soif Documentation

Release 1.0.3

Guillaume Schworer

August 21, 2016

CONTENTS

CHAPTER

ONE

SOIF PACKAGE

1.1 Subpackages

1.1.1 soif.test package

Submodules

soif.test.test__core module

soif.test.test__core.**test_abs2**(*seed=42*)

soif.test.test__core.**test_gen_seed**()

Module contents

1.2 Submodules

1.3 soif.oidata module

class soif.oidata.**Oidata**(*src*, *hduidx*, *datatype*, *hduwlidx*, *indices=[]*, *wlindices=[]*, *degree=True*, *flatten=False*, *significant_figures=5*, ***kwargs*)

Bases: *soif.oidata.OidataEmpty*

data

error

flat

flatten(***kwargs*)

is_angle

is_t3

mask

shapedata

shapeuv

u

update()
Given u, v, wl and flag information as object properties, this function updates the Oidata object: the data masking (from the new mask property) and the bl, pa, blwl properties (from u, v and wl properties)

v

wl

wl_d

class soif.oidata.**OidataEmpty**(*datatype*, ***kwargs*)
Bases: object

useit

class soif.oidata.**Oifits**(*src, datafilter, wl=[None, None], erb_sigma=None, sigma_erb=None, systematic_prior=None, systematic_bounds=None, flatten=False, degree=True, significant_figures=5, **kwargs*)
Bases: object

This class opens, reads and sorts data contained in the file 'src' (oifits format).

vis2, vis and t3 are lists of which corresponding data indeces to extract from the file.

addData(*src, datafilter={}, flatten=False, degree=True, significant_figures=5, wl=[None, None], **kwargs*)

flatten()
Flattens all data contained in the Oidata object. This can be useful in order to add several bits of data that do not have the same shapes

remorph(*viscomp*)

save(*filename*, *append=False*, *clobber=False*)

systematic_fit

systematic_p0()

update()
Updates all data contained in the Oidata object

class soif.oidata.**Oigrab**(*src*, ***kwargs*)
Bases: object

Opens, reads and filters data contained in the OIFITS file src.

Args:

- src (str): the path (relative or absolute) to the OIFITS file

Kwargs:

- raiseError (bool): if True, raises errors, otherwise prints them

Raises:

- NoTargetTable: if the OIFITS file has no OI_TARGET table

```
>>> import soif.oidata as oidata
>>> data = oidata.Oigrab('./data/datafile.oifits')
```

extract(*tgt=None, mjd=[None, None], wl=[None, None], hduNums=[], vis2=True, t3phi=True, t3amp=True, visphi=True, visamp=True, flatten=False, degree=True, significant_figures=5, erb_sigma=None, sigma_erb=None, systematic_prior=None, systematic_bounds=None, verbose=False, **kwargs*)

show_filtered(*tgt=None, mjd=[None, None], hduNums=[], vis2=True, t3phi=True, t3amp=True, visphi=True, visamp=True, verbose=False, **kwargs*)

Given an oifits file 'src' and filtering parameters on the target name (OI_TARGET table), the instrument name (OI_WAVELENGTH table), the array name (OI_ARRAY table), the observation wavelength (OI_WAVELENGTH table) and the acquisition time [t_min, t_max] (OI_VIS2, OI_VIS, OI_T3 tabkes), this function returns the data indices of the data matching all of these different filters. These lists are used to load the data within an Oidata object.

Leave input parameter to 'None' to discard filtering on that particular parameter.

Returns: VIS2, T3, VIS indeces as a tuple of 3 lists

show_specs(*ret=False*, **kwargs*)

Gets the target list and the data details from the OIFITS file.

