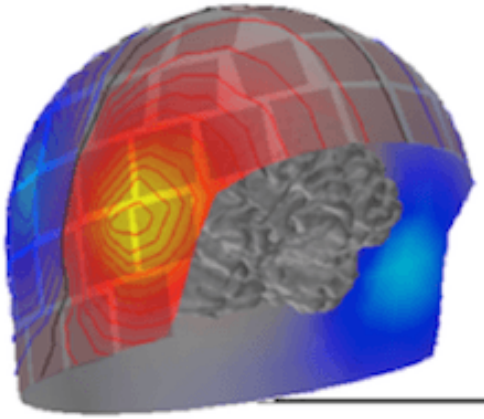


MNE

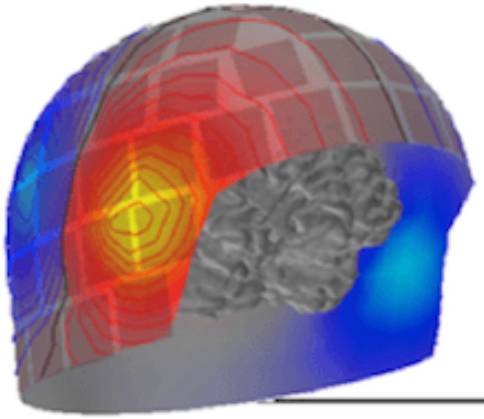
MNE from shell scripts and Unix commands to Python

Lorenzo De Santis

Université Paris-Sud - CNRS LPN

Supervisor: Alexandre Gramfort

Telecom ParisTech - CNRS LTCI - CEA Neurospin



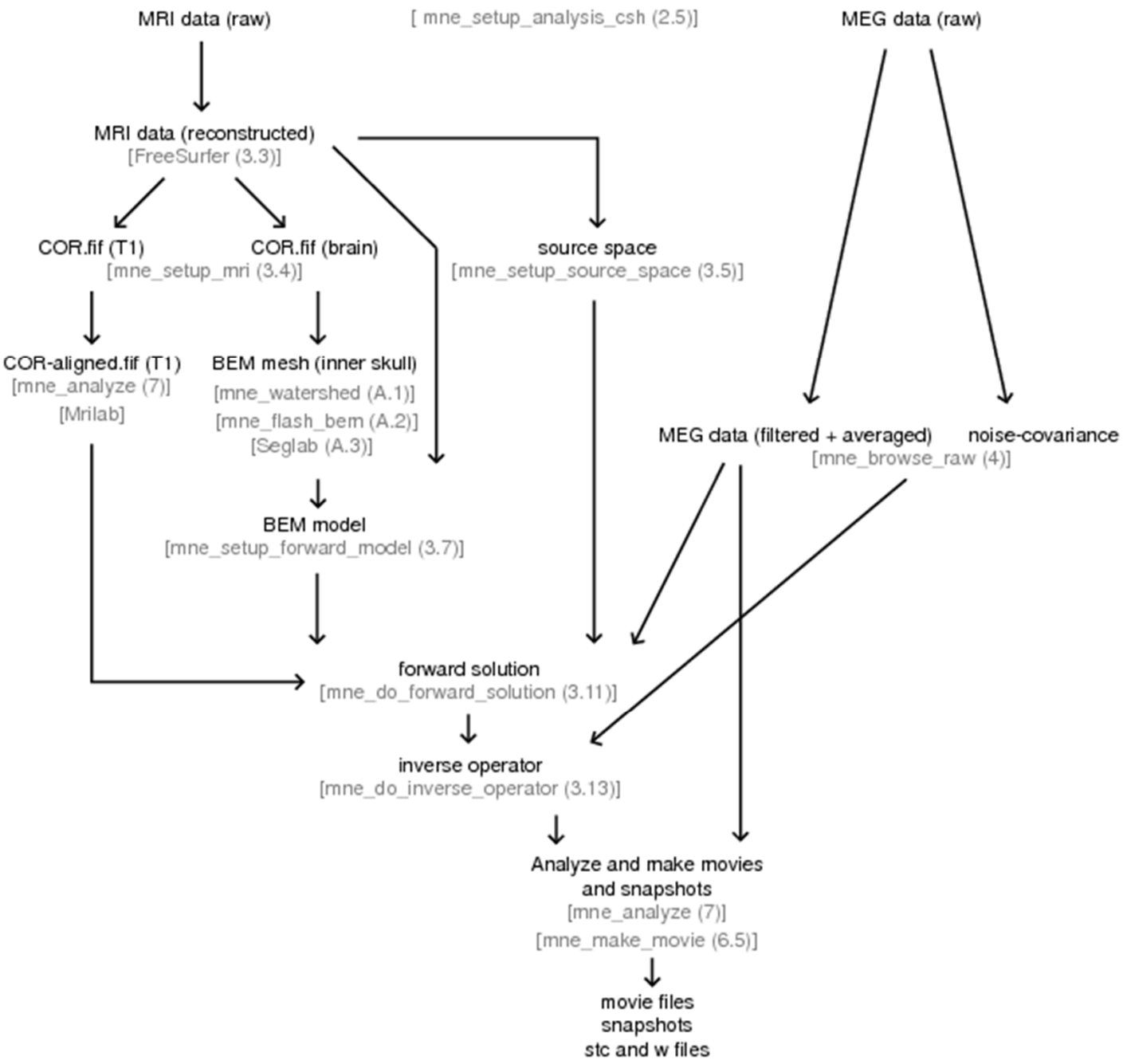
MNE

MNE software

software package for processing MagnetoEncephaloGraphy (MEG) and ElectroEncephaloGraphy (EEG) data and constructing cortically-constrained Minimum-Norm current Estimates

