MEG Pipeline

Release 0.1

Hadi Zaatiti hadi.zaatiti@nyu.edu

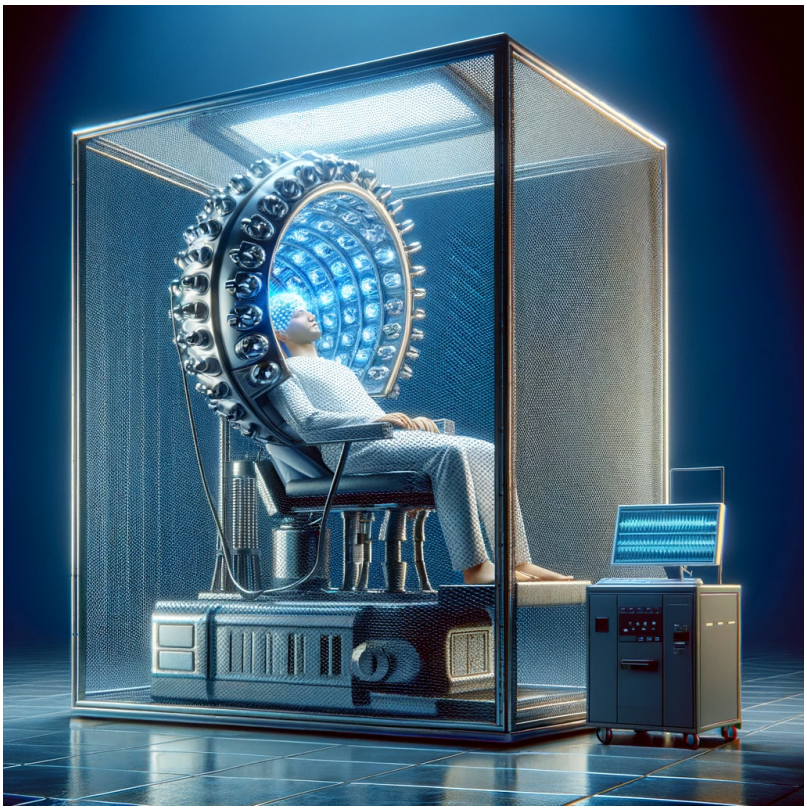
May 15, 2024

SYSTEMS OVERVIEW

Status of the documentation build

MEG/EEG-pipeline provides documentation for the Magnetoencephalography (MEG) and ElectroEncephaloGraphy (EEG) systems in the MEG laboratory and EEG setup within Brain Imaging Core Technology Platform. It offers a *simple* and *intuitive* overview on how MEG machines work, the specification of the system, what kind of data are generated and how to process them using *ready-to-use* pipelines. This documentation additionally provides a guide to build your own MEG-system experiment and what is required to successfully execute the experiment.

If you like to get a .PDF document of this website, click here Download PDF



Check out the *MEG System description: KIT Machine* section for further information, including how to *Data preparation* the project.

Note: This project is under active development.

CHAPTER

ONE

MEG SYSTEM DESCRIPTION: KIT MACHINE

1.1 System Overview

MagnetoEncephaloGraphy (MEG) systems are machines capable of measuring the magnetic field generated by the brain. They provide high temporal and spatial resolutions. They are non-invasive, similar to a microphone listening to your voice, MEG listens for the brain activity. MEGs are equipped with highly sensitive sensors called SQUIDs. In order for SQUIDS to operate, they need to be cooled down to -277 degres, to achieve this temperature liquid Helium is needed. Unlike MRI, MEG cannot show the anatomical structure of the brain, therefore MRI scans are combined with MEG measurements to identify the parts of the brain responsible for the measured brain activity. The magnetically shielded room is a product of Vacuumschmelze (Hanau, Germany). The shielding effect is provided by two layers of mu metal; the inner layer is 3 mm and the outer layer is 2 mm thick. Predicted shielding performance was rated to be -60 dB at 1 Hz; actual performance exceeds this prediction. The exterior dimensions of the room are 2.9 x 3.5 x 2.9 m, and the inner dimensions are 2.4 x 3.0 x 2.4 m. We refer to our system as having 160 channels, but in actuality it contains: 157 axial gradiometers used to measure brain activity, 3 orthogonally-oriented (reference) magnetometers located in the dewar but away from the brain area, used to measure and reduce external? noise offline, and 32 open positions, of which we currently use 8 to record stimulus triggers and the other 24 channels to record Eye Tracker data directly, auditory signals from our mixer and vocalization information from our optoacoustic fiber-optic microphone.

The system is located inside a magnetically shielded room. KIT refers to Kanazawa Institute of Technology, the manufacturer of the system.

1.2 LAB setup

Computers:

- MEG Main PC: used to acquire the MEG data
- stimulus1 pc: used to run the experiment
- stimulus 2 pc: used to put the experiment

1.3 MEG-Channels

(This part needs to be rewritten) Channels 0 to 222: Gradiometers squids

Channels 208-223: Magnetometers for reference magnetic field (these are used to denoising and to understand the ambiant magnetic field the environment)

224: Lightsensor 1

225: Lightsensor 2

228: Microphone

229: Event marker bit 0

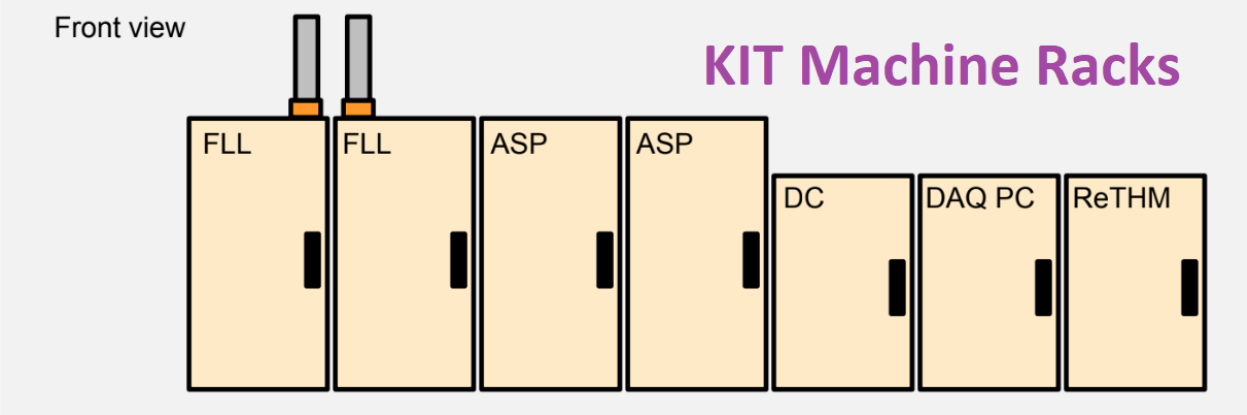
230: Event marker bit 1

231: Event marker bit 2

One of the channels (In the 80's) displays a digital signal, this is because one of the sensors are shut off and not used. Processing pipeline should include this exclusion and not process data from this channel. (channel name to be identified).

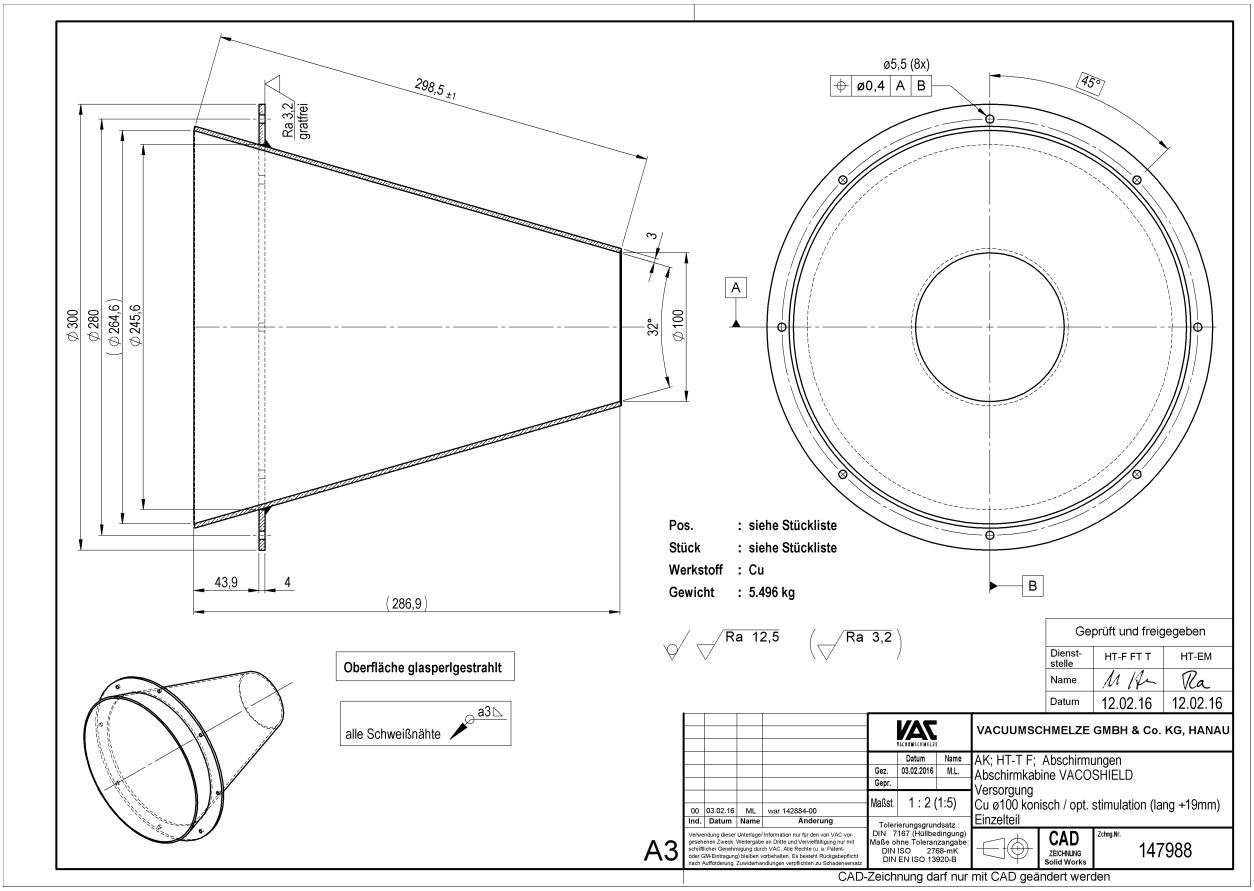
1.4 MEG-Racks

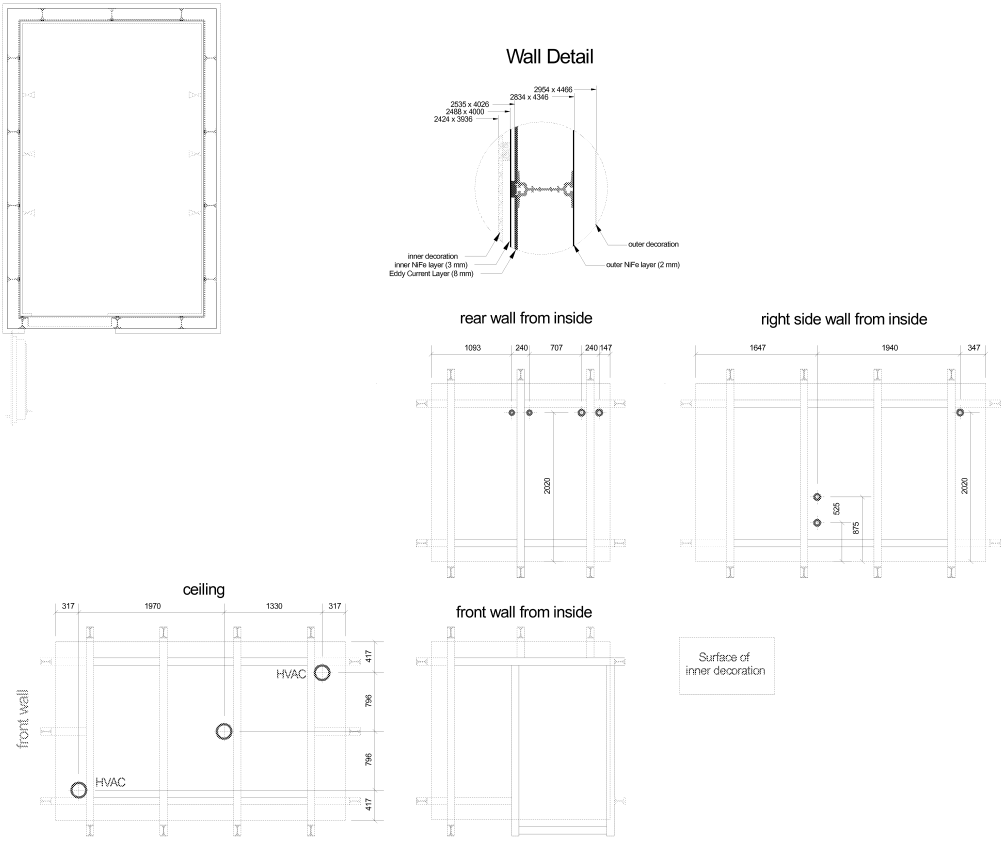
The KIT-MEG system has 7 racks



1.5 MSR: Magnetically Shielded Room

The KIT-MEG is located in an MSR built by VacuumShmelze





1.6 Contact

Name	Email	Number	Role
Hadi Zaatiti	hz3752@nyu.edu	+971 56 275 4921	Research Scientist
Osama Abdullah	osama.abdullah@nyu.edu	NA	Senior Scientist
Yoshiaki Adachi	adachi@ael.kanazawa-it.ac.jp	NA	MEG-KIT machine constructor reference

1.7 References

The following is a list of references for further understanding on MEG systems

- **MNE-Python: Overview and tutorials**
  - https://mne.tools/stable/auto_tutorials/intro/10_overview.html#sphx-glr-auto-tutorials-intro-10-overview-py
- **Marijn van Vliet's "Introduction to MNE-Python"**
  - https://mybinder.org/v2/gh/wmvanvliet/neuroscience_tutorials/master?filepath=mne-intro%2Findex.ipynb
- **Processing and analysis scripts from various Nellab members/alumni**

  - https://github.com/benebular/mne-python-preproc-templates
  - https://github.com/jdirani/MEGmvpa
  - https://github.com/jdirani/mne-preprocessing-template
  - https://github.com/jdirani/meg-analysis-templates
  - https://github.com/grahamflick/Nellab-MRI-Pipeline
  - https://github.com/grahamflick/Tools-for-Combined-MEG-and-Eye-tracking

- **Kit2fiff and ICA examples:**
  - https://docs.google.com/document/d/1zoxPCngUmyXuKYTNWM8W-_ncTld9okRuYncGXdVUtV0/edit?usp=sharing
  - https://docs.google.com/document/d/1OrVP9ts1gTGB5fhzx8YcK3JKZQgm0HM4Ic3hKtVzHzA/edit?usp=sharing
  - https://docs.google.com/document/d/1X9Tj28ekJ93TubJ52TnrebDvIh8zeXHLp2aMURNV40Y/edit?usp=sharing

- **Books:**
  - Hansen, Peter & Kringelbach, Morten & Salmelin, Riitta. (2010). MEG: An introduction to methods. 10.1093/acprof:oso/9780195307238.001.0001.

CHAPTER

TWO

MEG SYSTEM DESCRIPTION: OPM-MEG MACHINE

2.1 System Overview

The OPM has been installed on the 4th of march 2024.

2.2 Sensor locations on helmet

To access the sensors locations on the helmet: under the OPM computer, go to */usr/share/hedscan/doc/Beta2SensorLocations.png*

CHAPTER

THREE

EEG SYSTEM SETUP

Preliminary notes:

- scripts developed with Psychtoolbox should set the screen number set to ‘1’
- make sure that “Rear” projection mode is enabled, so that the participant sees text correctly from left to right

CHAPTER

FOUR

MEG QUIZZ

Test your knowledge. Click the next link to start the quizz, you will need to sign-in to your email.

Quizz form access

CHAPTER

FIVE

AUTHORIZATION DOCUMENTS

The following documents need to be signed by the participant prior to the study:

- Consent to participate in a research study

Every project owner must have an IRB for his study approved before getting participants to do their experiment

- IRB approval for project owner

CHAPTER

SIX

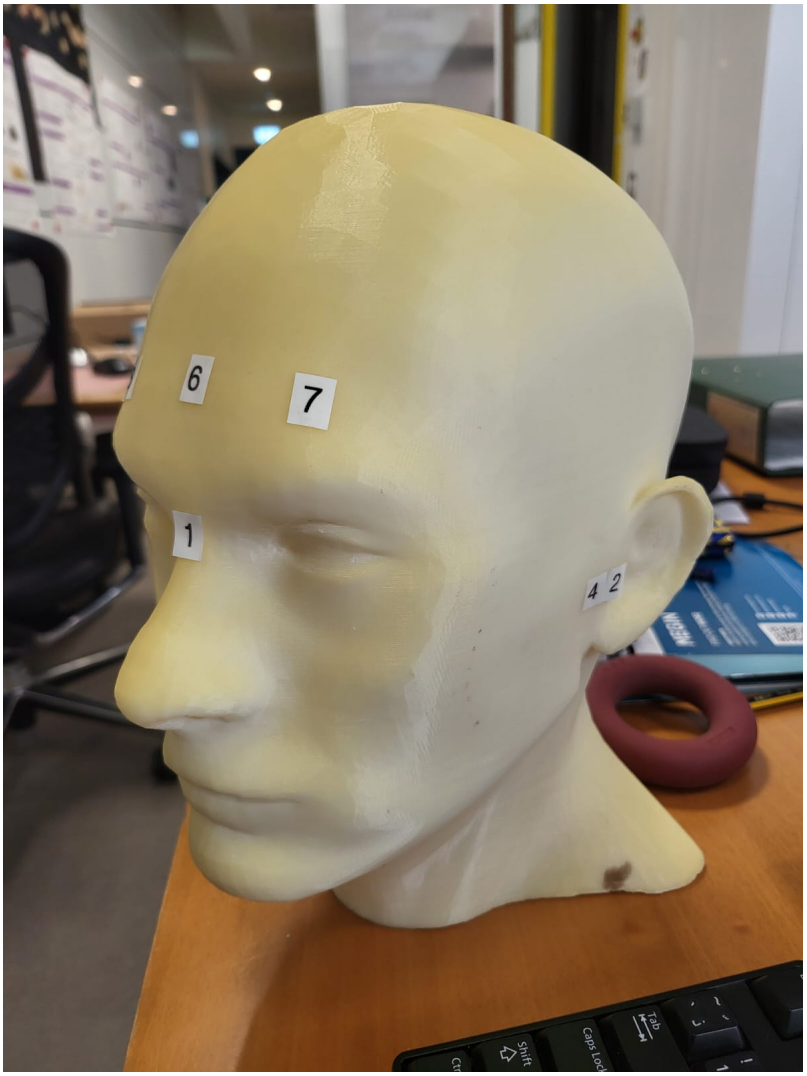
OPERATION PROTOCOL

Lead author:

Step 1 is to acquire a scan of the head surface generating a .ext (to be added) file for the participant

Step 2 is to

6.1 Stylus location and markers



The following table sumarises the position of each registered stylus location and whether or not a KIT coil will be placed on that position.

Index	Body Part	Marker Coil Information
1	Nosion	KIT: NO, OPM:
2	Left Traps	KIT: NO, OPM:
3	Right Traps	KIT: NO, OPM:
4	Left Ear	KIT: YES, OPM:
5	Right Ear	KIT: YES, OPM:
6	Center Forehead	KIT: YES, OPM:
7	Left Forehead	KIT: YES, OPM:
8	Right Forehead	KIT: YES, OPM:

CHAPTER

SEVEN

IMPLEMENTING YOUR EXPERIMENT

7.1 Purpose

This section provides information to help you out designing your MEG experiment. What is meant by experiment, is the stimuli involving usually visual and auditory or other perception-type stimulus. The experiment defines the timing of display of the stimuli, tracks responses from the participants and controls the different settings related to the content being presented to the participant. This section also provides the requirements that should be met to run your experiment in the NYUAD MEG Lab.