Args:

- ret (bool): if `True`, returns the information, otherwise prints it

Returns:

- a dictionary {'hdu index:info'} where info corresponds to a list of (Acquisition index, Target ID, MJD, N(UV), N(wl)) tuples

```
>>> import soif.oidata as oidata
>>> data = oidata.Oigrab('./data/datafile.oifits')
>>> data.showspecs()
TARGETS:
1: IRS_48
2: Elia_2-15
```

VIS2 [hdu=3]: Acq. Index | Target ID | MJD | UVs | N wl ————————————————

0 | 1 | 55636.3827746 | 21 | 1 1 | 2 | 55636.3827989 | 21 | 1 2 | 1 | 55636.3828232 | 21 | 1

targets

1.4 soif.oiexception module

exception `soif.oiexception.`**BadMaskShape**(*shape*, **args*, ***kwargs*)

Bases: *soif.oiexception.OIException*

If the mask shape does not match the data shape

exception `soif.oiexception.`**HduDatatypeMismatch**(*hduhead*, *datatype*, **args*, ***kwargs*)

Bases: *soif.oiexception.OIException*

If the data type and the hdu provided do not match

exception `soif.oiexception.`**IncompatibleData**(*typ1*, *typ2*, **args*, ***kwargs*)

Bases: *soif.oiexception.OIException*

If the data type and the hdu provided do not match

exception `soif.oiexception.`**InvalidDataType**(*datatype*, **args*, ***kwargs*)

Bases: *soif.oiexception.OIException*

If the data type provided does not exist

exception soif.oiexception.**NoSystematicsFit**(**args*, ***kwargs*)
Bases: *soif.oiexception.OIException*

If the user did not set on the fit of systematics

exception soif.oiexception.**NoTargetTable**(*src=''*, **args*, ***kwargs*)
Bases: *soif.oiexception.OIException*

If the file has no OITARGET table

exception soif.oiexception.**NoWavelengthTable**(*src=''*, **args*, ***kwargs*)
Bases: *soif.oiexception.OIException*

If the file has no OITARGET table

exception soif.oiexception.**NotADataHdu**(*idx*, *src*, **args*, ***kwargs*)
Bases: *soif.oiexception.OIException*

If the hdu provided does not contain data

exception soif.oiexception.**NotCallable**(*fct*, **args*, ***kwargs*)
Bases: *soif.oiexception.OIException*

If the function is callable

exception soif.oiexception.**OIException**(**args*, ***kwargs*)
Bases: exceptions.Exception

Root for SOIF Exceptions, only used to trigger any soif errors, never raised

exception soif.oiexception.**ReadOnly**(*attr*, **args*, ***kwargs*)
Bases: *soif.oiexception.OIException*

If the parameter is read-only

exception soif.oiexception.**WrongData**(*typ*, **args*, ***kwargs*)
Bases: *soif.oiexception.OIException*

If the data provided has the wrong data type

soif.oiexception.**doraise**(*obj*, ***kwargs*)

soif.oiexception.**raiseIt**(*exc*, *raiseoupas*, **args*, ***kwargs*)

1.5 soif.oifiting module

class soif.oifiting.**Oifiting**(*model*, *nwalkers=100*, *niters=500*, *burnInIts=100*, *threads=1*, *customlike=None*, ***kwargs*)
Bases: MCres.MCres.MCres

MCmap(*param_x*, *param_y*, *bin_x=50*, *bin_y=50*, *cmap='jet'*, *cm_min=None*, *cm_max=None*, *axescolor='w'*, *polar=False*, *showmax=True*, *radec=False*, ***kwargs*)
Return a 2D histogram of the MC chain, showing the walker density per bin

Pbmap(*param_x*, *param_y*, *bin_x=50*, *bin_y=50*, *cmap='jet'*, *cm_min=None*, *cm_max=None*, *axescolor='w'*, *polar=False*, *showmax=True*, *radec=False*, ***kwargs*)
Return a 2D histogram of the MC chain, showing the best loglikelihood per bin

image(*params=None*, *sepmax=None*, *wl=None*, *masperpx=None*, *nbpts=101*, *cmap='jet'*, *cm_min=None*, *cm_max=None*, *ret=False*, *visu=None*, ***visuargs*)

imagefft (*sepmax=None*, *wl=None*, *params=None*, *nbpts=101*, *cmap='jet'*, *cm_min=None*, *cm_max=None*, *ret=False*, *visu=None*, ***visuargs*)

residual (*params=None*, *c=None*, *cmap='jet'*, *cm_min=None*, *cm_max=None*, *datatype='All'*)

run (*niters*, *burnInIts=0*)

