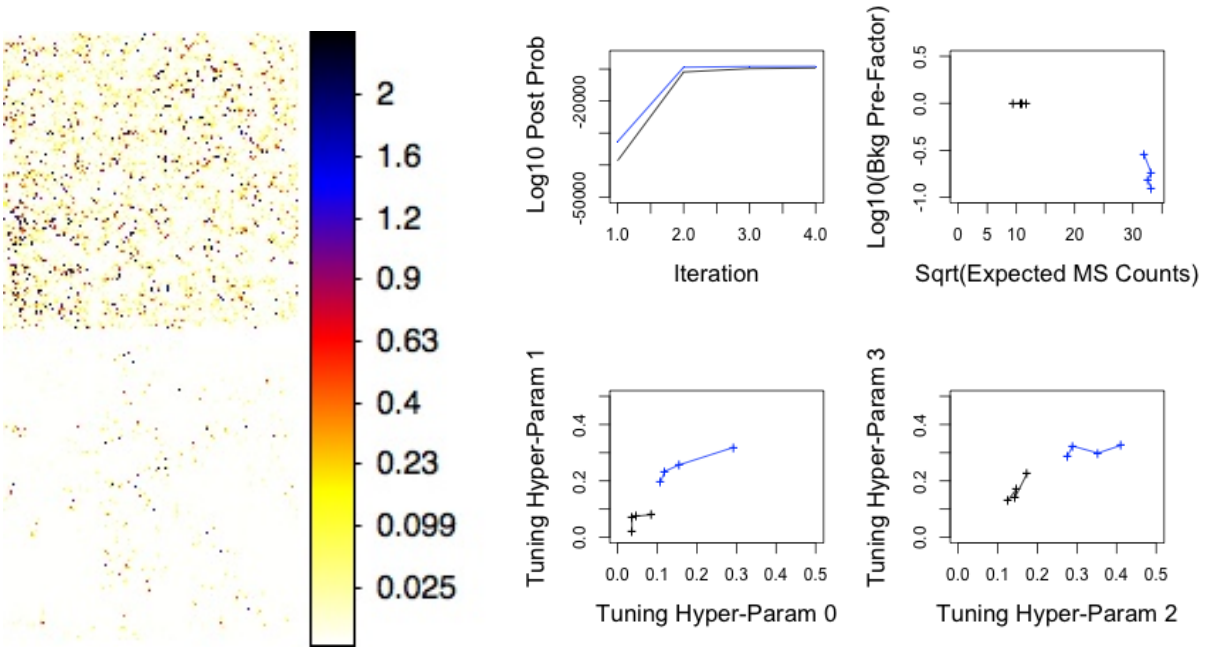
Iteration 001 Two start values: high (top image; blue); low (bottom image; black).



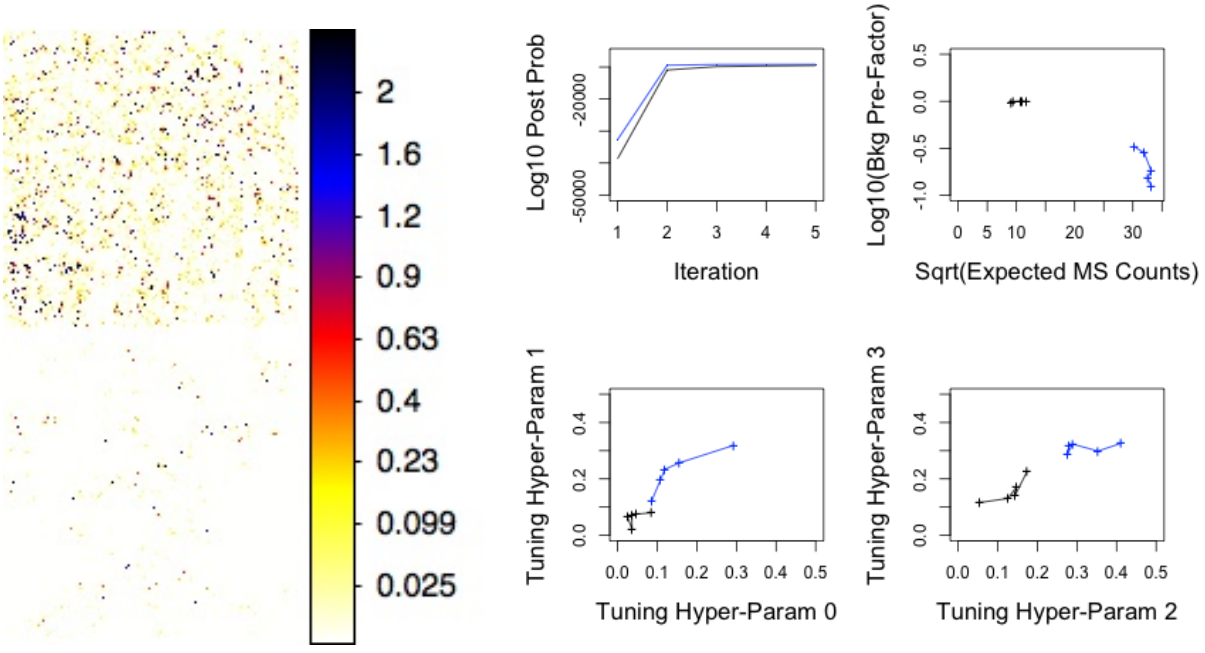
Iteration 002 Two start values: high (top image; blue); low (bottom image; black).



