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
from os import path as op
import pickle
from copy import deepcopy
import warnings
from do_stats import do_stats
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]:
```
project = 'MINDLAB2020_MEG-AuditoryPatternRecognition'
project_dir = '/projects/' + project
os.environ['MINDLABPROJ']= project
os.environ['MNE_ROOT']= '~/miniconda3/envs/mne' # for surfer
os.environ['MESA_GL_VERSION_OVERRIDE'] = '3.2'

avg_path = project_dir + '/scratch/working_memory/averages/data/'
stats_dir = project_dir + '/scratch/working_memory/results/stats/'
qr = Query(project)
sub_codes = qr.get_subjects()
```

In [3]:
```
## load data
sub_Ns = np.arange(21,91)
exclude = np.array([55,60,73,82]) # subjects with low maintenance accur
acy
sdata = {}
scount = 0
for sub in sub_Ns:
    sub_code = sub_codes[sub-1]
    if sub not in exclude:
        try:
            print('loading subject {}'.format(sub_code))
            score_fname = op.join(avg_path,sub_code + '_scores_imagine
d.p')
            score_file = open(score_fname,'rb')
            score = pickle.load(score_file)
            score_file.close()
            scount = scount +1
            for c in score:

                if scount == 1:
                    sdata[c] = []
                    cshape = score[c].data.shape[0]
                if score[c].shape[0] == cshape:
                    sdata[c].append(score[c].data)
        except Exception as e:
            print('could not load subject {}'.format(sub_code))
            print(e)
            continue
```

```
loading subject 0021_LZW
loading subject 0022_BHR
loading subject 0023_ZPC
loading subject 0024_JSV
loading subject 0025_62P
loading subject 0026_S1H
loading subject 0027_TLV
loading subject 0028_2NK
loading subject 0029_PAX
loading subject 0030_2MP
could not load subject 0030_2MP
[Errno 2] No such file or directory: '/projects/MINDLAB2020_MEG-Audit
oryPatternRecognition/scratch/working_memory/averages/data/0030_2MP_s
cores_imagined.p'
loading subject 0031_WZD
loading subject 0032_BLF
could not load subject 0032_BLF
[Errno 2] No such file or directory: '/projects/MINDLAB2020_MEG-Audit
oryPatternRecognition/scratch/working_memory/averages/data/0032_BLF_s
cores_imagined.p'
loading subject 0033_MXJ
could not load subject 0033_MXJ
[Errno 2] No such file or directory: '/projects/MINDLAB2020_MEG-Audit
oryPatternRecognition/scratch/working_memory/averages/data/0033_MXJ_s
cores_imagined.p'
loading subject 0034_V8Q
loading subject 0035_FTI
loading subject 0036_9EA
loading subject 0037_KPS
loading subject 0038_J7W
loading subject 0039_MLI
loading subject 0040_LDN
loading subject 0041_RSP
loading subject 0042_WOB
could not load subject 0042_WOB
[Errno 2] No such file or directory: '/projects/MINDLAB2020_MEG-Audit
oryPatternRecognition/scratch/working_memory/averages/data/0042_WOB_s
cores_imagined.p'
loading subject 0043_XKK
loading subject 0044_4QT
loading subject 0045_GVP
loading subject 0046_QXA
loading subject 0047_KXX
loading subject 0048_7BH
loading subject 0049_KFD
loading subject 0050_BBQ
loading subject 0051_NUI
could not load subject 0051_NUI
[Errno 2] No such file or directory: '/projects/MINDLAB2020_MEG-Audit
oryPatternRecognition/scratch/working_memory/averages/data/0051_NUI_s
cores_imagined.p'
loading subject 0052_GWV
loading subject 0053_BUS
loading subject 0054_TPU
```

