Introduction to MEG and EEG analysis with MNE

Denis A. Engemann
denis.engemann@gmail.com
github: dengemann
twitter: dngman

Bar Ilan - October 2015

Disclaimer ...

7 (Shocking?) facts about MNE

- cortically constrained *M*inmum *N*orm *E*stimates
- MNE based on C code developed for more than a decade by Matti Hamalainen
- MNE-Python started in Dec. 2010 at MGH, Boston
- MNE-Python is not a wrapper around Mattis code
- … is not a toollbox for MNE inverse solvers
- … is used by Kaggle competition winners
- … is used by EEG people —> *M*EG *'n' E*EG

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

In a Nutshell, MNE-Python...

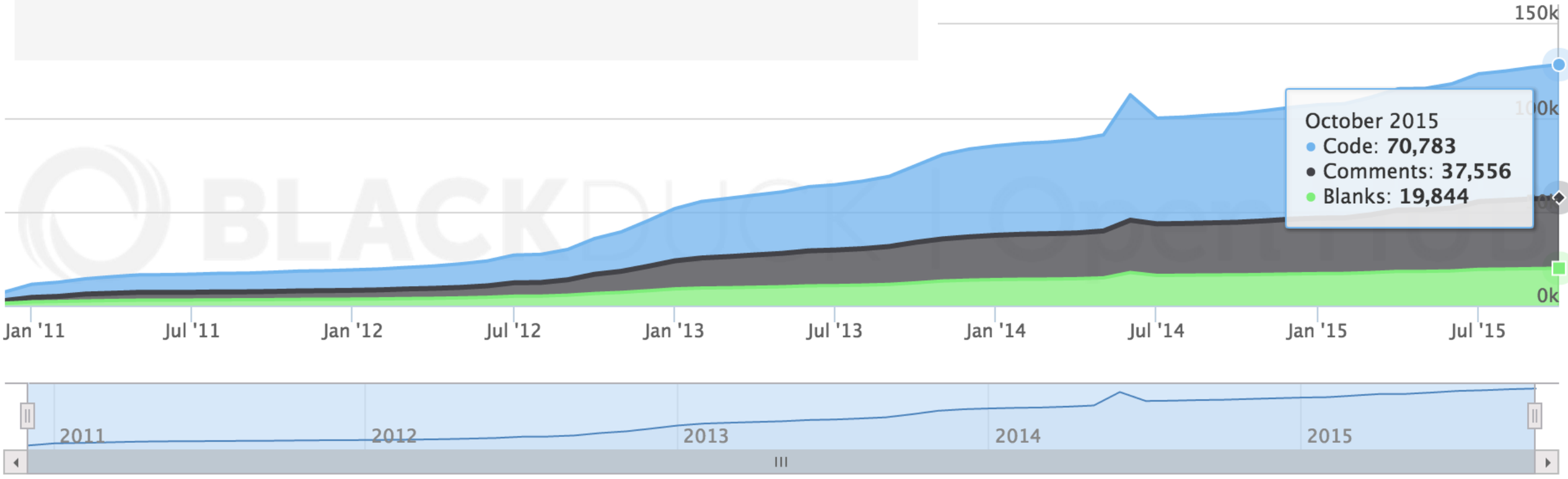
... has had 9,573 commits made by 98 contributors
representing 70,783 lines of code

... 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 9 days ago

MNE
MEG + EEG ANALYSIS & VISUALIZATION



main features

- I/O for many data formats
- preprocessing
- basic and advanced MEG and EEG analyses in sensor and sourse space
- multivariate and classical statistics
- visualization
- Python
- open development

Open dev process

https://github.com/mne-tools/mne-python

This repository | Search or type a command | Explore | Gist | Blog | Help | agramfort

mne-tools / mne-python

Unwatch 22 | Unstar 52 | Fork 53

Browse Issues | Milestones | New Issue

Everyone's Issues 30

Assigned to you 0

Created by you 7

Mentioning you 11

No milestone selected

Labels

ENH 11

BUG 0

DOC 0

FIX 0

30 Open | 905 Closed | Sort: Newest

Close | Label | Assignee | Milestone

ENH/FIX: default EEG layout when no headshape ore coil positions are present #926
Opened by dengemann 2 days ago 8 comments

Segmentation fault on unit test #922
Opened by lulopolar 2 days ago 13 comments

BUG: coverage is not working with Travis .... #919
Opened by dengemann 4 days ago

ENH: Improve test coverage #909
Opened by Eric89GXL 7 days ago 11 comments

DOC: setting up Python #900
Opened by christianmbrodbeck 11 days ago 5 comments

Development of the MNE software

http://martinos.org/mne/stable/contributing.html

People

@agramfort

@Eric89GXL

@dengemann

@mainakjas

@teonlamont

@joewalter

@christianmbrodbeck

@leggitta

@wmvanvliet

@haribharadwaj

@lauriparkkonen

@mshamalainen

… and many more …

@you?

Code is peer-reviewed

.travis.yml — outdated diff ✖

```
    ((13 lines not shown))
48  58
49  59   after_success:
50      -    - if [ ${PYTHON:0:1} == "2" ]; then coveralls; fi
        60   +    # Need to run from source dir to exectue "git" commands
        61   +    - if [ "${PYTHON}" == "2.7" ] && [ "${DEPS}" == "full" ]; then
```

5

dengemann repo collab — 2 days ago

can we add a comment here?

there's no reason not to run coverage on python 3.3 other than measuring coverage on the full test suite.

Eric89GXL repo collab — 2 days ago

I did this because I didn't think it made sense to send three coverage reports using coveralls for each commit.

dengemann repo collab — 2 days ago

yes, obviously, and I agree. I was suggesting to leave a message for future readers of this file.

Eric89GXL repo collab — 2 days ago

If that logic makes sense, I'll add a comment.

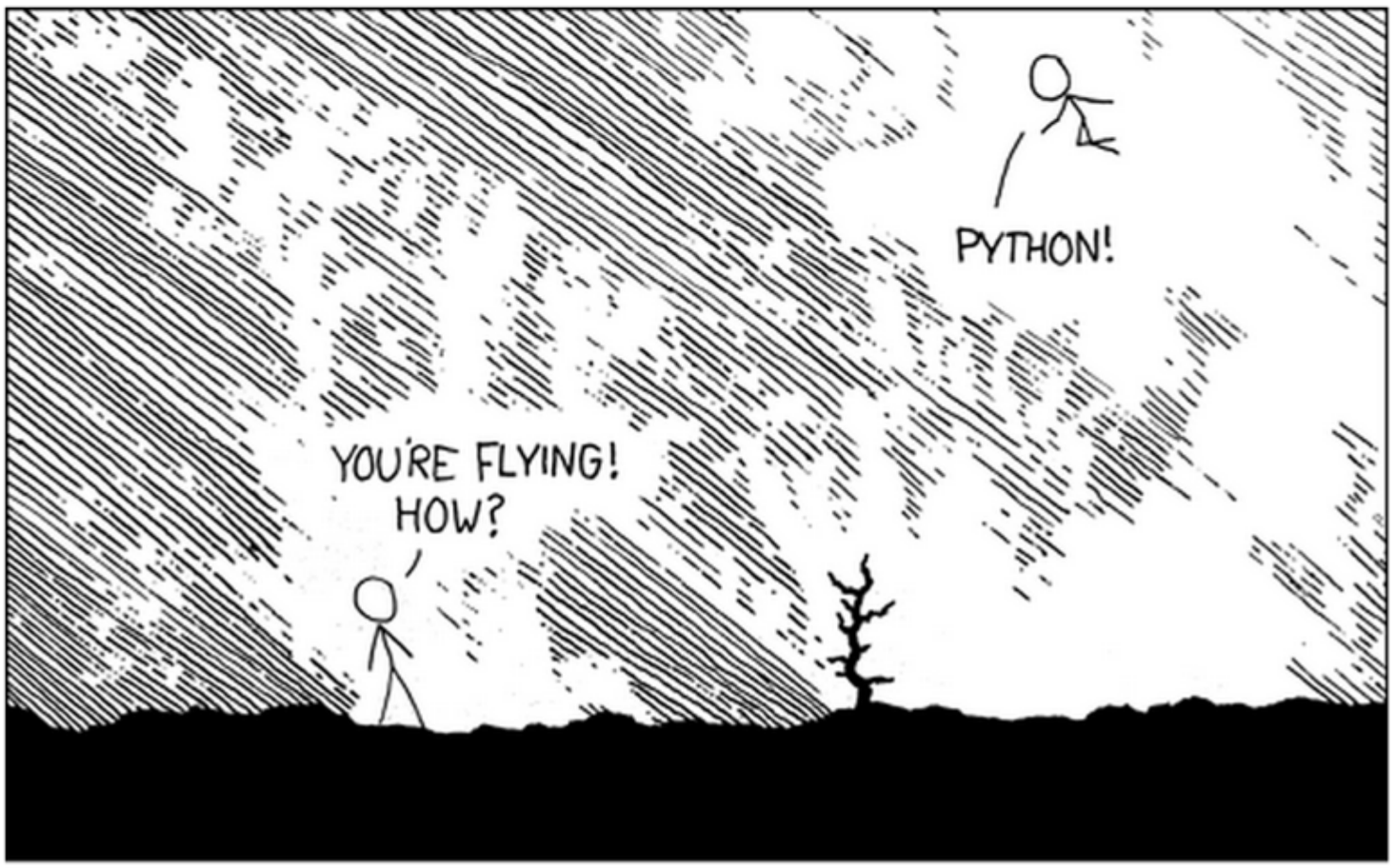
xkcd

A webcomic of romance, sarcasm, math, and language.

xkcd updates every Monday, Wednesday, and Friday.