There are three tools primarily used for designing the experiment

- Psychotoolbox
- Presentation
- Psychopy

7.2 Definning the hardware needs for your experiment

Depending on your study you might need different require different hardware, the following use cases can be identified:

- Show visual stimuli to participants for a certain amount of time
- Allow participant to send their input via buttons
- Get eyetracking information from the eyetracker device
- Provide audio to the user
- Record audio from the user's voice

7.3 Hardware involved in experiment

- Propixx
- Datapixx
- Eyetracker

Datapixx pixel mode Pixel mode.

The eyetracker sends three different signals to the MEG/EEG channels:

- The X-coordinates of the eye as function of time

- The Y-coordinates of the eye as function of time
- The Area of the pupil of the eye as function of time

7.4 Files produced by the experiment design

- An experiment in PsychToolBox is a *.m* MATLAB script.
- Presentation provides a *.exp* file, an experiment file.
- PsychoPy is a *.py* experiment file.

If using python library PsychoPy:

- Open the file with .psyexp extension
- you can run from within the psycopy builder the experiment file with .psyexp extension c

7.5 Experiments

7.5.1 Experiments example 1 (Psychtoolbox): Resting state

- Resting state experiment: Using PsychToolBox the following script executes a resting state experiment.

The participant is asked to close their eyes for some time, then to open their eyes while fixing a centered shape for a same duration. Two triggers are sent from the 'Datapixx3' to the KIT-MEG on channels 224 (closing eyes) and 225 (opening eyes). The code for the experiment can be found here: Source file link resting_state_meg.m.

```
clearvars
%Screen('Preference', 'SkipSyncTests', 0);
AssertOpenGL;


vpix_use = 1; % 0 if vpixx is not conected

% KEYBOARD SETUP
responseKeys = {'2', '3', 'y', 'n'};
KbName('UnifyKeyNames');
KbCheckList = [KbName('space'),KbName('ESCAPE'), KbName('leftarrow'), KbName('rightarrow
↪')];
for i = 1:length(responseKeys)
    KbCheckList = [KbName(responseKeys{i}),KbCheckList];
end

% SCREEN SETUP
screens = Screen('Screens');

s = max(screens);

black = [0 0 0];

[w, rect] = Screen('Openwindow',s,black)
```

(continues on next page)

(continued from previous page)

```
Priority(MaxPriority(w));

Screen('BlendFunction', w, GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
pixelSizes=Screen('PixelSizes', s);
fps=Screen('FrameRate',w);
ifi=Screen('GetFlipInterval', w);

[wx, wy] = RectCenter(rect);
Screen('Flip', w)


% Uncomment if Vpixx is connected, or else experiment will crash

if vpix_use == 1
    %VIEW PIXX SETUP
    Datapixx('Open');
    Datapixx('EnablePixelMode');  % to use topleft pixel to code trigger information,␣
↪see https://vpixx.com/vocal/pixelmode/
    Datapixx('RegWr');
end

% TRIGGERS SETUP
trigRect = [0 0 1 1]; % Top left pixel that controls triggers in PixelMode
%centeredRect_trigger = CenterRectOnPointd(baseRect_trigger, 0.5, 0.5);


% RGB color for top left pixel to trigger a channel on MEG

% output of Vpixx will be triggered
% Ref: https://docs.vpixx.com/vocal/defining-triggers-using-pixel-mode

% % triggers as color (RGB) of tope-left pixel in the screen
% Reference of all triggers for KIT MEG in NYUAD:
% trigger_224 = [4  0 0]; % 224 meg channel
% trigger_225 = [16 0 0]; % 225 meg channel
% trigger_226 = [64 0 0]; % 226 meg channel
% trigger_227 = [0  1 0]; % 227 meg channel
% trigger_228 = [0  4 0]; % 228 meg channel
% trigger_229 = [0 16 0]; % 229 meg channel
% trigger_230 = [0 64 0]; % 230 meg channel
% trigger_231 = [0 0  1]; % 231 meg channel
% example:
%   [16 0 0]  in binary is [10000 0 0] ==> Means pin number 4 on digital
%   [64 0 0] in binary is [1000000 0 0] ==> Means pin number 7 will be triggered

% Define triggers for closing eyes and opening eyes
trig.closed = [4  0  0]; %224 meg channel
trig.open = [16  0  0];  %225 meg channel

%Ensure all vpixx digital output are set to 0 by putting the trigger pixel
%to black [0 0 0]
Screen('FillRect', w, black, trigRect);
```

(continues on next page)

(continued from previous page)

```
% STIMULI SETUP

fixRadius = 30;
fixRect = CenterRectOnPoint([0, 0, fixRadius*2, fixRadius*2], wx, wy);
fixColor = [150 150 150];

time2rest = 60*5;

% START EXPERIMENT

Screen('DrawText', w, 'PRESS SPACE AND START CLOSED EYES REST',  wx-250, wy, [255,255,
↪255]);
Screen('Flip', w);

KbWait([],2)


Screen('FillRect', w, trig.closed, trigRect);
Screen('Flip', w);
WaitSecs(time2rest)

Screen('DrawText', w, 'PRESS SPACE AND START OPEN EYES REST',  wx-250, wy, [255,255,
↪255]);
Screen('Flip', w);
KbWait([],2)


Screen('FillRect', w, trig.open, trigRect);
Screen('FillOval', w, fixColor, fixRect);
Screen('Flip', w);
WaitSecs(time2rest)

Screen('CloseAll');

if vpix_use == 1
    %VIEW PIXX SETUP
    Datapixx('Close');
end
```

7.5.2 Experiments example 2 (Psychtoolbox): Triggering all channels on KIT

- Triggering all channels on the KIT machine one by one

The following script triggers each event channel on the KIT from 224 to 231,

test_all_meg_channels_triggers.m.

```
%% This script should send a trigger to each MEG channel, with a 1 second delay between␣
↪each trigger
```

(continues on next page)

(continued from previous page)

```
clearvars
Screen('Preference', 'SkipSyncTests', 1);
AssertOpenGL;


% Configuration parameters

vpix_use = 1; % 0 if vpixx is not conected
trigger_test = 1;
% if 0, trigger is 1 pixel,
% if 1 trigger is bigger (to be able to see it)

% SCREEN SETUP
s = Screen('Screens');

s = 2;

%Colors definition in RGB
black = [0 0 0];
white = [255 255 255];

% Get pointer to screen window and a point
[w, rect] = Screen('Openwindow',s,black)
Priority(MaxPriority(w));
Screen('Flip', w)
[wx, wy] = RectCenter(rect);



if vpix_use == 1
    %VIEW PIXX SETUP
    Datapixx('Open');
    Datapixx('EnablePixelMode');  % to use topleft pixel to code trigger information,␣
↪see https://vpixx.com/vocal/pixelmode/
    Datapixx('RegWr');
end


% TRIGGERS SETUP

% Top left pixel that controls triggers in PixelMode
if trigger_test == 0
    trigRect = [0 0 1 1];
    %centeredRect_trigger = CenterRectOnPointd(trigRect, 0.5, 0.5);
elseif trigger_test == 1
    trigRect = [0 0 100 100];
    %centeredRect_trigger = CenterRectOnPointd(trigRect, 25, 25);
end



%centeredRect_trigger = CenterRectOnPointd(baseRect_trigger, 0.5, 0.5);
```

(continues on next page)

(continued from previous page)

```
% Define trigger pixels for all usable MEG channels
trig.ch224 = [4  0  0]; %224 meg channel
trig.ch225 = [16  0  0];  %225 meg channel
trig.ch226 = [64 0 0]; % 226 meg channel
trig.ch227 = [0  1 0]; % 227 meg channel
trig.ch228 = [0  4 0]; % 228 meg channel
trig.ch229 = [0 16 0]; % 229 meg channel
trig.ch230 = [0 64 0]; % 230 meg channel
trig.ch231 = [0 0  1]; % 231 meg channel

fields = fieldnames(trig); % Get the field names of the structure

time2trigger = 5;



times = 3;
for j = 1:times

    for i = 1:numel(fields)
        fieldName = fields{i}; % Get the field name
        fieldValue = trig.(fieldName); % Get the value of the field

        fprintf('%s: [%d %d %d]\n', fieldName, fieldValue); % Print the field name and␣
↪value

        message = ['One trigger every ', int2str(time2trigger),' seconds.' ...
            'Channel name getting triggered now: ', fieldName];
        Screen('DrawText', w, message,  wx-250, wy, [255,255,255]);

        Screen('FillRect', w, fieldValue, trigRect);
        Screen('Flip', w);
        Screen('DrawText', w, message,  wx-250, wy, [255,255,255]);
        Screen('FillRect', w, black, trigRect);
        Screen('Flip', w);
        WaitSecs(time2trigger);
    end
end

Screen('CloseAll');

if vpix_use == 1
    %VIEW PIXX SETUP
    Datapixx('Close');
end
```

7.5.3 Experiments example 3 (Psychopy): Triggering all channels on KIT

- Triggering all channels on the KIT machine one by one

The following script triggers each event channel on the KIT from 224 to 231 using PsychoPy script trigger_test_psychopy.py.

```
from psychopy import visual, core
from pypixxlib import _libdpx as dp



# Define trigger pixels for all usable MEG channels
#trig.ch224 = [4  0  0]; %224 meg channel
#trig.ch225 = [16  0  0];  %225 meg channel
#trig.ch226 = [64 0 0]; % 226 meg channel
#trig.ch227 = [0  1 0]; % 227 meg channel
#trig.ch228 = [0  4 0]; % 228 meg channel
#trig.ch229 = [0 16 0]; % 229 meg channel
#trig.ch230 = [0 64 0]; % 230 meg channel
#trig.ch231 = [0 0  1]; % 231 meg channel



def drawPixelModeTrigger(win, pixelValue):
    # takes a pixel colour and draws it as a single pixel in the top left corner of the␣
↪window
    # window must cover top left of screen to work
    # interpolate must be set to FALSE before color is set
    # call this just before flip to ensure pixel is drawn over other stimuli

    topLeftCorner = [-win.size[0] / 2, win.size[1] / 2]
    line = visual.Line(
        win=win,
        units='pix',
        start=topLeftCorner,
        end=[topLeftCorner[0] + 1, topLeftCorner[1]],
        interpolate=False,
        colorSpace='rgb255',
        lineColor=pixelValue)
    line.draw()


def RGB2Trigger(color):
    # helper function determines expected trigger from a given RGB 255 colour value
    # operates by converting individual colours into binary strings and stitching them␣
↪together
    # and interpreting the result as an integer

    # return triggerVal
    return int((color[2] << 16) + (color[1] << 8) + color[0])  # dhk
```

(continues on next page)

(continued from previous page)

```
def Trigger2RGB(trigger):
    # helper function determines pixel mode RGB 255 colour value based on 24-bit trigger␣
↪(in decimal, base 10)
    # returns a list with R, G and B elements

    # return [red, green, blue]
    return [trigger % 256, (trigger >> 8) % 256, (trigger >> 16) % 256]  # dhk


##
# START
# Initialize connection and set up some default parameters:
dp.DPxOpen()
dp.DPxEnableDoutPixelMode()
dp.DPxWriteRegCache()

win = visual.Window(
    screen=1,  # change here to 1 to display on second screen!
    monitor=None,
    size=[1920, 1080],  # dhk: PsychoPy drew a grey (49,49,49) border around this small␣
↪window
    # fullscr=False,      # therefore, top-left pixel was drawn with incorrect color.
    fullscr=False,  # using a full screen window resolved this issue
    pos=[0, 0],
    color='black',
    units="pix"
)

testvals = [0, 64, 128, 191, 255]


# KIT MEG Channels triggered via Pixel Model by setting top left pixel to a specific␣
↪color
#trig.ch224 = [4  0  0]; %224 meg channel
#trig.ch225 = [16  0  0];  %225 meg channel
#trig.ch226 = [64 0 0]; % 226 meg channel
#trig.ch227 = [0  1 0]; % 227 meg channel
#trig.ch228 = [0  4 0]; % 228 meg channel
#trig.ch229 = [0 16 0]; % 229 meg channel
#trig.ch230 = [0 64 0]; % 230 meg channel
#trig.ch231 = [0 0  1]; % 231 meg channel

trigger = [[4, 0, 0], [16, 0, 0], [64, 0, 0], [0, 1, 0], [0, 4, 0], [0, 16, 0], [0, 64,␣
↪0], [0, 0, 1]]
channel_names  = ['224', '225', '226', '227', '228', '229', '230', '231']
black = [0, 0, 0]


failcount = 0
print('\nStarting Pixel Mode Test\n\nTest#\tRGB225 Color\t    Expected Dout    Returned␣
↪Dout')
```

(continues on next page)

(continued from previous page)

```
for i in range(5):
    for index, color in enumerate(trigger):

            print('Testing channel', channel_names[index])
            drawPixelModeTrigger(win, color)
            win.flip()
            color = black
            drawPixelModeTrigger(win, color)
            win.flip()
            core.wait(5)
            dp.DPxUpdateRegCache()

win.close()
dp.DPxDisableDoutPixelMode()
dp.DPxWriteRegCache()
dp.DPxClose()
```

7.5.4 Experiments example 4 (Psychopy): FaceInversion

- Face Inversion PsychoPy experiment

Run the *FaceInversion_Localizer.psyexp* within

FaceInversion Experiment.

7.5.5 Experiments example 5: Response buttons experiment

The MEG Lab has two response boxes which allow the user to provide their input during an experiment.

The *Left box* is the grey box and the *Right box* is the blue box.

- To test the response boxes you can run the following script

```
% DatapixxDinBasicDemo()
%
% Demonstrates the basic functions of the DATAPixx TTL digital inputs.
% Prints the number of TTL inputs in the system,
% then logs button presses until user hits a key.
%
% Also see: DatapixxSimonGame
%
% History:
%
% Oct 1, 2009  paa     Written
% Oct 29, 2014 dml     Revised

AssertOpenGL;   % We use PTB-3

% Open Datapixx, and stop any schedules which might already be running
Datapixx('Open');
```

(continues on next page)

(continued from previous page)

```
Datapixx('StopAllSchedules');
Datapixx('RegWrRd');    % Synchronize DATAPixx registers to local register cache

% Show how many TTL input bits are in the Datapixx
nBits = Datapixx('GetDinNumBits');
fprintf('\nDATAPixx has %d TTL input bits\n', nBits);

% RESPONSEPixx has 5 illuminated buttons.
% We drive those button lights by turning around 5 DIN bits to outputs.
% Test paradigms could illuminate only the buttons which are valid in context.
% (eg: 1 button when waiting for subject to initiate a trial,
% 2 other buttons when waiting for 2-alternative forced-choice response).
Datapixx('SetDinDataDirection', hex2dec('1F0000'));
Datapixx('SetDinDataOut', hex2dec('1F0000'));
Datapixx('SetDinDataOutStrength', 1);   % Set brightness of buttons

% We'll say that we want to calculate response times
% from a stimulus appearing at the next vertical sync.
Datapixx('SetMarker');
Datapixx('RegWrRdVideoSync');
stimulusOnsetTime = Datapixx('GetMarker');

% Fire up the logger
Datapixx('EnableDinDebounce');      % Filter out button bounce
%Datapixx('DisableDinDebounce');    % Uncomment this line to log gruesome details of␣
↪button bounce
Datapixx('SetDinLog');              % Configure logging with default values
Datapixx('StartDinLog');
Datapixx('RegWrRd');

% Show initial state of all the digital input bits
fprintf('Initial digital input states = ');
initialValues = Datapixx('GetDinValues');
for bit = nBits-1:-1:0              % Easier to understand if we show in binary
    if (bitand(initialValues, 2^bit) > 0)
        fprintf('1');
    else
        fprintf('0');
    end
end
fprintf('\n');

% Report logged button activity until keyboard is pressed
fprintf('\nPlug button box into Digital IN db-25\n');
fprintf('Press buttons to see new log entries\n');
fprintf('Hit any key to stop...\n');
if (exist('OCTAVE_VERSION'))
    fflush(stdout);
end
while ~KbCheck
    Datapixx('RegWrRd');
    status = Datapixx('GetDinStatus');
```

(continues on next page)

(continued from previous page)

```
    if (status.newLogFrames > 0)
        [data tt] = Datapixx('ReadDinLog');
        for i = 1:status.newLogFrames
            fprintf('responseTime = %f', tt(i)-stimulusOnsetTime);
            fprintf(', button states = ');
            for bit = 15:-1:0               % Easier to understand if we show in binary
                if (bitand(data(i), 2^bit) > 0)
                    fprintf('1');
                else
                    fprintf('0');
                end
            end
            fprintf('\n');
        end
        if (exist('OCTAVE_VERSION'))
            fflush(stdout);
        end
    end
end

% Show final status of digital input logger
fprintf('\nStatus information for digital input logger:\n');
disp(Datapixx('GetDinStatus'));

% Job done
Datapixx('StopDinLog');
Datapixx('RegWrRd');
Datapixx('Close');
fprintf('\n\nDemo completed\n\n');
```

- To get the response of a user while performing your experiment, you can use the following MATLAB function getButton.m.

```
function [resp, time] = getButton()

while true
    Datapixx('RegWrRd');
    kbcheck = dec2bin(Datapixx('GetDinValues'));
    if kbcheck(end) == '1' || kbcheck(end-1) == '1' || kbcheck(end-2) == '1' ||␣
↪kbcheck(end-3) == '1' || kbcheck(end-5) == '1' || kbcheck(end-6)  == '1' ||␣
↪kbcheck(end-7) == '1' || kbcheck(end-8)  == '1'
        for i_but = 1:9
            buttonBox(i_but) = str2num(kbcheck(end-9+i_but));
        end

        resp = find(buttonBox);
        time = GetSecs;
        if length(resp) == 1
            break;
        end
    end
end
```

The above function will return an integer *resp* which you will have to translate using the following table to identify the color that has been pressed.

Box	Button Color	Button States	Response Number (resp)	Offset button
Left Box	Red	11111111111110000000001	9	0
Left Box	Yellow	11111111111110000000010	8	1
Left Box	Green	11111111111110000000100	7	2
Left Box	Blue	11111111111110000001000	6	3
Right Box	Red	11111111111110000100000	4	5
Right Box	Yellow	11111111111110001000000	3	6
Right Box	Green	11111111111110010000000	2	7
Right Box	Blue	11111111111110100000000	1	8

```
function [resp, time] = listenButton(offset)

while true
    Datapixx('RegWrRd');
    kbcheck = dec2bin(Datapixx('GetDinValues'));

    if kbcheck(end-offset) == '1'
        for i_but = 1:9
            buttonBox(i_but) = str2num(kbcheck(end-9+i_but));
        end

        resp = find(buttonBox);
        time = GetSecs;
        if length(resp) == 1
            break;
        end
    end
end
```

The above function listens to a specific button press depending on the offset variable given as input, if the specific button is not pushed, the function stays in the while loop.

CHAPTER

EIGHT

BUILDING THE REQUIREMENT OF YOUR EXPERIMENT

CHAPTER

NINE

IDENTIFYING YOUR USAGE

(Add usage form)

CHAPTER

TEN

BOOKING SYSTEM AND SCHEDULING

While scheduling your experiment, avoid rush hours 8:30am and 5:30pm

rush hour has alot of noise avoid experiments during this time

Important: Please schedule your experiment in the MEG lab on https://corelabs.abudhabi.nyu.edu/

CHAPTER

ELEVEN

PIPELINE DESCRIPTION

11.1 General overview

11.2 Data preparation

Head surface scan generates: - _basic.txt - _points.txt - .fsn

MEGLab acquisition generates: - .con file - _NR.con file (after analysing noise reduction) - .mrk : an experiment will produce atleast 2 .mrk files, they contain the markers data

All data generated from KIT or OPM are saved on NYU Box Data access

Note: The link is invitation based only and not publicly available.