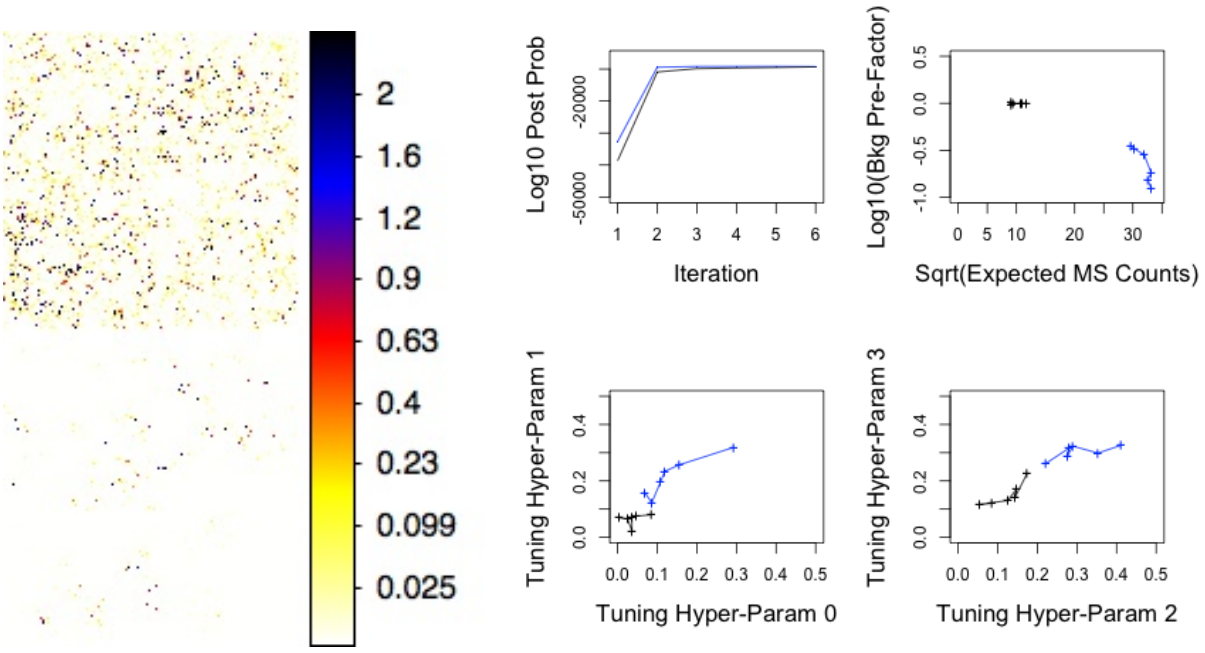
Iteration 003 Two start values: high (top image; blue); low (bottom image; black).



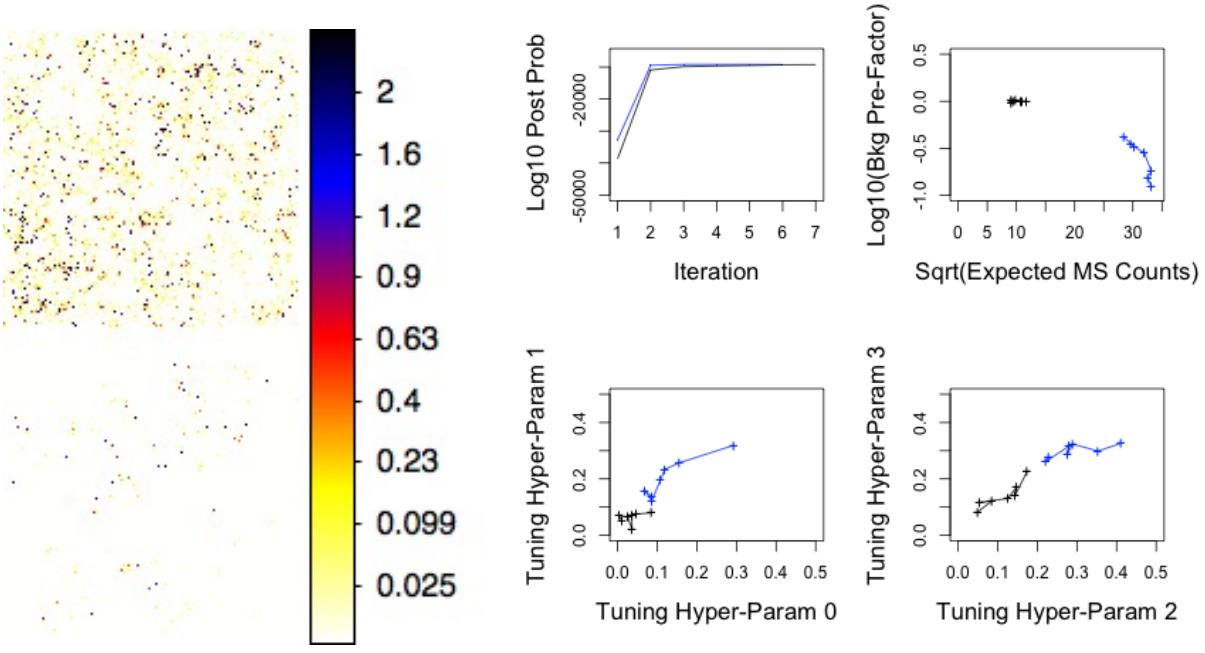
Iteration 004 Two start values: high (top image; blue); low (bottom image; black).



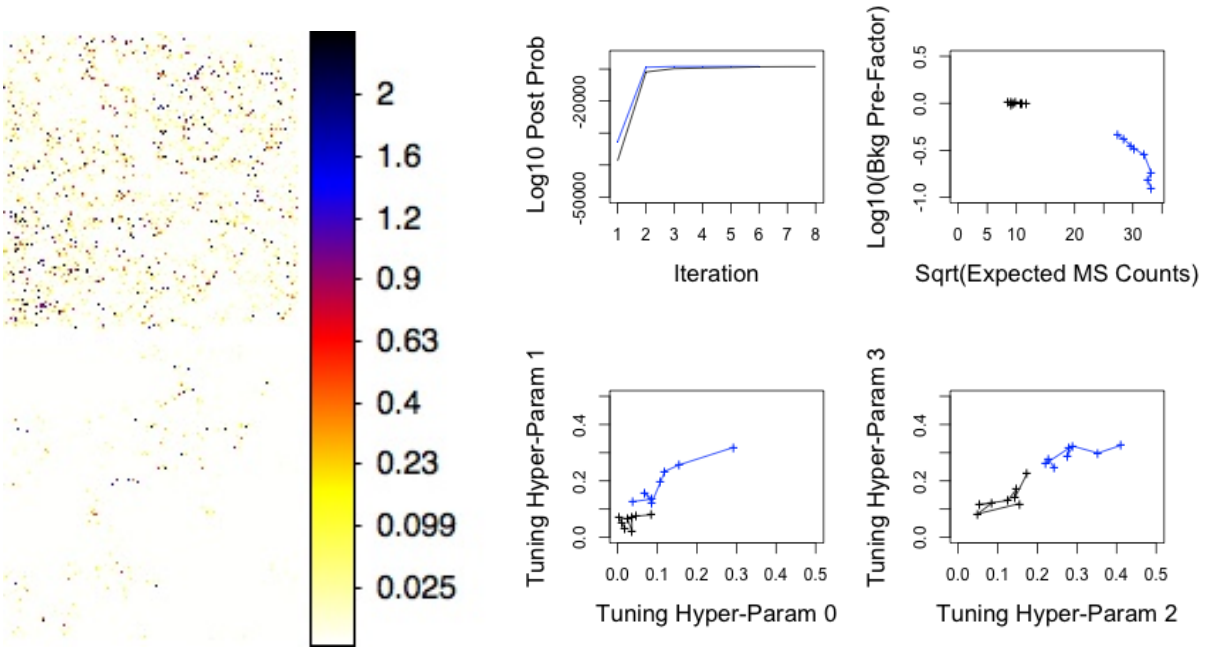
Iteration 005 Two start values: high (top image; blue); low (bottom image; black).