Python

|< < Prev Random Next > >|

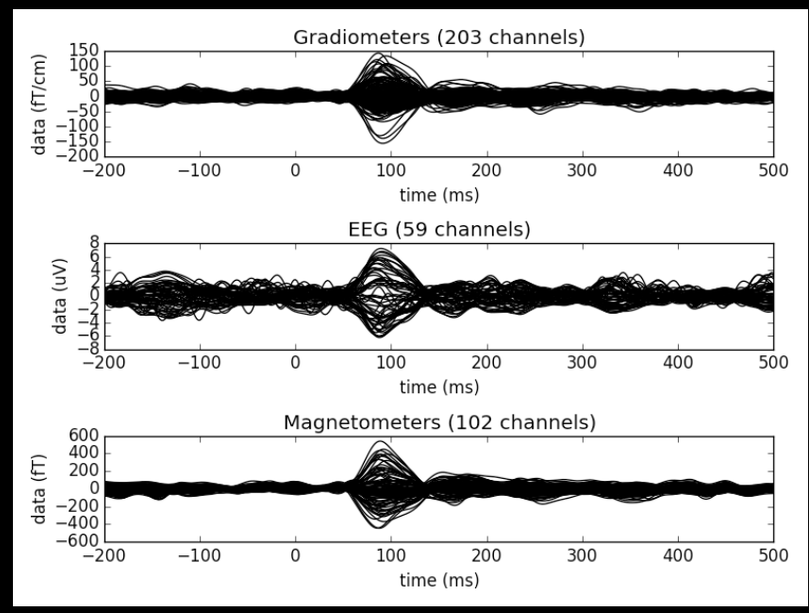


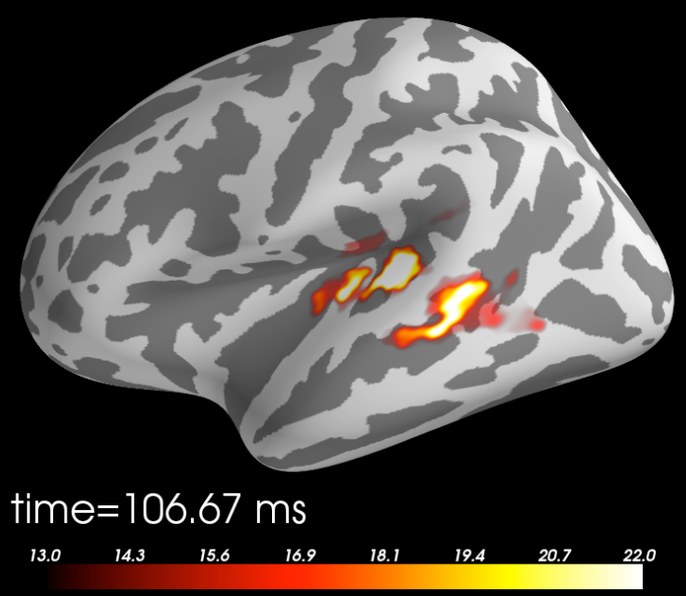
- is a **general purpose** **programming language**
- it's is **free** (as in beer and as in speech)
- increasingly popular in science and "big data"

- clean syntax, easy to read, easy to learn
- fast prototyping
- well suited for high performance numerical computing
- ... for visualization, text processing, web-programming, **...**

From raw data to dSPM solutions in ~30 lines

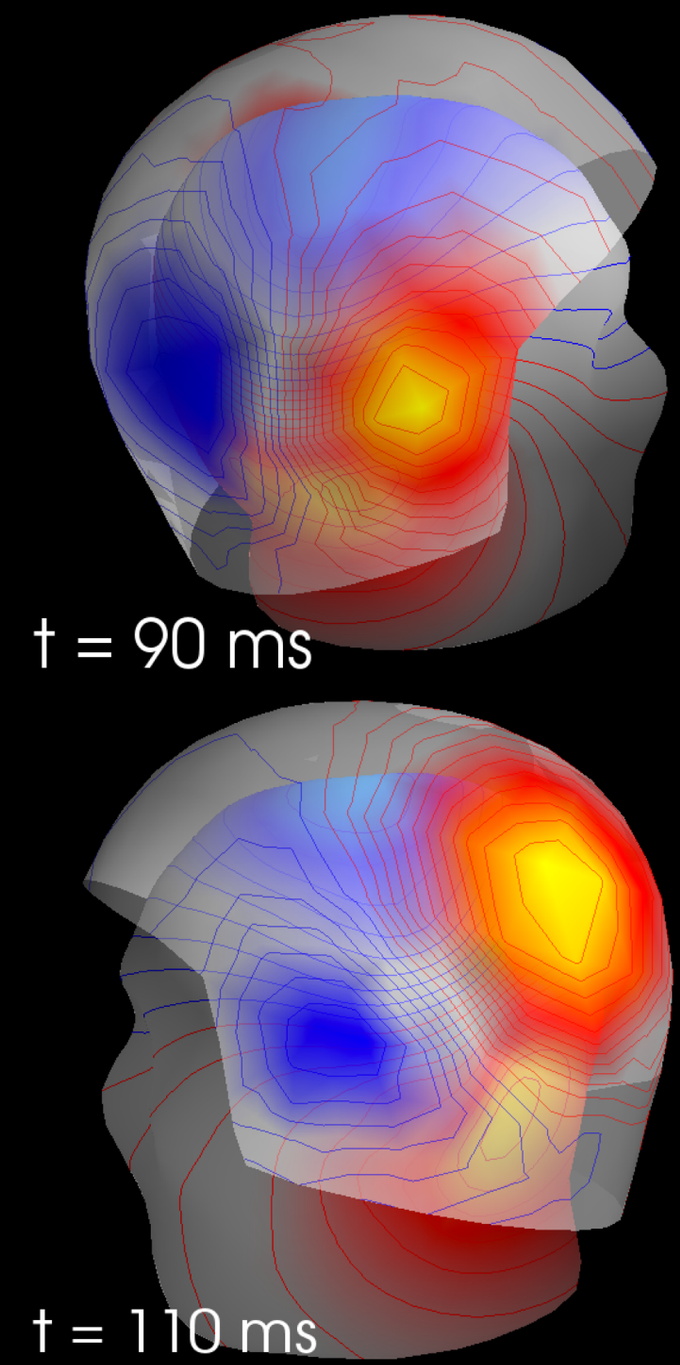
```
import mne

# load data
fname = 'raw.fif'
raw = mne.fiff.Raw(fname)
raw.info['bads'] = ['MEG 2443', 'EEG 053']  # mark bad channels

# band-pass filter data in beta band, and save it
raw.filter(13.0, 30.0, filter_length=4096, n_jobs='cuda')
raw.save(fname[:-4] + '_beta.fif')

# extract epochs
picks = mne.fiff.pick_types(raw.info, meg=True, eeg=True, eog=True)
events = mne.find_events(raw)
epochs = mne.Epochs(raw, events, event_id=1, tmin=-0.2, tmax=0.5, proj=True,
                    picks=picks, baseline=(None, 0), preload=True,
                    reject=dict(grad=4000e-13, mag=4e-12, eog=150e-6))

# compute evoked response and noise covariance, and plot evoked
evoked = epochs.average()
cov = mne.compute_covariance(epochs, tmax=0)
evoked.plot()

# compute inverse operator
fwd_fname = 'sample_audvis-meg-eeg-oct-6-fwd.fif'
fwd = mne.read_forward_solution(fwd_fname, surf_ori=True)
inv = mne.minimum_norm.make_inverse_operator(raw.info, fwd, cov, loose=0.2)

# compute inverse solution
stc = mne.minimum_norm.apply_inverse(evoked, inv, lambda2=1 / 3.0 ** 2,
                                     method='dSPM')

# morph it to average brain for group study
stc_avg = mne.morph_data('sample', 'fsaverage', stc, 5, smooth=5)
stc_avg.plot()
```