11.3 Installation

To use MEG-Pipeline, first install it using pip:

```
(.venv) $ pip install megpipeline
```

11.4 Reading the Raw Data

The `kind` parameter should be either "`raw`", "`fif`", or "`fll`".

```
#Import raw MEG files from the KIT machine
#Saves as .fif
#Lead author: Hadi Zaatiti


import mne

SUBJECTS_ID = ['Y0409', 'Y0440']

experiments = ['01','02']
```

(continues on next page)

(continued from previous page)

```
for experiment in experiments:
    for subject in SUBJECTS_ID:

        DIR = '../MEG_DATA/' + subject + '/'
        MEG_DATA_DIR = DIR + subject + '_'+experiment+'.con'
        HSP_DIR = DIR + subject + '_basic.txt' #Head Shape DIR
        STYLUS_DIR = DIR + subject + '_points_no_grad.txt'
        MARKERS = [subject + '-1.mrk', subject + '-2.mrk', subject + '-3.mrk']

        MARKERS_DIRS = [DIR + marker_path for marker_path in MARKERS]


        RAW_DATA = mne.io.read_raw_kit(input_fname=MEG_DATA_DIR,
                            mrk= MARKERS_DIRS,
                            elp = STYLUS_DIR,
                            hsp = HSP_DIR)


        RAW_DATA.save('../output/' + subject + '/' + subject +'_'+experiment+'_meg.fif')
```

The above script will later be implemented as part of the following class `MEGpipeline` and function `megpipeline.get_raw_data()`.

`megpipeline.MEGpipeline.`**`get_raw_data`**(*self*, *file_name*)

Return a list of random as strings.

Parameters
kind (*list[str] or None*) – Optional “kind” .

Returns
the converted data list.

Return type
list[str]

The `kind` parameter should be either `"raw"`, `"fif"`, or `"fll"`. Otherwise, `megpipeline.get_raw_data()` will raise an exception.

For example:

```
>>> import megpipeline
>>> megpipeline.get_raw_data()
['a', 'b', 'c']
```

CHAPTER

TWELVE

DATA STORAGE

12.1 MEG Data storage

The MEG data is securely stored on NYU BOX, access is given through invitations. The *Data* folder is structured in the **BIDS** standardized format. Please raise an issue on *github* repository if you think the structure does not conform to **BIDS**.

Sign in on NYU Box in the below link

Link to MEG data (Box Invitation Only)

https://nyu.box.com/v/meg-datafiles

Or directly from the below widget

If you are unable to access the datasets it means you do not have the permission to. Kindly contact us to get permission.

12.2 MRI Data storage

MRI data is hosted on Flywheel

Link to MRI data (Box Invitation Only)

https://flywheel.abudhabi.nyu.edu/#/login

CHAPTER

THIRTEEN

DATA NAMING AND UPLOADING PROTOCOL

In the following, [SUB_ID] should be replaced with the ID of the subject for naming purposes. The different data files generated from a MEG experiment are the following.

Note: If you have suggestions to make the naming convention better, please raise an issue on github or create a pull request with your proposed modifications.

13.1 Laser scan files

1. A .fsn filename that should be named `sub-\[SUB_ID\]_scan.fsn` : This file is obtained by saving the whole fastscan laser project (File Save)
2. **Several .txt**
    - `sub-[SUB_ID]_scan.txt` is the head scan of the participant
    - `sub-[SUB_ID]_scan_stylus.txt` is the stylus location file of the participant

13.2 KIT-MEG files

Depending on the experiment, many .con files can be produced by the KIT machine.

1. .con files are named: * `sub-[SUB_ID]_[date].con`
2. .mrk files are named: * `sub-[SUB_ID]_[marker_number]_[date].mrk` where [marker_number] is replaced with the number of the marker (representing the time-order of acquisition of that marker)

13.3 OPM files

The OPM system generates a BIDS directory with the .fif files

13.4 Data uploading

Data will be uploaded to NYU BOX, to the following link

Link to MEG data (Box Invitation Only)

https://nyu.box.com/v/meg-datafiles

Steps

1. Access the folder of NYU Box
2. Identify the project that the data belongs to
3. Access the folder for that project or create a new folder if non-existent
4. **Follow the BIDS structure format, within each project file, we find subjects file named by their ID**
    - Within each subjects folder we find, *anat*, *meg-kit*, *meg-opm* folders
    - Place the corresponding files into the right folder: all headscan files go to *anat*, the .con and .mrk goes to *meg-kit*, the .fif goes to *meg-opm*
5. Make sure that all files have been uploaded to the folder

CHAPTER

FOURTEEN

SETTING UP YOUR ENVIRONMENT FOR PROCESSING

We recommend to install MNE as a standalone installer.

The data of the MNE environment will be under `C:\ProgramData\mne-python\1.6.1_0`

You can setup Pycharm to use this environment as a Python environment for your pipeline project.

The configuration file for setting `SUBJECTS_DIR`, the directory to the data of your subject, can be set in: `C:\Users\user_name\.mne\mne-python.json`

CHAPTER

FIFTEEN

INSTALLING FREESURFER ON WINDOWS

Configure WSL2 on windows Access the files of the ubuntu distribution by typing `\\wsl$\Ubuntu` in the file explorer in windows.

Install freesurfer following their documentation page. https://surfer.nmr.mgh.harvard.edu/fswiki/FS7_wsl_ubuntu

CHAPTER

SIXTEEN

KIT2FIFF TUTORIAL

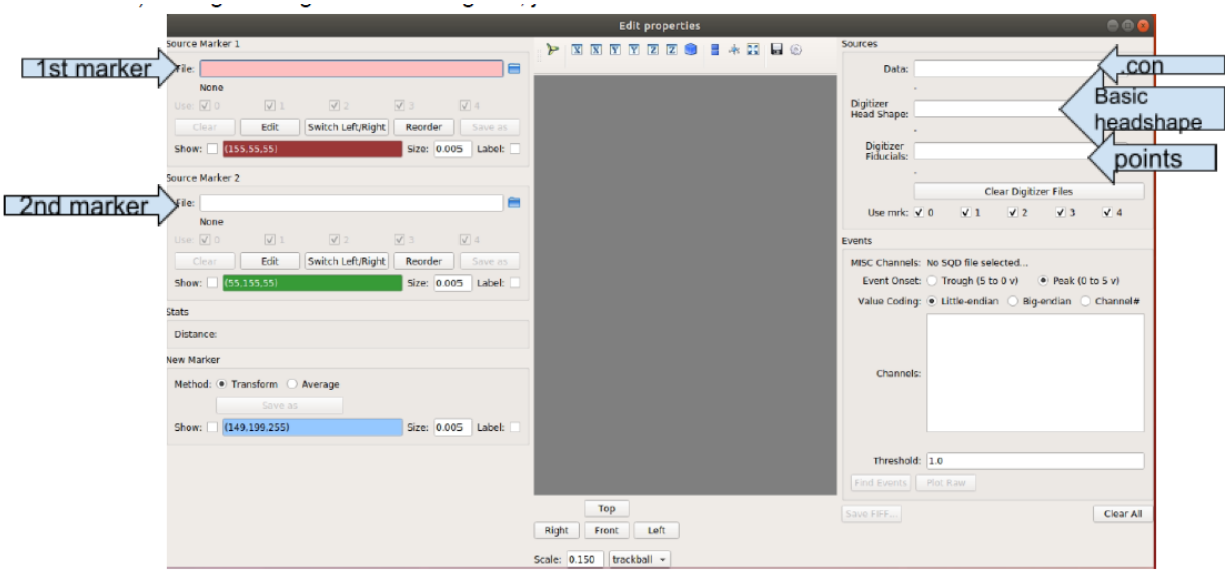
There are multiple files produced before and during magnetoencephalography. We will use the following here:

- Headscan basic surface .txt
- Headscan points .txt
- HPI coils *Marker* measurements (x2 files atleast) .mrk (each .mrk contains the position of 5 fiducial points on the face)
- MEG recording .con

16.1 Kit2Fiff

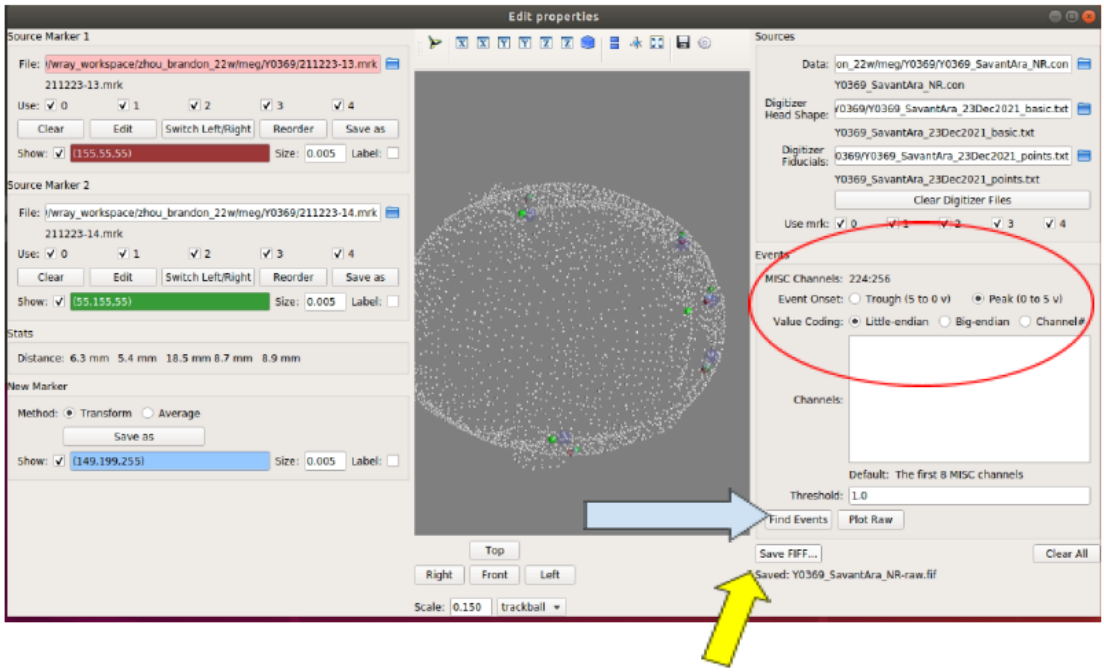
The first step is to convert the recording into a standard format for analysis in MNE, the premier software suite for M/EEG analysis.

1. Launch your terminal and activate your anaconda environment for MNE analysis. If you haven't set up an environment yet, do so.
2. In your terminal, run `mne kit2fiff`. This will launch a GUI with the following interface:
3. Using the diagram here as a guide, join the files listed above:



After the files are all loaded, you will see the headscan plotted in the gray panel in the middle of the GUI. You can rotate it around. A small sanity check should be performed here to see if the markers are in their expected position around the head.

Note: The parameters indicated by the red circle below should be set according to the experiment control software.

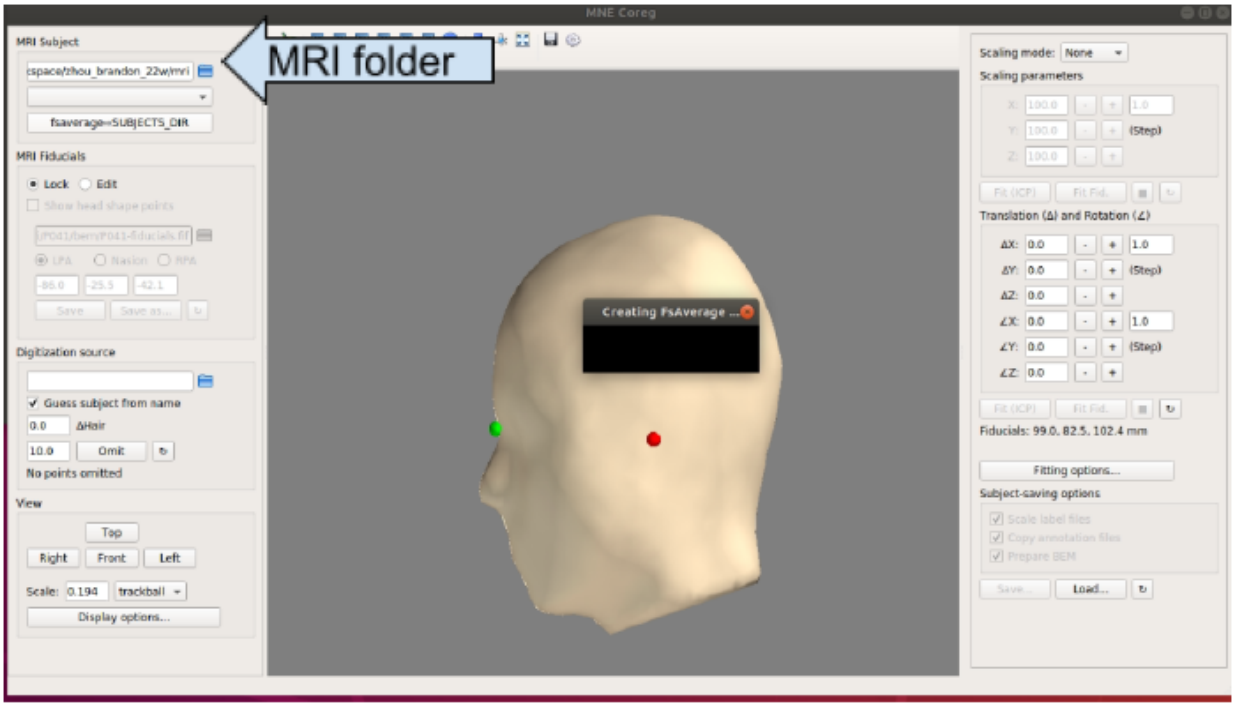


For experiments run in PsycoPy, the events should be indicated as "Trough", and for experiments run in Presentation, the triggers should be indicated as "Peak". Click "Find Events". If you find a list of events, you probably do have triggers indicating stimuli times. Hooray! If not, make sure the parameters in red are set as shown here. If that doesn't fix it, triggers were not sent properly.

4. After making sure the correct event type is selected for the experiment control software used, save the file by clicking on "Save Fiff". MNE suggests a filename; it is good practice to use the following naming convention: `subjID_experimentname-raw.fiff`.
5. When the .fiff has been saved, close the GUI.

16.2 Coreg without native MRI

In your terminal, run `mne coreg`. This will launch a GUI with the following interface.



1. Navigate to the MRI folder for your experiment in the spot indicated by the blue arrow. If this is the first coreg you are processing for this dataset, you will need to put the fsaverage in the MRI folder to serve as a basis for transformations of your subjects' heads.
2. In Digitization Source, put the .fiff created from earlier for the appropriate subject.

Note: This part of the preprocessing takes the most subjective judgment and hard work thus far.

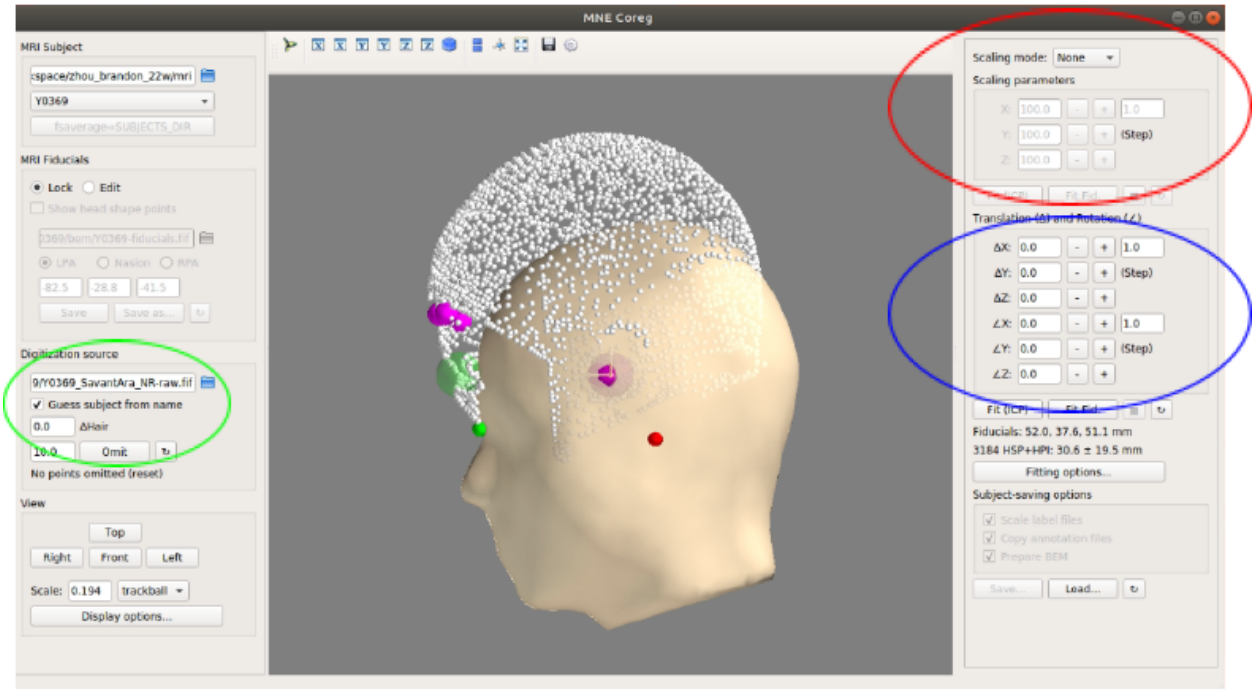
You will need to align the white net of dots (representing the MEG recording linked with the subject headshape) to the fsaverage headshape. You will do this by manipulating two parameters: translation of the net and transformation of the fsaverage headshape. The former is done with the controls in blue. Current versions of MNE allow the translation to be performed automatically by hitting the buttons marked "Fit (ICP)" and "Fit Fid.". Fit (ICP) will fit the white dots to the headshape. Fit Fid will fit the markers/points to the headshape markers/points. This approach should be alternated with transforming the headshape using the controls in red. First, you should change Scaling mode to "3-axis". This will allow the headshape to be transformed in three dimensions independently. To transform, hit Fit (ICP) within red.

Note: If a subject had a particularly thick hairstyle, you can add hair by putting a number (in mm) in green. You can also omit white dots that are too far

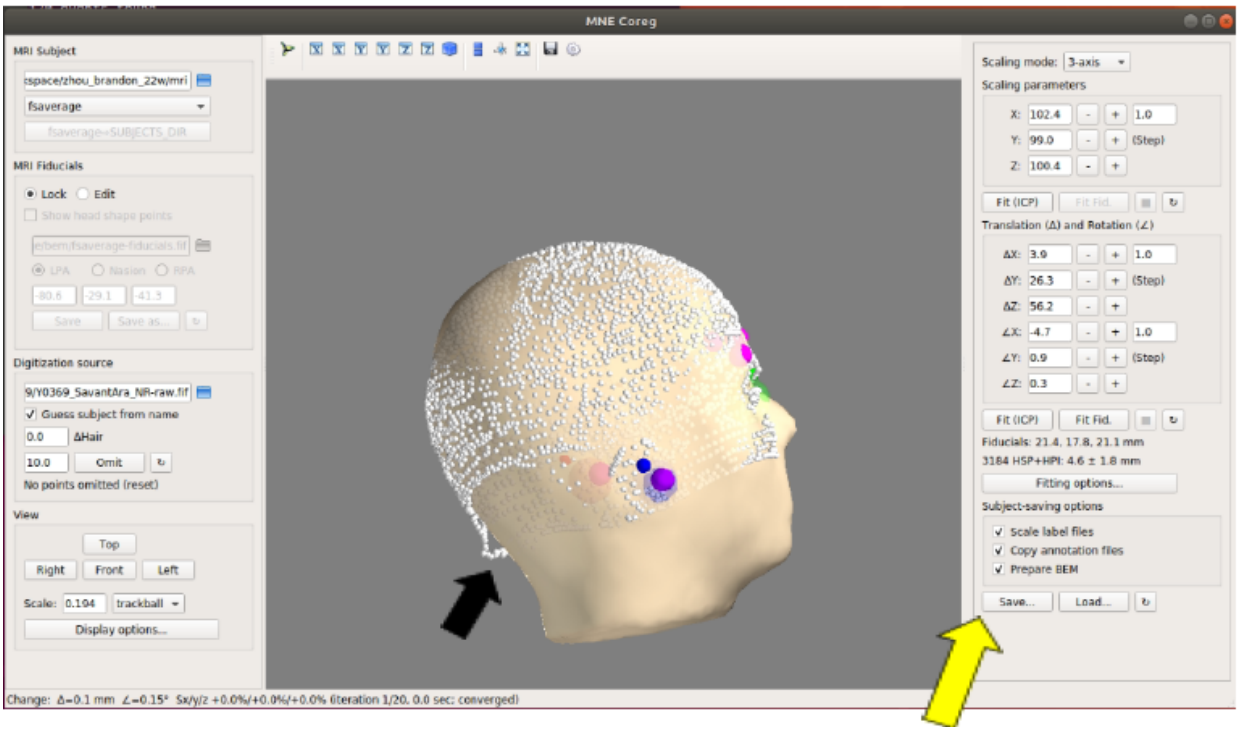
---

3. Navigate to the MRI folder for your experiment in the spot indicated by the blue arrow. If this is the first coreg you are processing for this dataset, you will need to put the fsaverage (average headshape and MRI) in the MRI folder to serve as a basis for transformations of your subjects' heads. To do this, under the MRI folder, there is a button for fsaverage=SUBJECTS_DIR. You'll need to set fsaverage as the headshape using the dropdown menu below the MRI folder selection; if there are any processed datasets already in the MRI folder, it will try to set those subjects as the base. Make sure your base is always fsaverage. In Digitization Source, put the fiff created from earlier for the

appropriate subject 4. This part of the preprocessing takes the most subjective judgment and hard work thus far. You will need to align the white net of dots (representing the MEG recording linked with the subject headshape) to the fsaverage headshape. You will do this by manipulating two parameters: translation of the net and transformation of the fsaverage headshape. The former is done with the controls in blue. Current versions of MNE allow the translation to be performed automatically by hitting the buttons marked "Fit (ICP)" and "Fit Fid.". Fit (ICP) will fit the white dots to the headshape. Fit Fid will fit the markers/points to the headshape markers/points. This approach should be alternated with transforming the headshape using the controls in red. First, you should change Scaling mode to "3-axis". This will allow the headshape to be transformed in three dimensions independently. To transform, hit Fit (ICP) within red. If a subject had a particularly thick hairstyle, you can add hair by putting a number (in mm) in green. You can also omit white dots that are too far from the headshape that occasionally result from a bad headscan.