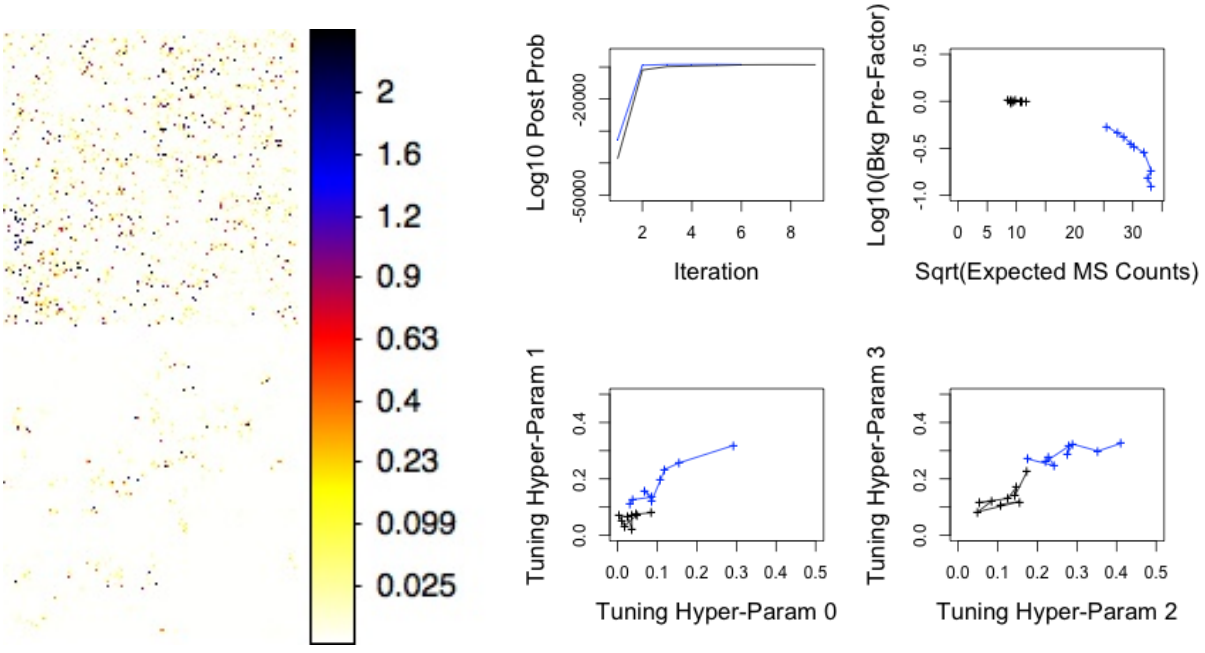
Iteration 006 Two start values: high (top image; blue); low (bottom image; black).



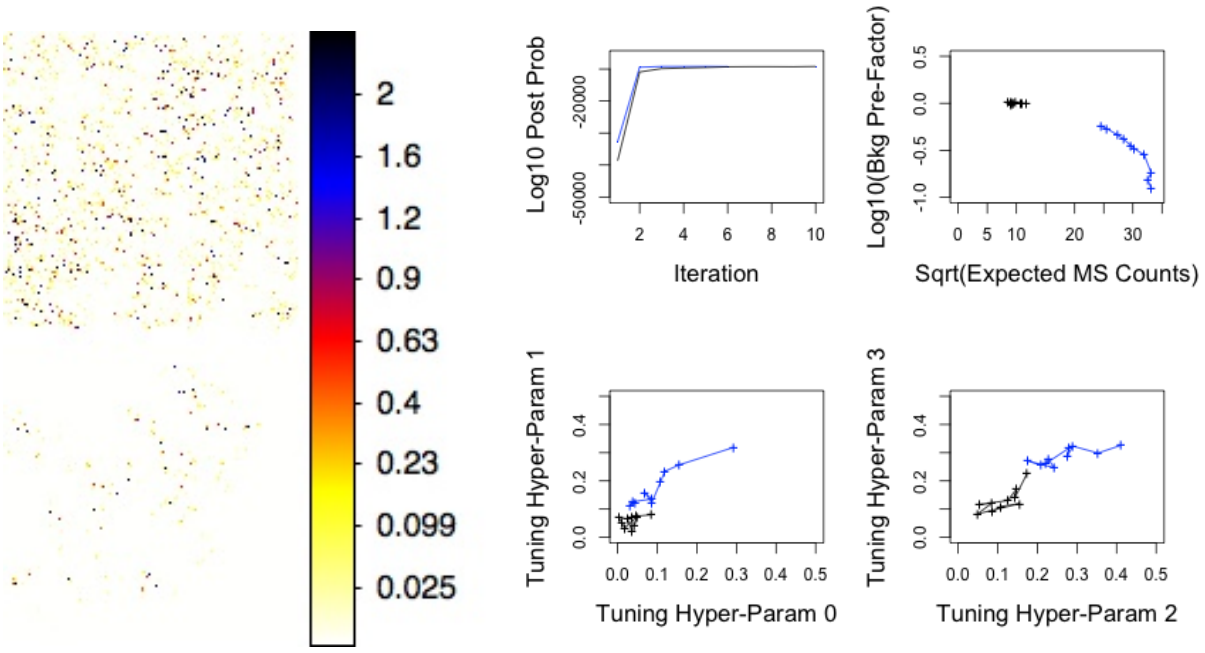
Iteration 007 Two start values: high (top image; blue); low (bottom image; black).



