

---

# **autodmri Documentation**

***Release v0.2.3***

**Samuel St-Jean**

**Mar 09, 2021**



---

## Modules:

---

<b>1</b>	<b>autodmri</b>	<b>3</b>
1.1	autodmri package . . . . .	3
1.1.1	Submodules . . . . .	3
1.1.2	autodmri.blocks module . . . . .	3
1.1.3	autodmri.estimator module . . . . .	4
1.1.4	autodmri.gamma module . . . . .	4
1.1.5	Module contents . . . . .	5
<b>2</b>	<b>Indices and tables</b>	<b>7</b>
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



This is the documentation detailing the internal of Automated characterization of noise distributions in diffusion MRI data, which is available at <https://github.com/samuelstjean/autodmri>.

You can find the original paper and full details of the algorithm as presented in

```
St-Jean S, De Luca A, Tax CMW, Viergever MA, Leemans A.  
Automated characterization of noise distributions in diffusion MRI data.  
Med Image Anal. June 2020:101758. doi:10.1016/j.media.2020.101758
```

Which you can grab a copy from [media](#).

There is also the previous conference manuscript which details a different set of experiments

```
St-Jean, Samuel, De Luca, Alberto, Viergever, Max A. and Leemans, Alexander  
'Automatic, Fast and Robust Characterization of Noise Distributions for Diffusion_  
↔MRI',  
MICCAI 2018. Springer International Publishing, pp. 304-312. doi: 10.1007/978-3-030-  
↔00928-1_35.
```

Which is also available on the publisher website [miccai](#) or [arxiv\\_miccai](#).

The datasets are also available at [zenodo](#).

You can find below the documentation for each modules.



## 1.1 autodmri package

### 1.1.1 Submodules

### 1.1.2 autodmri.blocks module

`autodmri.blocks.extract_patches` (*arr*, *patch\_shape*, *extraction\_step*, *flatten=True*)

Extracts patches of any n-dimensional array in place using strides. Given an n-dimensional array it will return a 2n-dimensional array with the first n dimensions indexing patch position and the last n indexing the patch content. This operation is immediate (O(1)). A reshape performed on the first n dimensions will cause numpy to copy data, leading to a list of extracted patches.

#### Parameters

- **arr** (*ndarray*) – n-dimensional array of which patches are to be extracted
- **patch\_shape** (*integer or tuple of length arr.ndim*) – Indicates the shape of the patches to be extracted. If an integer is given, the shape will be a hypercube of sidelength given by its value.
- **extraction\_step** (*integer or tuple of length arr.ndim*) – Indicates step size at which extraction shall be performed. If integer is given, then the step is uniform in all dimensions.

**Returns patches** – 2n-dimensional array indexing patches on first n dimensions and containing patches on the last n dimensions. These dimensions are fake, but this way no data is copied. A simple reshape invokes a copying operation to obtain a list of patches: `result.reshape([-1] + list(patch_shape))`

**Return type** strided ndarray

### 1.1.3 autodmri.estimate module

```
autodmri.estimate_from_dwis(data, axis=-2, return_mask=False, exclude_mask=None, ncores=-1, method='moments', verbose=0, fast_median=False)
```

Given the data, splits over each slice to compute parameters of the gamma distribution

**data** : array, input volume used to identify noise voxels and estimate the noise distribution

**axis** : (0, 1 or 2) the axis to consider as a slab of uniform noise profile

**return\_mask** : bool, if True returns the identified noise voxels as a mask

**exclude\_mask** : array, mask indicating voxels to remove from all the computations, such as those containing huge artifacts.

**ncores** : int, number of cores to use for multiprocessing

**method='moments'** or **method='maxlk'** : which algorithm to use to estimate sigma and N

**verbose** : print progress for joblib, can be an integer to increase verbosity

**fast\_median** : Computes the median of medians from each volume. Useful for large datasets with many volumes (e.g. HCP) since the median requires a full copy of the data and sorting.

sigma, N, mask (optional)

```
autodmri.estimate_from_nmaps(data, size=5, return_mask=True, method='moments', full=False, ncores=-1, use_rejection=False, verbose=0)
```

Given the data, estimates parameters of the gamma distribution in small 3D windows.

**data** : noise maps to use for parameter estimation.

**size** : size of the 3D windows (default 5)

**return\_mask** : bool, if True returns the identified noise voxels as a mask

**method='moments'** or **method='maxlk'** : which algorithm to use to estimate sigma and N

**full** : bool, if True estimates are made in overlapping windows

**ncores** : int, number of cores to use for multiprocessing

**use\_rejection** : if True, iterate to reject voxels in each estimated window, but this is much slower than just using all of the data.

**verbose** : print progress for joblib, can be an integer to increase verbosity

sigma, N, mask (optional)

```
autodmri.estimate.proc_inner(cur_map, median, size, method, use_rejection)
```

### 1.1.4 autodmri.gamma module

```
autodmri.gamma.get_noise_distribution(data, method='moments')
```

Computes sigma and N from an array of gamma distributed data

**data** A numpy array of gamma distributed values

**method='moments'** or **method='maxlk'** Use either the moments or maximum likelihood equations to estimate the parameters.



**sigma**, **N** parameters related to the original Gaussian noise distribution

`autodmri.gamma.inv_digamma` (*y*, *eps*=*1e-08*, *max\_iter*=*100*)

Numerical inverse to the digamma function by root finding

`autodmri.gamma.maxlk_sigma` (*m*, *xold*=*None*, *eps*=*1e-08*, *max\_iter*=*100*)

Maximum likelihood equation to estimate sigma from gamma distributed values

### 1.1.5 Module contents



## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**a**

autodmri, 5  
autodmri.blocks, 3  
autodmri.estimate, 4  
autodmri.gamma, 4



## A

`autodmri` (*module*), 5  
`autodmri.blocks` (*module*), 3  
`autodmri.estimator` (*module*), 4  
`autodmri.gamma` (*module*), 4

## E

`estimate_from_dwls()` (*in module autodmri.estimator*), 4  
`estimate_from_nmaps()` (*in module autodmri.estimator*), 4  
`extract_patches()` (*in module autodmri.blocks*), 3

## G

`get_noise_distribution()` (*in module autodmri.gamma*), 4

## I

`inv_digamma()` (*in module autodmri.gamma*), 5

## M

`maxlk_sigma()` (*in module autodmri.gamma*), 5

## P

`proc_inner()` (*in module autodmri.estimator*), 4