- provided as compiled C code for the LINUX and Mac OSX operating systems
- includes a Matlab toolbox for facilitated access and custom analysis
- newest component: **MNE-Python**

Workflow of the MNE software



Source: martinos.org/mne/stable/index.html

Workflow of the MNE software

MRI data (raw)

[mne_setup_analysis_csh (2.5)]

MEG data (raw)

MRI data (reconstructed)
[FreeSurfer (3.3)]

COR.fif (T1)
[mne_setup_mri (3.4)]

COR.fif (brain)
[mne_setup_mri (3.4)]

source space
[mne_setup_source_space (3.5)]

COR-aligned.fif (T1)
[mne_analyze (7)]
[Mrilab]

BEM mesh (inner skull)
[mne_watershed (A.1)]
[mne_flash_bem (A.2)]
[Seglab (A.3)]

BEM model
[mne_setup_forward_model (3.7)]

MEG data (filtered + averaged)
[mne_browse_raw (4)]

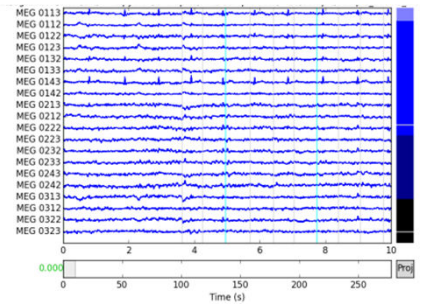
noise-covariance

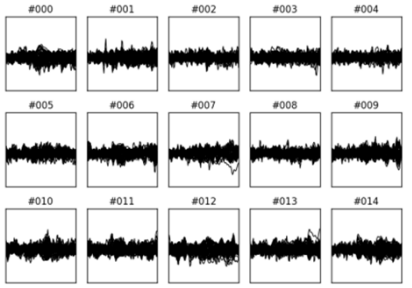
forward solution
[mne_do_forward_solution (3.11)]

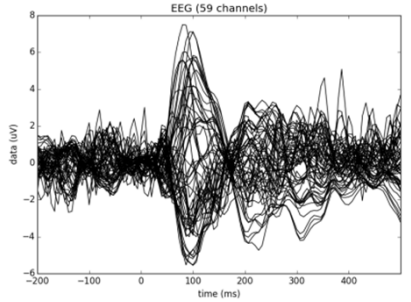
inverse operator
[mne_do_inverse_operator (3.13)]

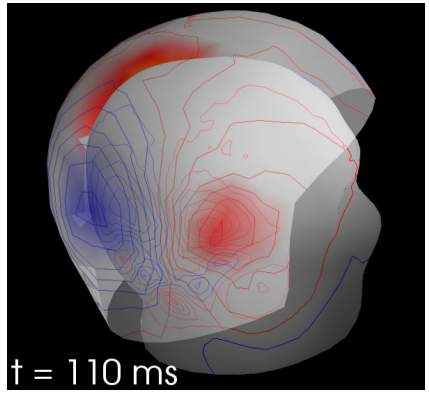
Analyze and make movies
and snapshots
[mne_analyze (7)]
[mne_make_movie (6.5)]

movie files
snapshots
stc and w files

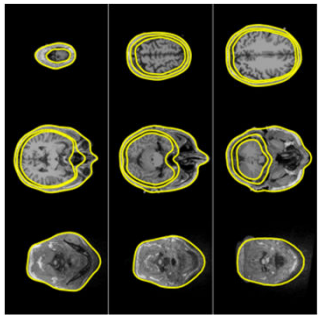


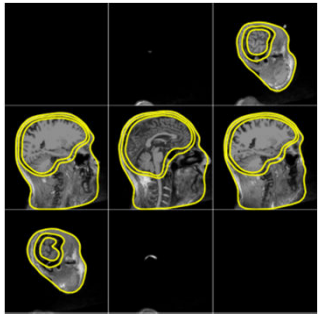


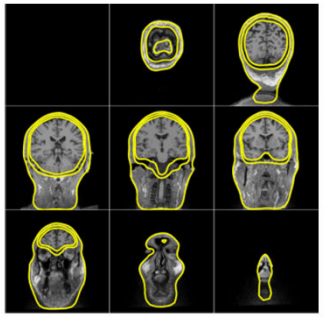




Workflow of the MNE software







MRI data (raw)

[mne_setup_analysis_csh (2.5)]

MEG data (raw)

MRI data (reconstructed)
[FreeSurfer (3.3)]

COR.fif (T1)
[mne_setup_mri (3.4)]