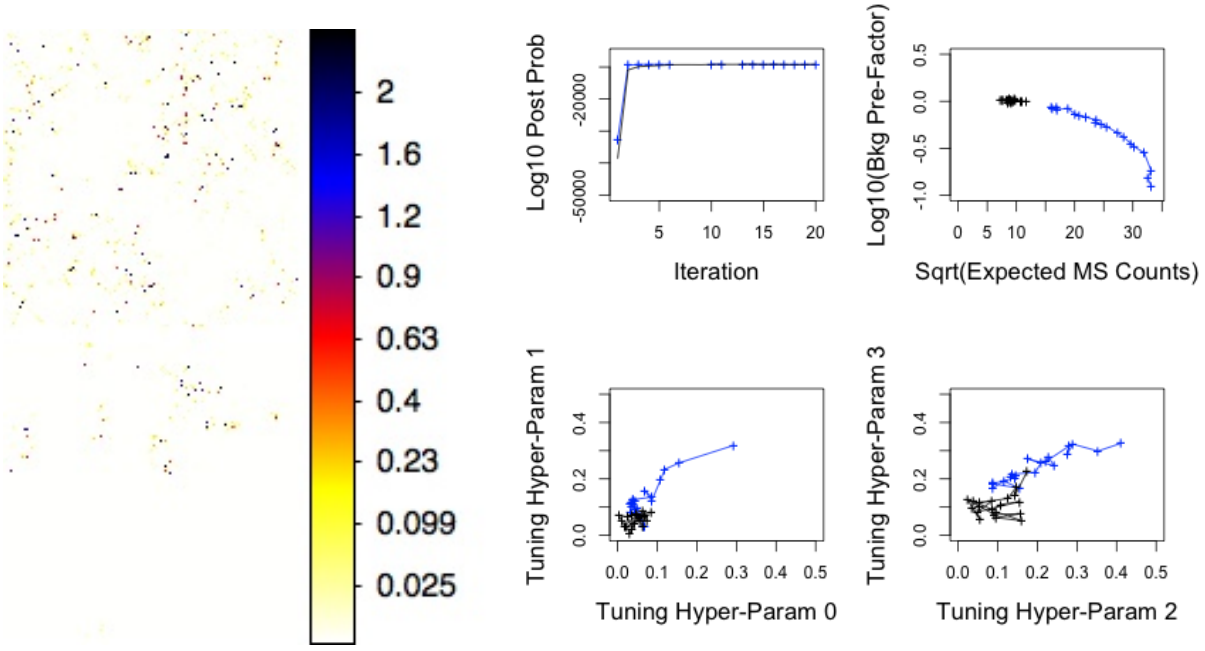
Iteration 008 Two start values: high (top image; blue); low (bottom image; black).



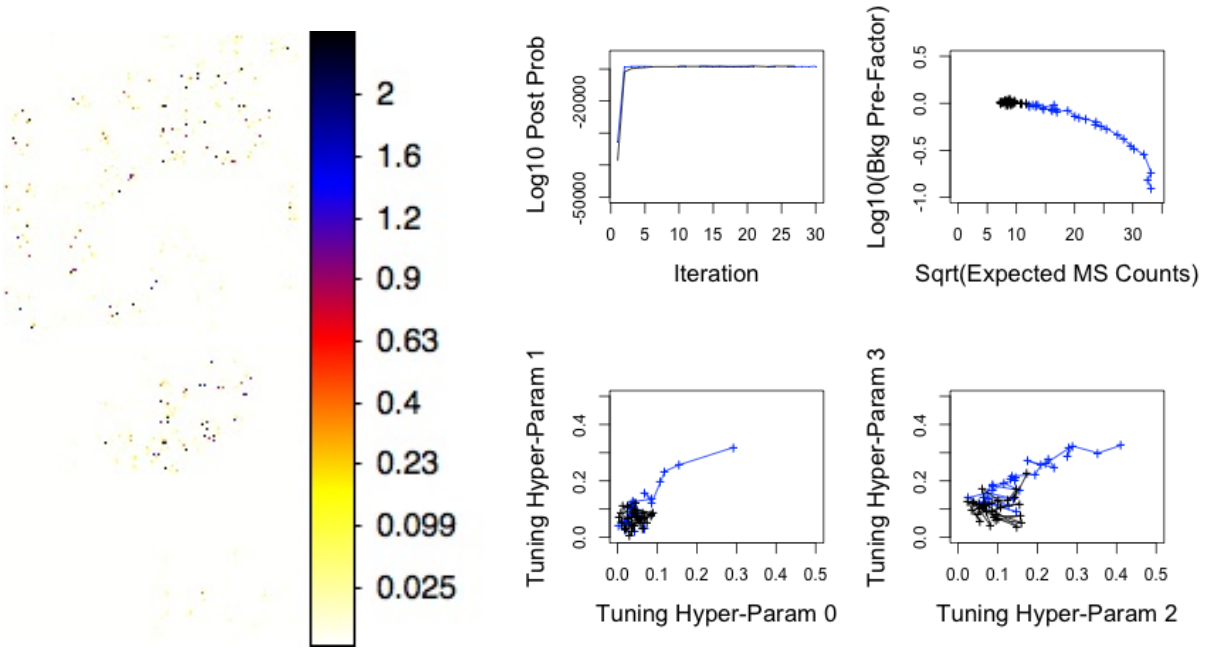
Iteration 009 Two start values: high (top image; blue); low (bottom image; black).