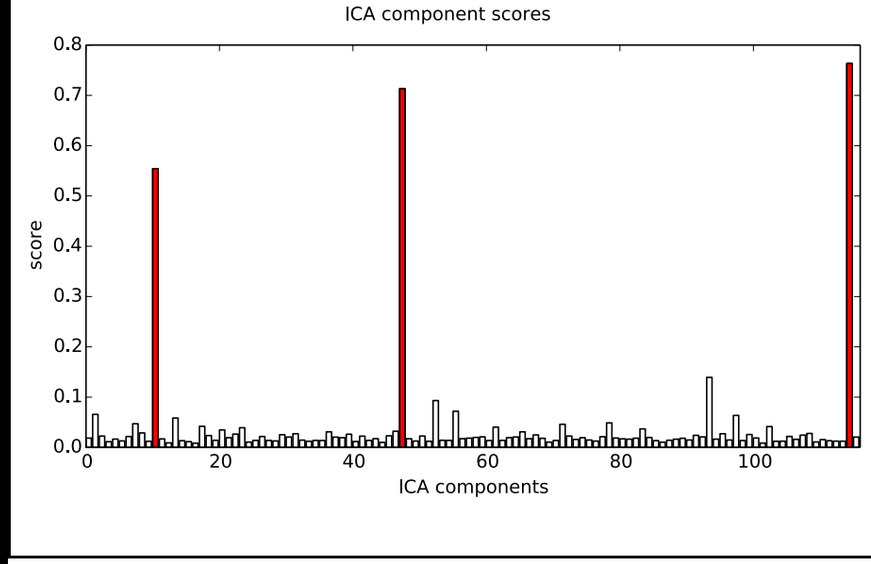
```
"""
=====================
Plot M/EEG field lines
=====================

In this example, M/EEG data are remapped onto the
MEG helmet (MEG) and subject's head surface (EEG).
This process can be computationally intensive.
"""

# Authors: Eric Larson <larson.eric.d@gmail.com>
#          Denis A. Engemann <d.engemann@fz-juelich.de>
#          Alexandre Gramfort <alexandre.gramfort@telecom-paristech.fr>
# License: BSD (3-clause)
import mne

data_path = mne.datasets.sample.data_path()
subjects_dir = data_path + '/subjects'
evoked_fname = data_path + '/MEG/sample/sample_audvis-ave.fif'
trans_fname = data_path + '/MEG/sample/sample_audvis_raw-trans.fif'

# If trans_fname is set to None then only MEG estimates can be visualized
setno = 'Left Auditory'

evoked = mne.fiff.read_evoked(evoked_fname, setno=setno,
                              baseline=(-0.2, 0.0))

# Compute the field maps to project MEG and EEG data to MEG helmet
# and scalp surface
maps = mne.make_field_map(evoked, trans_fname=trans_fname,
                          subject='sample', subjects_dir=subjects_dir,
                          n_jobs=1)

# explore several points in time
[evoked.plot_field(maps, time=time) for time in [0.09, .11]]
```

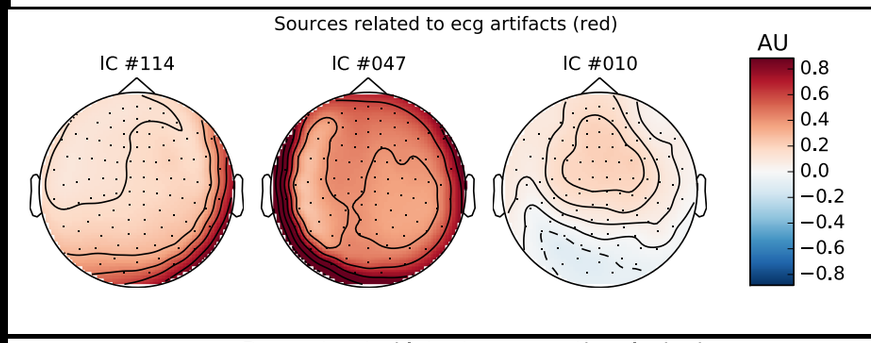
Project multiples sensor types on field map

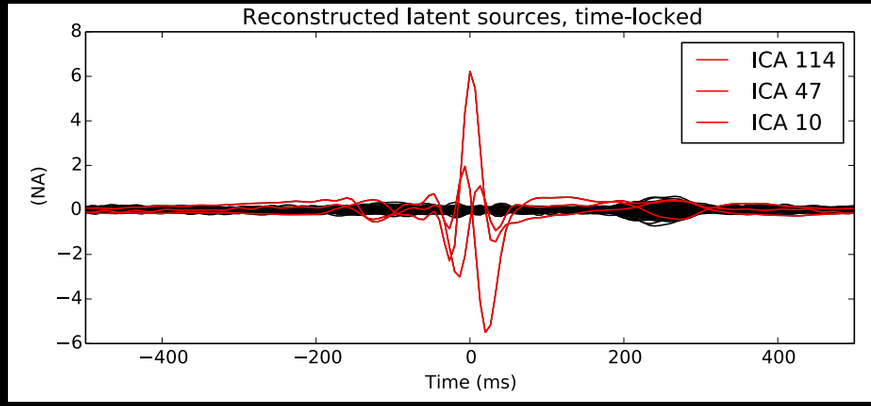


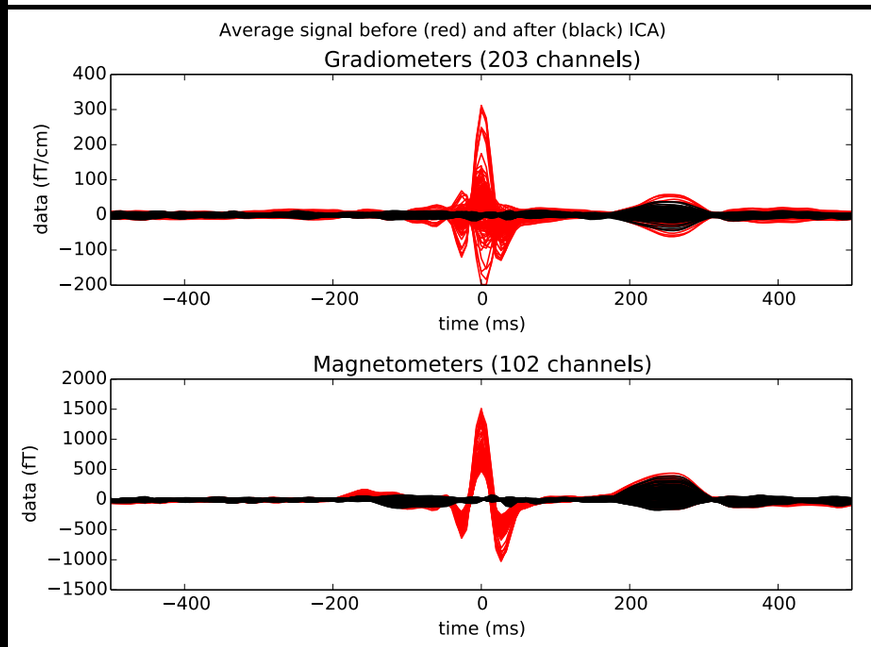
Remove artifacts using ICA

```
# Author: Denis Engemann <denis.engemann@gmail.com>
#
# License: BSD (3-clause)

raw = Raw(raw_fname, preload=True)
raw.filter(1, 30, method='iir')
picks = mne.pick_types(raw.info, meg=True, eeg=False, eog=True, ecg=True,
                       stim=False, exclude='bads')

events = mne.find_events(raw, stim_channel='STI 014')
event_id = dict(aud_l=1, aud_r=2, vis_l=3, vis_r=4)
reject = dict(eog=250e-6)
tmin, tmax = -0.5, 0.5

# 1) Fit ICA model using the FastICA algorithm
ica = ICA(n_components=0.95, method='fastica').fit(epochs)

# 2) Find ECG Artifacts
# generate ECG epochs to improve detection by correlation
ecg_epochs = create_ecg_epochs(raw, tmin=-.5, tmax=.5, picks=picks)
ecg_inds, scores = ica.find_bads_ecg(ecg_epochs)
ica.plot_scores(scores, exclude=ecg_inds)

title = 'Sources related to %s artifacts (red)'
ica.plot_components(ecg_inds, title=title % 'ecg')

# by default we expect 3 reliable ECG components
ica.exclude += ecg_inds[:3]

# 3) Assess component selection and unmixing quality
ecg_evoked = ecg_epochs.average() # estimate average artifact
ica.plot_sources(ecg_evoked)  # plot ECG sources + selection
ica.plot_overlay(ecg_evoked)  # plot ECG cleaning
```



