In [1]:
```
import mne
#%gui qt
#import matplotlib

#%matplotlib qt
import numpy as np
from matplotlib import pyplot as plt
from stormdb.access import Query
from pickle import load
from scipy import stats
from mne.datasets import sample
from mne.stats import spatio_temporal_cluster_1samp_test
import os
import pickle
from copy import deepcopy
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
# os.environ['ETS_TOOLKIT'] = 'qt4'
# os.environ['QT_API'] = 'pyqt5'
# %gui qt
#mne.viz.set_3d_backend("notebook")
```

In [2]:
```
proj_name = 'MINDLAB2020_MEG-AuditoryPatternRecognition'
wdir = '/projects/' + proj_name + '/scratch/working_memory/'
stats_dir = wdir + 'results/stats/'
data_dir = wdir + 'averages/data/'
```

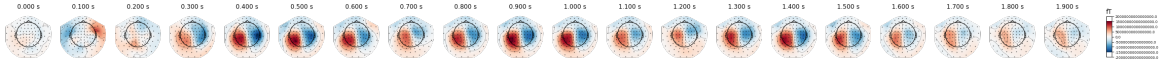
In [3]:
```
# Read stats
sfname = '{}ERF_sensor_stats.p'.format(stats_dir)
sfile = open(sfname,'rb')
stats_results = pickle.load(sfile)
sfile.close()
```

In [4]:
```
# Load an ERF
dfname = data_dir + '0021_LZW_evoked.p'
dfile = open(dfname,'rb')
ERF_dict = pickle.load(dfile)
ERF = ERF_dict['main']['same']
```

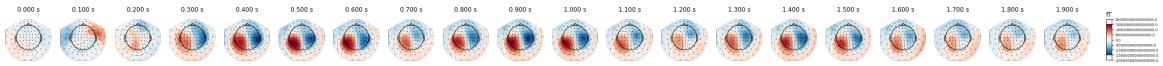
In [9]:
```
# Plot all topomaps
periods = {'encoding': [0,2], 'delay': [2,4], 'retrieval': [4,6]}
ch_type = ['mag','grad']
for cht in ch_type:
    print('\n############### {} ###############\n'.format(cht))
    for p in periods:
        print('############ {} ###############\n'.format(p))
        for c in stats_results:
            tmin = periods[p][0]
            tmax = periods[p][1]
            cERF = ERF.copy().pick_types(meg=cht).crop(tmin=tmin, tmax=
tmax)
            tidx = np.where([x and y for x,y in zip(ERF.times >= tmin,
ERF.times <= tmax)])[0]
            chidx = np.array(mne.pick_types(ERF.info, meg = cht))

            cpvals = stats_results[c]['pvals'].copy()
            cpvals = cpvals[tidx, :]
            cpvals = cpvals[:,chidx].T

            cdata = stats_results[c]['tvals'].copy()
            cdata = cdata[chidx,:]
            cdata = cdata[:,tidx]

            gmask, adj_pvals = mne.stats.fdr_correction(cpvals, 0.025)
            cERF.data = cdata#*gmask #stats_results[c]['mask']
            plot_times = np.arange(0,2,0.1) + tmin

            print(c)
            cERF.plot_topomap(times=plot_times)
```

```
############### mag ################

############# encoding ################
```

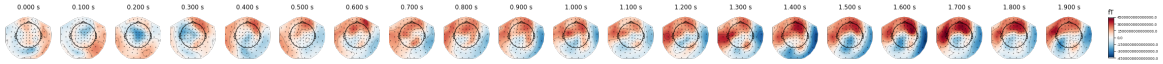
maintenance



manipulation

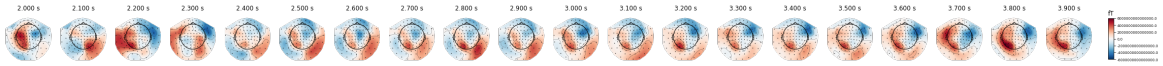


difference

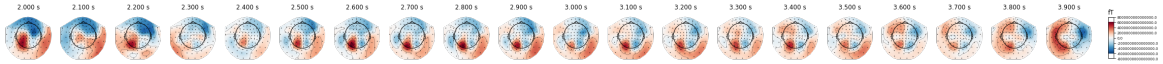


```
############# delay ################
```