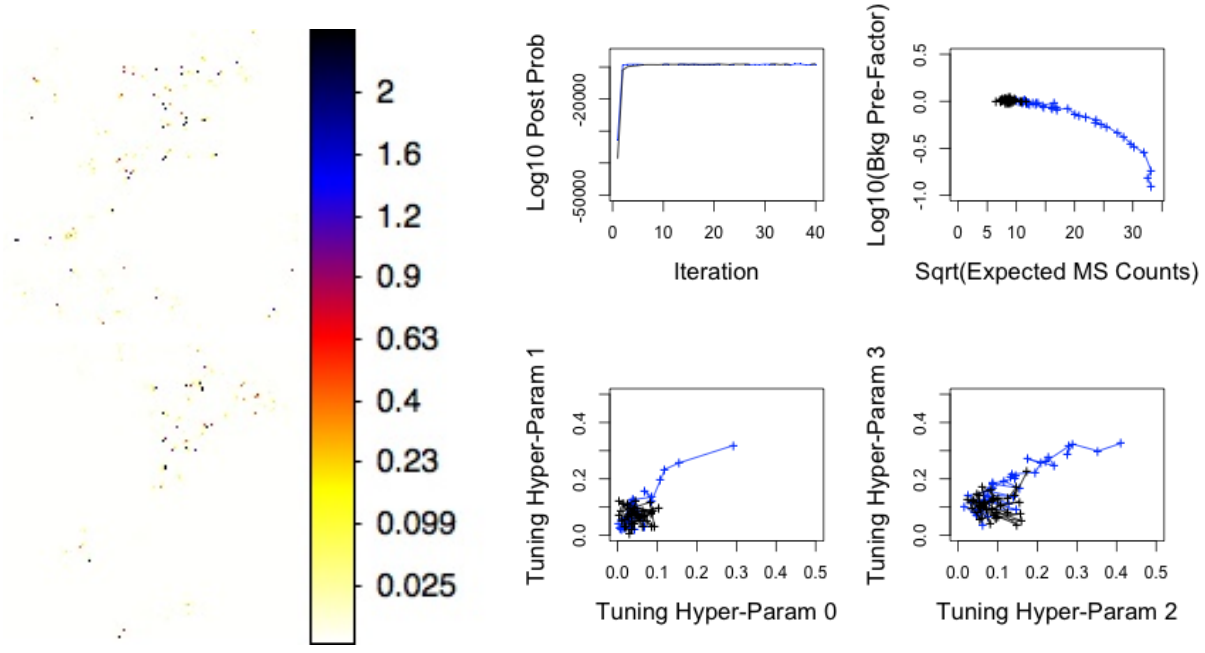
Iteration 010 Two start values: high (top image; blue); low (bottom image; black).



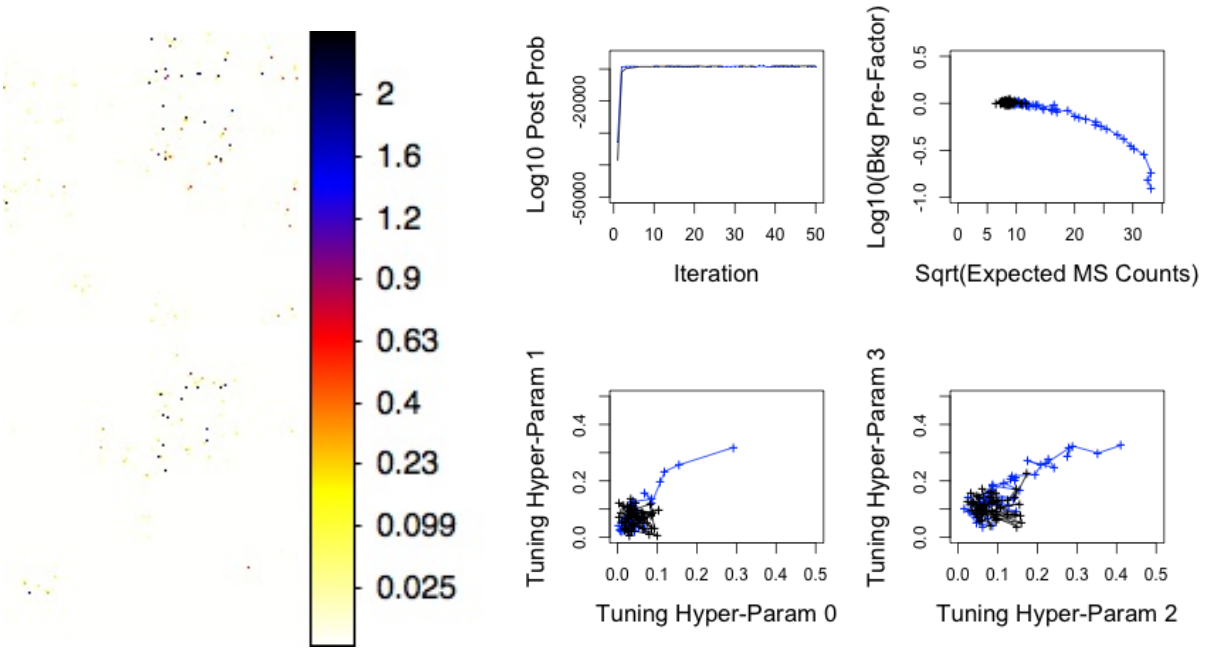
Iteration 020 Two start values: high (top image; blue); low (bottom image; black).