COR.fif (brain)
[mne_setup_mri (3.4)]

source space
[mne_setup_source_space (3.5)]

COR-aligned.fif (T1)
[mne_analyze (7)]
[Mrilab]

BEM mesh (inner skull)
[mne_watershed (A.1)]
[mne_flash_bem (A.2)]
[Seglab (A.3)]

BEM model
[mne_setup_forward_model (3.7)]

MEG data (filtered + averaged)
[mne_browse_raw (4)]

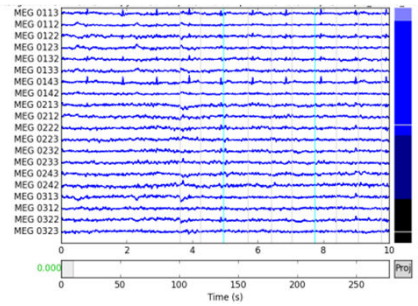
noise-covariance

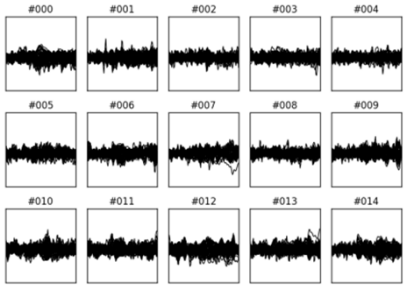
forward solution
[mne_do_forward_solution (3.11)]

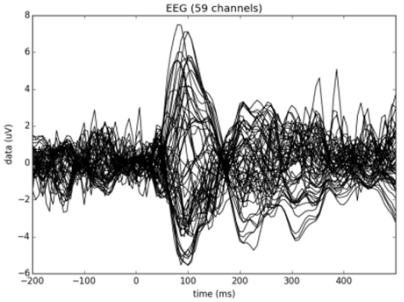
inverse operator
[mne_do_inverse_operator (3.13)]

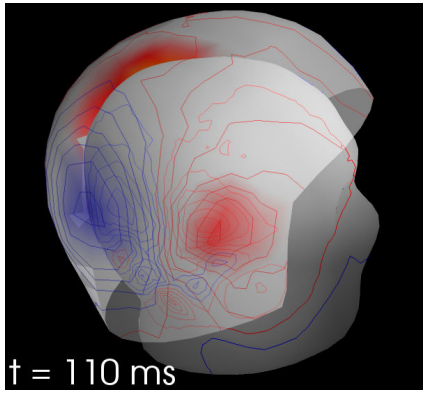
Analyze and make movies
and snapshots
[mne_analyze (7)]
[mne_make_movie (6.5)]

movie files
snapshots
stc and w files

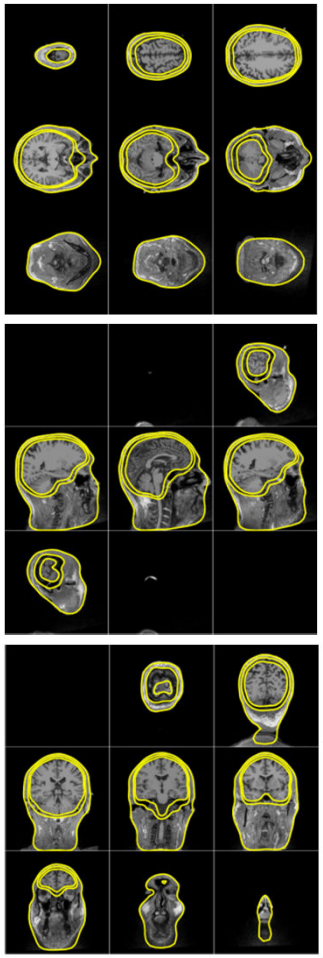








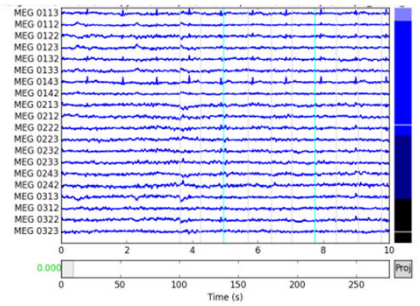
Workflow of the MNE software



MRI data (raw)

[mne_setup_analysis_csh (2.5)]

MEG data (raw)

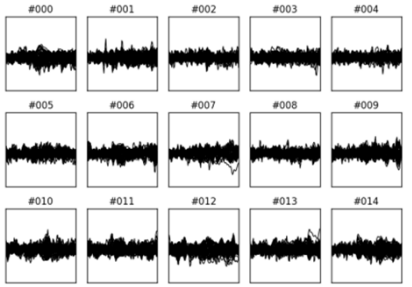


MRI data (reconstructed)
[FreeSurfer (3.3)]

COR.fif (T1)
[mne_setup_mri (3.4)]

COR.fif (brain)

source space
[mne_setup_source_space (3.5)]