5. You can check the fit of the headshape by rotating the head around in the grey panel with your mouse. The goal is to have the white net of dots lying flush with the surface of the head with minimal gaps between the dots and headshape, and with minimal embedding of the dots inside the headshape. Don't be too concerned with aligning the point of the net marked with the black arrow below; that isn't part of the subject's head. It is part of the neckbrace.



6. When you are satisfied with the fit, hit Save. This produces many files, and takes a fair amount of time. It generates the BEM (Boundary Element Model)1 files, the anatomical files, and a .trans file that maps the anatomicals of the fsaverage to the subject. 7. When this is finished, close the GUI

To see if something needs to be kit2fiffed, see if there is a -raw.fif file. To see if something needs to be coreged, see if there is a -trans.fif file

1. Fit(ICP)
2. Scaling mode = 3-axis
3. Fit(ICP) scaling parameters
4. Back and forth Fit
5. Screenshot all five views to put in coreg reports

CHAPTER

SEVENTEEN

SOFTWARE STACK

MEG data analysis:

- *LabMaestroSimulator*
- BEESA
- MNE Python library

17.1 Example:

Samantha's experiment called Arabic Tark_VpixxEdit contains a .sce, .exp, .tem

What is Tark Localizer?

they are called Tark_Localiser.sce Task_localiser.exp Tark_Loc_Main_Trial_GR.tem

When you open the .sce, you see a code that define the name of the scenario, font size, active buttons

Everytime the experiment is ran, a logfile seems to be created in

Output:

On the computer of the MEG MAIN PC, an experiment can yield different files:

- **a .con file shows the signals on top of each other, and the strength of the magnetic field on what part of the brain the unit can be**
  - pT: picoTesla
  - fT: femtoTesla
- a .mrk file

This website adds quite a few details to these extensions https://mne.tools/stable/auto_tutorials/io/10_reading_meg_data.html

The files can be opened with *MEG Lab*

CHAPTER

EIGHTEEN

BESA SOFTWARE

The following steps are primary to process MEG data using the BESA MRI and BESA Research suite

18.1 You have MRI data of your participant

Open BESA MRI, start a new segmentation project, check all the segmentation options (especially BEM and FEM), pick the landmarks for segmentation and start the process. Once done, BESA will save the segmentation, BEM, FEM model outputs.

In BESA MRI, start a new coregistration project.

Open BESA Research, load your MEG data from a .fif format.

18.2 Generic processing pipeline

18.3 Manual labelling of “bad” channels

18.4 Denoising

Awareness of the many sources of noise:

- Related to the site in which the MEG system is installed
- Related to conditions that could happen from time to time (parking garage nearby,)

Once the reasons are understood, we can identify the pattern that the noise makes.

With training data of the different possible noises, it is very possible to train a neural classifier that could identify the noise coming from the different sources and be able to denoise it from the MEG data.

18.5 Independent component analysis

Independent component analysis (ICA) is commonly used to generate what is supposed a set of independent signals from a given set of assumingly correlated signals.

The signals produced by MEG are highly correlated, therefore ICA is suitable to reduce correlation. Given a set of MEG signal X(t), ICA learns a matrix W and the output signals S(t) such that

add latex here: X(t) = W.S(t)

ICA can perform well to identify the noise signals that has a certain long lasting continuous-time pattern, but less efficient when the noise is a single event, happening at irregular periods of time.

Listing 1: Calling ICA withint a Python pipeline

```
projs, raw.info['projs'] = raw.info['projs'], []
ica.fit(raw)
raw.info['projs'] = projs
```

18.6 Frequency Analysis

Fast-oscillating signals means high frequencies, while slow oscillations are low frequencies. In fourier space (signal represented by its Fourier transform) we can see the frequency components constituting the signal. FFT (Fast Fourier Transform) algorithm is commonly to identify the frequency components.

Research showed that signals at different frequencies have different functions at different locations of the brain. In other words, given a region of the brain, signals of frequency 8Hz are responsible of an activity that is much different than signals with frequency 20 Hz

18.7 Brain Source Estimate

When neurons become active, they do so in large groups.

18.8 Code Overview

The code for an example.

Listing 2: This installs dependencies

```
# Install required Meg-pipeline dependencies
import matplotlib as plt
import mne
```

CHAPTER

NINETEEN

PIPELINE NOTEBOOKS

19.1 Resting State Processing Pipeline

Author: Hadi Zaatiti hadi.zaatiti@nyu.edu

In this script we will be processing data generated from the KIT-MEG for a resting state experiment.

The experiment has been conducted using the code in Link Text involving a 5 minutes eyes closed followed by a 5 minutes eyes open of the participant.

The data can be accessed at Data Storage we will be using the *resting_state/sub-01* BIDS dataset

After downloading the data, be familiar with the different files in the *meg-kit* folder:

- The .con are the raw files produced by the KIT machine, has the MEG time series for each channel
- The .con that has *analysis* in their name, have already applied a filter to account for the noisy magnetic field in the MSR, the latter is measured with the magnetometers of the KIT, this file contains the filtered MEG time series for each channel
- The .mrk has the markers position from KIT
- The .fif is produced by applying *KIT2FIFF* command provided by the MNE environment from the filtered .con file and the .mrk files (Check the KIT2FIF tutorial in this documentation)
- We obtained two .fif files, one for eyes closed and one for eyes open

Import mne and set visualisations to show in the notebook

[1]:
```
%matplotlib inline
import matplotlib.pyplot as plt

import mne
```

Load your .fif file for eyes closed

[2]:
```
# Load your FIFF file
raw = mne.io.read_raw_fif(r"C:\Users\hz3752\Box\MEG\Data\resting-state\sub-01\meg-kit\
↪sub-01_01-eyes-closed-raw.fif", verbose=False)
```

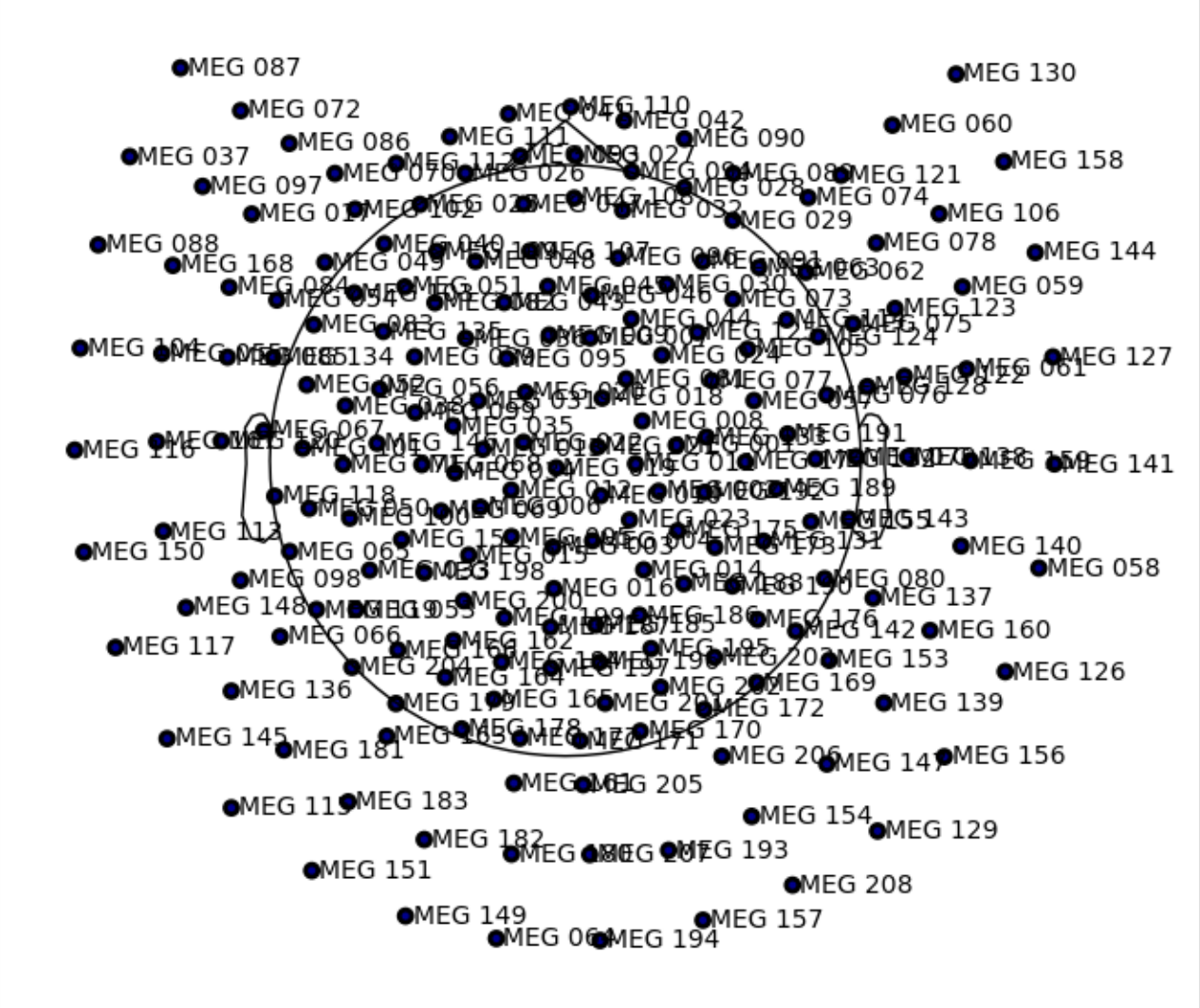
Display the info structure from the data

[3]:
```
print(raw.info)
```

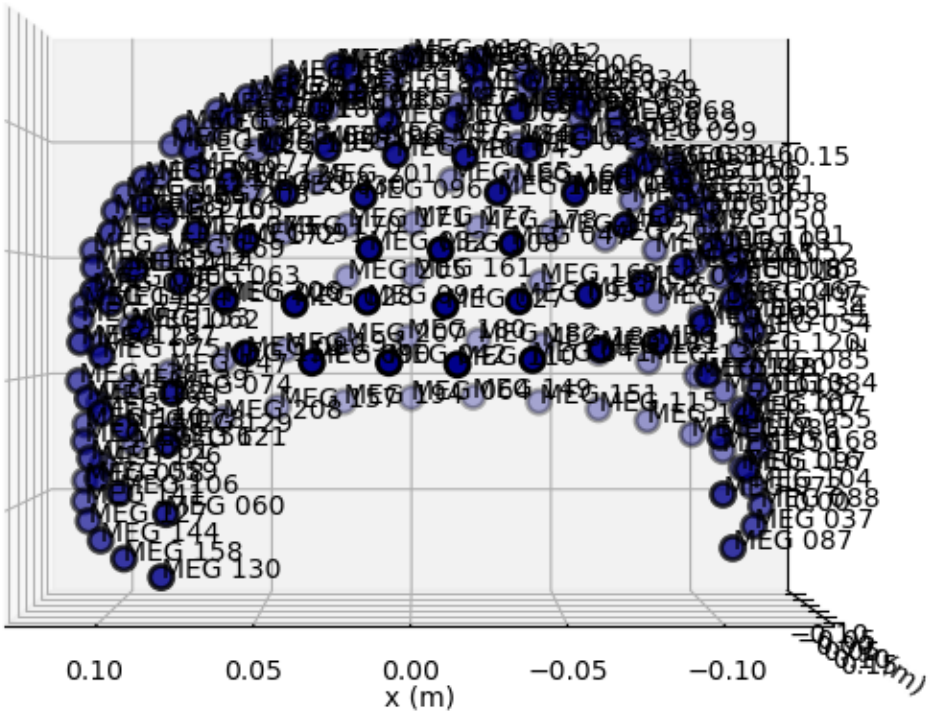
```
<Info | 13 non-empty values
 bads: []
 ch_names: MEG 001, MEG 002, MEG 003, MEG 004, MEG 005, MEG 006, MEG 007, ...
 chs: 207 Magnetometers, 17 Reference Magnetometers, 32 misc, 1 Stimulus
 custom_ref_applied: False
 description: New York University Abu Dhabi/224-channel MEG System (442) ...
 dev_head_t: MEG device -> head transform
 dig: 3459 items (3 Cardinal, 5 HPI, 3451 Extra)
 file_id: 4 items (dict)
 highpass: 0.0 Hz
 kit_system_id: 442 (New York University Abu Dhabi, 2014-)
 lowpass: 500.0 Hz
 meas_date: 2024-04-19 09:05:59 UTC
 meas_id: 4 items (dict)
 nchan: 257
 projs: []
 sfreq: 1000.0 Hz
>
```

Let us plot the sensor layout of the system used in the data acquisition (KIT-MEG)

[4]:
```
# For a 2D topographic plot of the sensor locations
raw.plot_sensors(kind='topomap', show_names=True);
```



[5]:
```
# For a 3D plot, you can also do:
fig = raw.plot_sensors(kind='3d', show_names=True)
```



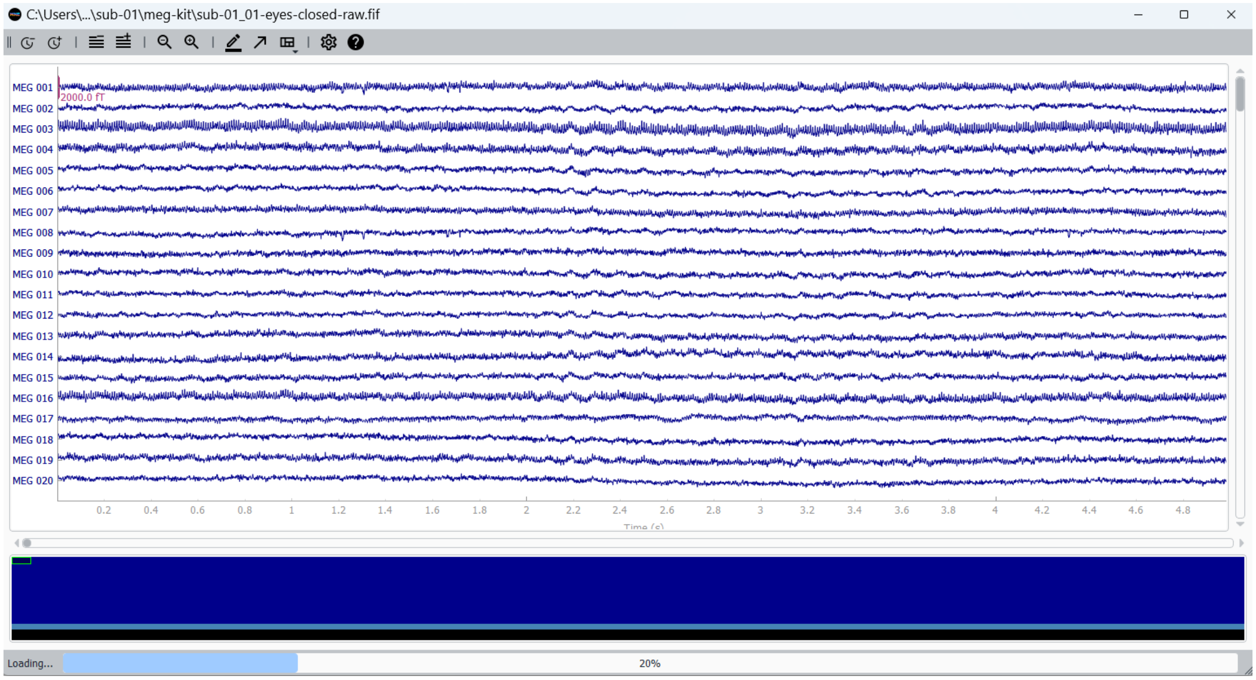
Let us visualise the data and plot the first 5 seconds of the data interactively)

[6]:
```
%matplotlib inline
# Plot the first 5 seconds of the data
raw.plot(start=0, duration=5)
```

```
Using qt as 2D backend.
```

[6]:
```
<mne_qt_browser._pg_figure.MNEQtBrowser at 0x21f2a46fac0>
```

You should see the following interactive window



From the above interactive window, you can mark channels as bad by visual inspection

Notice that in MNE, the channels have different types

1. MEG, for our system a .fif will show `MEG 001` to `MEG 208` are of type `mag`
2. MISC, for our system this will show from `MISC 001` to `MISC 032` are of type `misc`
3. STIM, for our system this will show as `STI 014` are of type `stim`

Let us print the channel types from our raw data.