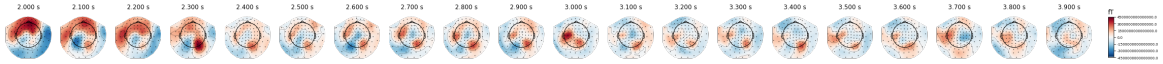
maintenance



manipulation

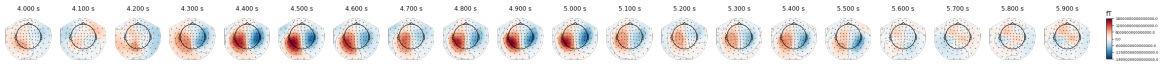


difference

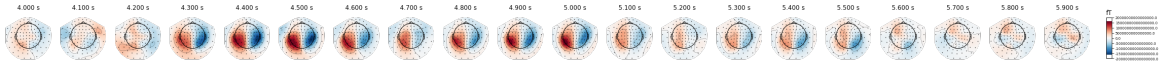


```
############# retrieval ################
```

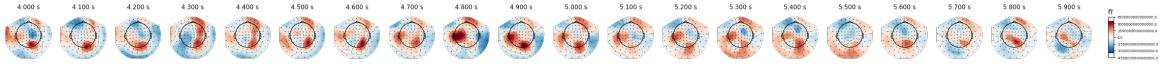
maintenance



manipulation

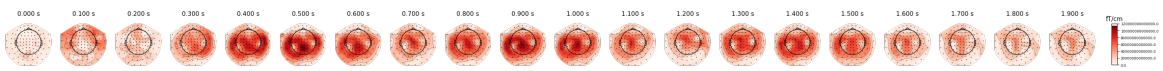


difference

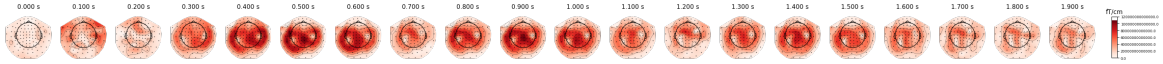


```
############### grad ################

############# encoding ################
```