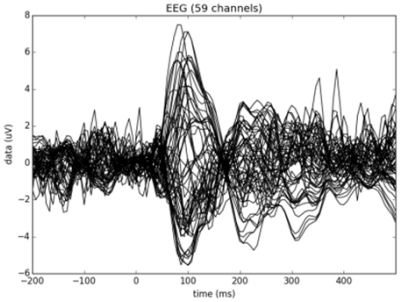
COR-aligned.fif (T1)
[mne_analyze (7)]
[Mrilab]

BEM mesh (inner skull)
[mne_watershed (A.1)]
[mne_flash_bem (A.2)]
[Seglab (A.3)]

MEG data (filtered + averaged)
[mne_browse_raw (4)]

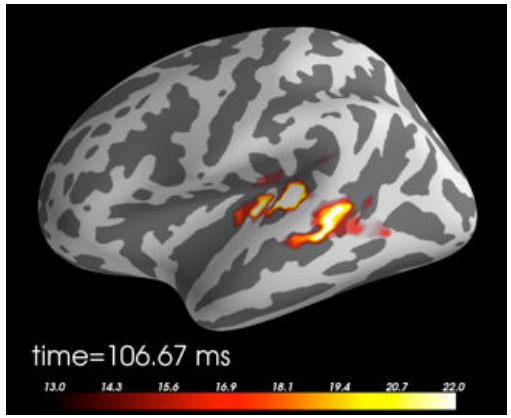
noise-covariance

BEM model
[mne_setup_forward_model (3.7)]



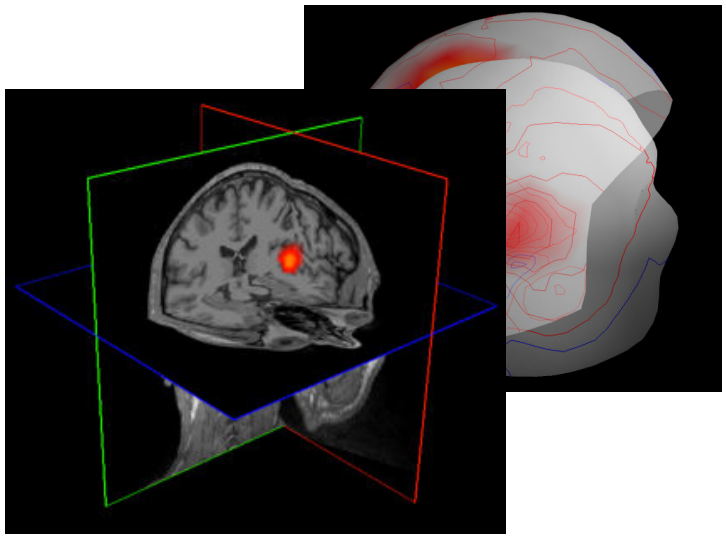
forward solution
[mne_do_forward_solution (3.11)]

inverse operator
[mne_do_inverse_operator (3.13)]



Analyze and make movies
and snapshots
[mne_analyze (7)]
[mne_make_movie (6.5)]

movie files
snapshots
stc and w files



The MNE-Python project

started in Dec. 2010 at MGH, Boston

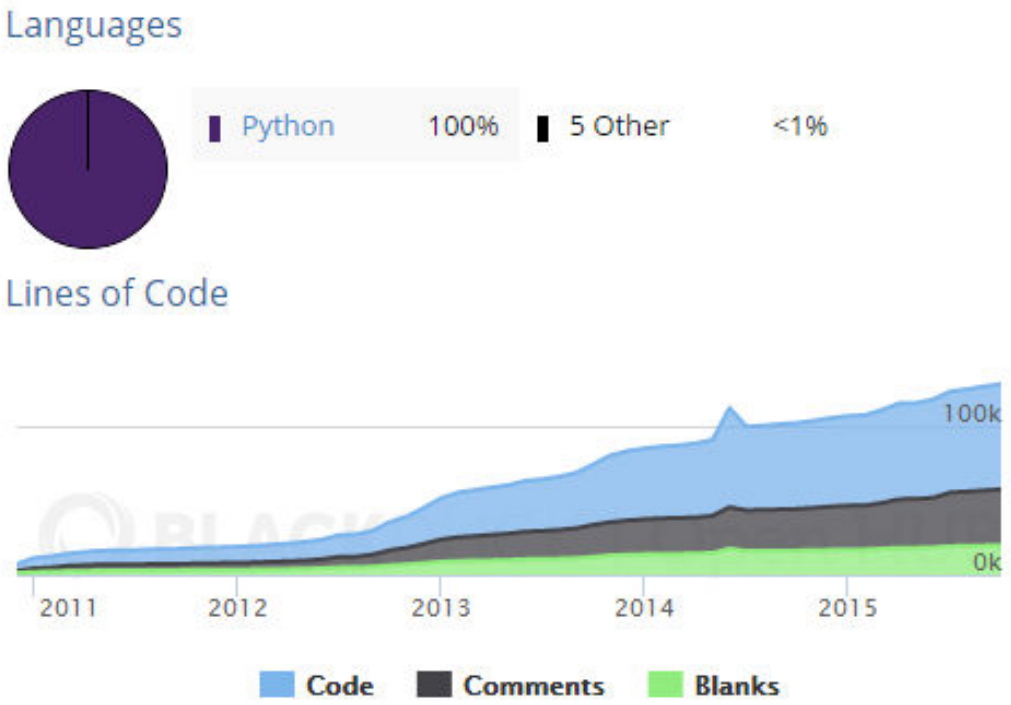
In a Nutshell, MNE-Python...