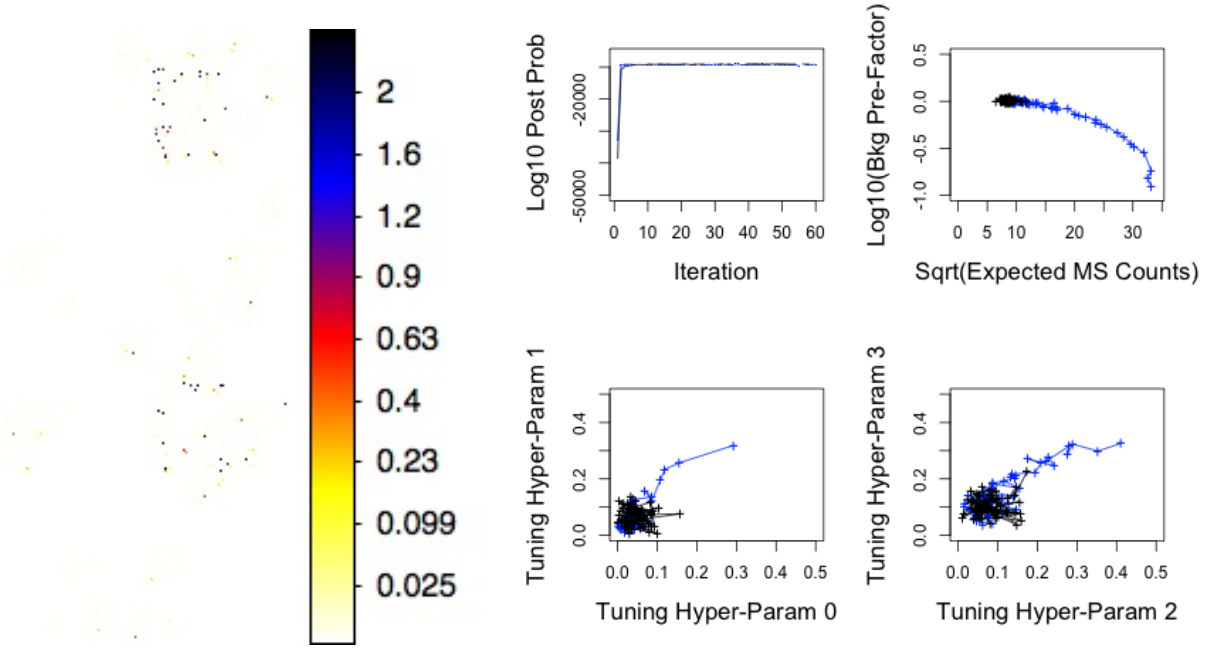
Iteration 030 Two start values: high (top image; blue); low (bottom image; black).



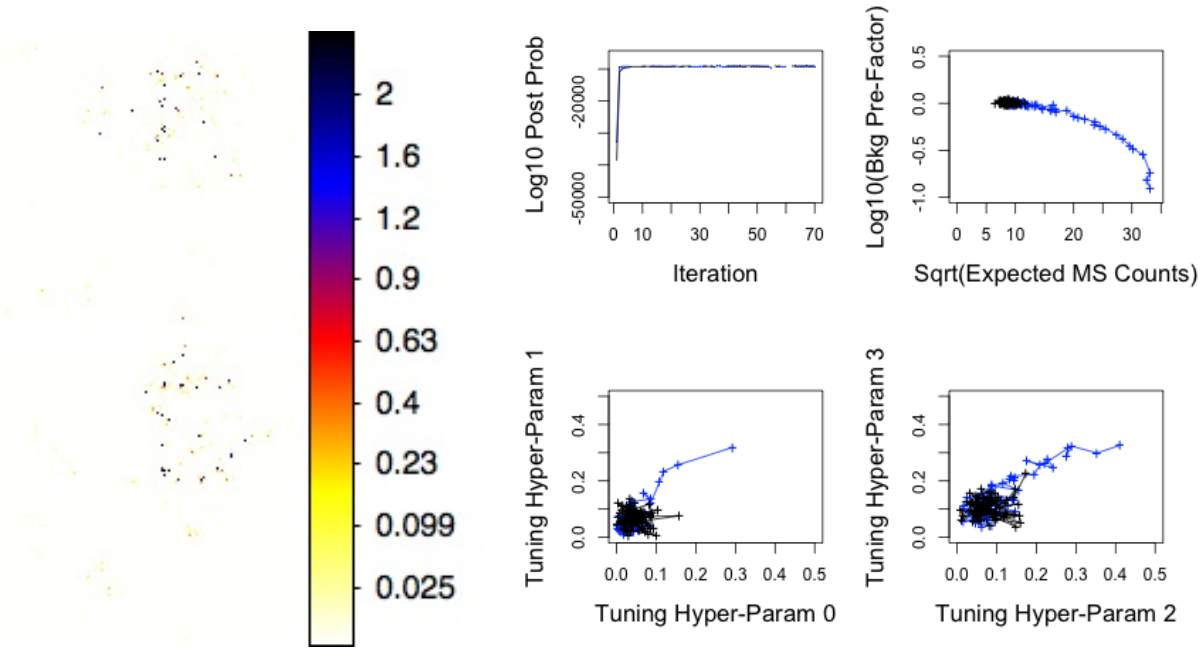
Iteration 040 Two start values: high (top image; blue); low (bottom image; black).



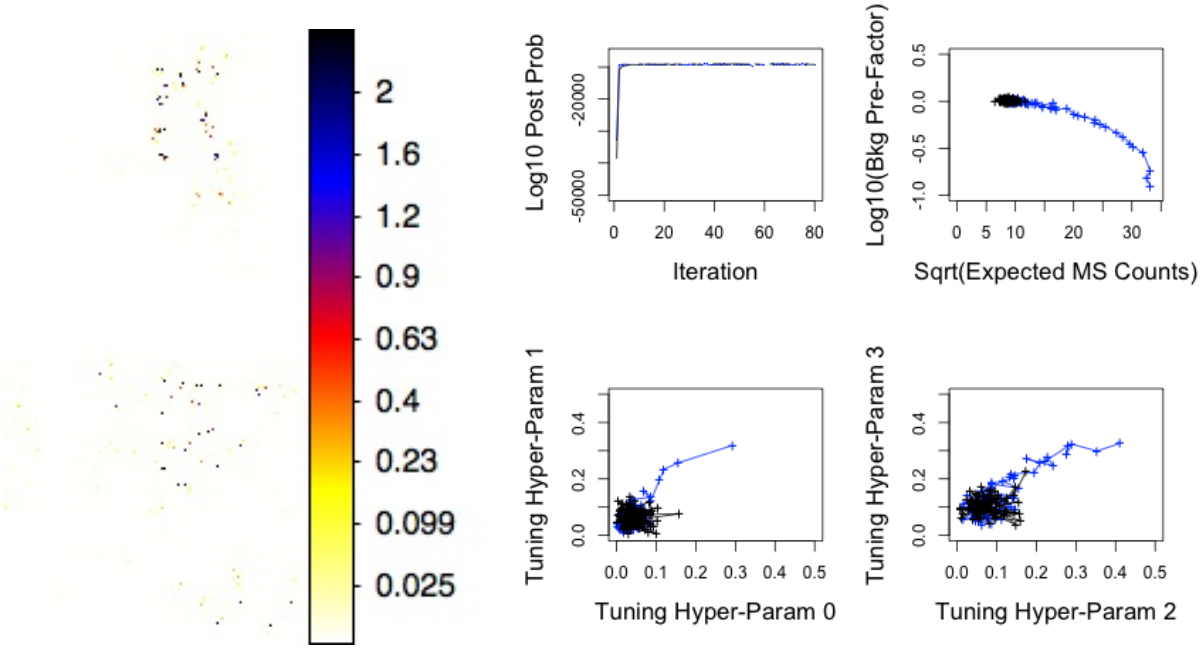
Iteration 050 Two start values: high (top image; blue); low (bottom image; black).