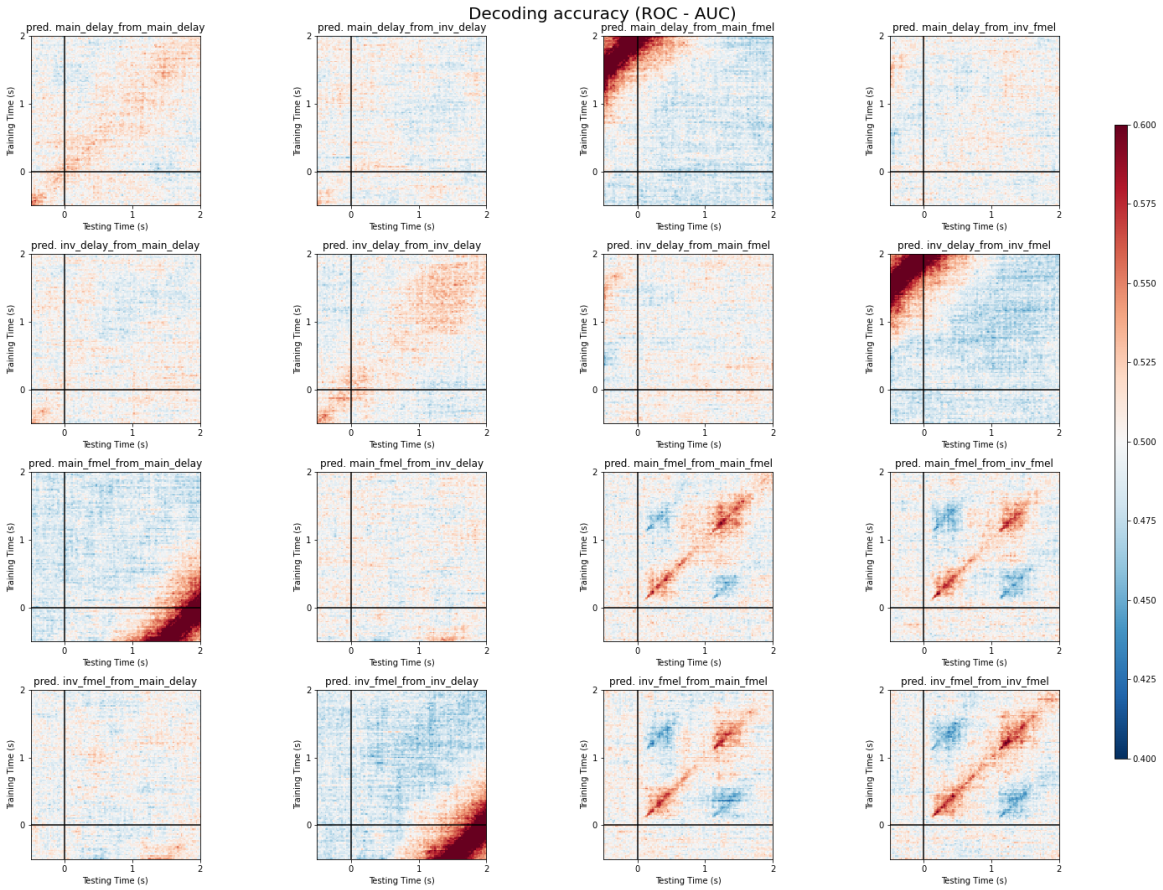
```
loading subject 0056_QLB
loading subject 0057_LOR
loading subject 0058_L3X
loading subject 0059_CYC
loading subject 0061_NMF
loading subject 0062_WKJ
loading subject 0063_NNY
loading subject 0064_GMS
loading subject 0065_DEH
loading subject 0066_T0O
loading subject 0067_PQF
loading subject 0068_Q6G
could not load subject 0068_Q6G
[Errno 2] No such file or directory: '/projects/MINDLAB2020_MEG-Audit
oryPatternRecognition/scratch/working_memory/averages/data/0068_Q6G_s
cores_imagined.p'
loading subject 0069_ECC
loading subject 0070_EWA
loading subject 0071_WGY
loading subject 0072_ZHI
loading subject 0074_CQ7
loading subject 0075_CIE
loading subject 0076_E8Z
loading subject 0077_YXW
loading subject 0078_PAK
loading subject 0079_DFV
could not load subject 0079_DFV
[Errno 2] No such file or directory: '/projects/MINDLAB2020_MEG-Audit
oryPatternRecognition/scratch/working_memory/averages/data/0079_DFV_s
cores_imagined.p'
loading subject 0080_G9V
loading subject 0081_V1D
loading subject 0083_B62
loading subject 0084_DZM
loading subject 0085_EDC
loading subject 0086_HBD
loading subject 0087_MM7
loading subject 0088_AVP
loading subject 0089_FHG
loading subject 0090_0MJ
```

In [4]:
```
# grand averages:
smean, sstd, smedian, siqr_lower, siqr_upper = {},{},{},{},{}
for s in sdata:
    sdata[s] = np.array(sdata[s])
    print(sdata[s].shape)
    smean[s] = np.mean(sdata[s],0)
    smedian[s] = np.median(sdata[s],0)
    sstd[s] = np.std(sdata[s],0)
    siqr_lower[s] = np.percentile(sdata[s],25,0)
    siqr_upper[s] = np.percentile(sdata[s],75,0)
```

```
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
(59, 126, 126)
```