... has had 9,573 commits made by 98 contributors representing 70,783 lines of code

... is mostly written in Python with a well-commented source code

... has a codebase with a long source history maintained by a very large development team with increasing Y-O-Y commits

... took an estimated 18 years of effort (COCOMO model) starting with its first commit in December, 2010 ending with its most recent commit 8 days ago



Activity

30 Day Summary

Sep 17 2015 — Oct 17 2015

187 Commits

14 Contributors

including 2 new contributors

12 Month Summary

Oct 17 2014 — Oct 17 2015

3251 Commits

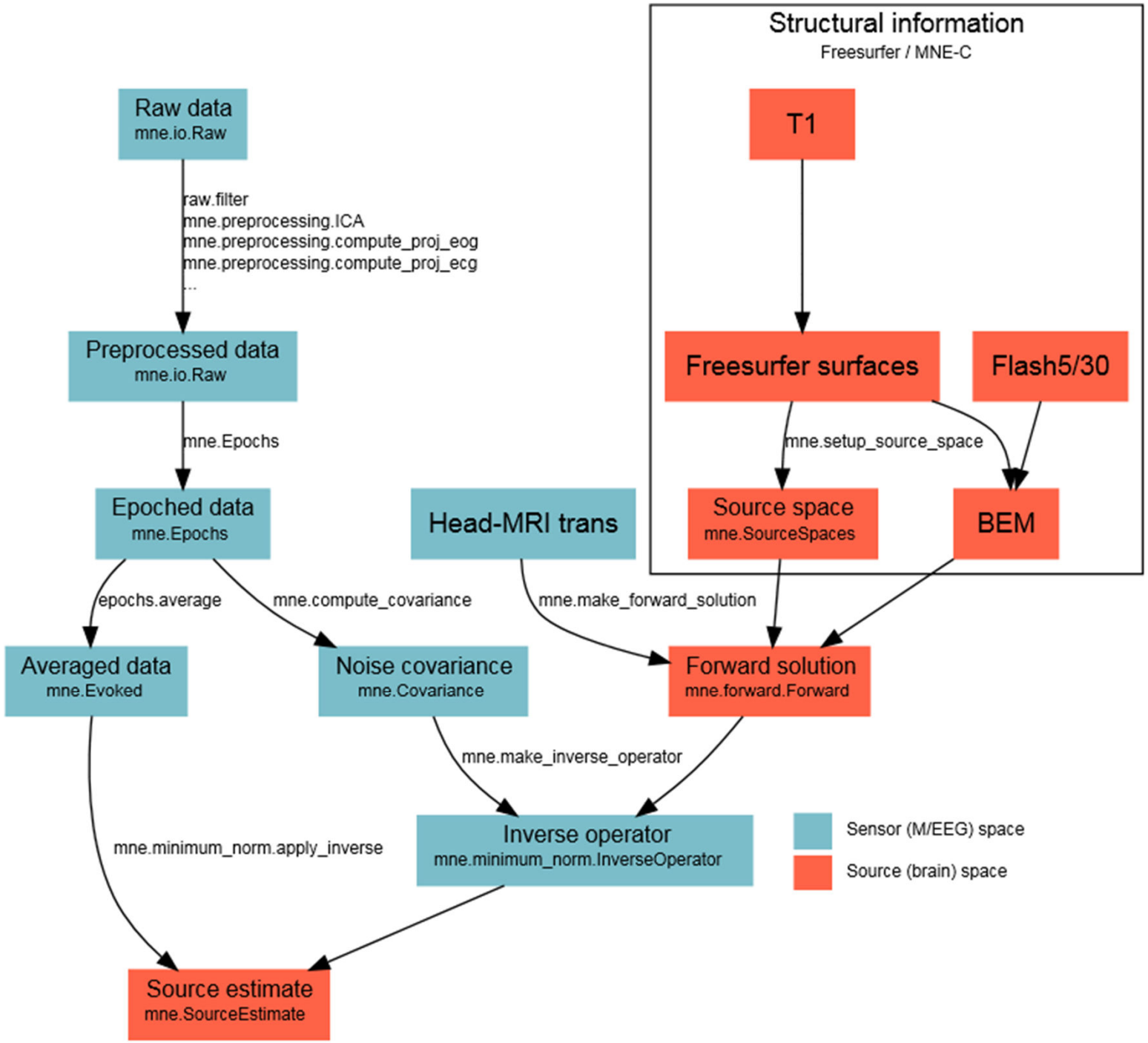
Up + 749 (29%) from previous 12 months

58 Contributors

Up + 11 (23%) from previous 12 months

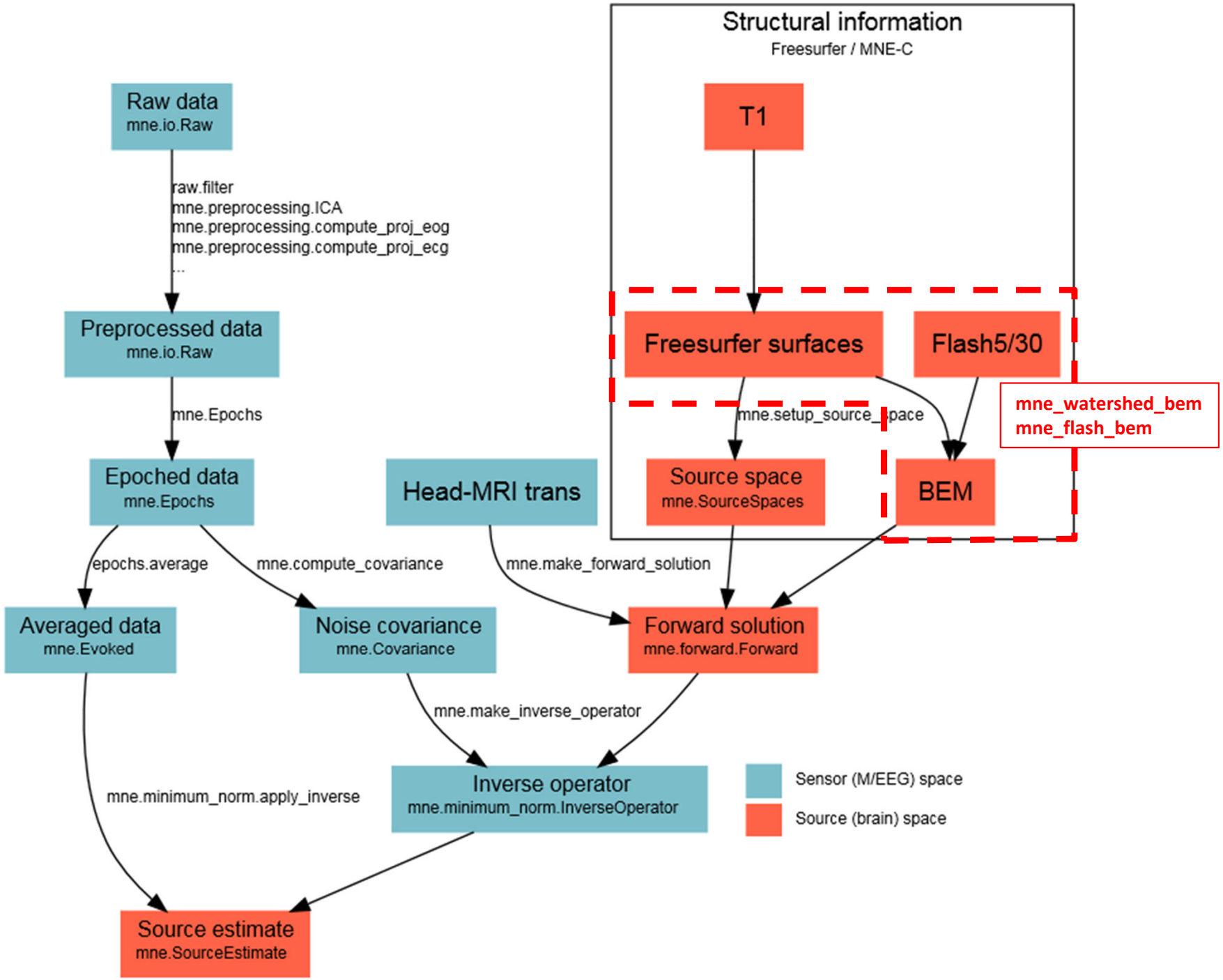
Source: https://www.ohloh.net/p/MNE

MNE-Python - workflow



Source: martinos.org/mne/stable/index.html

MNE-Python - workflow



Source: martinos.org/mne/stable/index.html

MNE-Python - Creating the BEM model meshes

mne_watershed_bem

the shell script

- **mri_watershed** $preflood $atlas -useSRAS -surf $ws_dir/$SUBJECT $T1_dir $ws_dir/w

  create BEM surfaces using the watershed algorithm included with FreeSurfer

- **mne_convert_surface** --surf $s --mghmri $T1_mgz --surfout $s
- **mne_surf2bem** --surf $ws_dir/"$SUBJECT"_outer_skin_surface --id 4 --fif $SUBJECT-head.fif

```
[...]
usage()
{
        echo "usage: $0 [options]"
        echo
        echo "    --overwrite           (to write over existing files)"
        echo "    --subject subject     (defaults to SUBJECT environment variable)"
        echo "    --volume  name        (defaults to T1)"
        echo "    --atlas               specify the --atlas option for mri_watershed"
        echo "    --gcaatlas            use the subcortical atlas«
        echo "    --preflood number     change the preflood height"
        echo
        echo "Minimal invocation:"
        echo
        echo "$0                  (SUBJECT environment variable set)"
        echo "$0 --subject subject   (define subject on the command line)"
        echo
}
[...]
```

calling the shell script:

```
$ mne_watershed_bem --subject sample --overwrite
```