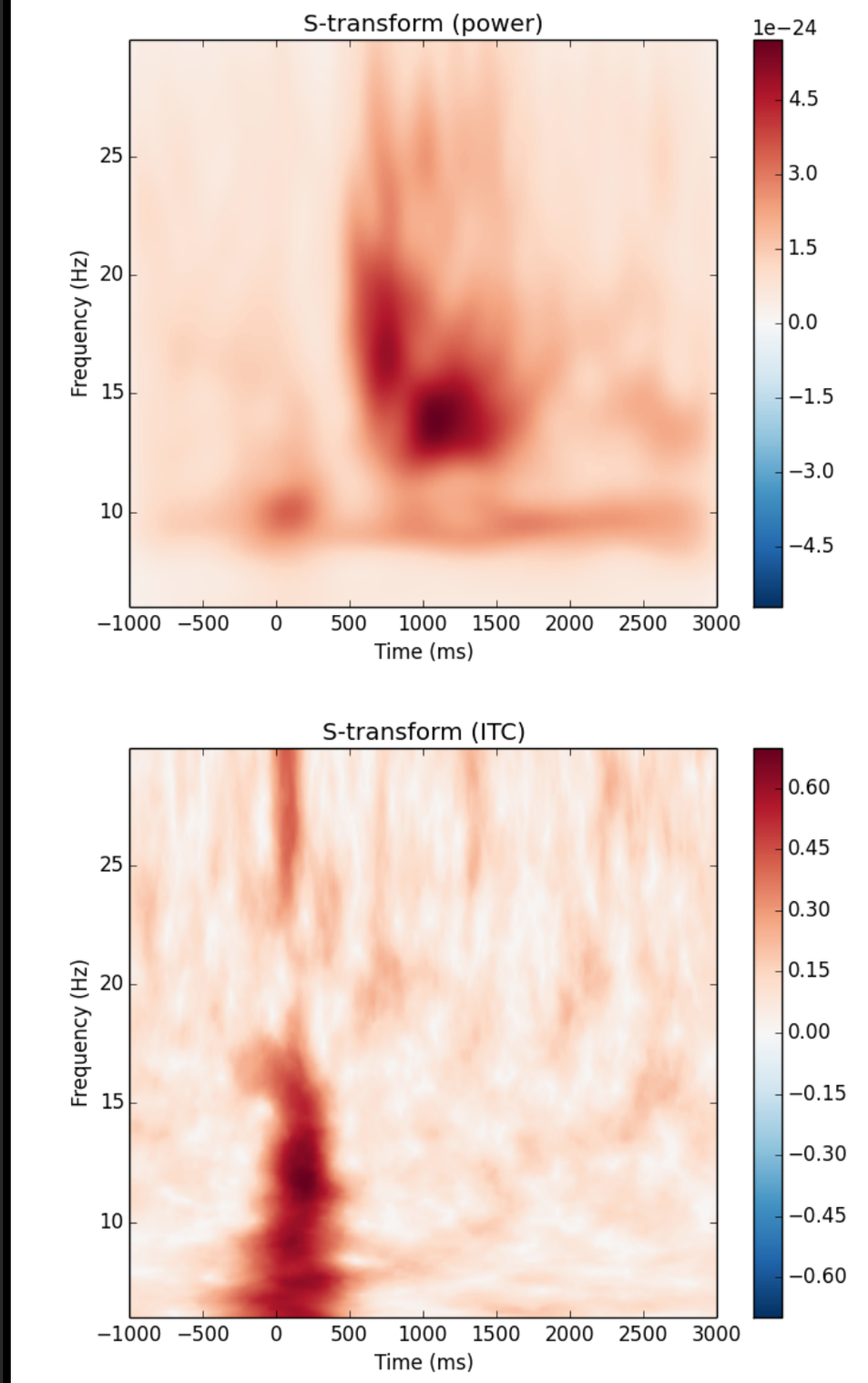




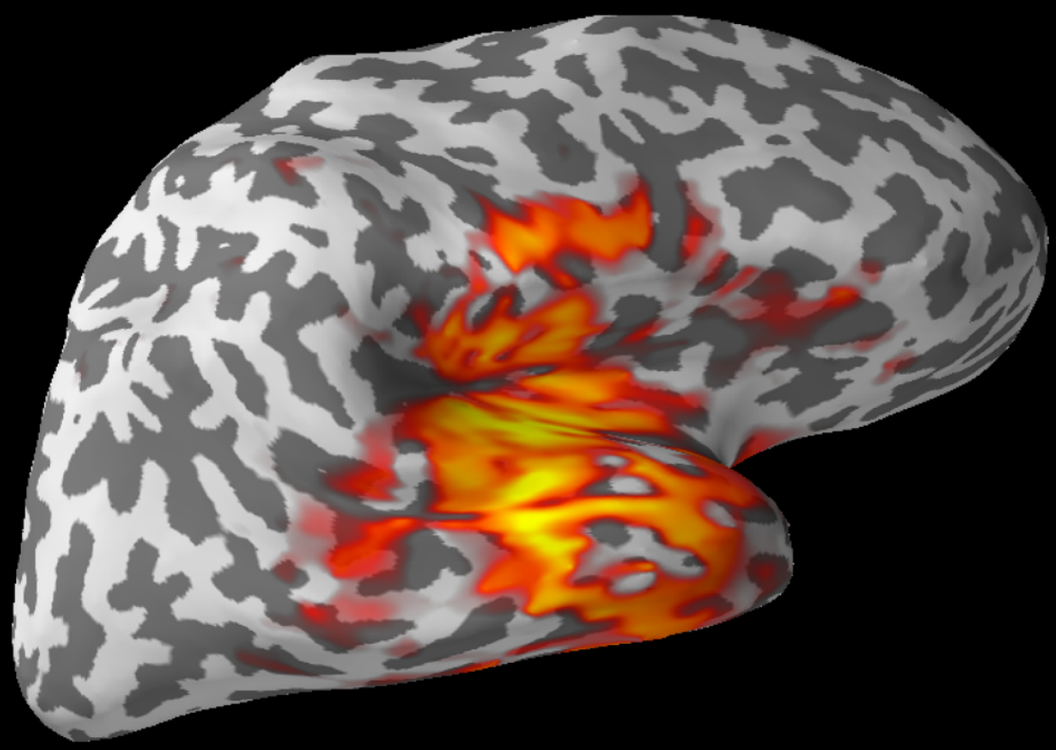


```
"""
========================================================
Time frequency with Stockwell transform in sensor space
========================================================

This script shows how to compute induced power and intertrial coherence
using the Stockwell transform, a.k.a. S-Transform.

"""
# Authors: Denis A. Engemann <denis.engemann@gmail.com>
#          Alexandre Gramfort <alexandre.gramfort@telecom-paristech.fr>
#
# License: BSD (3-clause)

import mne
from mne import io
from mne.time_frequency import tfr_stockwell
from mne.datasets import somato

###############################################################################
# Set parameters
data_path = somato.data_path()
raw_fname = data_path + '/MEG/somato/sef_raw_sss.fif'
event_id, tmin, tmax = 1, -1., 3.

# Setup for reading the raw data
raw = io.Raw(raw_fname)
baseline = (None, 0)
events = mne.find_events(raw, stim_channel='STI 014')

# picks MEG gradiometers
picks = mne.pick_types(raw.info, meg='grad', eeg=False, eog=True, stim=False)

epochs = mne.Epochs(raw, events, event_id, tmin, tmax, picks=picks,
                    baseline=baseline, reject=dict(grad=4000e-13, eog=350e-6),
                    preload=True)

###############################################################################
# Calculate power and intertrial coherence

epochs = epochs.pick_channels([epochs.ch_names[82]])  # reduce computation

power, itc = tfr_stockwell(epochs, fmin=6., fmax=30., decim=4, n_jobs=2,
                           width=.3, return_itc=True)

power.plot([0], baseline=(-0.5, 0), mode=None, title='S-transform (power)')

itc.plot([0], baseline=None, mode=None, title='S-transform (ITC)')
```