[7]:
```
print(raw.info.get_channel_types())
```

```
['mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag
↪', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'ref_meg', 'mag', 'mag', 'mag', 'mag', 'mag',
↪ 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag', 'mag',
↪'mag', 'mag', 'mag', 'ref_meg', 'ref_meg', 'ref_meg', 'ref_meg', 'ref_meg', 'ref_meg',
↪'ref_meg', 'ref_meg', 'ref_meg', 'ref_meg', 'ref_meg', 'ref_meg', 'ref_meg', 'ref_meg',
↪ 'ref_meg', 'ref_meg', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc',
↪'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc',
```

(continues on next page)

(continued from previous page)

```
↪ 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc', 'misc
↪', 'misc', 'misc', 'stim']
```

Let us print all the channel names

[8]:
```
print(raw.ch_names)
```

```
['MEG 001', 'MEG 002', 'MEG 003', 'MEG 004', 'MEG 005', 'MEG 006', 'MEG 007', 'MEG 008',
↪'MEG 009', 'MEG 010', 'MEG 011', 'MEG 012', 'MEG 013', 'MEG 014', 'MEG 015', 'MEG 016',
↪ 'MEG 017', 'MEG 018', 'MEG 019', 'MEG 020', 'MEG 021', 'MEG 022', 'MEG 023', 'MEG 024
↪', 'MEG 025', 'MEG 026', 'MEG 027', 'MEG 028', 'MEG 029', 'MEG 030', 'MEG 031', 'MEG␣
↪032', 'MEG 033', 'MEG 034', 'MEG 035', 'MEG 036', 'MEG 037', 'MEG 038', 'MEG 039',
↪'MEG 040', 'MEG 041', 'MEG 042', 'MEG 043', 'MEG 044', 'MEG 045', 'MEG 046', 'MEG 047',
↪ 'MEG 048', 'MEG 049', 'MEG 050', 'MEG 051', 'MEG 052', 'MEG 053', 'MEG 054', 'MEG 055
↪', 'MEG 056', 'MEG 057', 'MEG 058', 'MEG 059', 'MEG 060', 'MEG 061', 'MEG 062', 'MEG␣
↪063', 'MEG 064', 'MEG 065', 'MEG 066', 'MEG 067', 'MEG 068', 'MEG 069', 'MEG 070',
↪'MEG 071', 'MEG 072', 'MEG 073', 'MEG 074', 'MEG 075', 'MEG 076', 'MEG 077', 'MEG 078',
↪ 'MEG 079', 'MEG 080', 'MEG 081', 'MEG 082', 'MEG 083', 'MEG 084', 'MEG 085', 'MEG 086
↪', 'MEG 087', 'MEG 088', 'MEG 089', 'MEG 090', 'MEG 091', 'MEG 092', 'MEG 093', 'MEG␣
↪094', 'MEG 095', 'MEG 096', 'MEG 097', 'MEG 098', 'MEG 099', 'MEG 100', 'MEG 101',
↪'MEG 102', 'MEG 103', 'MEG 104', 'MEG 105', 'MEG 106', 'MEG 107', 'MEG 108', 'MEG 109',
↪ 'MEG 110', 'MEG 111', 'MEG 112', 'MEG 113', 'MEG 114', 'MEG 115', 'MEG 116', 'MEG 117
↪', 'MEG 118', 'MEG 119', 'MEG 120', 'MEG 121', 'MEG 122', 'MEG 123', 'MEG 124', 'MEG␣
↪125', 'MEG 126', 'MEG 127', 'MEG 128', 'MEG 129', 'MEG 130', 'MEG 131', 'MEG 132',
↪'MEG 133', 'MEG 134', 'MEG 135', 'MEG 136', 'MEG 137', 'MEG 138', 'MEG 139', 'MEG 140',
↪ 'MEG 141', 'MEG 142', 'MEG 143', 'MEG 144', 'MEG 145', 'MEG 146', 'MEG 147', 'MEG 148
↪', 'MEG 149', 'MEG 150', 'MEG 151', 'MEG 152', 'MEG 153', 'MEG 154', 'MEG 155', 'MEG␣
↪156', 'MEG 157', 'MEG 158', 'MEG 159', 'MEG 160', 'MEG 161', 'MEG 162', 'MEG 163',
↪'MEG 164', 'MEG 165', 'MEG 166', 'MEG 167', 'MEG 168', 'MEG 169', 'MEG 170', 'MEG 171',
↪ 'MEG 172', 'MEG 173', 'MEG 174', 'MEG 175', 'MEG 176', 'MEG 177', 'MEG 178', 'MEG 179
↪', 'MEG 180', 'MEG 181', 'MEG 182', 'MEG 183', 'MEG 184', 'MEG 185', 'MEG 186', 'MEG␣
↪187', 'MEG 188', 'MEG 189', 'MEG 190', 'MEG 191', 'MEG 192', 'MEG 193', 'MEG 194',
↪'MEG 195', 'MEG 196', 'MEG 197', 'MEG 198', 'MEG 199', 'MEG 200', 'MEG 201', 'MEG 202',
↪ 'MEG 203', 'MEG 204', 'MEG 205', 'MEG 206', 'MEG 207', 'MEG 208', 'MEG 209', 'MEG 210
↪', 'MEG 211', 'MEG 212', 'MEG 213', 'MEG 214', 'MEG 215', 'MEG 216', 'MEG 217', 'MEG␣
↪218', 'MEG 219', 'MEG 220', 'MEG 221', 'MEG 222', 'MEG 223', 'MEG 224', 'MISC 001',
↪'MISC 002', 'MISC 003', 'MISC 004', 'MISC 005', 'MISC 006', 'MISC 007', 'MISC 008',
↪'MISC 009', 'MISC 010', 'MISC 011', 'MISC 012', 'MISC 013', 'MISC 014', 'MISC 015',
↪'MISC 016', 'MISC 017', 'MISC 018', 'MISC 019', 'MISC 020', 'MISC 021', 'MISC 022',
↪'MISC 023', 'MISC 024', 'MISC 025', 'MISC 026', 'MISC 027', 'MISC 028', 'MISC 029',
↪'MISC 030', 'MISC 031', 'MISC 032', 'STI 014']
```

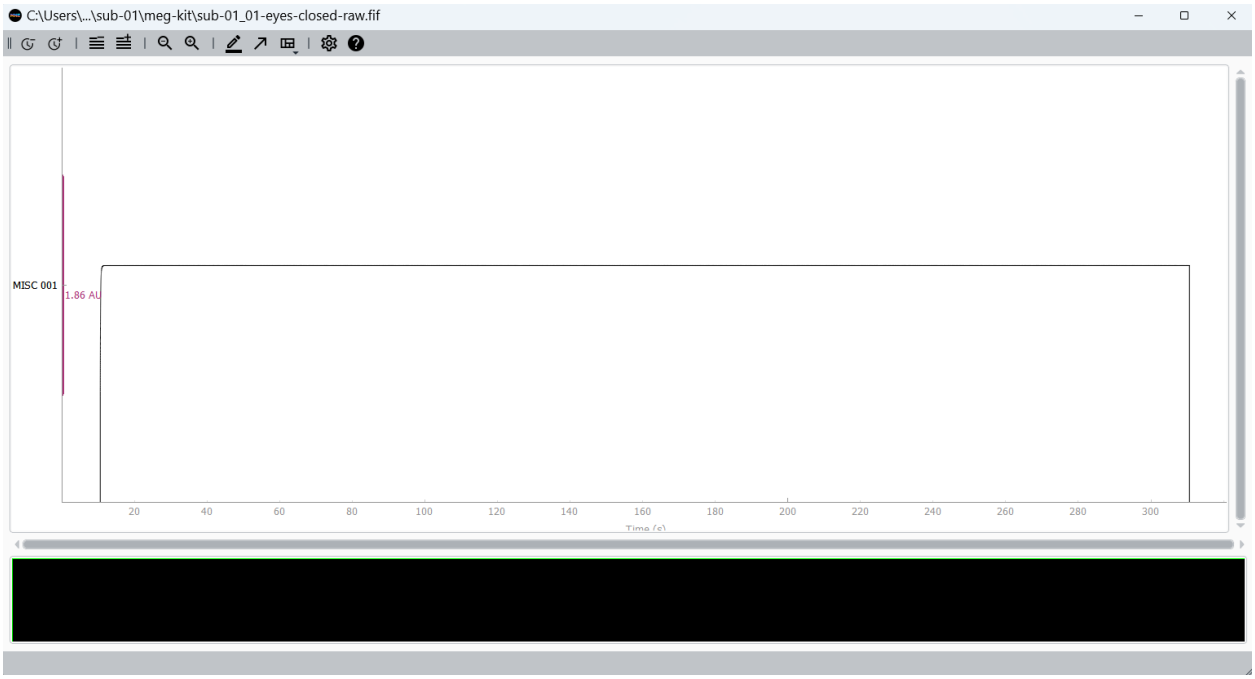
Let us plot the `MISC 001` channel that contains the trigger, it is a good practice to copy the raw data before picking a specific channel since the picking operation changes

[9]:
```
%matplotlib inline
channel_name = 'MISC 001'
raw_picked = raw.copy().pick_channels([channel_name])
scalings = {'misc':0.1}

raw_picked.plot(scalings = scalings, duration=315, start=0, n_channels=1)
```

```
NOTE: pick_channels() is a legacy function. New code should use inst.pick(...).
```

[9]:
```
<mne_qt_browser._pg_figure.MNEQtBrowser at 0x21f2bdf3ac0>
```

You should see the trigger channel going from 0 to 1 over a period of 5 minutes, corresponding to the eyes closed period as in the following image. The AU unit on the Y-axis correspond to arbitrary unit.



Note the times of beginning and end of the trigger as respectively 11 seconds and 310 seconds, let's do FFT on the data for this time

[10]:
```
croped_data = raw.copy()
```

[11]:
```
croped_data.crop(11, 310)
```

[11]:
```
<Raw | sub-01_01-eyes-closed-raw.fif, 257 x 299001 (299.0 s), ~1.4 MB, data not loaded>
```

[12]:
```
%matplotlib inline
croped_data.plot(start=0, duration =5)
```

[12]:
```
<mne_qt_browser._pg_figure.MNEQtBrowser at 0x21f2cf97640>
```

[13]:
```
data, times = raw[:]
wsize=256
fourier_transform_data = mne.time_frequency.stft(data, wsize=wsize)
```

```
Number of frequencies: 129
Number of time steps: 2516
```

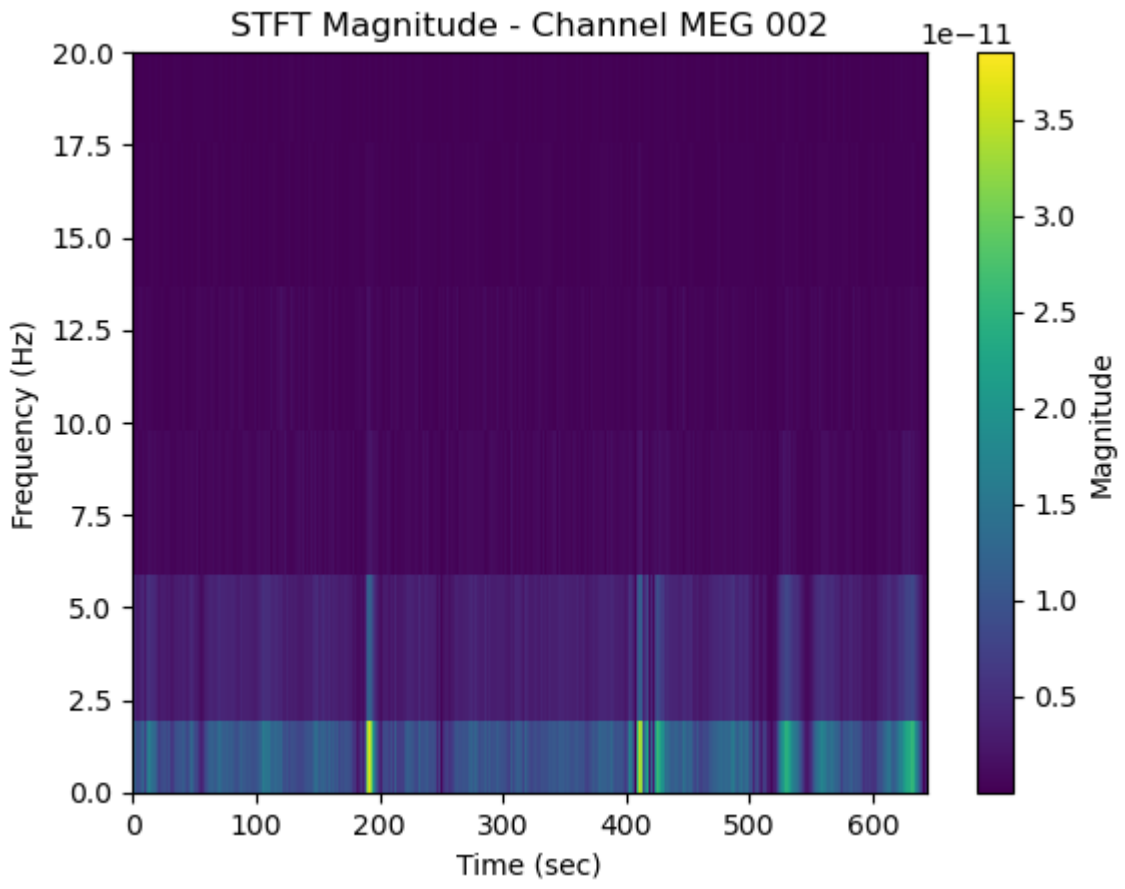
[14]:
```
%matplotlib inline
# Select the channel index you're interested in
channel_index = 1
import numpy as np
# Compute magnitude of STFT results
magnitude = np.abs(fourier_transform_data[channel_index])
```

(continues on next page)

(continued from previous page)

```
# Prepare frequencies and time vectors for plotting
sfreq = raw.info['sfreq']
print('Sampling frequency', sfreq)
frequencies = np.linspace(0, sfreq / 2, magnitude.shape[0], endpoint=True)
t_secs = np.arange(magnitude.shape[1]) * (wsize / sfreq)

# Plotting
plt.figure()
plt.pcolormesh(t_secs, frequencies, magnitude, shading='auto')
plt.title(f'STFT Magnitude - Channel {raw.info["ch_names"][channel_index]}')
plt.ylabel('Frequency (Hz)')
plt.ylim(0, 20)
plt.xlabel('Time (sec)')
plt.colorbar(label='Magnitude')
plt.show()
```

```
Sampling frequency 1000.0
```

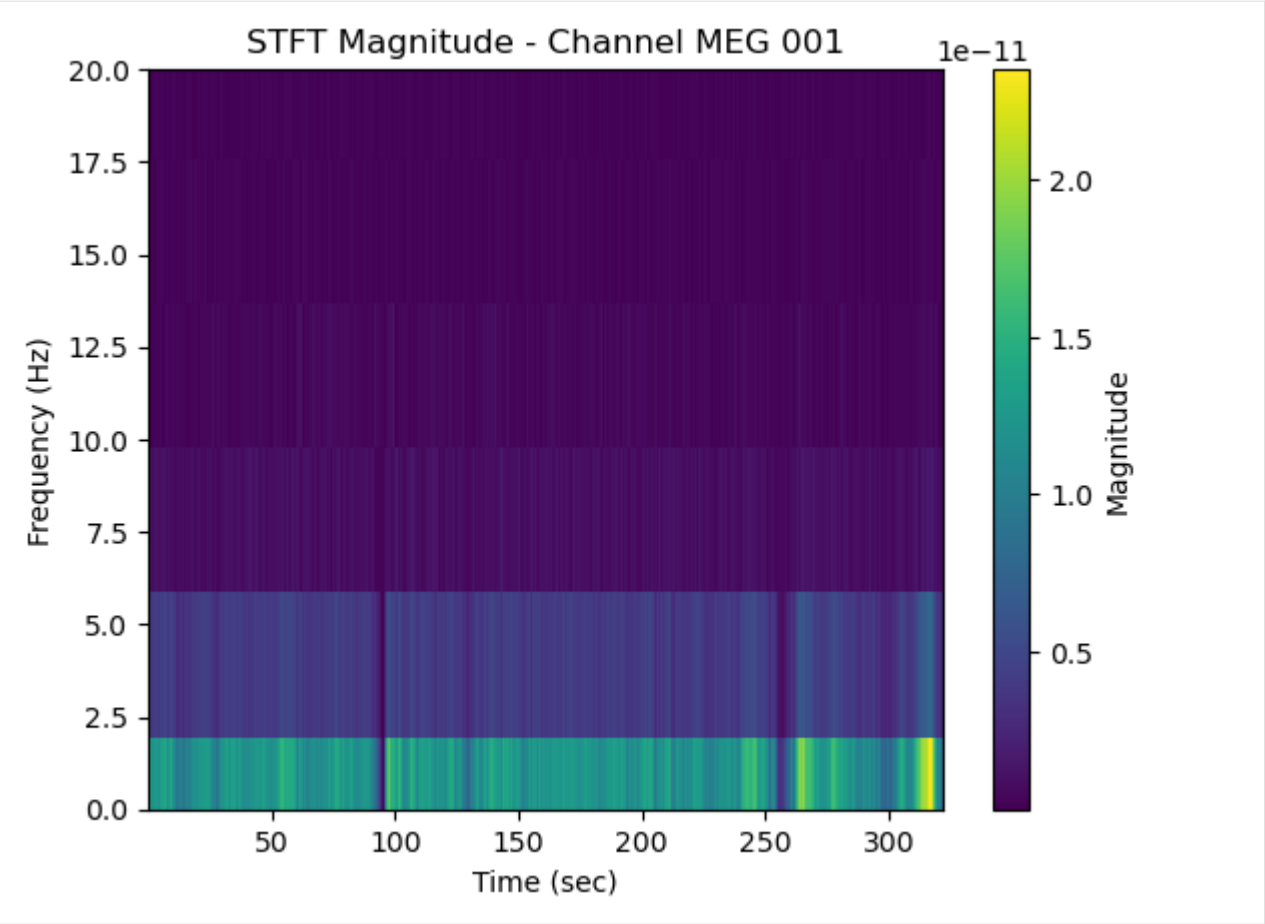


[15]:
```
import numpy as np
import matplotlib.pyplot as plt

# Assuming 'stft_result' from previous STFT calculation and 'raw' object are available
```

(continues on next page)

(continued from previous page)

```
# Parameters
channel_index = 0
sfreq = raw.info['sfreq']
wsize = 256  # Window size used in STFT
hop_size = int(wsize / 2)  # 50% overlap

# Compute magnitude of STFT results
magnitude = np.abs(fourier_transform_data[channel_index])

# Prepare frequencies and time vectors for plotting
frequencies = np.linspace(0, sfreq / 2, magnitude.shape[0], endpoint=True)

# Calculate correct start times for each window and then add half the window size to␣
↪center it
start_times = np.arange(0, magnitude.shape[1] * hop_size, hop_size)
t_secs = (start_times + wsize / 2) / sfreq

# Limit frequency data to 100 Hz
freq_limit = 20
freq_indices = frequencies <= freq_limit
frequencies = frequencies[freq_indices]
magnitude = magnitude[freq_indices, :]

# Plotting
plt.figure()
plt.pcolormesh(t_secs, frequencies, magnitude, shading='auto')
plt.title(f'STFT Magnitude - Channel {raw.info["ch_names"][channel_index]}')
plt.ylabel('Frequency (Hz)')
plt.xlabel('Time (sec)')
plt.ylim(0, freq_limit)  # Set y-axis limit to 100 Hz
plt.colorbar(label='Magnitude')
plt.show()
```



[ ]:

19.2 Data preparation and coregistration after data acquisition

Author: Hadi Zaatiti hadi.zaatiti@nyu.edu

In this notebook, we shall do the initial data processing of the raw data generated by an MEG experiment. Prepare the following files to go through this notebook: Prerequisits: Obtained from the laser scan:

- Headscan basic surface .txt
- Headscan points .txt

Obtained from the KIT-MEG machine:

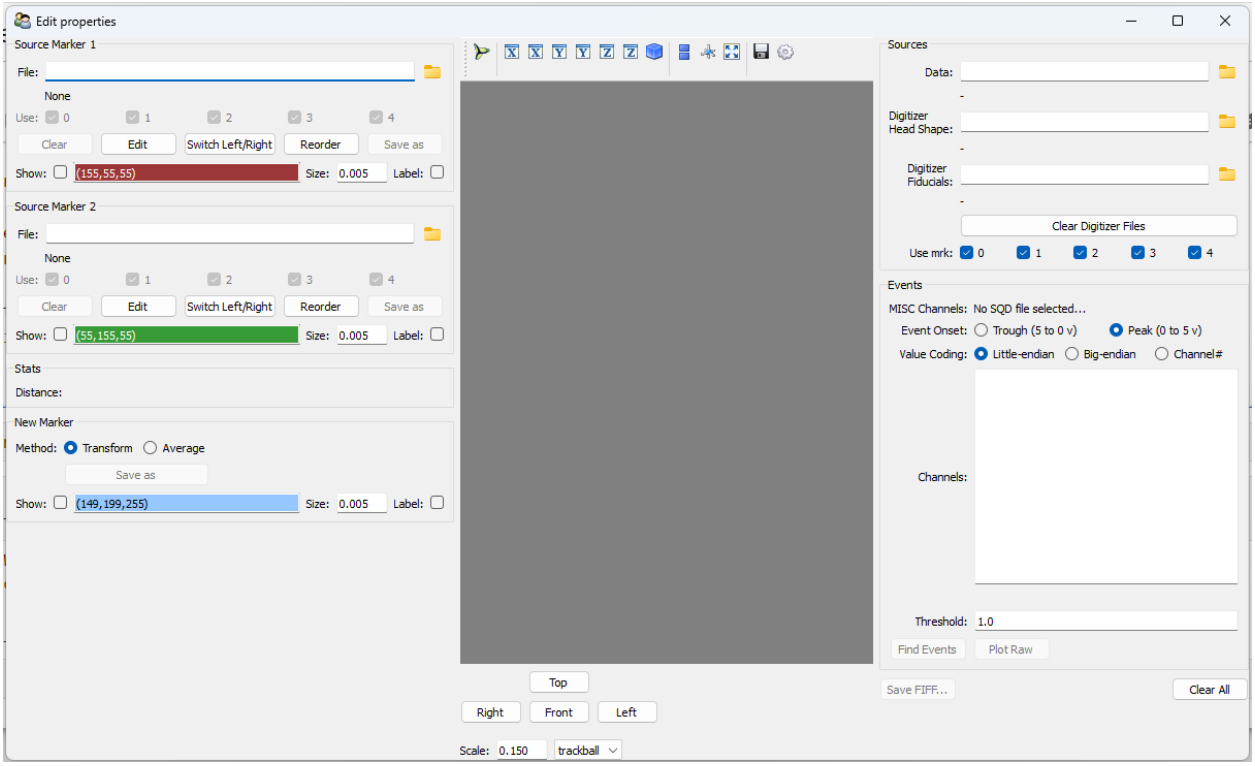
- Marker measurement (x2) .mrk
- MEG recording con.