MNE-Python - Creating the BEM model meshes

mne_watershed_bem

from the shell script to the Python function *make_watershed_bem*

```
@verbose
def make_watershed_bem(subject, subjects_dir=None, overwrite=False,
                       volume='T1', atlas=False, gcaatlas=False, preflood=None,
                       verbose=None):
[…]
# put together the command
    cmd = ['mri_watershed']
    if preflood:
        cmd += ["-h",  "%s" % int(preflood)]
    if gcaatlas:
        cmd += ['-atlas', '-T1', '-brain_atlas', env['FREESURFER_HOME'] +
                '/average/RB_all_withskull_2007-08-08.gca',
                subject_dir + '/mri/transforms/talairach_with_skull.lta']
    elif atlas:
        cmd += ['-atlas']
    if op.exists(T1_mgz):
        cmd += ['-useSRAS', '-surf', op.join(ws_dir, subject), T1_mgz,
                op.join(ws_dir, 'ws')]
    else:
        cmd += ['-useSRAS', '-surf', op.join(ws_dir, subject), T1_dir,
                op.join(ws_dir, 'ws')]
    # report and run
    logger.info('\nRunning mri_watershed for BEM segmentation with the '
                'following parameters:\n\n'
                'SUBJECTS_DIR = %s\n'
                'SUBJECT = %s\n'
                'Results dir = %s\n' % (subjects_dir, subject, ws_dir))
    os.makedirs(op.join(ws_dir, 'ws'))
    run_subprocess(cmd, env=env, stdout=sys.stdout)
[…]
```

MNE-Python - Creating the BEM model meshes

mne_watershed_bem

wrapping the function to a Python script

```
from mne.bem import make_watershed_bem

def run():
    from mne.commands.utils import get_optparser

    parser = get_optparser(__file__)

    parser.add_option("-s", "--subject", dest="subject",
                      help="Subject name (required)", default=None)
    parser.add_option("-d", "--subjects-dir", dest="subjects_dir",
                      help="Subjects directory", default=None)
    parser.add_option("-o", "--overwrite", dest="overwrite",
                      help="Write over existing files", action="store_true")
    parser.add_option("-v", "--volume", dest="volume",
                      help="Defaults to T1", default='T1')
    parser.add_option("-a", "--atlas", dest="atlas",
                      help="Specify the --atlas option for mri_watershed",
                      default=False, action="store_true")
    parser.add_option("-g", "--gcaatlas", dest="gcaatlas",
                      help="Use the subcortical atlas", default=False,
                      action="store_true")
    parser.add_option("-p", "--preflood", dest="preflood",
                      help="Change the preflood height", default=None)
    parser.add_option("--verbose", dest="verbose",
                      help="If not None, override default verbose level",
                      default=None)

    options, args = parser.parse_args()

    if options.subject is None:
        parser.print_help()
        sys.exit(1)

    subject = options.subject
    subjects_dir = options.subjects_dir
    overwrite = options.overwrite
    volume = options.volume
    atlas = options.atlas
    gcaatlas = options.gcaatlas
    preflood = options.preflood
    verbose = options.verbose

    make_watershed_bem(subject=subject, subjects_dir=subjects_dir,
                       overwrite=overwrite, volume=volume, atlas=atlas,
                       gcaatlas=gcaatlas, preflood=preflood,
verbose=verbose)

is_main = (__name__ == '__main__')
if is_main:
    run()
```

MNE-Python - Creating the BEM model meshes

mne_watershed_bem

testing the Python command

```
@ultra_slow_test
@requires_freesurfer
@testing.requires_testing_data
def test_watershed_bem():
    """Test mne watershed bem"""
    check_usage(mne_watershed_bem)
    # Copy necessary files to tempdir
    tempdir = _TempDir()
    mridata_path = op.join(subjects_dir, 'sample', 'mri')
    mridata_path_new = op.join(tempdir, 'sample', 'mri')
    os.mkdir(op.join(tempdir, 'sample'))
    os.mkdir(mridata_path_new)
    if op.exists(op.join(mridata_path, 'T1')):
        shutil.copytree(op.join(mridata_path, 'T1'), op.join(mridata_path_new,
                        'T1'))
    if op.exists(op.join(mridata_path, 'T1.mgz')):
        shutil.copyfile(op.join(mridata_path, 'T1.mgz'),
                        op.join(mridata_path_new, 'T1.mgz'))

    with ArgvSetter(('-d', tempdir, '-s', 'sample', '-o'),
                    disable_stdout=False, disable_stderr=False):
        mne_watershed_bem.run()
```

calling the Python command

```
$ mne watershed_bem --subject sample --overwrite
```

MNE-Python - Creating the BEM model meshes

mne_flash_bem

the shell script

extracts the BEM surfaces (outer skull, inner skull, and outer skin) from multiecho FLASH MRI data with spin angles of 5 and 30 degrees

```
[...]
usage()
{
        echo "usage: $0 [options]"
        echo
        echo "    --noflash30       Only 5 deg flash angle is available"
        echo "    --noconvert       Assume that the images have already been converted"
        echo "    --unwarp option   Unwarp the images using grad_unwarp with this unwarping option"
        echo "    --help            List this info"
        echo "    --usage           List this info"
        echo
}
[...]
```

calling the shell script:

```
$ mne_flash_bem
```

MNE-Python - Creating the BEM model meshes

mne_flash_bem

from the shell script to the Python function *make_flash_bem*

extracts the BEM surfaces (outer skull, inner skull, and outer skin) from multiecho FLASH MRI data with spin angles of 5 and 30 degrees