In [5]:
```
ncols = 4
fig, axes = plt.subplots(ncols=ncols,nrows=4, figsize = (20,15)) #,grid
spec_kw=dict(width_ratios=[1,1,1,1]) )
for sidx,s in enumerate(smean):
    f,se = s.split('_from_')
    ext = [-0.5,2,
           -.5,2]
    rix, cix = sidx//ncols,sidx%ncols
    im = axes[rix, cix].matshow(smean[s], vmin = .4, vmax = .6,#vmin=0.
18, vmax=0.48,
                                        cmap='RdBu_r', origin='lower', ex
tent=ext)
    axes[rix, cix].axhline(0., color='k')
    axes[rix, cix].axvline(0., color='k')
    axes[rix, cix].xaxis.set_ticks_position('bottom')
    axes[rix, cix].set_xlabel('Testing Time (s)')
    axes[rix, cix].set_ylabel('Training Time (s)')
    axes[rix, cix].set_anchor('W')
    axes[rix, cix].set_title('pred. {}'.format(s),{'horizontalalignment
': 'center'})
cbar_ax = fig.add_axes([0.925,0.15,0.01,0.7])
fig.colorbar(im,cax=cbar_ax)
fig.suptitle('Decoding accuracy (ROC - AUC)', fontsize =  20)
plt.tight_layout()
#plt.savefig(avg_path + '/figures/{}_accuracies_imagined.pdf'.format(su
b),orientation='landscape')
```

```
/tmp/ipykernel_16790/147766028.py:20: UserWarning: This figure includ
es Axes that are not compatible with tight_layout, so results might b
e incorrect.
  plt.tight_layout()
```

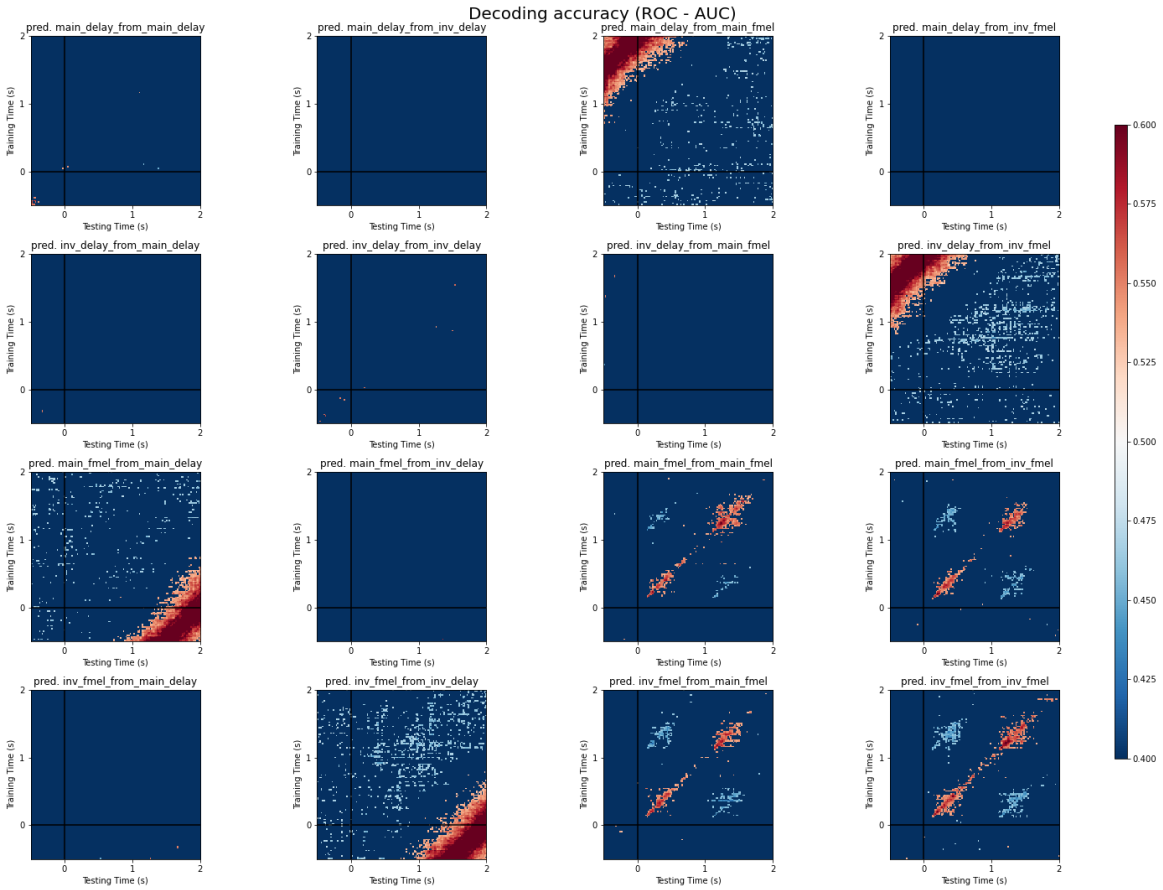


In [6]:
```
FDR_stats = {}
for s in sdata:
    FDR_stats[s] = do_stats(sdata[s],'FDR',h0 = .5)
```

```
Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.

Performing FDR correction
.
```