The Stockwell transform

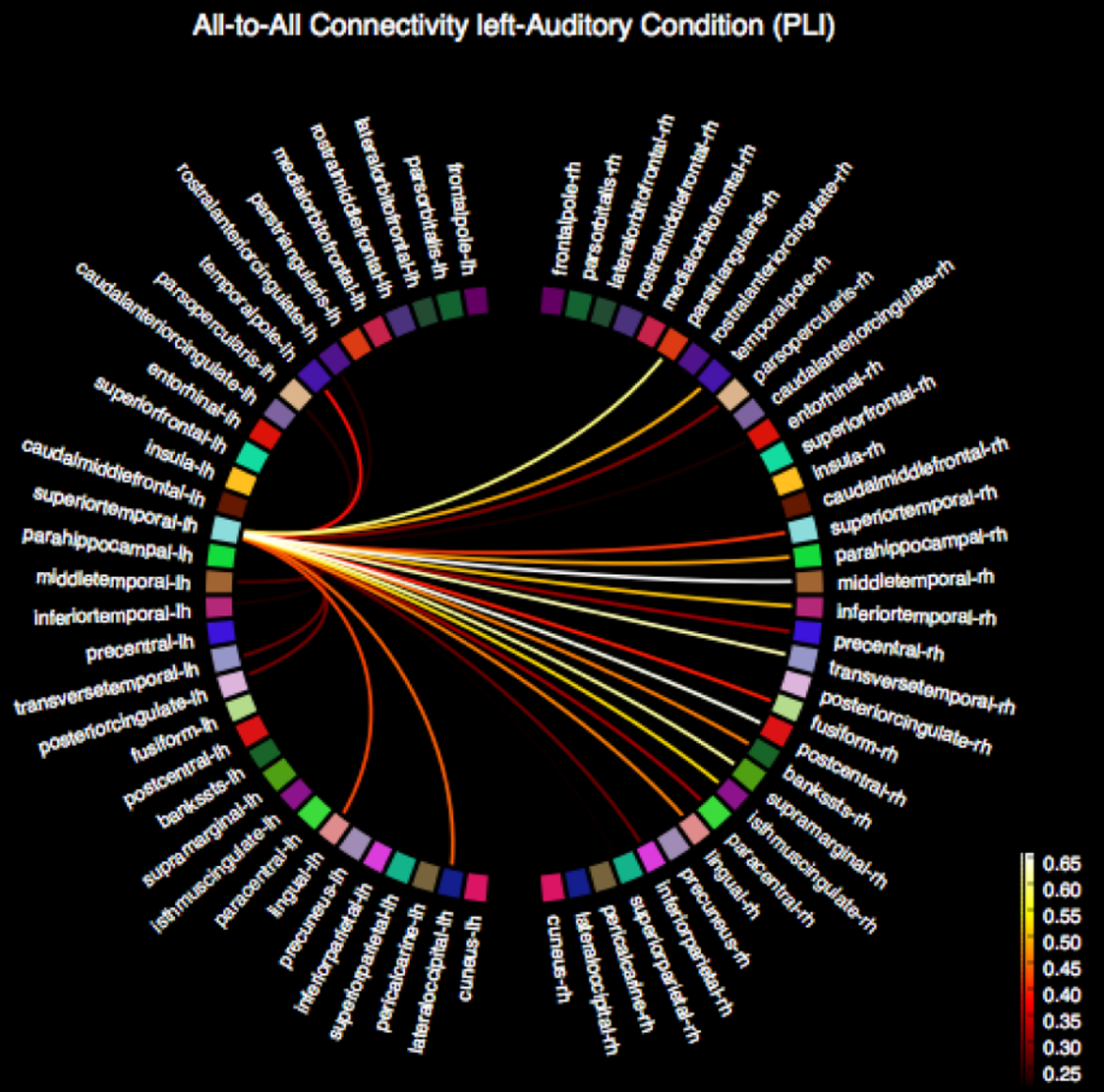


Functional Connectivity

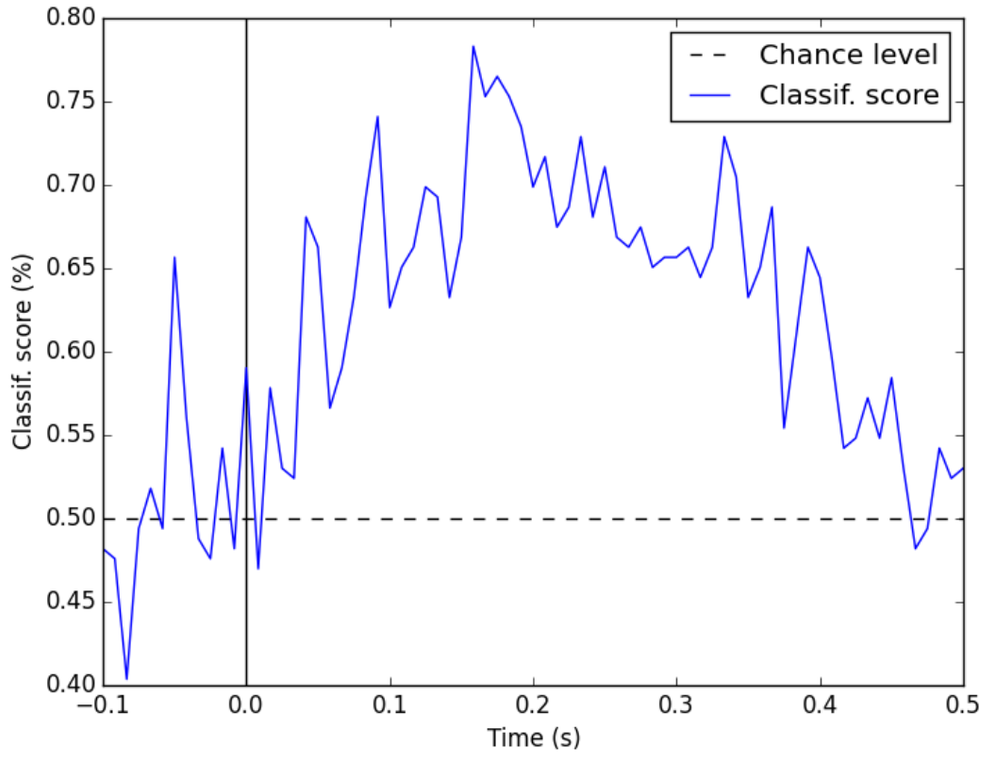


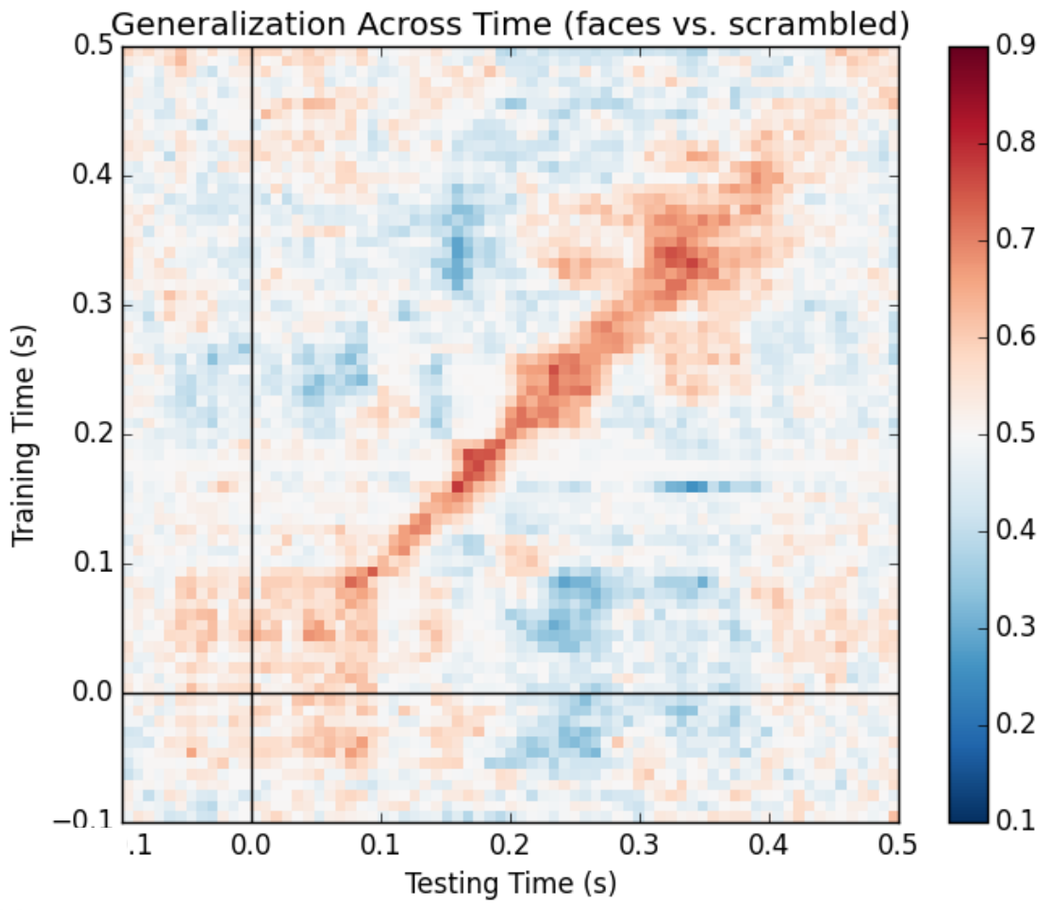
Coherence 10.6 Hz

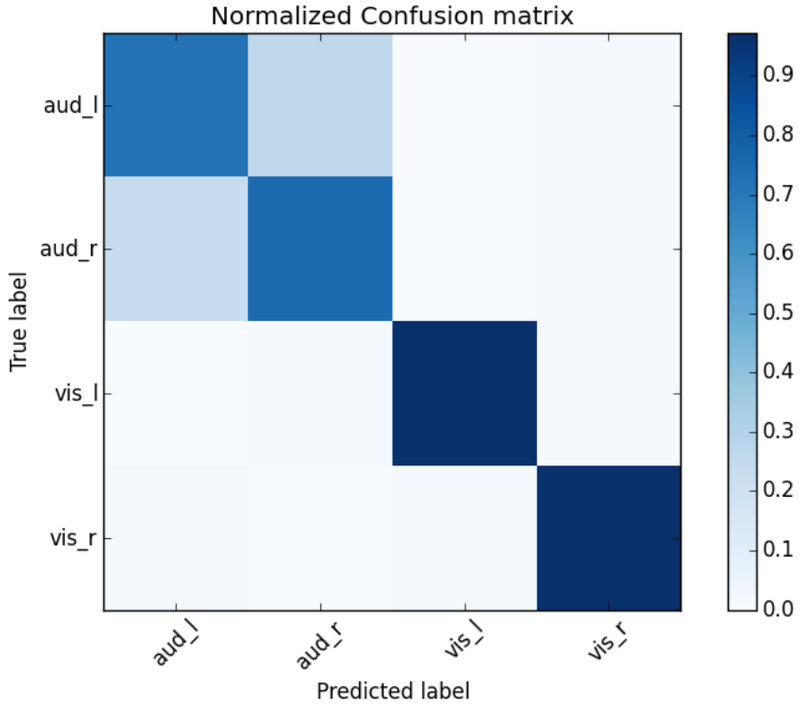
0.250 0.307 0.364 0.421 0.479 0.536 0.593 0.650