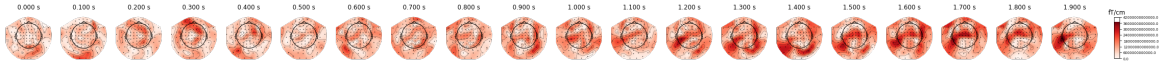
maintenance



manipulation

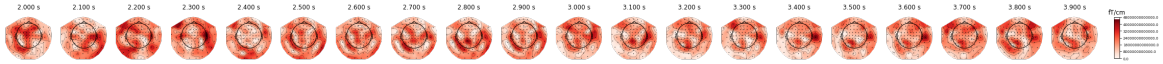


difference

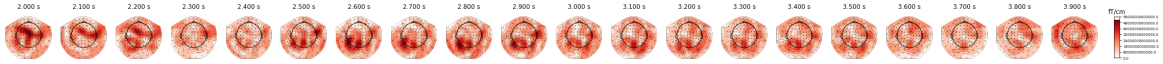


############ delay ################

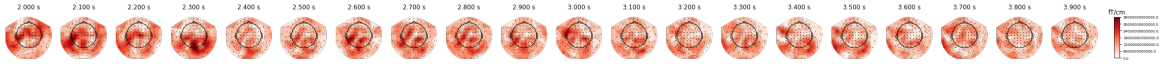
maintenance



manipulation

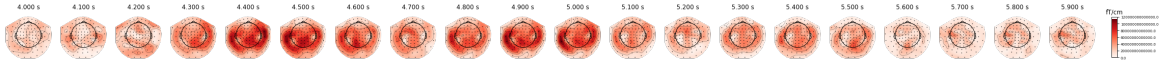


difference

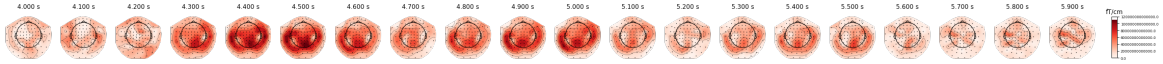


############ retrieval ################

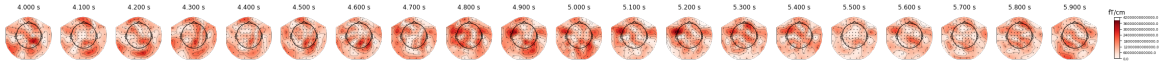
maintenance



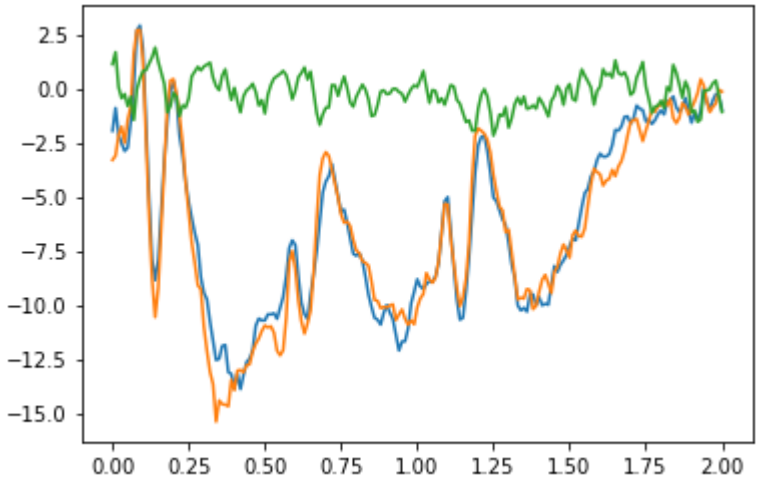
manipulation

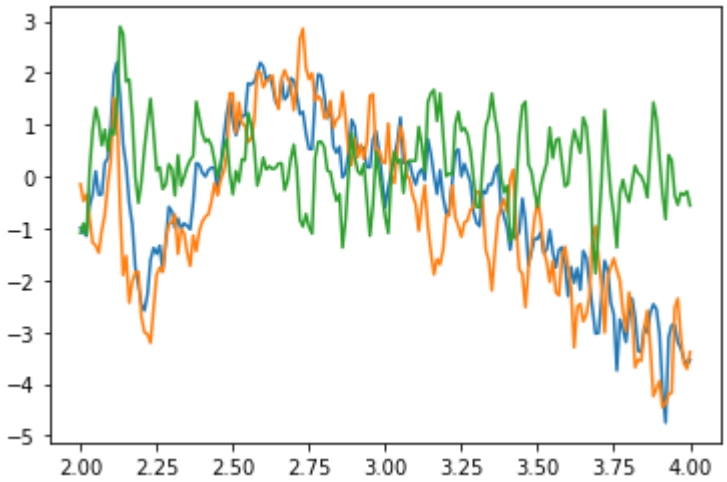


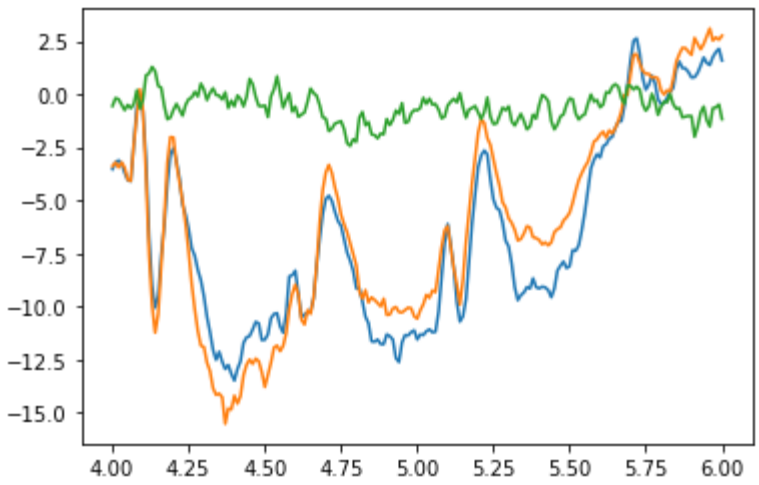
difference



In [6]:
```
# Plot channel pairs
periods = {'encoding': [0,2], 'delay': [2,4], 'retrieval': [4,6]}
channels = ['MEG1331']
for cht in channels:
    for p in periods:
        fig = plt.figure()
        for c in stats_results:
            tmin = periods[p][0]
            tmax = periods[p][1]
            cERF = ERF.copy().pick_channels([cht]).crop(tmin=tmin, tmax
=tmax)
            tidx = np.where([x and y for x,y in zip(ERF.times >= tmin,
ERF.times <= tmax)])[0]
            chidx = np.array(mne.pick_channels(ERF.info['ch_names'], [c
ht]))

            cpvals = stats_results[c]['pvals'].copy()
            cpvals = cpvals[tidx, :]
            cpvals = cpvals[:,chidx].T

            cdata = stats_results[c]['tvals'].copy()
            cdata = cdata[chidx,:]
            cdata = cdata[:,tidx]

            gmask, adj_pvals = mne.stats.fdr_correction(cpvals, 0.025)
            cERF.data = cdata#*gmask #stats_results[c]['mask']
            plt.plot(ERF.times[tidx],np.squeeze(cERF.data))
```







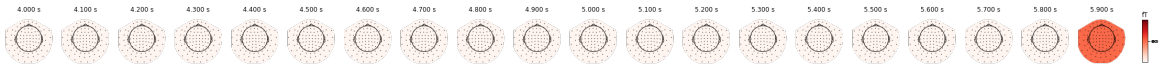
In [7]:
```
# Read stats
tfname = '{}ERF_target_sensor_stats.p'.format(stats_dir)
tfile = open(tfname,'rb')
target_results = pickle.load(tfile)
tfile.close()
```