Start a MCMC simulation on as many CPUs as threads parameter. Use customlike parameter to use a custom li

e.g.: mycustomlike(modeledData, model, **kwargs)

save (*name*, *clobber=False*)

statut (*params=None*, *customlike=None*, ***kwargs*)

uvimage (*blmax=None*, *wl=None*, *params=None*, *datatype='Data'*, *nbpts=101*, *withdata=True*, *cmap='jet'*, *cm_min=None*, *cm_max=None*, *ret=False*)

Datatype in ['vis2', 'phase', 'vis']

1.6 soif.oiload module

1.7 soif.oimainobject module

class soif.oimainobject.**Oimainobject** (*name*, *priors={}*, *bounds={}*, *verbose=False*, **args*, ***kwargs*)

Bases: object

compVis (*oidata*, *params=None*, *flat=False*)

Does the paperwork before calculating the complex visibilities of the object

getP0 ()

Returns a list of initial values for each parameter in the object, according the to parameter keys order

keys

oscil (*u*, *v*, *wl*)

params

Returns a list of the parameters values, according to the parameter keys order

save (*filename*, *append=False*, *clobber=False*)

setParams (*params*, *priors=False*)

Affects to the object the registered values from params, according the to parameter keys order

show ()

to_pospx (*sepmax*, *nbpts*, *integer=False*)

to_radec ()

Returns (ra, dec) in radian from sep and pa

typ

1.8 soif.oimodel module

class soif.oimodel.**Oimodel** (*oidata*, *objs=[]*, *tweakparams=None*, ***kwargs*)

Bases: object

add_obj(*typ*, *name=None*, *params={}*, *prior={}*)
Add an object to the model

compVis(*params=None*)
Calculate the complex visibility of the model from each separate object

compimage(*params=None*, *sepmax=None*, *wl=None*, *masperpx=None*, *nbpts=101*, *psfConvolve=None*, ***kwargs*)
psfConvolve in mas (lambda/D)

compuvimage(*blmax*, *wl=None*, *params=None*, *nbpts=101*)

del_obj(*idobj*)
Delete an object from the model. idobj can be the name of the object or its index in the model list

getP0()
Return an initialized param list

image(*params=None*, *sepmax=None*, *wl=None*, *masperpx=None*, *nbpts=101*, *cmap='jet'*, *cm_min=None*, *cm_max=None*, *ret=False*, *visu=None*, *psfConvolve=None*, ***visuargs*)
psfConvolve in mas (lambda/D)

likelihood(*params*, *customlike=None*, *chi2=False*, ***kwargs*)

nparams

nparamsObjs

params
Return current params

paramstr

reinit(*params*)

residual(*params*, *c=None*, *cmap='jet'*, *cm_min=None*, *cm_max=None*, *datatype='All'*)

save(*filename*, *clobber=False*)

setParams(*params*, *priors=False*)

statut(*params*, *customlike=None*, ***kwargs*)

uvimage(*params=None*, *blmax=None*, *wl=None*, *typ='vis2'*, *nbpts=101*, *cmap='jet'*, *cm_min=None*, *cm_max=None*, *ret=False*, *visu=None*, ***visuargs*)
typ can be: vis, vis2, phase

1.9 soif.oipriors module

soif.oipriors.**lnnormal**(*x*, *x0=0.0*, *sigma=1.0*, **args*, ***kwargs*)
Returns the probability of drawing x

soif.oipriors.**lnnormalbumpy**(*x*, *x0=0.0*, *rangeshrink=10*, **args*, ***kwargs*)
Returns the probability of drawing x

soif.oipriors.**lntriangle**(*x*, *x0=0*, *slope=4*, **args*, ***kwargs*)
Returns the log-probability of drawing any value in the range of the prior

soif.oipriors.**lnuniform**(*x*, *prior_lninvrange=0*, **args*, ***kwargs*)
Returns the log-probability of drawing any value in the range of the prior

soif.oipriors.**normal**(*x*, *x0=0.0*, *sigma=1.0*, **args*, ***kwargs*)
Returns the probability of drawing x

soif.oipriors.**normalbumpy**(*x*, *x0=0.0*, *rangeshrink=10*, **args*, ***kwargs*)
Returns the probability of drawing x

soif.oipriors.**triangle**(*x*, *x0=0*, *slope=4*, **args*, ***kwargs*)
Returns the probability of drawing any value in the range of the prior

soif.oipriors.**uniform**(*x*, *prior_invrange=0*, **args*, ***kwargs*)
Returns the probability of drawing any value in the range of the prior