Environment: Have MNE with all dependencies installedc From these files we will n .create .fif files, the base file format for MNE

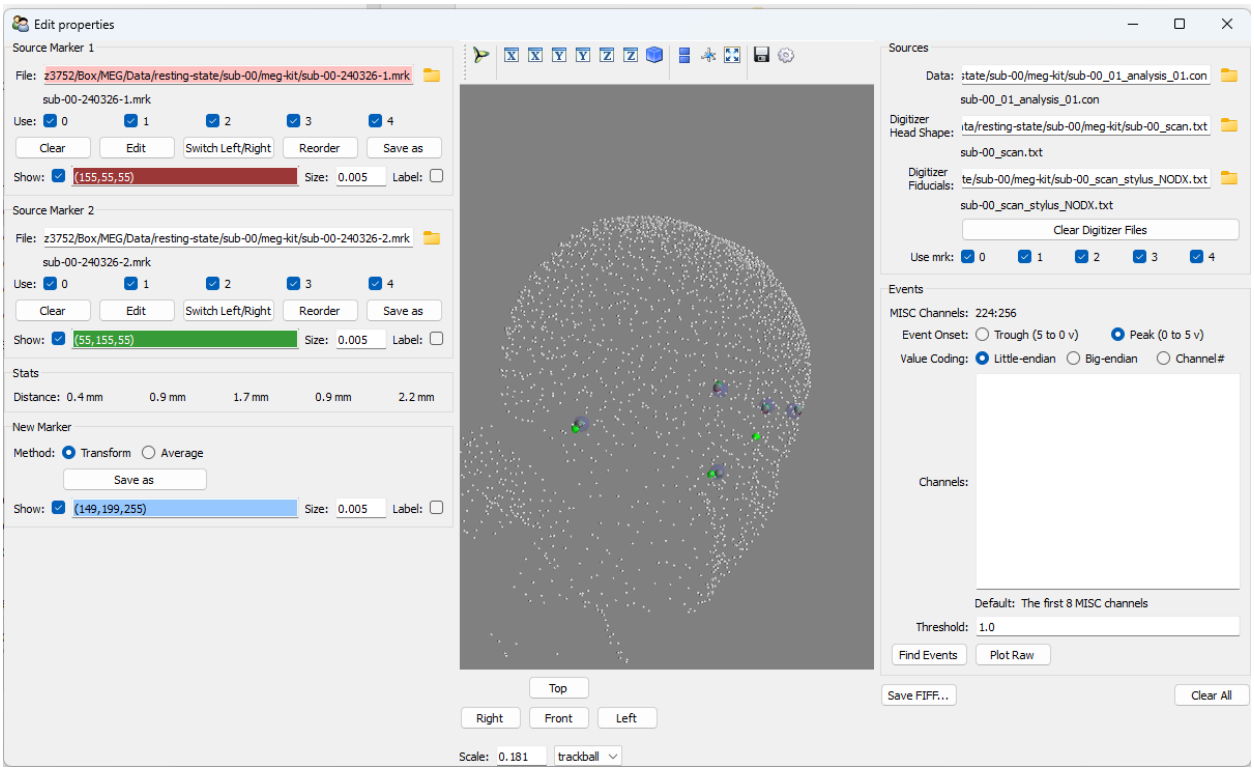
In a terminal, run the following command, the following is an example on Windows CMD terminal.

[1]:
```
%%cmd
mne kit2fiff
```

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

(mne-1.6.1_0) C:\Users\hz3752\PycharmProjects\meg-pipeline\docs\source\4-pipeline\
↪notebooks>mne kit2fiff
********************************************************************************
WARNING: Imported VTK version (9.3) does not match the one used
         to build the TVTK classes (9.2). This may cause problems.
         Please rebuild TVTK.
********************************************************************************


(mne-1.6.1_0) C:\Users\hz3752\PycharmProjects\meg-pipeline\docs\source\4-pipeline\
↪notebooks>
```



Place the two .mrk in the Source marker 1 and Source marker 2, then in Sources, place the .con in Data, the head scan .txt in Digitizer head shape and the head scan points .txt in Digitizer Fiducials



You should now see the head scan, markers and scan points. Press the SAVE FIFF to save all the data within a .fiff.

Let us now load the .fiff within python and check its structure of the .fif

[2]:
```
%matplotlib inline
import mne
import matplotlib.pyplot as plt

raw = mne.io.read_raw_fif(r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\
↪sub-01_01-eyes-closed-raw.fif')


raw.plot(duration = 15)
plt.show()
print(raw.info.get_channel_types())

plt.show()
```

```
Opening raw data file C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\sub-01_
↪01-eyes-closed-raw.fif...
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[2], line 5
      2 import mne
      3 import matplotlib.pyplot as plt
----> 5 raw =␣
↪mne.io.read_raw_fif(r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\sub-01_01-eyes-close
      8 raw.plot(duration = 15)
      9 plt.show()
```

(continues on next page)

(continued from previous page)

```
File C:\ProgramData\mne-python\1.6.1_0\Lib\site-packages\mne\io\fiff\raw.py:543, in read_
↪raw_fif(fname, allow_maxshield, preload, on_split_missing, verbose)
    502 @fill_doc
    503 def read_raw_fif(
    504     fname, allow_maxshield=False, preload=False, on_split_missing="raise",␣
↪verbose=None
    505 ):
    506     """Reader function for Raw FIF data.
    507
    508     Parameters
   (...)
    541     are updated accordingly.
    542     """
--> 543     return Raw(
    544         fname=fname,
    545         allow_maxshield=allow_maxshield,
    546         preload=preload,
    547         verbose=verbose,
    548         on_split_missing=on_split_missing,
    549     )

File <decorator-gen-179>:12, in __init__(self, fname, allow_maxshield, preload, on_split_
↪missing, verbose)

File C:\ProgramData\mne-python\1.6.1_0\Lib\site-packages\mne\io\fiff\raw.py:105, in Raw._
↪_init__(self, fname, allow_maxshield, preload, on_split_missing, verbose)
    103 next_fname = fname
    104 while next_fname is not None:
--> 105     raw, next_fname, buffer_size_sec = self._read_raw_file(
    106         next_fname, allow_maxshield, preload, do_check_ext
    107     )
    108     do_check_ext = False
    109     raws.append(raw)

File <decorator-gen-180>:12, in _read_raw_file(self, fname, allow_maxshield, preload, do_
↪check_ext, verbose)

File C:\ProgramData\mne-python\1.6.1_0\Lib\site-packages\mne\io\fiff\raw.py:187, in Raw._
↪read_raw_file(self, fname, allow_maxshield, preload, do_check_ext, verbose)
    185     check_fname(fname, "raw", endings)
    186 # filename
--> 187 fname = str(_check_fname(fname, "read", True, "fname"))
    188 ext = os.path.splitext(fname)[1].lower()
    189 whole_file = preload if ".gz" in ext else False

File <decorator-gen-0>:12, in _check_fname(fname, overwrite, must_exist, name, need_dir,␣
↪verbose)

File C:\ProgramData\mne-python\1.6.1_0\Lib\site-packages\mne\utils\check.py:263, in _
↪check_fname(fname, overwrite, must_exist, name, need_dir, verbose)
    261             raise PermissionError(f"{name} does not have read permissions:
↪{fname}")
```

(continues on next page)

(continued from previous page)

```
    262 elif must_exist:
--> 263     raise FileNotFoundError(f'{name} does not exist: "{fname}"')
    265 return fname

FileNotFoundError: fname does not exist: "C:\Users\hz3752\PycharmProjects\mne_bids_
↪pipeline\data\meg\sub-01_01-eyes-closed-raw.fif"
```

19.3 Coregistration after KIT2FIFF

Coregistration involves aligning the MEG sensor axis, with the axis of the MRI headscan.

Different transformations can be applied during this coregistration:

- Scaling: making the head bigger or smaller to adjust to the volume of the system
- linear transformation: this involves translating the head scan to match the center of the MEG helmet
- rotation

Non-linear transformations are not used in MEG coregistration.

19.3.1 You have MRI anatomical data of the participant

In this situation, the participant has had his head scanned in the MRI. Get the MRI ID of the participant and cross check it in the .csv file on NYU BOX, since the ID of the participants in MEG is different than the one in MRI.

T1w scans are needed and provided as input to the HPC brainsegmentation freesurfer pipeline. The MRI Lab team will provide you with the complete segmentation output folder, however this will not contain the BEM/FEM model needed for source localization.

At this point you can follow the tutorial on computing the BEM prior to proceeding.

We are now ready to coregister the participant head scan with the MEG sensors positions.

You can run the following script to launch a mne coregistration GUI

[ ]:
```
import mne
from PyQt5.QtWidgets import QApplication
import sys

app = QApplication.instance()  # checks if QApplication already exists
if not app:  # create QApplication if it doesnt exist
    app = QApplication(sys.argv)

mne.gui.coregistration(
    inst=r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\Sub-0037\sub-01_01-
↪eyes-closed-raw.fif',
    subject='Sub-0037',
    subjects_dir=r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w',  # contains a sub-folder for subject
    head_high_res=True,
)

app.exec_()
```

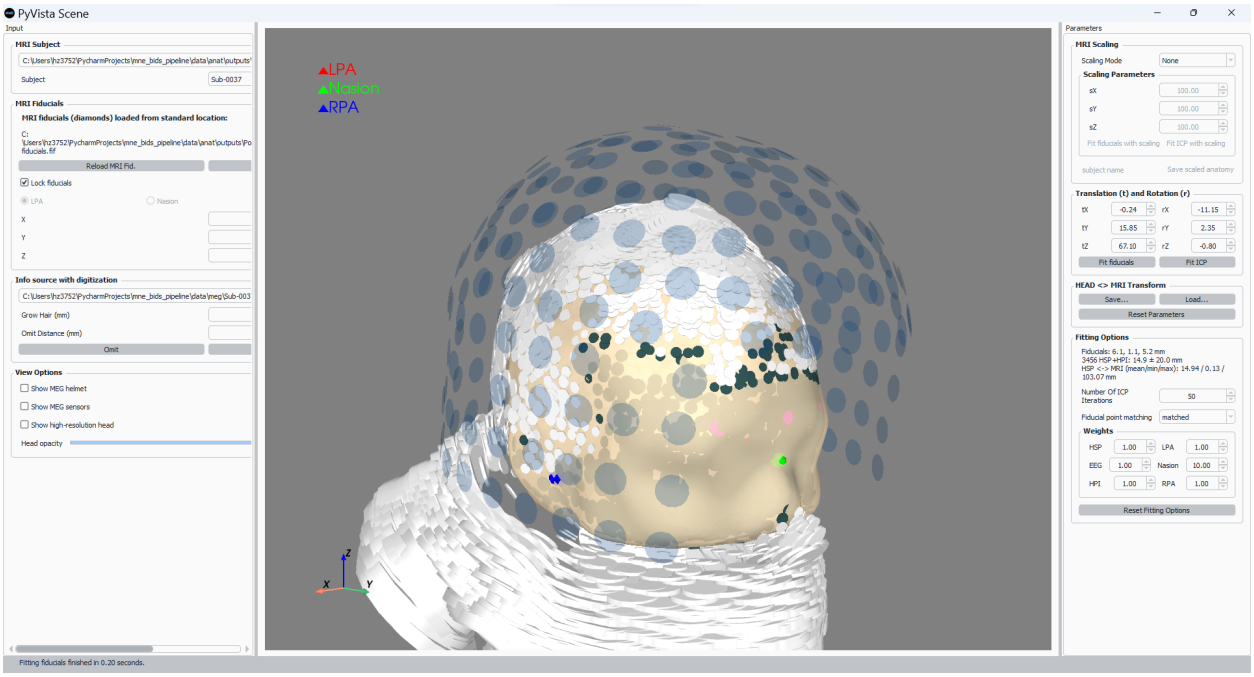
The MNE Coreg GUI is open at this point, if it is the first time you open the GUI you will get the message that there are no fiducials that have been found in the `bem` directory. In such case you will now need to set the fiducials
There is basically three points that needs to be set:

- the Nasion
- the LPA: the left pre-aucular
- the RPA: the right pre-aucular

Set each one of them and then save the fiducials. A file called `sub-id-fiducials.fif` is then created in the `bem` directory At this point you will see the digitized headscan from the laser scan, with the MRI headshape.

Apply the Fit fiducials and then the Fit ICp algorithms to match respectivaly the HPI coils markers and headscan with the three points previously.

Perform any additional translation, rotation that can seem needed to have the best alignment.



19.3.2 You do not have the MRI anatomical data of the participant

(Add part how to use fsaverage, to match a generic headshape to the positions of the HPI coil markers)

19.4 Conclusion

We now have the necesary .fif files to perform the source localization. You can now proceed to the source localization notebook.

[ ]:

19.5 Source estimation and localization

Author: Hadi Zaatiti hadi.zaatiti@nyu.edu

A Boundary Element Model (BEM) is a computational model used primarily in the field of neuroimaging, especially in magnetoencephalography (MEG) and electroencephalography (EEG), to help solve the forward problem, which involves calculating the electric or magnetic fields generated by neuronal activity in the brain at the sensors located on the scalp.

The NYUAD MRI lab will provide the MRI T1w scans of the participant, with segmentation of the different parts of the brain and different volumetry and other `freesurfer` analysis.

An example of such data is available under NYU BOX, https://nyu.box.com/v/meg-datafiles Access the directory: `Box\MEG\Data\resting-state\sub-01\anat`

Requirements: Having ran `mne kit2fiff` and `mne coreg` as explained previously

19.5.1 Generating the BEM

We will now compute and plot the BEM model for the participant. Set the freesurfer SUBJECTS_DIR to `Box\MEG\Data\resting-state\sub-01\anat\outputs\PostFreeSurfer\T1w\Sub-0037` where `Sub-0037` is the MRI ID of the subject.

Note that the ID in MEG for a participant are different than the ID in MRI.

Generate the head surfaces files using the command

mkheadsurf -subjid sub-0037

You should see the following output

```
INFO: log file is /home/USERNAME/freesurferproject/subjects//sub-0037/scripts/mkheadsurf.
↪log
---------------------------------
Wed May  8 15:15:41 +04 2024
/home/USERNAME/freesurferproject/subjects/sub-0037
mri_seghead --invol /home/USERNAME/freesurferproject/subjects//sub-0037/mri/T1.mgz --
↪outvol /home/USERNAME/freesurferproject/subjects//sub-0037/mri/seghead.mgz --fill 1 --
↪thresh1 20 --thresh2 20 --nhitsmin 2 --rescale --fill-holes-islands
---------------------------------
input volume:  /home/USERNAME/freesurferproject/subjects//sub-0037/mri/T1.mgz
output volume: /home/USERNAME/freesurferproject/subjects//sub-0037/mri/seghead.mgz
threshold1:    20
threshold2:    20
nhitsmin:      2
fill value:    1
Loading input volume
Changing type, rescale = 1, fhi=0.999
Filling Columns
Filling Rows
Filling Slices
Merging
nhits = 2546749
Removing islands
```

(continues on next page)

(continued from previous page)

```
MRIremoveVolumeIslands() thresh=0.5, nKeep=1, nclusters = 35
Removing Volume Holes
MRIremoveVolumeHoles() thresh=0.5, nKeep=1
MRIremoveVolumeIslands() thresh=0.5, nKeep=1, nclusters = 1
Filling axial Slice Holes
ostr LIA
slicedir 6 slicediruse 2
slicedim 256 1 256
MRIremoveSliceHodes() removed 35899 voxels
Filling sag Slice Holes
ostr LIA
slicedir 4 slicediruse 1
slicedim 1 256 256
MRIremoveSliceHodes() removed 3538 voxels
Filling cor Slice Holes
slicedir 2 slicediruse 2
slicedim 256 1 256
MRIremoveSliceHodes() removed 0 voxels
Counting
N Head Voxels = 2585413
N Back Voxels = 14191803
Avg. Back Intensity = 1.097717
Max. Back Intensity = 205.000000
Writing output
Done

constructing final surface...
(surface with 291644 faces and 145822 vertices)...done
computing the maximum edge length...1.414214 mm
reversing orientation of faces...
checking orientation of surface...
0.000 % of the vertices (0 vertices) exhibit an orientation change

counting number of connected components...
   145822 voxel in cpt #1: X=0 [v=145822,e=437466,f=291644] located at (1.178080, 5.
↪080982, -25.954636)
For the whole surface: X=0 [v=145822,e=437466,f=291644]
One single component has been found
nothing to do
writing out surface...done
-------------------------------
Wed May  8 15:15:59 +04 2024
/home/USERNAME/freesurferproject/subjects/sub-0037
mris_smooth -n 10 -nw /home/USERNAME/freesurferproject/subjects/sub-0037/surf/lh.seghead␣
↪/home/USERNAME/freesurferproject/subjects/sub-0037/surf/lh.seghead
-------------------------------
smoothing for 10 iterations
smoothing surface tessellation for 10 iterations...
smoothing complete - recomputing first and second fundamental forms...
Started at: Wed May 8 15:15:41 +04 2024
Ended   at: Wed May  8 15:16:01 +04 2024
mkheadsurf done
```

The command will generate the head segmentation file `mri\seghead.mgz`

We will now generate the boundary for the brain, the inner skull and outer skull and skin

`mne watershed_bem --subject sub-0037`

You should the following output

```
Running mri_watershed for BEM segmentation with the following parameters:

Results dir = /home/USERNAME/freesurferproject/subjects/sub-0037/bem/watershed
Command = mri_watershed -useSRAS -surf /home/USERNAME/freesurferproject/subjects/sub-
↪0037/bem/watershed/sub-0037 /home/USERNAME/freesurferproject/subjects/sub-0037/mri/T1.
↪mgz /home/USERNAME/freesurferproject/subjects/sub-0037/bem/watershed/ws

Running subprocess: mri_watershed -useSRAS -surf /home/USERNAME/freesurferproject/
↪subjects/sub-0037/bem/watershed/sub-0037 /home/USERNAME/freesurferproject/subjects/sub-
↪0037/mri/T1.mgz /home/USERNAME/freesurferproject/subjects/sub-0037/bem/watershed/ws

Mode:          use surfaceRAS to save surface vertex positions
Mode:          Saving out BEM surfaces

*******************************************************
The input file is /home/USERNAME/freesurferproject/subjects/sub-0037/mri/T1.mgz
The output file is /home/USERNAME/freesurferproject/subjects/sub-0037/bem/watershed/ws

************************WATERSHED*************************
Sorting...
      T1-weighted MRI image
      modification of the preflooding height to 15 percent
      Count how many 110 voxels are present :  263997

      Find the largest 110-component...
             heap usage = 468888 Kbytes.
             removing small segments (less than 1 percent of maxarea).done
      And identify it as the main brain basin...done
      Main component: 244657 voxels
      first estimation of the COG coord: x=125 y=137 z=127 r=70
      first estimation of the main basin volume: 1464330 voxels
      global maximum in x=113, y=115, z=93, Imax=255
      CSF=15, WM_intensity=110, WM_VARIANCE=5
      WM_MIN=110, WM_HALF_MIN=110, WM_HALF_MAX=110, WM_MAX=110
      preflooding height equal to 15 percent
done.
Analyze...

      main basin size= 1453495 voxels, voxel volume =1.000
                     = 1453495 mmm3 = 1453.495 cm3
done.
PostAnalyze...
      ***** 0 basin(s) merged in 1 iteration(s)
      ***** 0 voxel(s) added to the main basin
done.
```

(continues on next page)

(continued from previous page)

```
***************TEMPLATE DEFORMATION***************

      second estimation of the COG coord: x=127,y=140, z=126, r=9052 iterations
^^^^^^^^ couldn't find WM with original limits - expanding ^^^^^^

   GLOBAL      CSF_MIN=0, CSF_intensity=5, CSF_MAX=19 , nb = 43452
 Problem with the least square interpolation in GM_MIN calculation.