Decoding


Chance level
Classif. score
Classif. score (%)
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
Time (s)

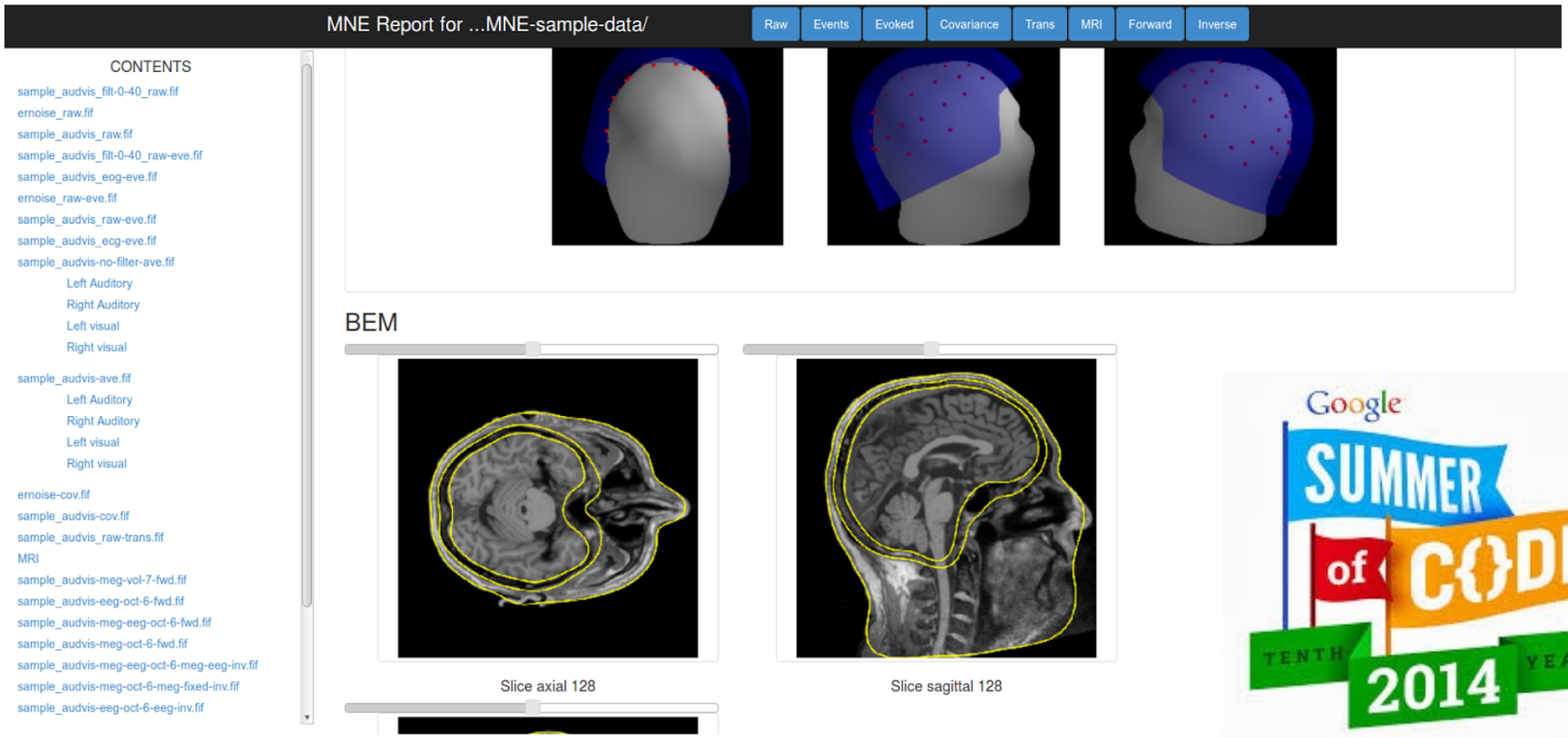

Generalization Across Time (faces vs. scrambled)
Training Time (s)
Testing Time (s)
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1


Normalized Confusion matrix
True label
Predicted label
aud_l
aud_r
vis_l
vis_r
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0


scikit
learn

the report tool

(MNE-Python 0.8)



Mainak Jas, Alex Gramfort and Denis Engemann

http://ml-py-meg.blogspot.fr/search/label/gsoc2014

```python
from mne.report import Report
report = Report(verbose=True)
```

initialize report

```python
from mne.datasets import sample
from mne import read_evokeds
fname = path + '/MEG/sample/sample_audvis-ave.fif'

path = sample.data_path()
report = Report(verbose=True)

# report.parse_folder(data_path=path, pattern='*-eve.fif')

evoked = read_evokeds(fname,
                      condition='Left Auditory',
                      baseline=(None, 0),
                      verbose=True)

fig = evoked.plot()

report.add_figs_to_section(
    fig, captions='Left Auditory', section='evoked')

report.save('report.html', overwrite=True)
```

```python
from mne.report import Report
report = Report(verbose=True)
```

do stuff

```python
from mne.datasets import sample
from mne import read_evokeds
fname = path + '/MEG/sample/sample_audvis-ave.fif'

path = sample.data_path()
report = Report(verbose=True)

# report.parse_folder(data_path=path, pattern='*-eve.fif')

evoked = read_evokeds(fname,
                      condition='Left Auditory',
                      baseline=(None, 0),
                      verbose=True)

fig = evoked.plot()

report.add_figs_to_section(
    fig, captions='Left Auditory', section='evoked')

report.save('report.html', overwrite=True)
```

```python
from mne.report import Report
report = Report(verbose=True)

from mne.datasets import sample
from mne import read_evokeds
fname = path + '/MEG/sample/sample_audvis-ave.fif'

path = sample.data_path()
report = Report(verbose=True)

# report.parse_folder(data_path=path, pattern='*-eve.fif')

evoked = read_evokeds(fname,
                      condition='Left Auditory',
                      baseline=(None, 0),
                      verbose=True)

fig = evoked.plot()

report.add_figs_to_section(
    fig, captions='Left Auditory', section='evoked')

report.save('report.html', overwrite=True)
```

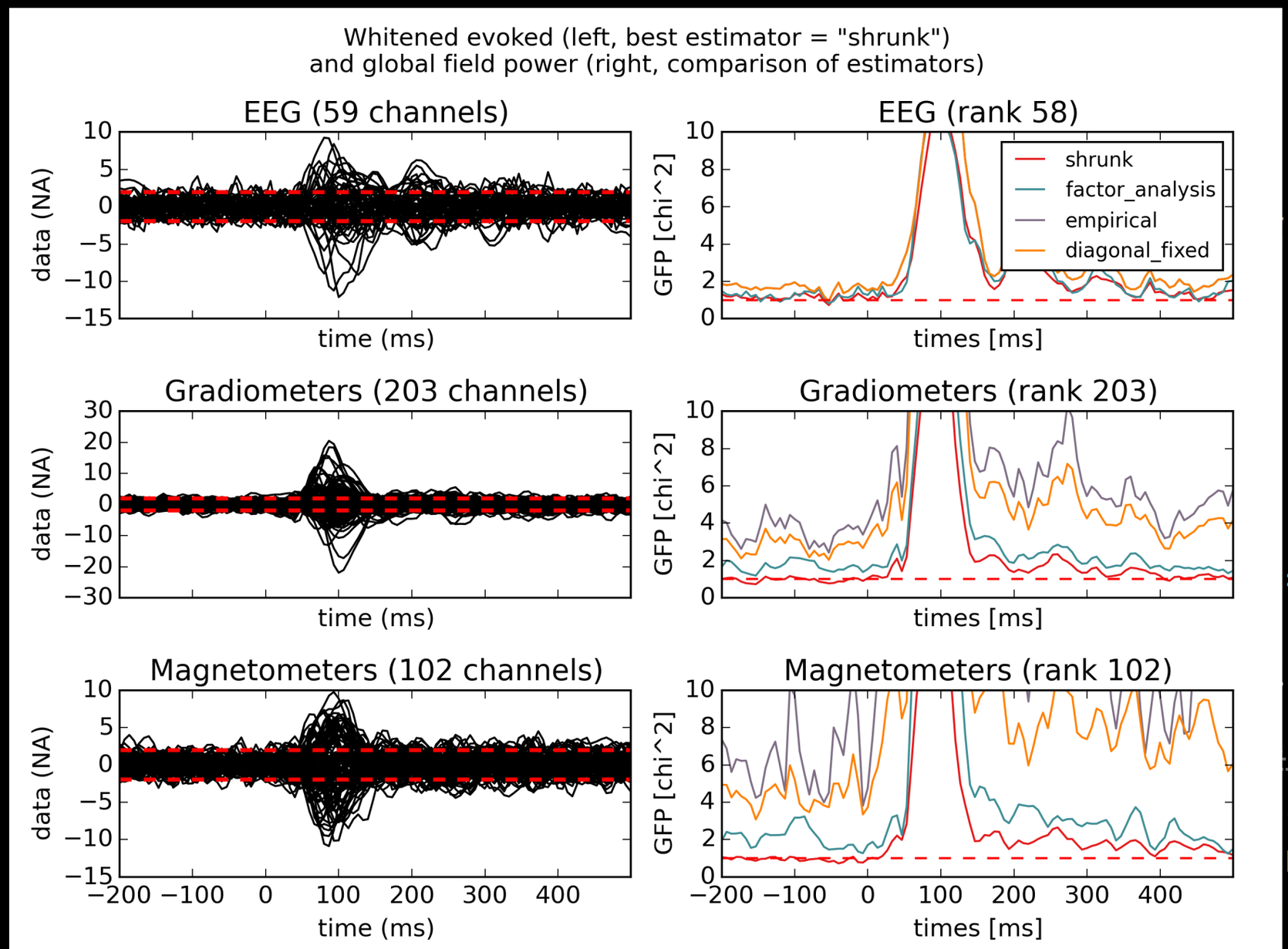
make a figure

```python
from mne.report import Report
report = Report(verbose=True)

from mne.datasets import sample
from mne import read_evokeds
fname = path + '/MEG/sample/sample_audvis-ave.fif'

path = sample.data_path()
report = Report(verbose=True)

# report.parse_folder(data_path=path, pattern='*-eve.fif')

evoked = read_evokeds(fname,
                      condition='Left Auditory',
                      baseline=(None, 0),
                      verbose=True)

fig = evoked.plot()

report.add_figs_to_section(
    fig, captions='Left Auditory', section='evoked')

report.save('report.html', overwrite=True)
```