1. **mri_convert** it creates an mgz file for each FLASH data file
2. **grad_unwarp** run the Freesurfer unwarp algorithm on mgz files
3. **mri_ms_fitparms** it creates parameter maps for the data
4. **mri_synthesize** it creates a synthetic 5-degree flip angle volume
5. **fsl_rigid_register** it creates a registered 5-degree flip angle volume to the T1 volume under mri
6. **mri_convert** it converts the registered volume to COR format under mri/flash5. If necessary, the T1 and brain volumes are also converted into the COR format.
7. **mri_make_bem_surfaces** it creates the BEM surface tessellations
8. **mne_convert_surface** it creates the FreeSurfer surface files in the same directory
9. Cleanup unwanted files

```
def make_flash_bem(subject, subjects_dir, no_flash30=False, no_convert=False,
                   unwarp=False, overwrite=False, show=False):
    """Create 3-Layers BEM model from Flash MRI images

    Parameters
    ----------
    subject : str
        Subject name.
```

MNE-Python - Creating the BEM model meshes

mne_flash_bem

wrapping the function to a Python script

```
from mne.bem import make_flash_bem

def run():
    from mne.commands.utils import get_optparser

    parser = get_optparser(__file__)

    parser.add_option("-s", "--subject", dest="subject",
                      help="Subject name", default=None)
    parser.add_option("-d", "--subjects-dir", dest="subjects_dir",
                      help="Subjects directory", default=None)
    parser.add_option("-3", "--noflash30", dest="noflash30",
                      action="store_true", default=False,
                      help=("Skip the 30-degree flip angle data"),)
    parser.add_option("-n", "--noconvert", dest="noconvert",
                      action="store_true", default=False,
                      help=("Assume that the Flash MRI images have already "
                          "been converted to mgz files"))
    parser.add_option("-u", "--unwarp", dest="unwarp",
                      action="store_true", default=False,
                      help=("Run grad_unwarp with -unwarp <type> option on "
                          "each of the converted data sets"))
    parser.add_option("-o", "--overwrite", dest="overwrite",
                      action="store_true", default=False,
                      help="Write over existing .surf files in bem folder")
    parser.add_option("-v", "--view", dest="show", action="store_true",
                      help="Show BEM model in 3D for visual inspection",
                      default=False)

    options, args = parser.parse_args()

    subject = options.subject
    subjects_dir = options.subjects_dir
    noflash30 = options.noflash30
    noconvert = options.noconvert
    unwarp = options.unwarp
    overwrite = options.overwrite
    show = options.show

    if options.subject is None:
        parser.print_help()
        raise RuntimeError('The subject argument must be set')

    make_flash_bem(subject=subject, subjects_dir=subjects_dir,
                   no_flash30=noflash30, no_convert=noconvert, unwarp=unwarp,
                   overwrite=overwrite, show=show)

is_main = (__name__ == '__main__')
if is_main:
    run()
```

MNE-Python - Creating the BEM model meshes

mne_flash_bem

testing the Python command

```
@slow_test
@requires_mne
@requires_freesurfer
@sample.requires_sample_data
def test_flash_bem():
    """Test mne flash_bem"""
    check_usage(mne_flash_bem, force_help=True)
    # Using the sample dataset
    subjects_dir = op.join(sample.data_path(download=False),
'subjects')
    # Copy necessary files to tempdir
    tempdir = _TempDir()
    mridata_path = op.join(subjects_dir, 'sample', 'mri')
    mridata_path_new = op.join(tempdir, 'sample', 'mri')
    os.makedirs(op.join(mridata_path_new, 'flash'))
    os.makedirs(op.join(tempdir, 'sample', 'bem'))
    shutil.copyfile(op.join(mridata_path, 'T1.mgz'),
                    op.join(mridata_path_new, 'T1.mgz'))
    shutil.copyfile(op.join(mridata_path, 'brain.mgz'),
                    op.join(mridata_path_new, 'brain.mgz'))
    # Copy the available mri/flash/mef*.mgz files from the dataset
    files = glob.glob(op.join(mridata_path, 'flash', 'mef*.mgz'))
    for infile in files:
        shutil.copyfile(infile, op.join(mridata_path_new, 'flash',
                                        op.basename(infile)))
    # Test mne flash_bem with --noconvert option
    # (since there are no DICOM Flash images in dataset)
    currdir = os.getcwd()
    with ArgvSetter(('-d', tempdir, '-s', 'sample', '-n'),
                    disable_stdout=False, disable_stderr=False):
        mne_flash_bem.run()
    os.chdir(currdir)
```

calling the Python command

```
$ mne flash_bem --subject sample
```

Thank you for your attention