                     CSF_MAX  TRANSITION  GM_MIN  GM
    GLOBAL
  before analyzing :    19,      37,        59,   76
  after  analyzing :    19,      51,        59,   57
      mri_strip_skull: done peeling brain
      highly tesselated surface with 10242 vertices
      matching...66 iterations

*********************VALIDATION*********************
curvature mean = -0.014, std = 0.011
curvature mean = 67.170, std = 7.285

No Rigid alignment: -atlas Mode Off (basic atlas / no registration)
      before rotation: sse = 5.20, sigma = 8.02
      after  rotation: sse = 5.20, sigma = 8.02
Localization of inacurate regions: Erosion-Dilation steps
      the sse mean is  5.43, its var is  7.01
      before Erosion-Dilatation  1.89% of inacurate vertices
      after  Erosion-Dilatation  0.00% of inacurate vertices
      Validation of the shape of the surface done.
Scaling of atlas fields onto current surface fields

********FINAL ITERATIVE TEMPLATE DEFORMATION********
Compute Local values csf/gray
Fine Segmentation...41 iterations


      mri_strip_skull: done peeling brain

Brain Size = 1454766 voxels, voxel volume = 1.000 mm3
           = 1454766 mmm3 = 1454.766 cm3

      outer skin surface matching...119 iterations

******************************
Saving /home/USERNAME/freesurferproject/subjects/sub-0037/bem/watershed/ws
done
mri_watershed done
error: unknown file type for file (/home/USERNAME/freesurferproject/subjects/sub-0037/
↪bem/watershed/ws)
Overwriting existing file.
Overwriting existing file.
Overwriting existing file.
```

(continues on next page)

(continued from previous page)

```
Overwriting existing file.
Symbolic links to .surf files created in bem folder

Thank you for waiting.
The BEM triangulations for this subject are now available at:
/home/USERNAME/freesurferproject/subjects/sub-0037/bem.
outer skin  CM is   1.20  -0.97 -16.05 mm
Surfaces passed the basic topology checks.
Created /home/USERNAME/freesurferproject/subjects/sub-0037/bem/sub-0037-head.fif

Complete.
```

You should see the following files in your subject directory - the brain boundary `bem\brain.surf` - the inner skull boundary `bem\inner_skull.surf` - the outer skull boundary `bem\outer_skull.surf` - the outer skin boundary `bem\outer_skin.surf`

A `sub-0037-head.fif` file should be generated aswell in the `bem` folder. Note that for MEG, the inner skull boundary would be enough to do source localization and estimation. However, for EEG 3 layers (inner skull, outer skull, and skin) are typically used. Let us now plot the boundaries that we generated using MNE.

19.5.2 Plotting the BEM for visual inspection

Let us now plot the boundaries that we generated using MNE

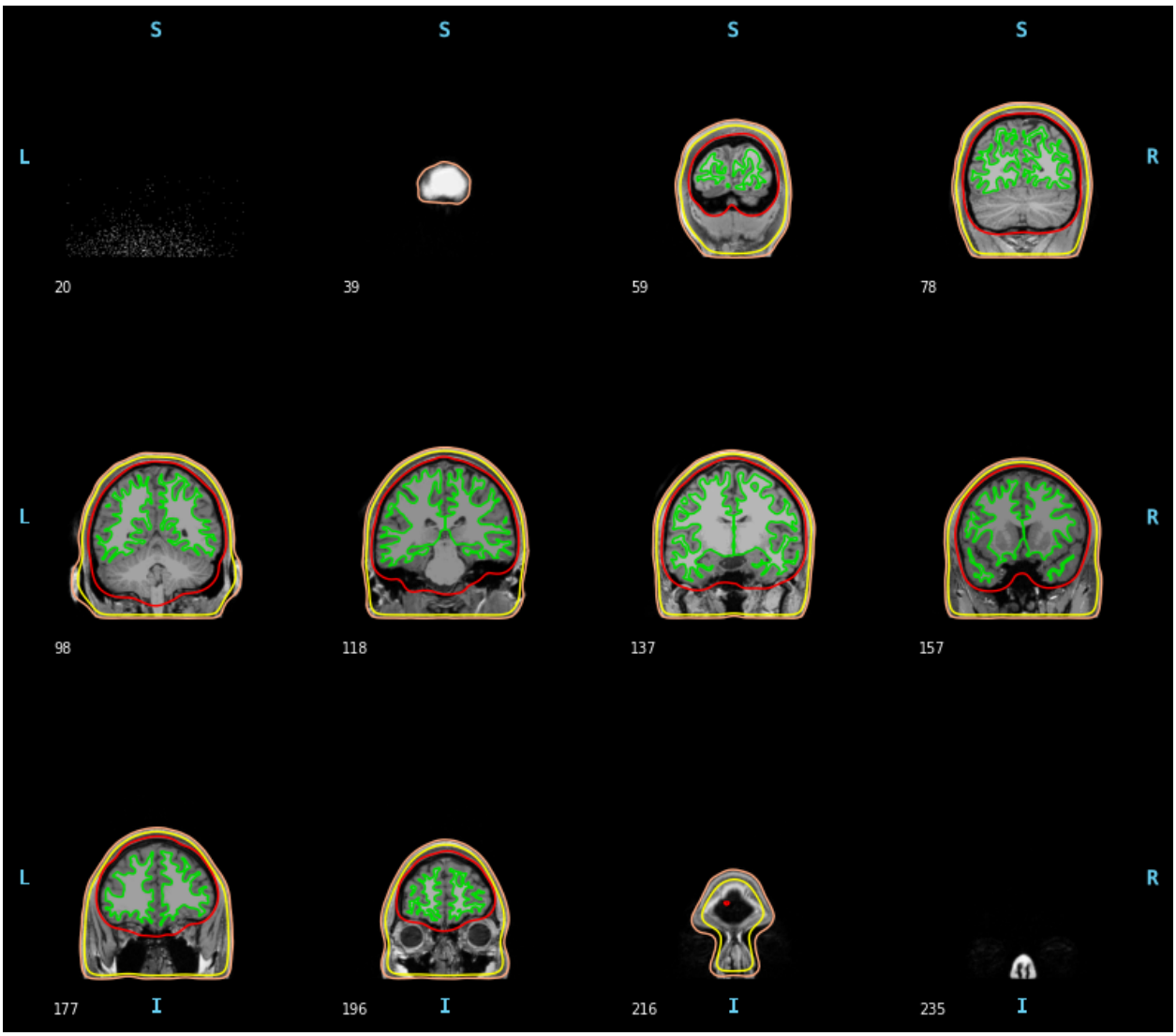
[1]:
```
%matplotlib inline
import mne
```

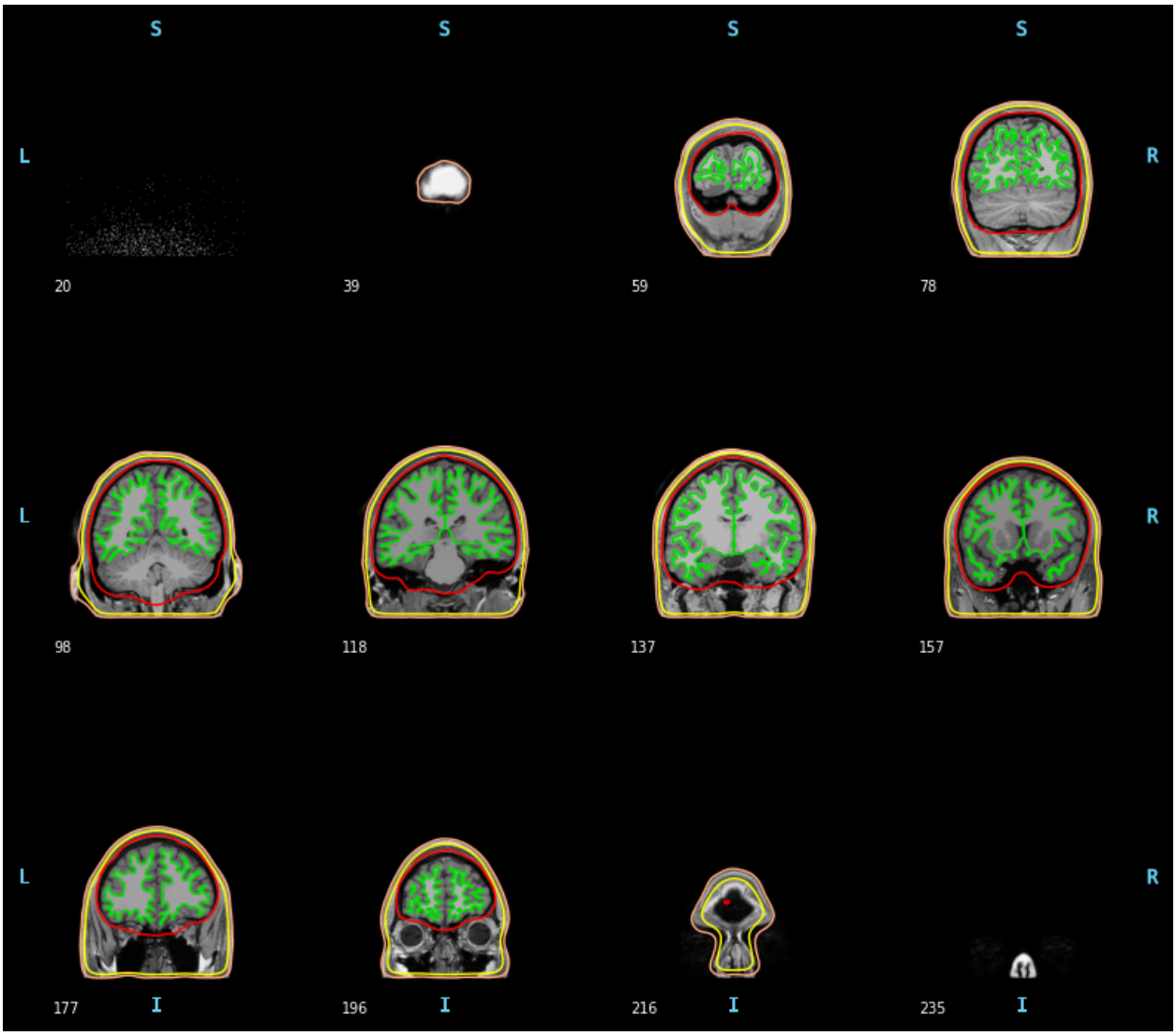
[2]:
```
fif_path = r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w\Sub-0037\bem\sub-0037-head.fif'

# Load the BEM surfaces from the generated .fif file
bem_surfaces = mne.read_bem_surfaces(fif_path)

# Plot the BEM surfaces
mne.viz.plot_bem(subject='sub-0037',
                 subjects_dir=r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\
↪anat\outputs\PostFreeSurfer\T1w',
                 brain_surfaces='white',
                 src=None,
                 orientation='coronal')
```

```
    1 BEM surfaces found
    Reading a surface...
[done]
    1 BEM surfaces read
Using surface: C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w\sub-0037\bem\inner_skull.surf
Using surface: C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w\sub-0037\bem\outer_skull.surf
Using surface: C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w\sub-0037\bem\outer_skin.surf
```

```
C:\ProgramData\mne-python\1.6.1_0\Lib\site-packages\mne\viz\utils.py:165: UserWarning:␣
↪FigureCanvasAgg is non-interactive, and thus cannot be shown
  (fig or plt).show(**kwargs)
```

[2]:





19.6 Source localization and estimation

Generating source space, refers to the process of creating a model of where in the brain the magnetic fields are being generated from. This model is essential for solving the inverse problem, which involves estimating the neuronal activity that causes the measured magnetic fields on the scalp.

[3]:
```
import mne
subject='sub-0037'
subjects_dir=r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w'

source_space = mne.setup_source_space(subject,spacing='ico4',subjects_dir=subjects_dir)

source_space.save(subjects_dir+'/%s/bem/%s-ico4-src.fif' %(subject,subject),␣
↪overwrite=True)
```

```
Setting up the source space with the following parameters:

SUBJECTS_DIR = C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w
Subject      = sub-0037
Surface      = white
Icosahedron subdivision grade 4

>>> 1. Creating the source space...

Doing the icosahedral vertex picking...
Loading C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w\sub-0037\surf\lh.white...
Mapping lh sub-0037 -> ico (4) ...
    Triangle neighbors and vertex normals...
Loading geometry from C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\
↪outputs\PostFreeSurfer\T1w\sub-0037\surf\lh.sphere...
Setting up the triangulation for the decimated surface...
loaded lh.white 2562/126910 selected to source space (ico = 4)

Loading C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w\sub-0037\surf\rh.white...
Mapping rh sub-0037 -> ico (4) ...
    Triangle neighbors and vertex normals...
Loading geometry from C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\
↪outputs\PostFreeSurfer\T1w\sub-0037\surf\rh.sphere...
Setting up the triangulation for the decimated surface...
loaded rh.white 2562/128713 selected to source space (ico = 4)

Calculating source space distances (limit=inf mm)...
    Computing patch statistics...
    Patch information added...
    Computing patch statistics...
    Patch information added...
You are now one step closer to computing the gain matrix
Overwriting existing file.
    Write a source space...
    [done]
    Write a source space...
    [done]
    2 source spaces written
```

We will now load and visualise the source space together with the BEM model

[4]:
```
src = mne.read_source_spaces(subjects_dir+'/%s/bem/%s-ico4-src.fif' %(subject,subject))
# Plot the bem with the sources
mne.viz.plot_bem(subject=subject, subjects_dir=subjects_dir,brain_surfaces='white',
src=src, orientation='coronal')
```

```
    Reading a source space...
    Computing patch statistics...
    Patch information added...
    Distance information added...
```

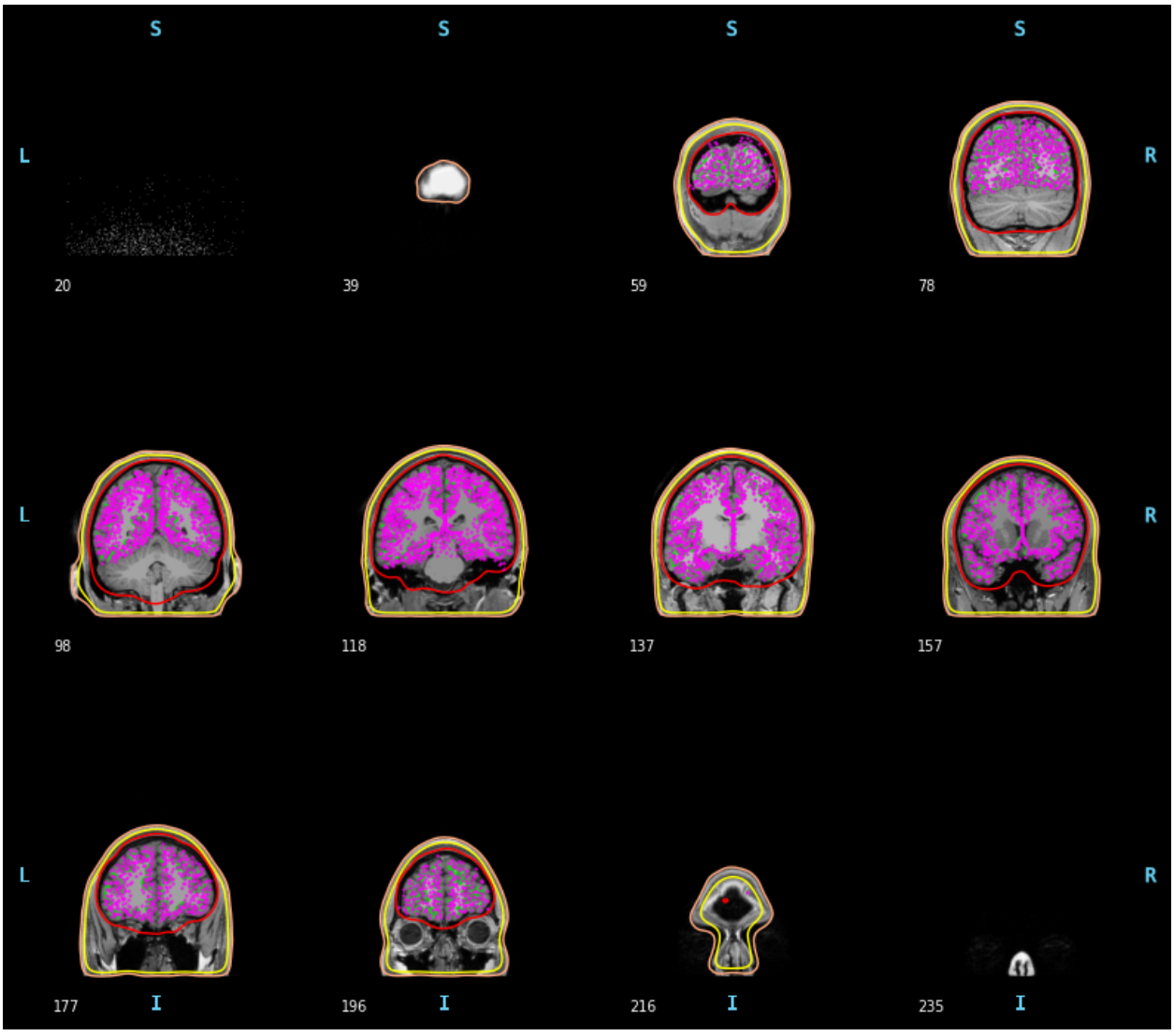
(continues on next page)

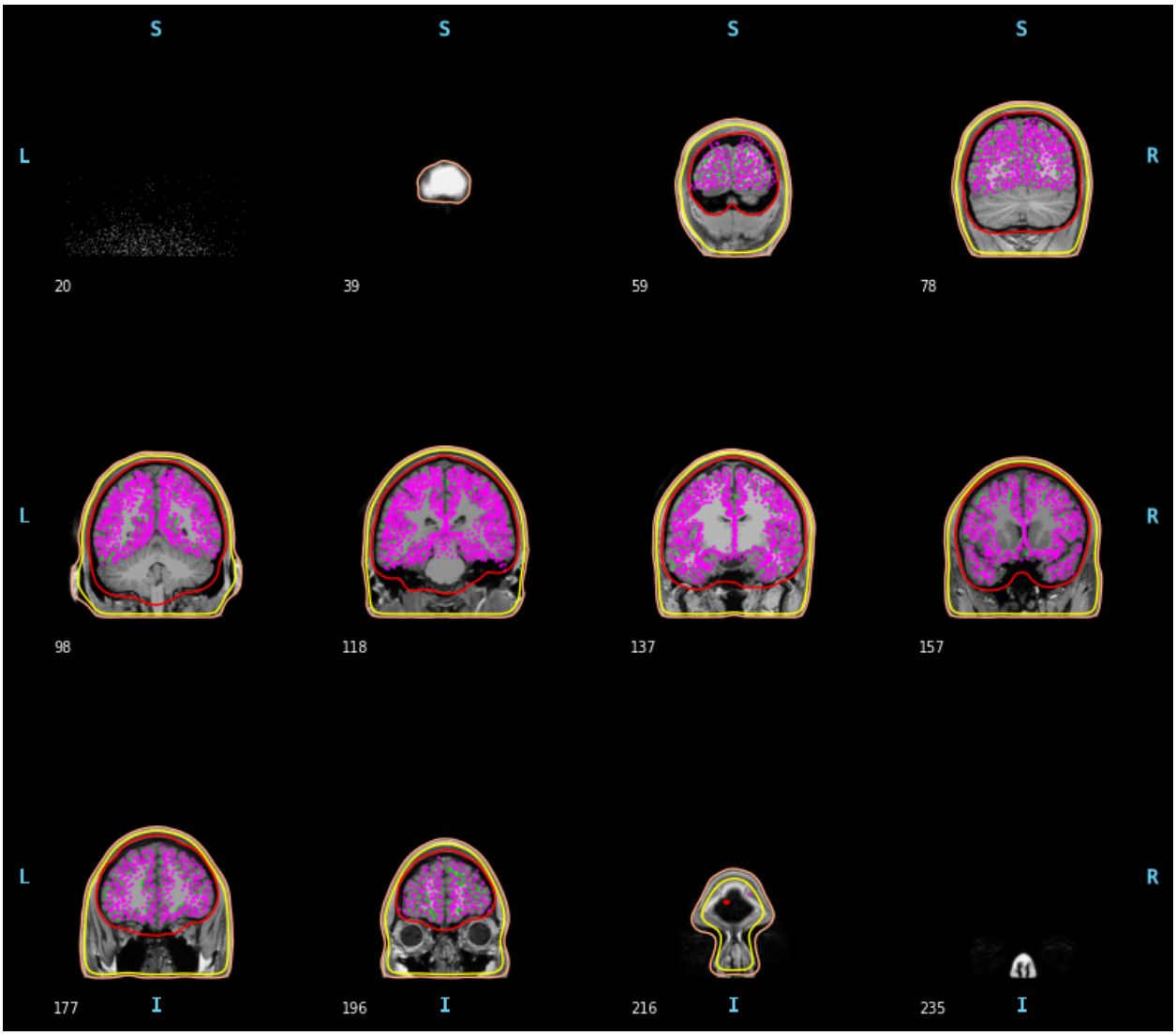
(continued from previous page)