Iteration 060 Two start values: high (top image; blue); low (bottom image; black).



Iteration 070 Two start values: high (top image; blue); low (bottom image; black).



Iteration 080 Two start values: high (top image; blue); low (bottom image; black).

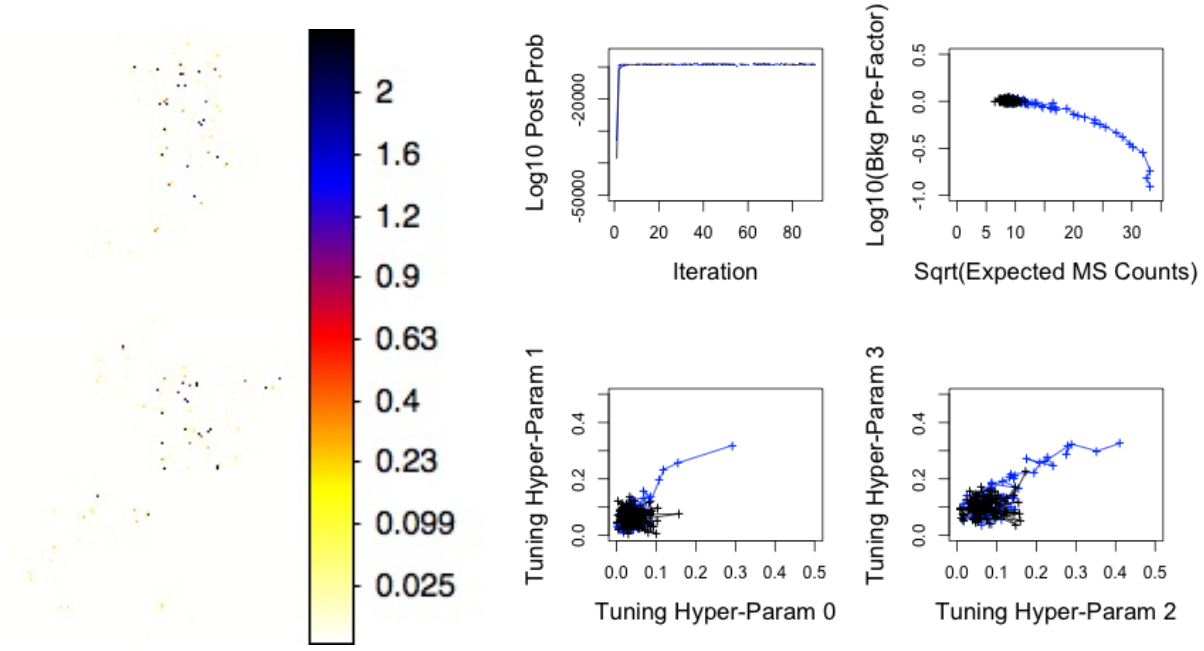


Fig. 23.— **Iteration 090** Two start values: high (top image; blue); low (bottom image; black).

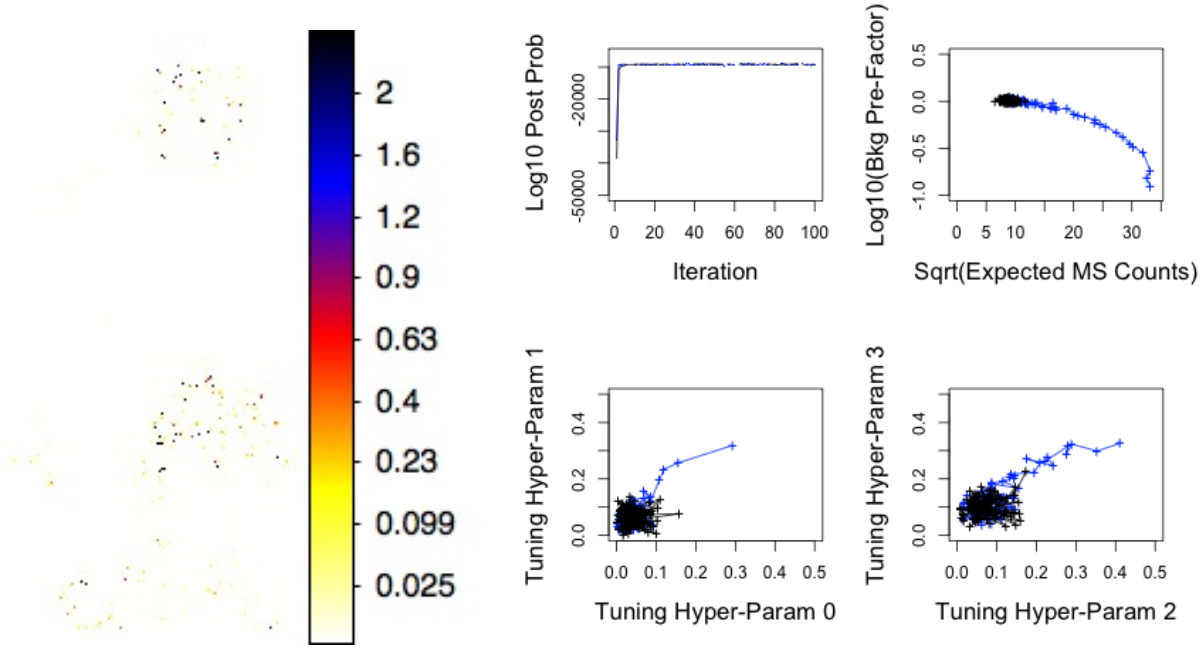


Fig. 24.— **Iteration 100** Two start values: high (top image; blue); low (bottom image; black).