make report

```
# Authors: Denis A. Engemann <denis.engemann@gmail.com>
#          Alexandre Gramfort <alexandre.gramfort@telecom-paristech.fr>
#
# License: BSD (3-clause)

import mne

data_path = mne.datasets.sample.data_path()
raw_fname = data_path + '/MEG/sample/sample_audvis_filt-0-40_raw.fif'
event_fname = data_path + '/MEG/sample/sample_audvis_filt-0-40_raw-eve.fif'

raw = mne.io.Raw(raw_fname, preload=True)
raw.info['bads'] += ['MEG 2443']
raw.filter(1, 30)

epochs = mne.Epochs(
    raw, events=mne.read_events(event_fname), event_id=1, tmin=-.2, tmax=0.5,
    picks=mne.pick_types(raw.info, meg=True, eeg=True, exclude='bads'),
    baseline=None, reject=dict(mag=4e-12, grad=4000e-13, eeg=80e-6))

###############################################################################
# Compute covariance using automated regularization and show whitening
noise_covs = mne.cov.compute_covariance(epochs[:20], tmax=0, method='auto',
                                        return_estimators=True)

evoked = epochs.average()
evoked.plot()  # plot evoked response
evoked.plot_white(noise_covs)  # compare estimators
```

```
# Authors: Denis A. Engemann <denis.engemann@gmail.com>
#          Alexandre Gramfort <alexandre.gramfort@telecom-paristech.fr>
#
# License: BSD (3-clause)
```

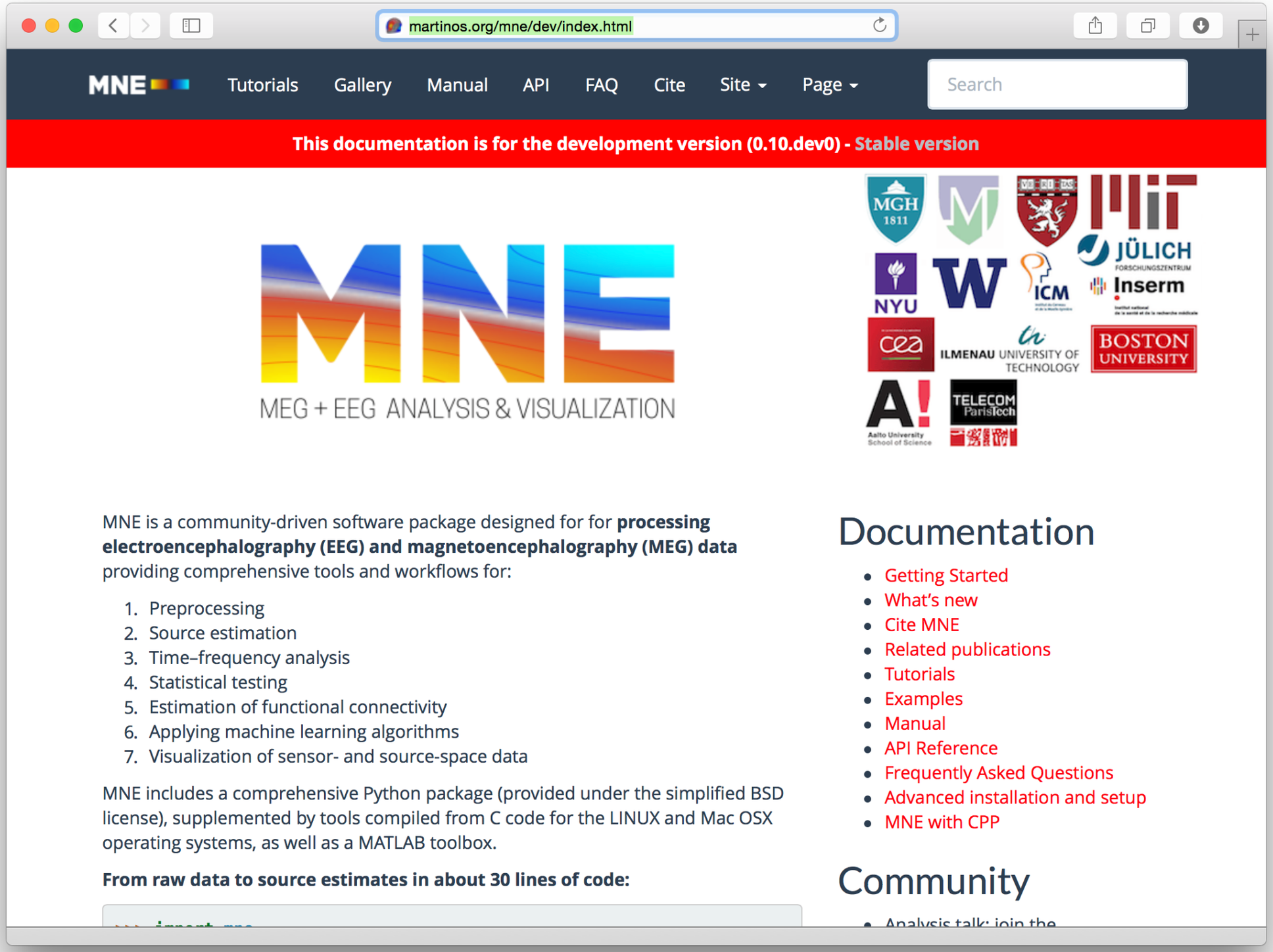



```
t-0-40_raw.fif'
ilt-0-40_raw-eve.fif'


=1, tmin=-.2, tmax=0.5,
 exclude='bads'),
3, eeg=80e-6))

##############################
d show whitening
max=0, method='auto',
ors=True)
```

```
evoked = epochs.average()
evoked.plot()  # plot evoked response
evoked.plot_white(noise_covs)  # compare estimators
```


KEEP
CALM
MNE-PYTHON
IS WITH
YOU

MNE Website



MNE software for processing MEG and EEG data, A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, M. Hämäläinen, Neuroimage 2014

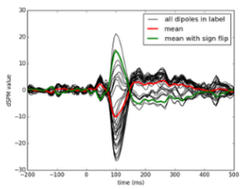
Getting inspired

martinos.org/mne/dev/auto_examples/index.html

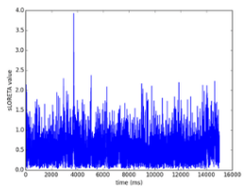
MNE Tutorials Gallery Manual API FAQ Cite Site Page Search

Inverse problem and source analysis

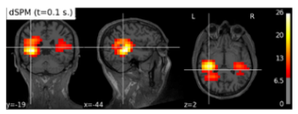
Estimate source activations, extract activations in labels, morph data between subjects etc.