In [8]:
```
# Plot all target topomaps
ch_type = ['mag','grad']
for cht in ch_type:
    print('\n############### {} ###############\n'.format(cht))
    for c in target_results:
        print('\n############### {} ###############\n'.format(c))
        for p in target_results[c]:
            cERF = ERF.copy().pick_types(meg=cht).crop(tmin=4, tmax=6)
            chidx = np.array(mne.pick_types(ERF.info, meg = cht))

            cpvals = target_results[c][p]['pvals'].copy()
            cpvals = cpvals[:,chidx].T

            cdata = target_results[c][p]['tvals'].copy()
            cdata = cdata[chidx,:]

            gmask, adj_pvals = mne.stats.fdr_correction(cpvals, 0.025)
            cERF.data = cdata*gmask #stats_results[c]['mask']
            plot_times = np.arange(4,6,0.1)

            print(p)
            cERF.plot_topomap(times=plot_times)
```

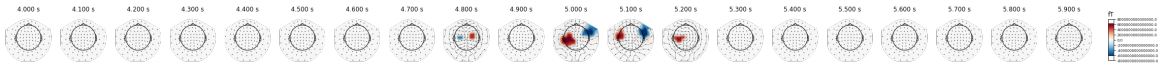
```
############### mag ################
```

```
############### main ################
```

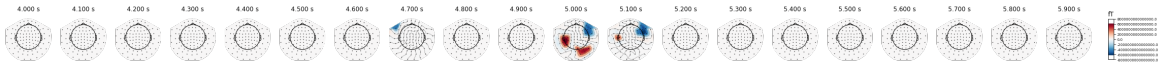
```
same-different1
```



```
same-different2
```

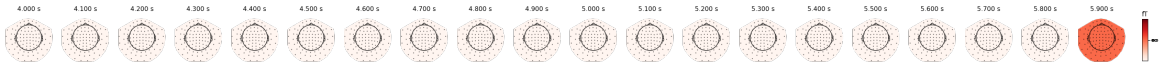


```
different1-different2
```

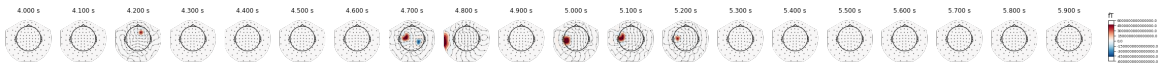


```
############### inv ################
```

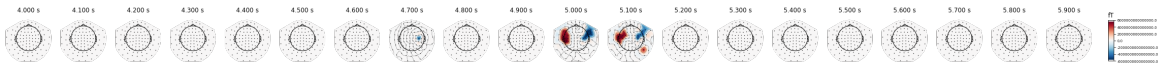
```
inverted-other1
```



```
inverted-other2
```



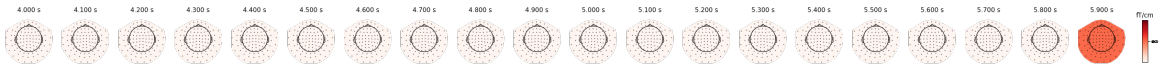
```
other1-other2
```



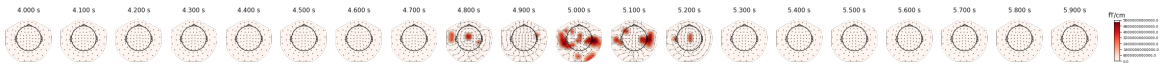
```
############### grad ################
```

```
############### main ################
```

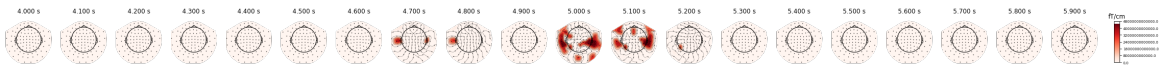
```
same-different1
```



```
same-different2
```

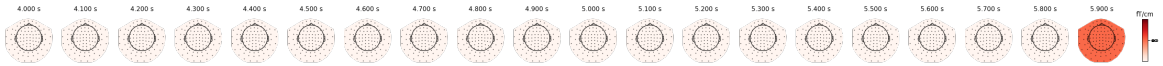


```
different1-different2
```

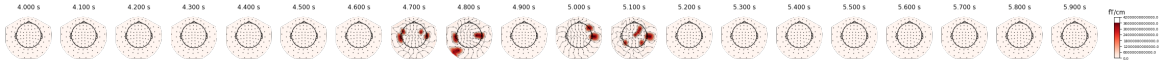


```
############### inv ################
```

```
inverted-other1
```



inverted-other2



other1-other2