APPENDIX B: Getting your own copy:

Contacts:

1. Nathan Stein : nmstein (fas dot harvard dot edu) or nathanmstein (gmail dot com)
2. Alanna Connors : aconnors (eurekabayes dot com)
3. David van Dyk : david.a.vandyk (gmail dot com)

Building and installing.

1. Install R from cran: `cran.r-project.org/doc/manuals/R-admin.html`
2. If you wish, install R's fitsIO package, from Harris of STSci. It is listed under 'packages', so you can use the handy R-GUI package installer (menu bar at top).
3. Get a LIRA tar-ball either from the contacts above, or from the svn repository: `hea-www.harvard.edu/codecomb/EMC2/trunk/EMC2/R-package/`
4. un-gzip and un-tar the file (tar xzvf lira-etc.gz)
5. Make sure your machine's gcc matches that of the R-build e.g. for Mac darwin: sudo select gcc 4
6. Change your working directory to `lira/`.
7. For unix/linux–like operating systems, do the usual R-package procedure:
   (a) R CMD BUILD
   (b) R CMD INSTALL

   NOTE: Mac users may prefer to "sudo R CMD BUILD" and sudo "R CMD INSTALL".
8. LIRA has not yet been built on a Windows machine. Theoretically one could follow the procedures documented in `cran.r-project.org/doc/manuals/R-admin.html`.
9. If you wish, you can try out some of the R-command files in `tests/`.
10. Check out the documentation! Hopefully during the INSTALL, the `docs/` directory moved from `inst/` (the *install* directory) to the main directory.

That's it! Have fun!

APPENDIX C: Quick Summary of Steps to Run LIRA

For those who already understand MMIs, MCMC, Low-count Poisson, Bayesian priors and hyperpriors, etc., etc., we offer this quick outline of how to run LIRA.

1. INPUTS/OUTPUTS
   (a) INPUTS:
      i. Required inputs:
         A. A $2^n \times 2^n$ matrix of data, in the form of cts/bin;
         B. Number of iterations, thinning, names of output files. Defaults are available for all of these if none were given.
      ii. Optional:
         A. A $2^n \times 2^n$ background or best-fit model or null model map
         B. An exposure map of matching dimensions. If there is an exposure map, then the units of the exposure map times the background or model map should be cts/bin, matching the data map.
         C. A map of the point-spread function, or instrument smearing. The bin-size should match that of the data, but the size can be arbitrary (but no larger than the data map).
   (b) OUTPUTS:
      i. A log file, *.Rout, that echoes the inputs and keeps track of the usage time. Any error messages should end up in here.
      ii. An ascii parameter file, typically *.param, that has a header designated by "#"; and a list of parameters and hyper-parameters for each MCMC iteration. If "thin" is set > 1, then only every "thin" iterations will be written out.
      iii. An ascii output of the MCMC images, typically *.out. If the input data map has dimension $2^n \times 2^n$, then each row in the file will contain 2^n ascii entries. If "thin" is set > 1, then only every "thin" iterations will be written out. Hence the entries written to the "param" file and the sample images written to this "*.out" file will match.
2. CHECKING:
   (a) Do a few short runs (iter at least a few hundred), with a minmum of three separated starting values
   (b) Plot traces and scatter plots. Looks OK? Burn-in? Correlation Length?
   (c) Look at movies (it's fun!). Do the images make sense?

(d) If wished, look at the mean and higher moments (after burn-in). Do these images make sense?

(e) Given the limits you want, estimate resources for the longer runs: How many separate runs with separate starting maps (at least three)? How many iterations for each run? How long will each take?

3. LONGER ANALYSIS RUN:

(a) If it all works out, do several longer runs (iter 1000 or more).

(b) Again, plot traces and scatter plots. All OK?

(c) Look at the movies from separate runs. All OK?

(d) Combine, after chopping burn-in, to calculate mean and higher moments

(e) Scrutinize combinations of the fit-parmaters. Which is the appropriate summary statistic for your problem?

(f) Sort the samples according to your summary statistic. Select the ones that are around n% and 100-n% of the tail. Averaging these two sets of images will give you your upper and lower confidence bounds.

4. COMPARING WITH NULL MODEL FOR GOODNESS OF FIT:

This is a short description, but running all the nulls can take a long time.

(a) Create several Posson realizations of your Null model.

(b) Analyze in exactly the same way as in 2 and 3, above.

(c) COMPARE the scatter plots and especially the summary statistic distributions. How much overlap?

(d) COMPARE the Null and Interesting images at the same values of the summary statistic that gave the +/- confidence bounds.

(e) Make sense of the result!

That's it! Have fun!

APPENDIX C: Special Notes for Users:

1. LIRA **REQUIRES** unprocessed counts, since it uses the Poisson formulation. The data CANNOT be background-subtracted (data-cuts are fine, though); or averaged or extrapolated (as in some CGRO/EGRET all-sky maps), or 'corrected' by an exposure factor (as is often done with TeV telescope data). All such processing belongs in the 'Instrument' model in the forward fitting: effective area; PSF; exposure map; and background. The process will simply NOT WORK (i.e. give you very weird results) if you try running on data that have been 'corrected' in some fashion.
2. As it is now written, LIRA REQUIRES the data to be in square maps of dimension 2^n. Any accompanying model, exposure, and background maps must match the data. In theory, Nowak and Kolaczyk have found a way, in one dimension, to lift the 2^n requirement - by implicitly embedding it in a larger 2^n vector. We have not done this work, in 2D, for LIRA.
3. Note that LIRA has smoothing intrinsically built into the multi-scale component. The multi-scale portion is NOT engineered to detect point-sources (although it can – but the fluxes will tend to be a little low).
4. No longer always convex, as we have introduced more competing model components (see next). SO should try multiple starting maps to make sure they all converge to the same final chain. (see next section.)
5. Although this is only important when running with a non-zero background model, it is often a good idea to include MCMC runs with: 1) starting values too high (i.e. puts all counts in the MS component, to start); and 2) starting values very low (i.e. puts all counts in the background component, to start). If there is going to be any problem with multiple modes, this can illustrate it rather quickly.