1.10 soif.oiunitmodels module

class soif.oiunitmodels.**BGimage**(*name*, *img=None*, *masperpx=None*, *priors={}*, *bounds={}*, *negRA=False*, *totFlux=None*, **args*, ***kwargs*)
Bases: *soif.oimainobject.Oimainobject*

image(*sepmax=None*, *masperpx=None*, *wl=None*, *nbpts=101*)

img

prepare(*oidata*, *force=False*)

class soif.oiunitmodels.**Gauss**(*name*, *priors={}*, *bounds={}*, *verbose=False*, **args*, ***kwargs*)
Bases: *soif.oimainobject.Oimainobject*

image(*sepmax*, *masperpx=None*, *wl=None*, *nbpts=101*)

class soif.oiunitmodels.**Gauss2D**(*name*, *priors={}*, *bounds={}*, *verbose=False*, **args*, ***kwargs*)
Bases: *soif.oiunitmodels.Gauss*

image(*sepmax*, *masperpx=None*, *wl=None*, *nbpts=101*)

class soif.oiunitmodels.**GaussDiff**(*name*, *priors={}*, *bounds={}*, *verbose=False*, **args*, ***kwargs*)
Bases: *soif.oimainobject.Oimainobject*

image(*sepmax*, *masperpx=None*, *wl=None*, *nbpts=101*)

class soif.oiunitmodels.**GaussDiff2D**(*name*, *priors={}*, *bounds={}*, *verbose=False*, **args*, ***kwargs*)
Bases: *soif.oiunitmodels.GaussDiff*

image(*sepmax*, *masperpx=None*, *wl=None*, *nbpts=101*)

class soif.oiunitmodels.**PointSource**(*name*, *priors={}*, *bounds={}*, **args*, ***kwargs*)
Bases: *soif.oimainobject.Oimainobject*

image(*sepmax*, *masperpx=None*, *wl=None*, *nbpts=101*)

class soif.oiunitmodels.**PointSourceSpectral**(*name*, *spectralKey*, *oidata*, *priors={}*, *bounds={}*, **args*, ***kwargs*)
Bases: *soif.oiunitmodels.Spectral*, *soif.oiunitmodels.PointSource*

class soif.oiunitmodels.**Spectral**(*oidata*, *spectralKey*, **args*, ***kwargs*)
Bases: object

spectralKey

class soif.oiunitmodels.**UniformDisk**(*name*, *priors={}*, *bounds={}*, **args*, ***kwargs*)
Bases: *soif.oimainobject.Oimainobject*

image(*sepmax*, *masperpx=None*, *wl=None*, *nbpts=101*)

class soif.oiunitmodels.**UniformDisk2D**(*name*, *priors={}*, *bounds={}*, **args*, ***kwargs*)
Bases: *soif.oiunitmodels.UniformDisk*

image(*sepmax*, *masperpx=None*, *wl=None*, *nbpts=101*)

class soif.oiunitmodels.**UniformDisk2DLinCR**(*name*, *priors={}*, *bounds={}*, **args*, ***kwargs*)
Bases: *soif.oiunitmodels.UniformDiskLinCR*

class soif.oiunitmodels.**UniformDiskLinCR**(*name*, *priors={}*, *bounds={}*, **args*, ***kwargs*)
Bases: *soif.oiunitmodels.UniformDisk*

cr

class soif.oiunitmodels.**UniformRing**(*name*, *priors={}*, *bounds={}*, **args*, ***kwargs*)
Bases: *soif.oimainobject.Oimainobject*

image(*sepmax*, *masperpx=None*, *wl=None*, *nbpts=101*)

class soif.oiunitmodels.**UniformRing2D**(*name*, *priors={}*, *bounds={}*, **args*, ***kwargs*)
Bases: *soif.oiunitmodels.UniformRing*

image(*sepmax*, *masperpx=None*, *wl=None*, *nbpts=101*)

1.11 Module contents

S

O

P

R

S

T

U

V

W