```
    [done]
    Reading a source space...
    Computing patch statistics...
    Patch information added...
    Distance information added...
    [done]
    2 source spaces read
Using surface: C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w\sub-0037\bem\inner_skull.surf
Using surface: C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w\sub-0037\bem\outer_skull.surf
Using surface: C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\anat\outputs\
↪PostFreeSurfer\T1w\sub-0037\bem\outer_skin.surf
```

```
C:\ProgramData\mne-python\1.6.1_0\Lib\site-packages\mne\viz\utils.py:165: UserWarning:␣
↪FigureCanvasAgg is non-interactive, and thus cannot be shown
  (fig or plt).show(**kwargs)
```

[4]:





[5]:
```
## Create the bem solution.
conductivity = (0.3,) # for single layer

bem_model = mne.make_bem_model(subject=subject,ico=4,conductivity=conductivity,subjects_
↪dir=subjects_dir)
```

```
Creating the BEM geometry...
Going from 5th to 4th subdivision of an icosahedron (n_tri: 20480 -> 5120)
inner skull CM is   0.92  -2.09 -11.14 mm
Surfaces passed the basic topology checks.
Complete.
```

[6]:
```
# Load the BEM surfaces from the generated .fif file
#bem_model = mne.read_bem_surfaces(fif_path)
```

(continues on next page)

(continued from previous page)

```
#Make solutions
bem = mne.make_bem_solution(bem_model)
mne.write_bem_solution(subjects_dir+'/%s/bem/%s-inner-skull.bem.fif' %(subject,subject),
↪bem, overwrite=True)
```

```
Homogeneous model surface loaded.
Computing the linear collocation solution...
    Matrix coefficients...
        inner skull (2562) -> inner skull (2562) ...
    Inverting the coefficient matrix...
Solution ready.
BEM geometry computations complete.
Overwriting existing file.
```

[7]:
```
bem = mne.read_bem_solution(subjects_dir+'/%s/bem/%s-inner-skull.bem.fif' %(subject,
↪subject))
```

```
Loading surfaces...

Loading the solution matrix...

Homogeneous model surface loaded.
Loaded linear collocation BEM solution from C:\Users\hz3752\PycharmProjects\mne_bids_
↪pipeline\data\anat\outputs\PostFreeSurfer\T1w\sub-0037\bem\sub-0037-inner-skull.bem.fif
```

We now have the necessary input to compute the forward solution operator. (Bear in mind that this will compute an operator and not apply it to the measurements to get the source time series.)

[8]:
```
fwd = mne.make_forward_solution(r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\
↪meg\Sub-0037\sub-01_01-eyes-closed-raw.fif',
                                trans=r'C:\Users\hz3752\PycharmProjects\mne_bids_
↪pipeline\data\meg\%s\%s-trans.fif' %(subject,subject),
                                src=src,
                                bem=bem,
                                meg=True,
                                eeg=False,
                                ignore_ref=True)

mne.write_forward_solution(r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\
↪%s\%s-fwd.fif' %(subject,subject),
                           fwd,
                           overwrite=True,
                           verbose=None)
```

```
Source space          : <SourceSpaces: [<surface (lh), n_vertices=126910, n_used=2562>,
↪<surface (rh), n_vertices=128713, n_used=2562>] MRI (surface RAS) coords, subject 'sub-
↪0037', ~22.2 MB>
MRI -> head transform : C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\sub-
↪0037\sub-0037-trans.fif
Measurement data      : sub-01_01-eyes-closed-raw.fif
Conductor model   : instance of ConductorModel
Accurate field computations
```

(continues on next page)

(continued from previous page)

```
Do computations in head coordinates
Free source orientations

Read 2 source spaces a total of 5124 active source locations

Coordinate transformation: MRI (surface RAS) -> head
     0.999060  0.005683  0.042983      -0.24 mm
    -0.013875  0.981144  0.192780      10.85 mm
    -0.041076 -0.193195  0.980300      67.10 mm
     0.000000  0.000000  0.000000       1.00

Read 207 MEG channels from info
105 coil definitions read
Coordinate transformation: MEG device -> head
     0.990440 -0.137012  0.016000      -1.48 mm
     0.126928  0.950617  0.283226      26.76 mm
    -0.054016 -0.278488  0.958920      64.19 mm
     0.000000  0.000000  0.000000       1.00
MEG coil definitions created in head coordinates.
Source spaces are now in head coordinates.

Employing the head->MRI coordinate transform with the BEM model.
BEM model instance of ConductorModel is now set up

Source spaces are in head coordinates.
Checking that the sources are inside the surface (will take a few...)
Checking surface interior status for 2562 points...
    Found  918/2562 points inside  an interior sphere of radius   47.5 mm
    Found    0/2562 points outside an exterior sphere of radius   91.3 mm
    Found    0/1644 points outside using surface Qhull
    Found    0/1644 points outside using solid angles
    Total 2562/2562 points inside the surface
Interior check completed in 498.4 ms
Checking surface interior status for 2562 points...
    Found  858/2562 points inside  an interior sphere of radius   47.5 mm
    Found    0/2562 points outside an exterior sphere of radius   91.3 mm
    Found    0/1704 points outside using surface Qhull
    Found    0/1704 points outside using solid angles
    Total 2562/2562 points inside the surface
Interior check completed in 534.4 ms

Checking surface interior status for 207 points...
    Found   0/207 points inside  an interior sphere of radius   47.5 mm
    Found 207/207 points outside an exterior sphere of radius   91.3 mm
    Found   0/  0 points outside using surface Qhull
    Found   0/  0 points outside using solid angles
    Total 0/207 points inside the surface
Interior check completed in 27.5 ms

Composing the field computation matrix...
Computing MEG at 5124 source locations (free orientations)...
```

(continues on next page)

(continued from previous page)

```
Finished.
    Write a source space...
    [done]
    Write a source space...
    [done]
    2 source spaces written
```

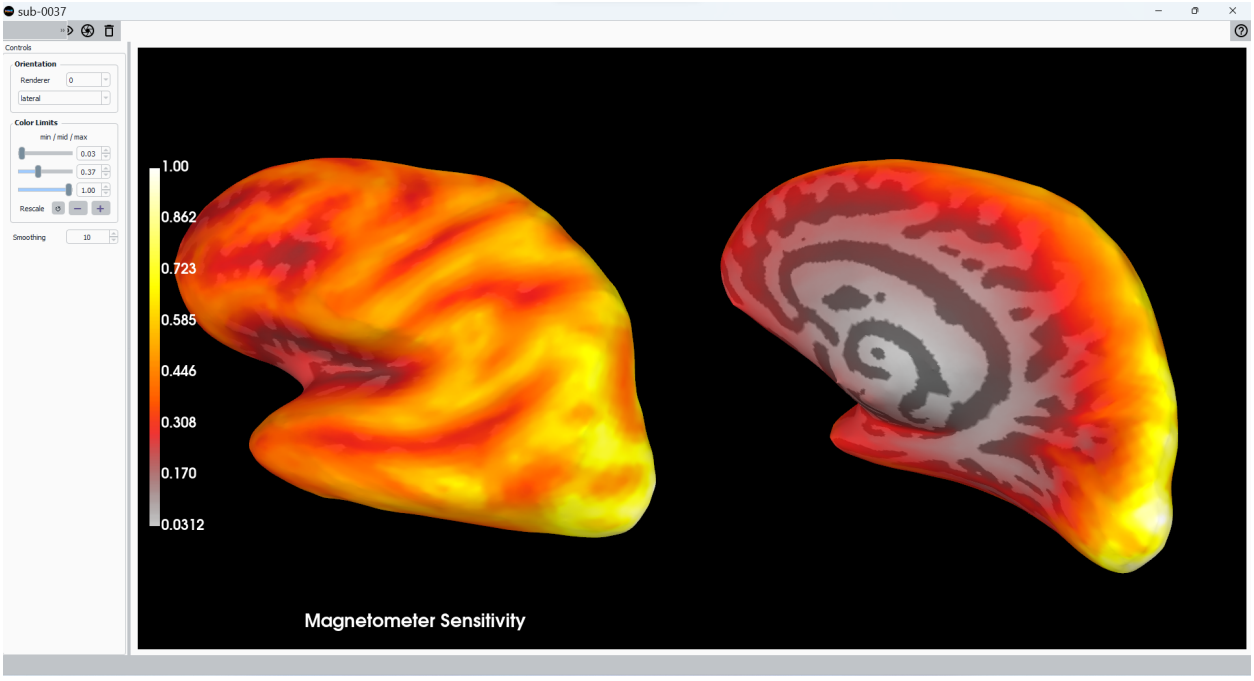
The number of modeled sources are 5124, there is 2562 source in each hemisphere

[10]:
```
mag_map = mne.sensitivity_map(fwd, ch_type='mag',mode='free')
mag_map.save(r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\%s\%s_
↪sensitivity-free' %(subject,subject), overwrite=True)
```

```
    207 out of 207 channels remain after picking
Writing STC to disk...
Overwriting existing file.
Overwriting existing file.
[done]
```

[ ]:
```
brainmap = mag_map.plot(time_label='Magnetometer Sensitivity', subjects_dir=subjects_dir,
↪clim=dict(lims=[0,50,100]),hemi='split')
```

We can now visualise the sensitivity map on both hemispheres



[ ]:

[ ]:

19.7 Close-up on forward solutions

In previous notebooks we computed the forward solution operator given the raw data, transformed data, BEM model and source space. In this notebook we investigate deeper the forward solution operator. Let us import the forward solution operator we previously computed.

[10]:
```
import mne

subjects_dir = r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\Sub-0037\sub-
↪0037-fwd.fif'

fwd = mne.read_forward_solution(subjects_dir, include=(), exclude=(), ordered=True,␣
↪verbose=None)
```

```
Reading forward solution from C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\
↪Sub-0037\sub-0037-fwd.fif...
    Reading a source space...
    Computing patch statistics...
    Patch information added...
    Distance information added...
    [done]
    Reading a source space...
    Computing patch statistics...
    Patch information added...
    Distance information added...
    [done]
    2 source spaces read
    Desired named matrix (kind = 3523) not available
    Read MEG forward solution (5124 sources, 207 channels, free orientations)
    Source spaces transformed to the forward solution coordinate frame
```

Two possible source orientations exist in MNE, documented in `source_ori` in `fwd`: - FIFF.FIFFV_MNE_FIXED_ORI: Fixed orientation mode, the orientation of the source currents is constrained to be perpendicular to the cortical surface. This means that the direction of the current is assumed to be known a priori and is fixed during the inverse solution process. The primary advantage of this approach is its simplicity and reduced computational load. By constraining the direction of the sources, the number of variables in the inverse problem is effectively halved, which can make the solutions more stable under conditions of noisy data or when the data are limited. - When to use fixed orientation? Fixed orientation is often used when a strong prior knowledge of the orientation of the cortical currents exists (such as orientation aligned with the local sulcal or gyral geometry) or when computational simplicity is required.

- FIFF.FIFFV_MNE_FREE_ORI: Free Orientation: In contrast, free orientation allows the orientation of the source currents to vary and be estimated as part of the source localization process. This mode does not impose any constraints on the direction of the currents, thereby enabling the estimation of the most probable orientation based on the data itself. This can potentially lead to more accurate reconstructions of the underlying neural activity, particularly in complex scenarios where the orientation of the sources cannot be easily predicted.
- When to use free orientation?: Free orientation is particularly useful in research settings where the precise modeling of neural activity is crucial, and where there is enough high-quality data to support the increased complexity of the inverse solution. For the data we red, let us print which source orientation was used:

[2]:
```
print(fwd['source_ori'])
```

```
2 (FIFFV_MNE_FREE_ORI)
```

The location of the sources are in `source_rr`

[3]:
```
print(fwd['source_rr'])
```

```
[[-0.0210264  -0.06770435  0.06570835]
 [-0.01708252 -0.06216683  0.07929588]
 [-0.01315619 -0.06514898  0.07275735]
 ...
 [ 0.00288066  0.02885441  0.04047859]
 [ 0.01054338  0.0624362   0.0432792 ]
 [ 0.01025996  0.08508232  0.04505657]]
```

The forward solution operator are in `sol` they are of shape (n_channels, n_sources * n_orientation) where n_orientation is the number of possible orientations of the source (3 orientations for the free-orientation setup and 1 for the fixed orientation setup), let us print the shape of the solution

[4]:
```
print(fwd['sol']['data'].shape)
```

```
(207, 15372)
```

This means that we used 15372/3 = 5124 sources and the solutions are computed for 207 MEG gradiometers

[2]:
```
import mne
# Load your raw data
raw = mne.io.read_raw_fif(r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\
↪Sub-0037\sub-01_01-eyes-closed-raw.fif', preload=True)
```

```
Opening raw data file C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\Sub-
↪0037\sub-01_01-eyes-closed-raw.fif...
    Range : 0 ... 321999 =      0.000 ...   321.999 secs
Ready.
Reading 0 ... 321999  =      0.000 ...   321.999 secs...
```

[7]:
```
#Add covariance matrix computation

#cov = mne.compute_covariance

cov = mne.compute_raw_covariance(raw)
cov.save(r'C:\Users\hz3752\PycharmProjects\mne_bids_pipeline\data\meg\Sub-0037\sub-01-
↪cov.fif')
```

```
Using up to 1610 segments
Number of samples used : 322000
[done]
```

[13]:
```
 inv = mne.minimum_norm.make_inverse_operator(raw.info, fwd, cov, depth=None, loose='auto
↪', fixed=False) #fixed=False: Ignoring dipole direction.
```

```
Converting forward solution to surface orientation
    Average patch normals will be employed in the rotation to the local surface␣
↪coordinates...
    Converting to surface-based source orientations...
    [done]
Computing inverse operator with 207 channels.
    207 out of 207 channels remain after picking
```

(continues on next page)

(continued from previous page)

```
Selected 207 channels
Applying loose dipole orientations to surface source spaces: 0.2
Whitening the forward solution.
Computing rank from covariance with rank=None
    Using tolerance 8.2e-13 (2.2e-16 eps * 207 dim * 18  max singular value)
    Estimated rank (mag): 207
    MAG: rank 207 computed from 207 data channels with 0 projectors
    Setting small MAG eigenvalues to zero (without PCA)
Creating the source covariance matrix
Adjusting source covariance matrix.
Computing SVD of whitened and weighted lead field matrix.
    largest singular value = 14.1305
    scaling factor to adjust the trace = 7.29474e+27 (nchan = 207 nzero = 3)
```

[ ]:
```
subj='sub-01'
        #-------------------------STCs--------------------------------#

print '%s: Creating STCs...'%subj
        os.makedirs('STC/%s' %subj)
        for ev in evoked:
            stc = mne.minimum_norm.apply_inverse(ev, inv, lambda2=lambda2, method='dSPM')
            # mophing stcs to the fsaverage using precomputed matrix method:
            vertices_to = mne.grade_to_vertices('fsaverage', grade=4, subjects_
↪dir=subjects_dir) #fsaverage's source space
            morph_mat = mne.compute_morph_matrix(subject_from=subj, subject_to='fsaverage
↪', vertices_from=stc.vertices, vertices_to=vertices_to, subjects_dir=subjects_dir)
            stc_morph = mne.morph_data_precomputed(subject_from=subj, subject_to=
↪'fsaverage', stc_from=stc, vertices_to=vertices_to, morph_mat=morph_mat)
            stc_morph.save('STC/%s/%s_%s_dSPM' %(subj,subj,ev.comment))
            del stc, stc_morph
        print '>> DONE CREATING STCS FOR SUBJ=%s'%subj
        print '----------------------------------------\n'

        #deleting variables
        del epochs_rej, evoked, info, trans, src, fwd, cov, inv
```

CHAPTER

TWENTY

MAINTENANCE OF MEG SYSTEM

20.1 Checks to be made

Helium level

- Every two days check Helium Level it should be higher than 50%
- **Check the ATP and ATL gas flow and pressure**
  - **If low Helium pressure (Low G FLOW on the ATL) then**
    - Remove the hose between ATP and ATL
    - Check if helium is passing through the hose
    - Plug the hose again, see if G Flow increases
    - Restart the ATL from the green button on the rear panel

20.2 Data retrieval from ATP and ATL for diagnostic

When the ATP and ATL helium recovery system exhibits low gas flow or unusual temperature/pressure, contact the references below and provide them the following data

1. Retrieve data from ATP and ATL by opening an FTP connection to 10.224.44.200 and 10.224.44.201
2. Connect to the NYU Abu Dhabi VPN
3. Use the following information username: qd, password: 79653
4. Navigate to the */StorageCard/DAT_Files/* directory for each of the previous IP addresses, send the latest *.dat* files

---

Note: documentation in the link: QD Documentation for Data retrieval Sheet to be filled by maintenance team for Helium levels Helium Filling Sheet

---

Check air pressure every month

- The air pressure inside the meg can be not so good, it must be 0.7, 0.8 this could be a compressor problem
- When the air pressure is low, the door could be stuck

MSR Door:

- Test the emergency button every week

- Test if the pressure release when using the emergency button is getting heavier or not releasing pressure as it is supposed to be
- If using the manual handle, make sure that to reset the door, you need to put the handle in the original position or else the door won't reset

20.3 Contacts table

Name	Email	Number	Role
Hadi Zaatiti	hz3752@nyu.edu	+971 56 275 4921	Research Scientist
Lawrence Torres	ljt7767@nyu.edu	NA	NA
Qiang Zhang	qz19@nyu.edu	NA	NA
QD Helium Recovery	heliumrecovery@qd-europe.com	NA	NA
QD Konstantin Voigt	voigt@qd-europe.com	NA	NA
QD Tobias Adler	adler@qd-europe.com	NA	NA
Ahmed Ansari	aa7703@nyu.edu	NA	Helium store manager (Primary contact for getting Helium tanks)
Mohammad Rakib	mr5527@nyu.edu	NA	Logistics and Sanitation Coordinator

CHAPTER

TWENTYONE

EMERGENCY PROCEDURES

Helium leak: - Contact maintenance team

MSR door locked: - push the Red emergency button - if that don’t work use the manual

CHAPTER

TWENTYTWO

GLOSSARY

Table 1: Glossary table

Word	Definition
fROI	Functional Region of Interest

CHAPTER

TWENTYTHREE

API DOCUMENTATION OF MEG-PIPELINE

megpipeline	megpipeline - Python base package for MEG data post-processing.

23.1 megpipeline

megpipeline - Python base package for MEG data post-processing.

Functions

`get_raw_data`(self, file_name)	Return a list of random as strings.

Classes

`MEGpipeline`()	Primary class for MEG pipeline.

PYTHON MODULE INDEX

INDEX

G

M