In [7]:
```
ncols = 4
fig, axes = plt.subplots(ncols=ncols,nrows=4, figsize = (20,15)) #,grid
spec_kw=dict(width_ratios=[1,1,1,1]) )
for sidx,s in enumerate(FDR_stats):
    f,se = s.split('_from_')
    ext = [-0.5,2,
           -.5,2]
    rix, cix = sidx//ncols,sidx%ncols
    mask = FDR_stats[s]['qvals'] <= .025
    #mask = FDR_stats[s]['mask']
    im = axes[rix, cix].matshow(FDR_stats[s]['data_mean'] * mask, vmin
= .4, vmax = .6,#vmin=0.18, vmax=0.48,
                                        cmap='RdBu_r', origin='lower', ex
tent=ext)
    axes[rix, cix].axhline(0., color='k')
    axes[rix, cix].axvline(0., color='k')
    axes[rix, cix].xaxis.set_ticks_position('bottom')
    axes[rix, cix].set_xlabel('Testing Time (s)')
    axes[rix, cix].set_ylabel('Training Time (s)')
    axes[rix, cix].set_anchor('W')
    axes[rix, cix].set_title('pred. {}'.format(s),{'horizontalalignment
': 'center'})
cbar_ax = fig.add_axes([0.925,0.15,0.01,0.7])
fig.colorbar(im,cax=cbar_ax)
fig.suptitle('Decoding accuracy (ROC - AUC)', fontsize =  20)
plt.tight_layout()
#plt.savefig(avg_path + '/figures/{}_accuracies_imagined.pdf'.format(su
b),orientation='landscape')
```

```
/tmp/ipykernel_16790/310591734.py:22: UserWarning: This figure includ
es Axes that are not compatible with tight_layout, so results might b
e incorrect.
  plt.tight_layout()
```



In [8]:
```
cluster_stats = {}
for s in sdata:
    cluster_stats[s] = do_stats(sdata[s],method='montecarlo',h0=.5,n_pe
rmutations=1000)
```