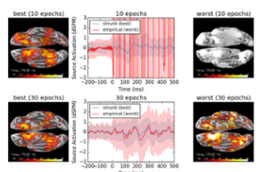
Compute MNE-dSPM inverse solution on single epochs



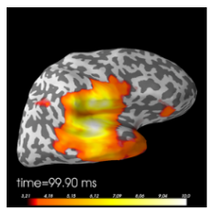
Compute sLORETA inverse solution on raw data



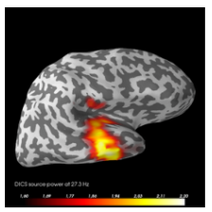
Compute MNE-dSPM inverse solution on evoked data in volume source space



Demonstrate impact of whitening on source estimates

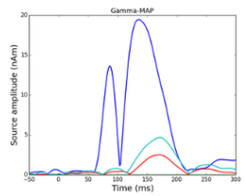


Compute DICS beamfomer on evoked data

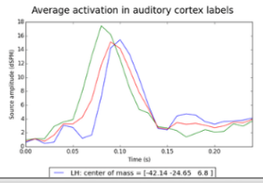


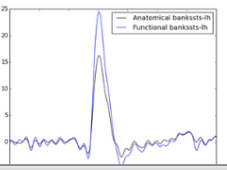
Compute source power using DICS beamfomer

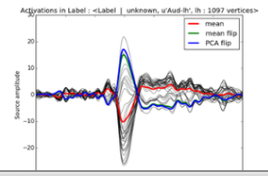
Do a dipole fit

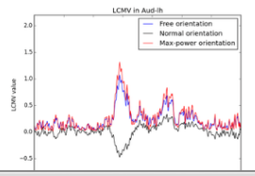


Compute a sparse inverse solution using the Gamma-Map empirical Bayesian





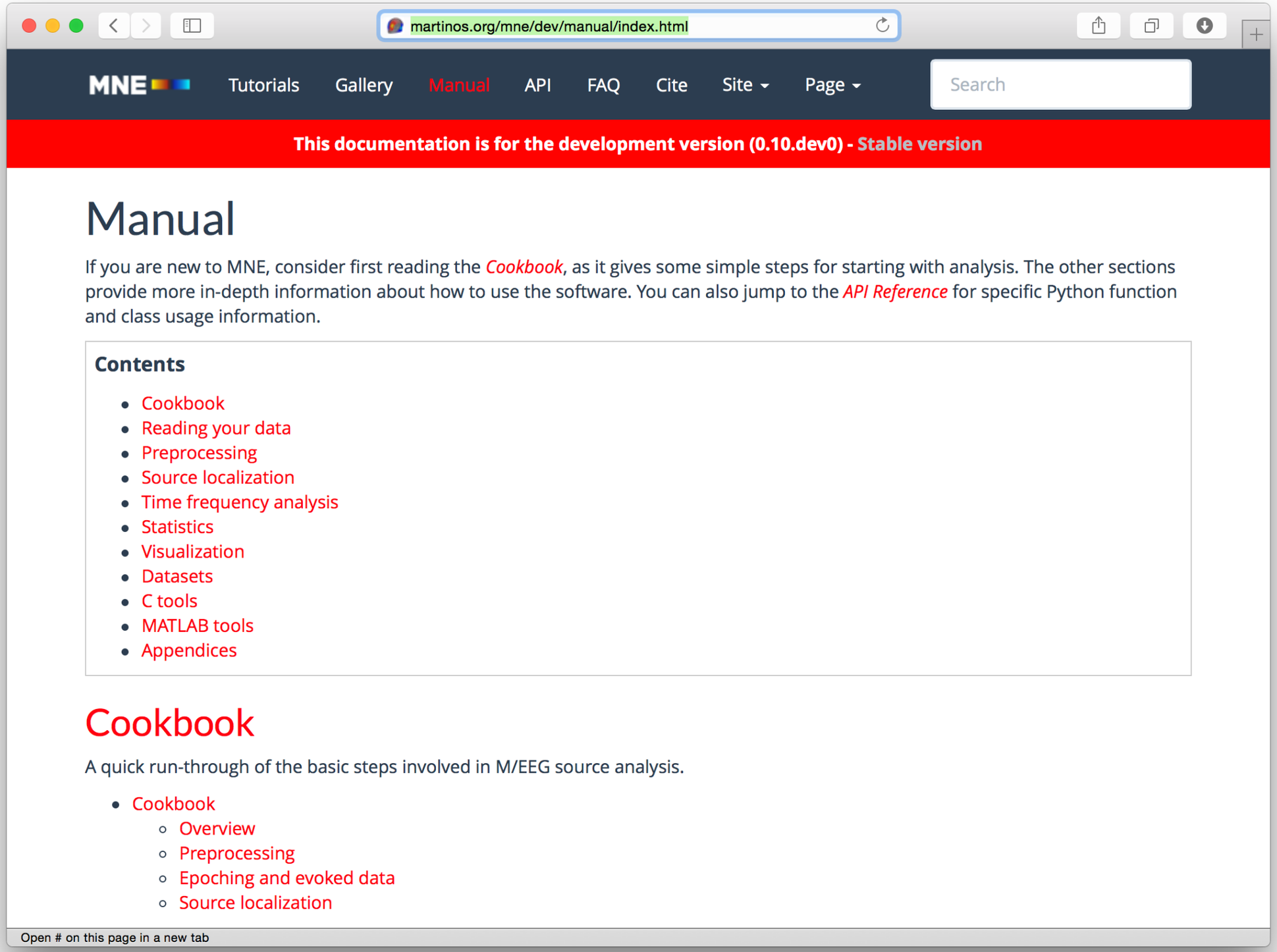




Open # on this page in a new tab

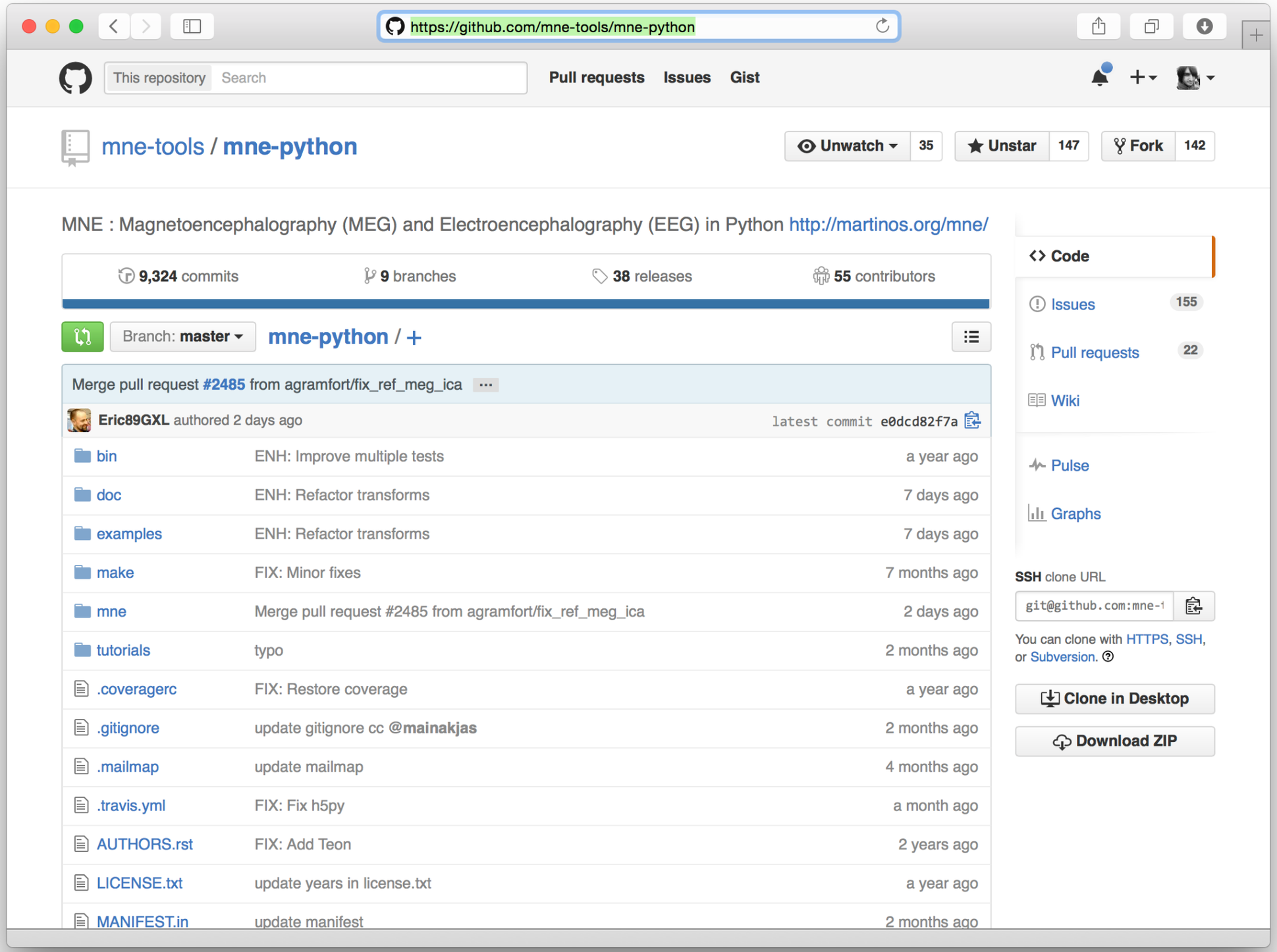
MNE software for processing MEG and EEG data, A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, M. Hämäläinen, Neuroimage 2014

How was it again?



MNE software for processing MEG and EEG data, A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, M. Hämäläinen, Neuroimage 2014

Contribute



MNE software for processing MEG and EEG data, A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, M. Hämäläinen, Neuroimage 2014