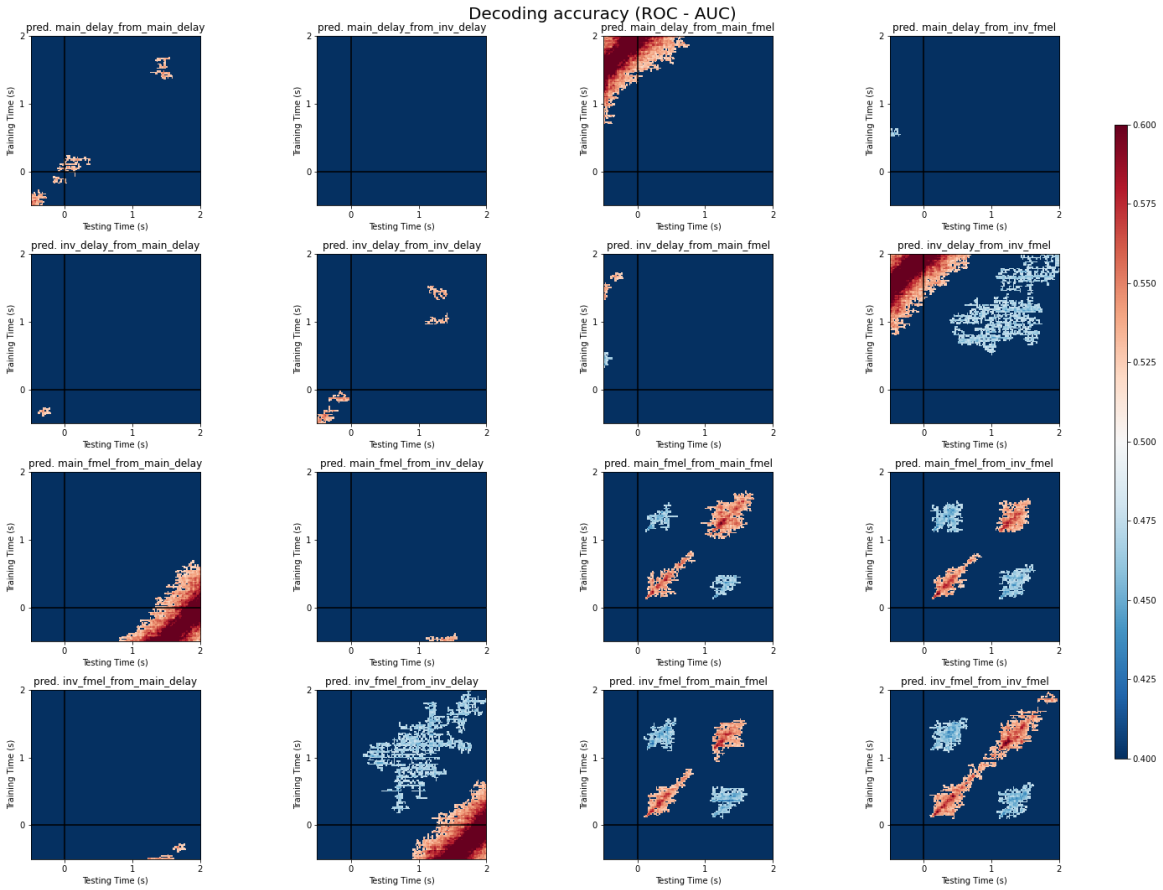
```
Clustering.
stat_fun(H1): min=-5.361005 max=6.779203
Running initial clustering
Found 662 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-4.639292 max=4.664819
Running initial clustering
Found 584 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-5.954016 max=197.962427
Running initial clustering
Found 734 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-4.423415 max=4.579189
Running initial clustering
Found 574 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-4.446595 max=5.529618
Running initial clustering
Found 560 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-4.120748 max=5.569066
Running initial clustering
Found 710 clusters
Permuting 999 times...
```

```
Computing cluster p-values

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-7.107615 max=182.513025
Running initial clustering
Found 698 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-5.783496 max=197.962427
Running initial clustering
Found 749 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-4.298014 max=8.542730
Running initial clustering
Found 552 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-5.943971 max=7.263998
Running initial clustering
Found 560 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-7.577320 max=8.542730
Running initial clustering
Found 557 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-5.088657 max=5.529618
Running initial clustering
Found 561 clusters
Permuting 999 times...
```

```
Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-5.784397 max=182.513025
Running initial clustering

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-7.091009 max=8.403537
Running initial clustering
Found 556 clusters
Permuting 999 times...

Computing cluster p-values
Done.
Clustering.
stat_fun(H1): min=-7.629682 max=8.374595
Running initial clustering
Found 553 clusters
Permuting 999 times...

Computing cluster p-values
Done.
```

In [9]:
```
ncols = 4
fig, axes = plt.subplots(ncols=ncols,nrows=4, figsize = (20,15)) #,grid
spec_kw=dict(width_ratios=[1,1,1,1]) )
for sidx,s in enumerate(cluster_stats):
    f,se = s.split('_from_')
    ext = [-0.5,2,
           -.5,2]
    rix, cix = sidx//ncols,sidx%ncols
    #mask = stats_results[s]['qvals'] <= .025
    mask = cluster_stats[s]['mask']
    im = axes[rix, cix].matshow(cluster_stats[s]['data_mean'] * mask, v
min = .4, vmax = .6,#vmin=0.18, vmax=0.48,
                                        cmap='RdBu_r', origin='lower', ex
tent=ext)
    axes[rix, cix].axhline(0., color='k')
    axes[rix, cix].axvline(0., color='k')
    axes[rix, cix].xaxis.set_ticks_position('bottom')
    axes[rix, cix].set_xlabel('Testing Time (s)')
    axes[rix, cix].set_ylabel('Training Time (s)')
    axes[rix, cix].set_anchor('W')
    axes[rix, cix].set_title('pred. {}'.format(s),{'horizontalalignment
': 'center'})
cbar_ax = fig.add_axes([0.925,0.15,0.01,0.7])
fig.colorbar(im,cax=cbar_ax)
fig.suptitle('Decoding accuracy (ROC - AUC)', fontsize =  20)
plt.tight_layout()
#plt.savefig(avg_path + '/figures/{}_accuracies_imagined.pdf'.format(su
b),orientation='landscape')
```

```
/tmp/ipykernel_16790/696177372.py:22: UserWarning: This figure includ
es Axes that are not compatible with tight_layout, so results might b
e incorrect.
  plt.tight_layout()
```



In [ ]: