In [1]:
```
%config InlineBackend.figure_format = 'svg'
```

In [2]:
```
import os
import shutil
import time

# prevent lengthy SPM output
from nipype.utils.logger import logging, logger, fmlogger, iflogger
#logger.setLevel(logging.getLevelName('CRITICAL'))
#fmlogger.setLevel(logging.getLevelName('CRITICAL'))
#iflogger.setLevel(logging.getLevelName('CRITICAL'))

import numpy as np
from scipy.stats.stats import pearsonr, spearmanr
from scipy.stats import wilcoxon
import sklearn as sk
from sklearn.linear_model.base import BaseEstimator, RegressorMixin
import sklearn.metrics as skm
import sklearn.cross_validation as cv
import matplotlib
#matplotlib.use('Agg')
#import matplotlib.pyplot as plt


#from nipype.utils.config import config
#config.enable_debug_mode()

import nipype.pipeline.engine as pe

from spm_2lvl import do_spm         #spm workflow --> give directory + confiles
from feature_selection import determine_model_all
from cluster_tools import get_clustermeans
from cfutils import get_subjects, get_subject_data
```

```
INFO:interface:stdout 2011-12-06T13:58:42.180976:/software/matlab_versions/2010b/bin//matlab
```

In [3]:
```
X = get_subjects()
_, pdata = get_subject_data(X)
X = pdata.subject
y = pdata.lsas_pre - pdata.lsas_post
dcsidx = np.nonzero(pdata.classtype==2)[0]
pcbidx = np.nonzero(pdata.classtype==3)[0]
```

In [4]:
```
#wf = do_spm(X, y, analname='all_subjects', run_workflow=False)
#wf.base_dir = os.path.realpath('..')
#wf.run()
```

get cluster coordinates

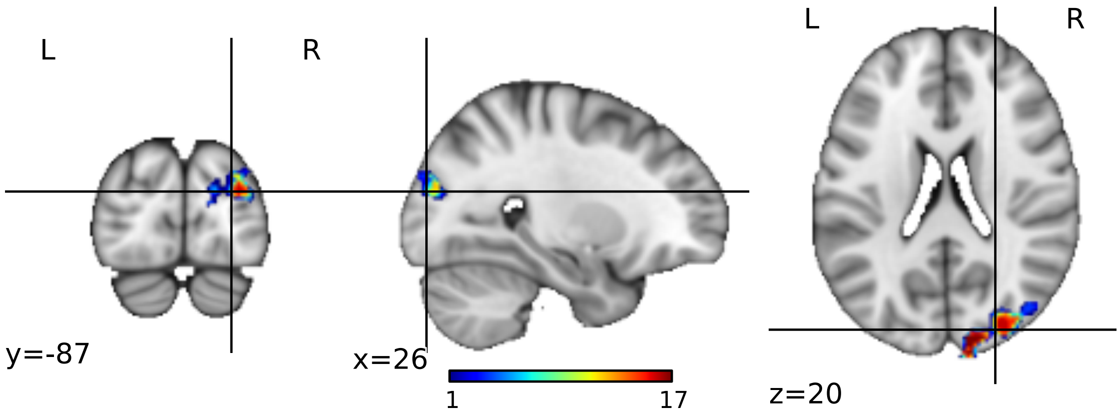
In [5]:
```
def get_coords(img, affine):
    coords = []
    labels = np.setdiff1d(np.unique(img.ravel()), [0])
    cs = []
    for label in labels:
        cs.append(np.sum(img==label))
    for label in labels[argsort(cs)[::-1]]:
        coords.append(np.dot(affine,
                             np.hstack((np.mean(np.asarray(np.nonzero(img==label)),
                                                axis = 1),
                                        1)))[:3].tolist())
    return coords
```

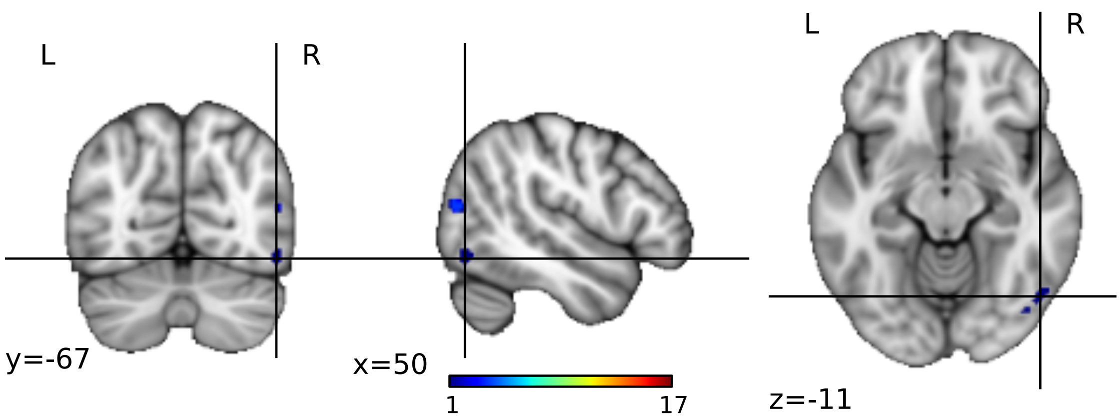
In [6]:
```
from nipy.labs import viz
from nibabel import load
def show_slices(img, coords=None, threshold=0.1, cmap=None, prefix=None,
                show_colorbar=None, formatter='%.2f'):
    if cmap is None:
        cmap = pylab.cm.hot
    data, aff = img.get_data(), img.get_affine()
    anatimg = load('/usr/share/fsl/data/standard/MNI152_T1_1mm_brain.nii.gz')
    anatdata, anataff = anatimg.get_data(), anatimg.get_affine()
    anatdata = anatdata.astype(np.float)
    anatdata[anatdata<10.] = np.nan
    outfile = 'cluster.svg'
    if prefix:
        outfile = '_'.join((prefix, outfile))
    outfile = os.path.join('figures', outfile)
    if coords is None:
        osl = viz.plot_map(np.asarray(data), aff, threshold=threshold,
                           cmap=cmap, black_bg=False)
        osl.frame_axes.figure.savefig(outfile, transparent=True)
    else:
        for idx,coord in enumerate(coords):
            outfile = 'cluster%02d' % idx
            if prefix:
                outfile = '_'.join((prefix, outfile))
            outfile = os.path.join('figures', outfile)
            osl = viz.plot_map(np.asarray(data), aff, anat=anatdata, anat_affine=anataff,
                               threshold=threshold, cmap=cmap,
                               black_bg=False, cut_coords=coord)
            if show_colorbar:
                cb = colorbar(gca().get_images()[1], cax=axes([0.4, 0.075, 0.2, 0.025]),
                         orientation='horizontal', format=formatter)
                cb.set_ticks([cb._values.min(), cb._values.max()])
                show()
            osl.frame_axes.figure.savefig(outfile+'.svg', bbox_inches='tight', transparent=Tru
            osl.frame_axes.figure.savefig(outfile+'.png', dpi=600, bbox_inches='tight', transp
```

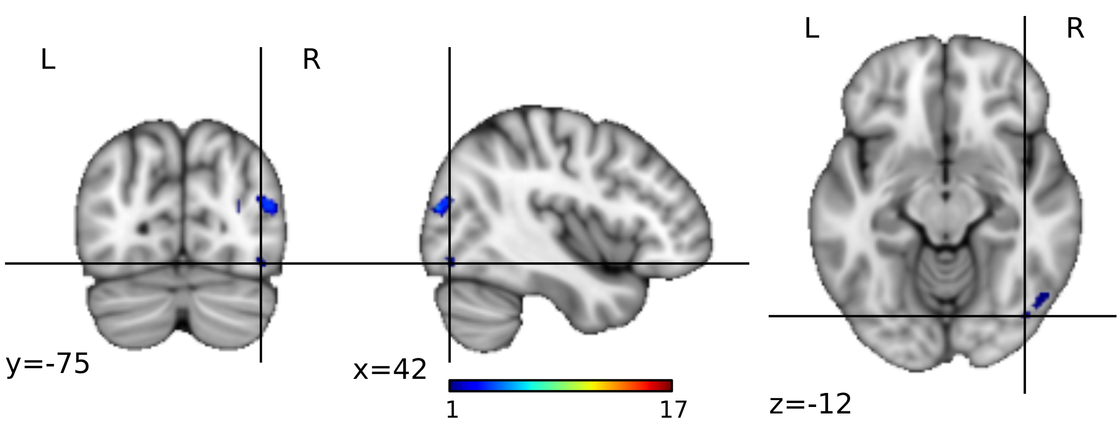
In [7]:
```
def plot_regression_line(x,y, xlim, color='r'):
    model=sk.linear_model.LinearRegression().fit(x[:,None],y)
    xplot = np.arange(xlim[0], xlim[1])[:,None]
    plot(xplot, model.predict(xplot), color=color)
```

In [8]:
```
import os
from scipy.ndimage import label
import scipy.stats as ss
def get_labels(data, min_extent=5):
    labels, nlabels = label(data)
    for idx in range(1, nlabels+1):
        if sum(labels==idx)<min_extent:
            labels[labels==idx] = 0
    return labels, nlabels
```

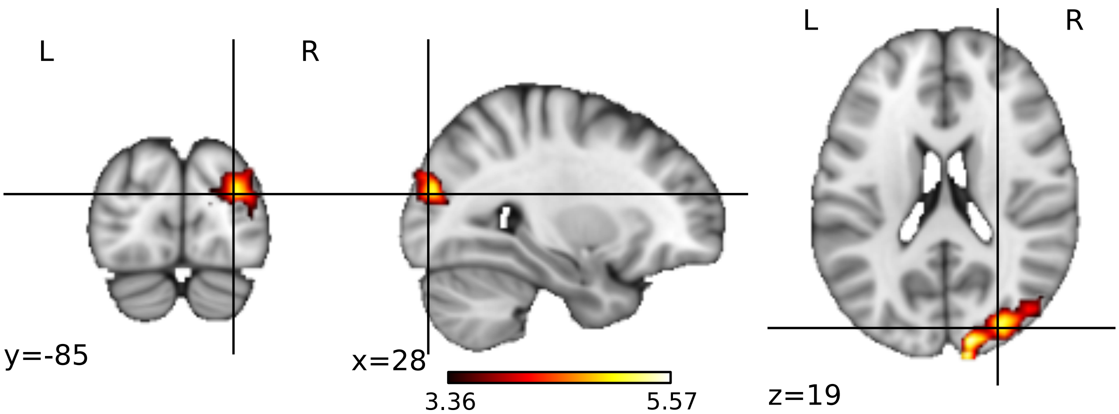
In [9]:
```
base_dir = '/mindhive/gablab/satra/sad/'
filename = os.path.join(base_dir, 'scripts', 'clustermean.nii.gz')
img=load(filename)
labels, nlabels = label(abs(img.get_data())>0)
coords = get_coords(labels, img.get_affine())
show_slices(img, coords, cmap=pylab.cm.jet, prefix='overlap', show_colorbar=True,
            formatter='%d')
```

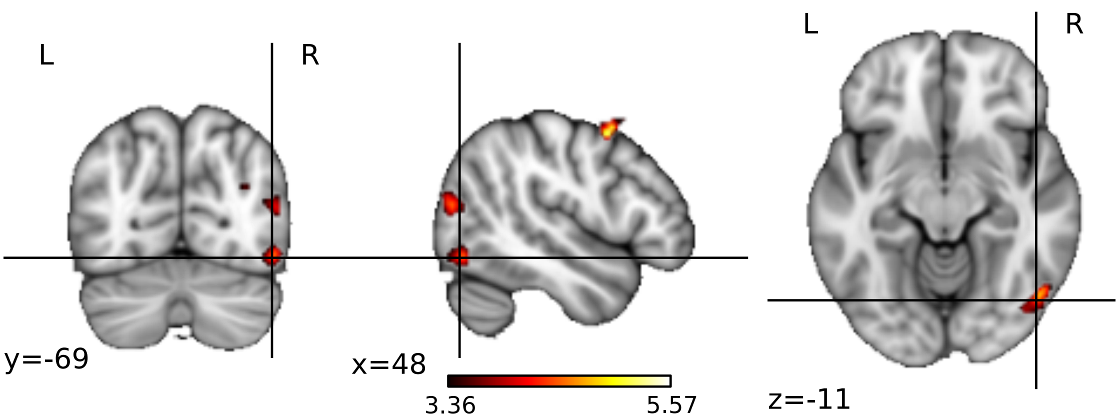


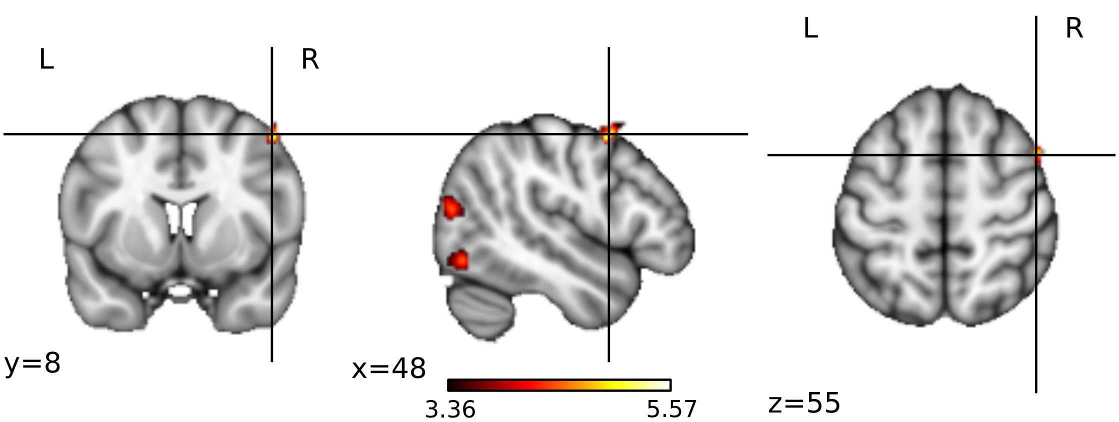




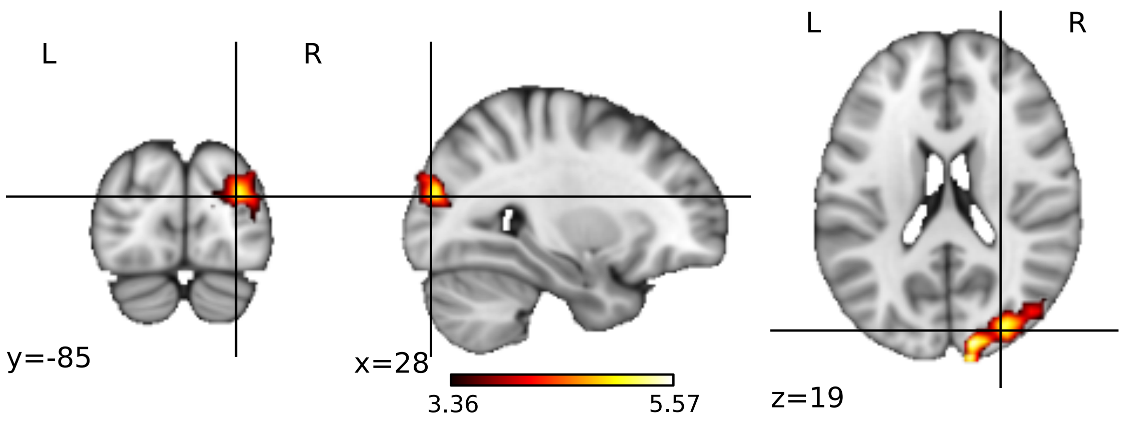
In [10]:
```
base_dir = '/mindhive/gablab/satra/sad/'
filename = os.path.join(base_dir, 'all_subjects', 'conest', 'spmT_0001.img')
img=load(filename)
labels, nlabels = get_labels(img.get_data()>ss.t.ppf(1-0.001,33), 20)
data = img.get_data()
data[labels==0] = 0
#cmeans = get_clustermeans(X, labels, nlabels)
coords = get_coords(labels, img.get_affine())
show_slices(img, coords, threshold=0.5, prefix='uncorrected', show_colorbar=True)
```

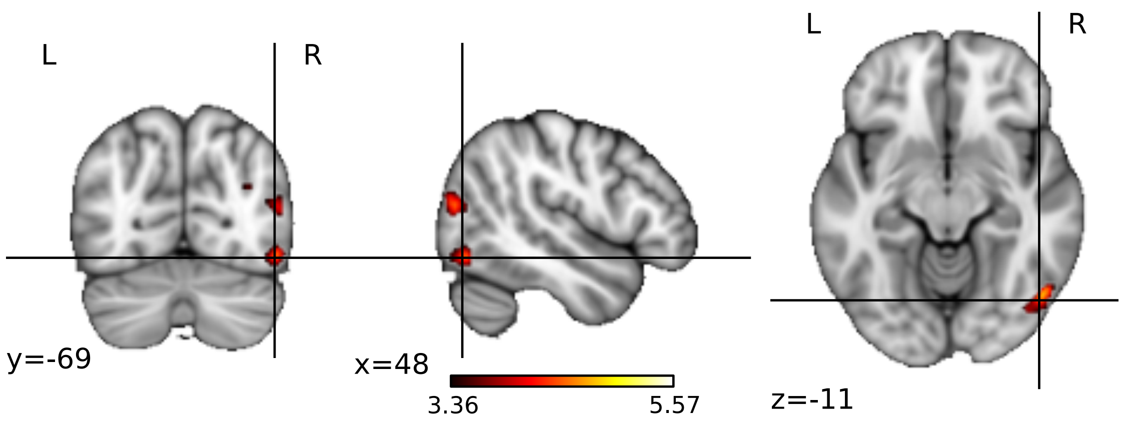






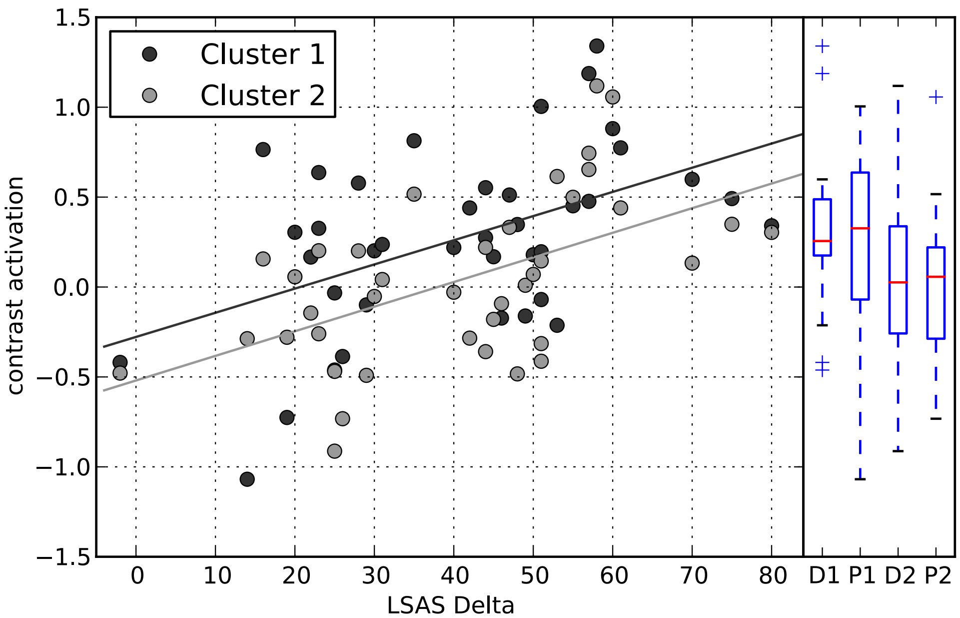
In [11]:
```
import os
from scipy.ndimage import label
base_dir = '/mindhive/gablab/satra/sad/'
filename = os.path.join(base_dir, 'all_subjects', 'thresh', 'spmT_0001_thr.img')
img=load(filename)
labels, nlabels = label(abs(img.get_data())>0)
cmeans = get_clustermeans(X, labels, nlabels)
coords = get_coords(labels, img.get_affine())
show_slices(img, coords, prefix='topocorrect', show_colorbar=True)
```





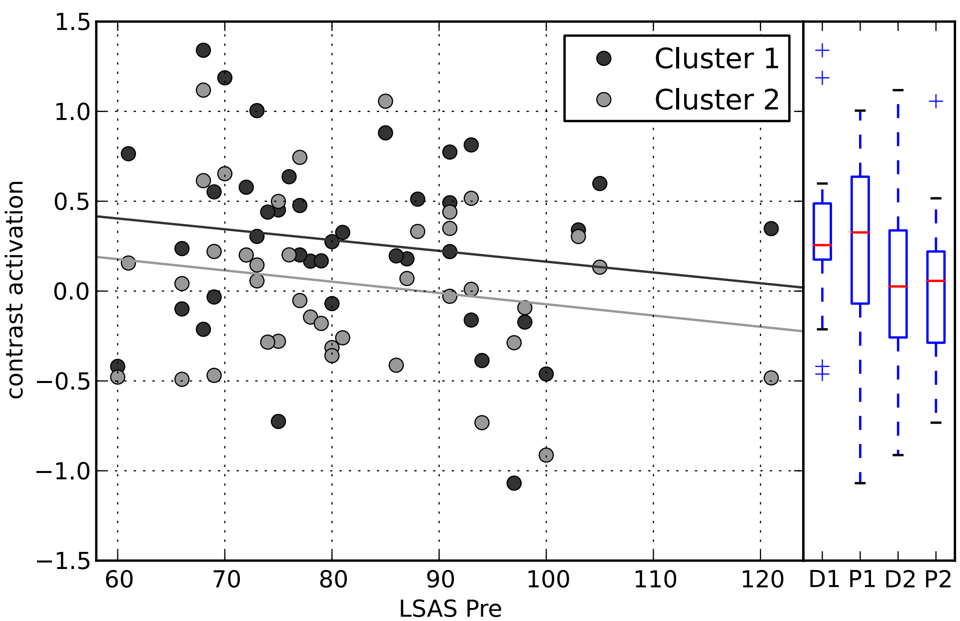
In [12]:
```
close('all')
axes([0.1,0.1,0.7,0.8])
plot(y, cmeans[:,0], 'o', color=[0.2,0.2,0.2])
plot(y, cmeans[:,1], 'o', color=[0.6,0.6,0.6])
xlim([-5, 84])
xlabel('LSAS Delta')
ylabel('contrast activation')
legend(('Cluster 1', 'Cluster 2'), 'best', numpoints=1)
plot_regression_line(y, cmeans[:,0], [-4,85], color=[0.2,0.2,0.2])
plot_regression_line(y, cmeans[:,1], [-4,85], color=[0.6,0.6,0.6])
grid()
axes([0.8,0.1,0.15,0.8])
boxplot([cmeans[dcsidx,0], cmeans[pcbidx,0], cmeans[dcsidx,1], cmeans[pcbidx,1]])
yticks([])
xticks([1,2,3,4], ['D1', 'P1', 'D2', 'P2'])
savefig('figures/scatter_means_all.svg')
savefig('figures/scatter_means_all.png', dpi=600)
print 'r: C1', pearsonr(cmeans[:,0], y)
print 'r: C2', pearsonr(cmeans[:,1], y)
```

```
r: C1 (0.48514858956652174, 0.0017459420489864509)
r: C2 (0.54136848262878923, 0.00037241303817518978)
```



In [13]:
```
close('all')
axes([0.1,0.1,0.7,0.8])
plot(pdata.lsas_pre, cmeans[:,0], 'o', color=[0.2,0.2,0.2])
plot(pdata.lsas_pre, cmeans[:,1], 'o', color=[0.6,0.6,0.6])
xlim([58, 124])
xlabel('LSAS Pre')
ylabel('contrast activation')
legend(('Cluster 1', 'Cluster 2'), 'best', numpoints=1)
plot_regression_line(pdata.lsas_pre, cmeans[:,0], [57, 125], color=[0.2,0.2,0.2])
plot_regression_line(pdata.lsas_pre, cmeans[:,1], [57, 125], color=[0.6,0.6,0.6])
grid()
axes([0.8,0.1,0.15,0.8])
boxplot([cmeans[dcsidx,0], cmeans[pcbidx,0], cmeans[dcsidx,1], cmeans[pcbidx,1]])
yticks([])
xticks([1,2,3,4], ['D1', 'P1', 'D2', 'P2'])
savefig('figures/scatter_means_all_lsaspre.svg')
savefig('figures/scatter_means_all_lsaspre.png', dpi=600)
print 'r: C1', pearsonr(cmeans[:,0], pdata.lsas_pre)
print 'r: C2', pearsonr(cmeans[:,1], pdata.lsas_pre)
print 'r: C1D', pearsonr(cmeans[dcsidx,0], pdata.lsas_pre[dcsidx])
print 'r: C2D', pearsonr(cmeans[dcsidx,1], pdata.lsas_pre[dcsidx])
print 'r: C1P', pearsonr(cmeans[pcbidx,0], pdata.lsas_pre[pcbidx])
print 'r: C2P', pearsonr(cmeans[pcbidx,1], pdata.lsas_pre[pcbidx])
```

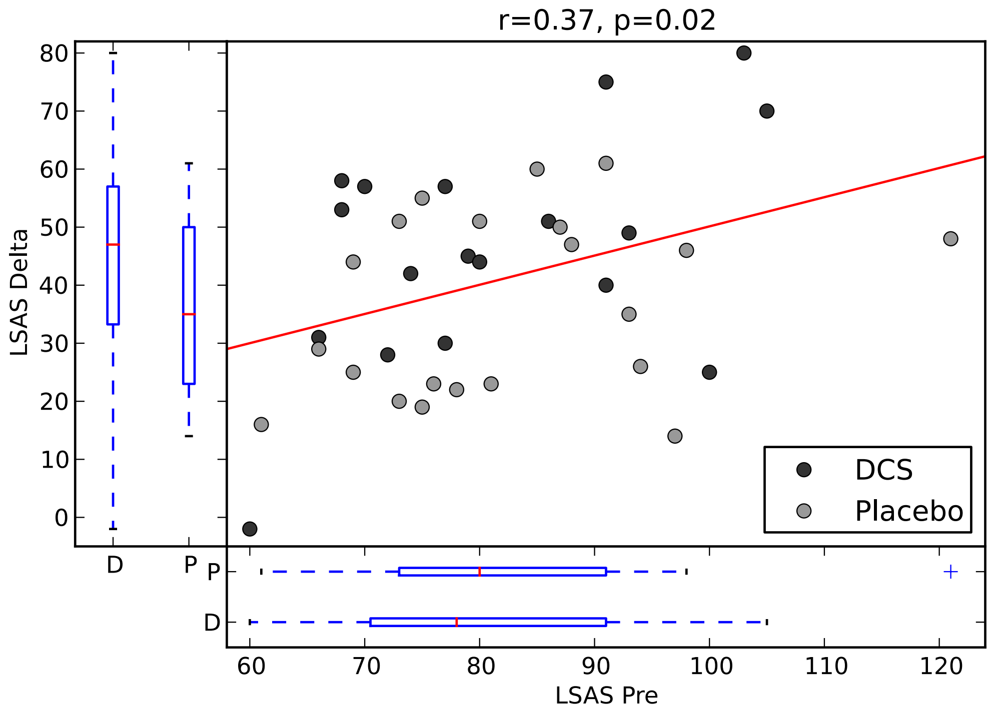
```
r: C1 (-0.16095048699429301, 0.32766201406019846)
r: C2 (-0.18417747913131668, 0.261692057110544)
r: C1D (-0.14358866223194502, 0.56975027637211828)
r: C2D (-0.24952790428404886, 0.31800409852280737)
r: C1P (-0.16990164301777813, 0.46155369765786414)
r: C2P (-0.1158078786499194, 0.61715284762787637)
```



In [14]:
```
def Rmodel(y_true, y_pred):
    robjects.globalenv['y_true'] = robjects.FloatVector(y_true)
    robjects.globalenv['y_pred'] = robjects.FloatVector(y_pred)
    robjects.r("model = lm('y_true~y_pred')")
    print robjects.r("summary(model)")
```

In [15]:
```
close('all')
a1 = axes([0.05, 0.2, 0.15, 0.75])
boxplot([y[dcsidx], y[pcbidx]])
ylim([-5, 82])
ylabel('LSAS Delta')
xticks([1,2],('D','P'))
a2 = axes([0.2, 0.05, 0.75, 0.15])
boxplot([pdata.lsas_pre[dcsidx], pdata.lsas_pre[pcbidx]],
        vert=False)
xlim([58, 124])
xlabel('LSAS Pre')
yticks([1,2],('D','P'))
a3 = axes([0.2, 0.2, 0.75, 0.75]) #, sharex=a2, sharey=a1)
plot(pdata.lsas_pre[dcsidx], y[dcsidx], 'o', color=(0.2, 0.2, 0.2))
plot(pdata.lsas_pre[pcbidx], y[pcbidx], 'o', color=(0.6, 0.6, 0.6))
plot_regression_line(pdata.lsas_pre, y, [57, 125])
a3.set_xticks([])
a3.set_yticks([])
ylim([-5, 82])
xlim([58, 124])
grid()
legend(('DCS', 'Placebo'), 'lower right', numpoints=1)
title('r=%.2f, p=%.2f' % pearsonr(y, pdata.lsas_pre))
savefig('figures/corr_pre_delta.svg')
savefig('figures/corr_pre_delta.png', dpi=600)
print 'D', mean(pdata.lsas_pre[dcsidx]), '+-', std(pdata.lsas_pre[dcsidx])
print 'P', mean(pdata.lsas_pre[pcbidx]), '+-', std(pdata.lsas_pre[pcbidx])
```

```
D 81.1111111111 +- 13.0847192
P 82.380952381 +- 13.393232799
```



In [16]:
```
from rpy2 import robjects
from rpy2.robjects.packages import importr
stats = importr('stats')
base = importr('base')
```

In [17]:
```
c1 = robjects.FloatVector(cmeans[:,0])
c2 = robjects.FloatVector(cmeans[:,1])
lsasd = robjects.FloatVector(y)
robjects.globalenv['c1'] = c1
robjects.globalenv['c2'] = c2
robjects.globalenv['lsaspre'] = robjects.FloatVector(pdata.lsas_pre)
robjects.globalenv['group'] = robjects.IntVector(pdata.classtype-2)
robjects.globalenv['lsasd'] = lsasd
m1 = robjects.r("model1 = lm('lsasd~c1 + c2 + lsaspre + lsaspre:group +c1:group + c2:group')")
m2 = robjects.r("model2 = lm('lsasd~lsaspre + lsaspre:group')")
m3 = robjects.r("model3 = lm('lsasd~lsaspre')")
```

In [18]:
```
print robjects.r("summary(model1)")
print robjects.r("summary(model2)")
print robjects.r("summary(model3)")
print robjects.r("anova(model3, model2)")
```

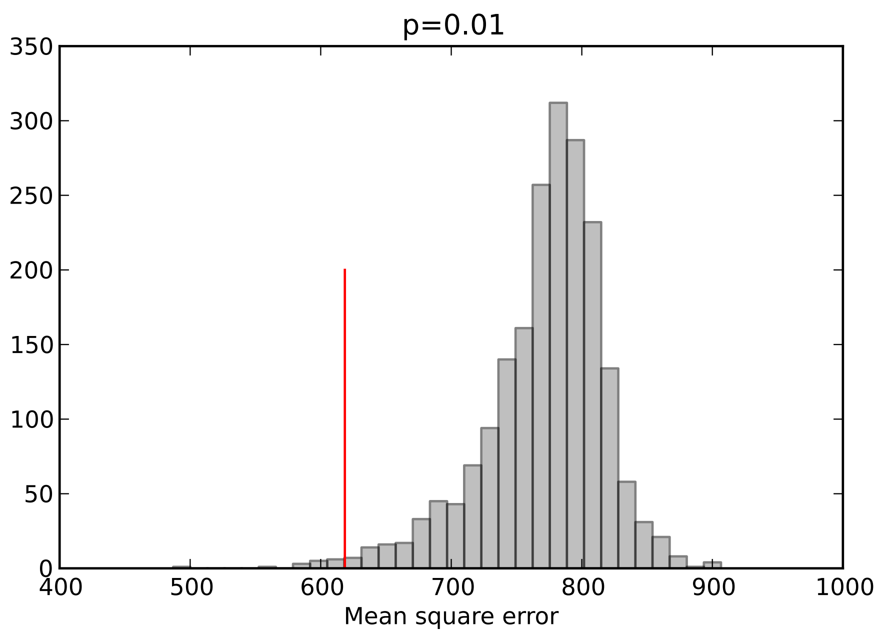
```
Call:
lm(formula = "lsasd~c1 + c2 + lsaspre + lsaspre:group +c1:group + c2:group")

Residuals:
     Min       1Q   Median       3Q      Max 
-19.7886  -9.7221   0.7661   8.5806  23.5959 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   -19.35098   12.25982  -1.578  0.12431    
c1              6.99024    8.57336   0.815  0.42090    
c2             22.81833    8.19825   2.783  0.00895 ** 
lsaspre         0.76585    0.15144   5.057 1.68e-05 ***
lsaspre:group  -0.12637    0.05612  -2.252  0.03133 *  
c1:group        3.65047   10.92309   0.334  0.74041    
c2:group      -11.17317   11.91793  -0.938  0.35552    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 11.81 on 32 degrees of freedom
Multiple R-squared: 0.6417,	Adjusted R-squared: 0.5745 
F-statistic: 9.551 on 6 and 32 DF,  p-value: 4.967e-06 



Call:
lm(formula = "lsasd~lsaspre + lsaspre:group")

Residuals:
    Min      1Q  Median      3Q     Max 
-35.973 -11.985  -0.339  14.132  22.628 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)   
(Intercept)   -1.88058   16.26468  -0.116  0.90859   
lsaspre        0.59756    0.20092   2.974  0.00522 **
lsaspre:group -0.13576    0.06312  -2.151  0.03828 * 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 16.24 on 36 degrees of freedom
Multiple R-squared: 0.2374,	Adjusted R-squared: 0.1951 
F-statistic: 5.604 on 2 and 36 DF,  p-value: 0.007604 



Call:
lm(formula = "lsasd~lsaspre")

Residuals:
    Min      1Q  Median      3Q     Max 
-34.623 -13.605   2.389  14.922  29.395 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)  -0.1687    17.0226  -0.010   0.9921  
lsaspre       0.5030     0.2054   2.449   0.0192 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 17.02 on 37 degrees of freedom
Multiple R-squared: 0.1394,	Adjusted R-squared: 0.1162 
```

In [19]:
```
from sklearn.linear_model import LinearRegression
import sklearn.cross_validation as cv
result = []
Xnew = np.vstack((pdata.lsas_pre, pdata.lsas_pre*(pdata.classtype-2))).T
for train, test in cv.StratifiedKFold(pdata.classtype, 18):
    model = LinearRegression()
    model.fit(Xnew[train], y[train])
    result.append([y[test], model.predict(Xnew[test])])
result_lsas = result
y_true = []; y_pred = []
for a,b in result:
    y_true.extend(a.tolist())
    y_pred.extend(b.tolist())
result = np.array(np.vstack((y_true, y_pred))).T
```

In [20]:
```
value, distribution, pvalue = cv.permutation_test_score(LinearRegression(), Xnew, y,
                                                        score_func=skm.mean_square_error,
                                                        cv=cv.StratifiedKFold(pdata.classtype,
                                                        n_permutations=2000,
                                                        )
```

In [21]:
```
hist(distribution, 32, alpha=0.5, color='gray')
plot([value, value], [0,200], 'r')
title('p=%.2f' % (1-pvalue))
xlabel('Mean square error')
```

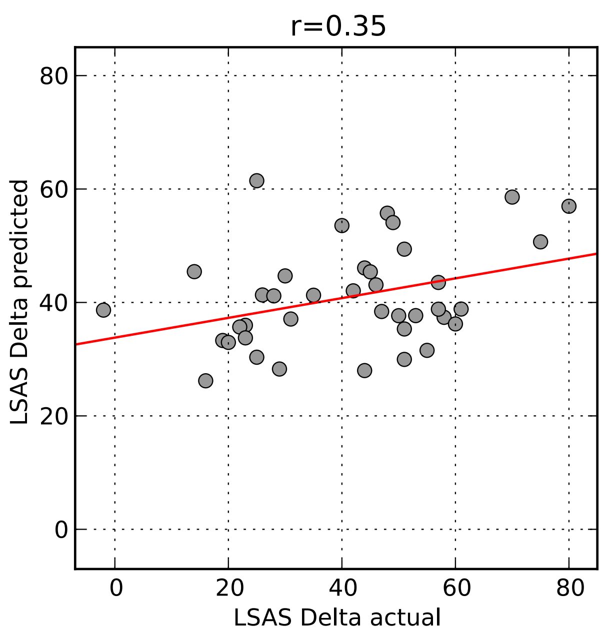
Out[21]: <matplotlib.text.Text at 0x7fd86f160f10>



In [22]:
```
print np.corrcoef(result.T)
Rmodel(result.T[0], result.T[1])
```

```
[[ 1.          0.35391993]
 [ 0.35391993  1.        ]]

Call:
lm(formula = "y_true~y_pred")

Residuals:
    Min      1Q  Median      3Q     Max 
-41.329 -13.401  -0.674  14.265  27.500 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)  11.5355    13.0814   0.882   0.3836  
y_pred        0.7192     0.3124   2.302   0.0271 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 17.16 on 37 degrees of freedom
Multiple R-squared: 0.1253,     Adjusted R-squared: 0.1016 
F-statistic: 5.298 on 1 and 37 DF,  p-value: 0.02708
```

In [23]:
```
plot(result[:,0], result[:,1], 'o', color=[0.6,0.6,0.6])
minv = np.min(result)-5
maxv = np.max(result)+5
plot_regression_line(result[:,0], result[:,1], [minv-1, maxv+1], color='r')
xlabel('LSAS Delta actual')
ylabel('LSAS Delta predicted')
axis('scaled')
ylim([minv, maxv])
xlim([minv, maxv])
grid()
title('r=%.2f' % np.corrcoef(result.T)[0,1])
savefig('figures/loo_lsaspre.svg')
savefig('figures/loo_lsaspre.png', dpi=600)
```



In [24]:
```
result = []
Xnew = np.hstack((np.vstack((pdata.lsas_pre, pdata.lsas_pre*(pdata.classtype-2))).T,
                  cmeans))
for train, test in cv.StratifiedKFold(pdata.classtype, 18):
    model = LinearRegression()
    model.fit(Xnew[train], y[train])
    result.append([y[test], model.predict(Xnew[test])])
y_true = []; y_pred = []
for a,b in result:
    y_true.extend(a.tolist())
    y_pred.extend(b.tolist())
result = np.array(np.vstack((y_true, y_pred))).T
```

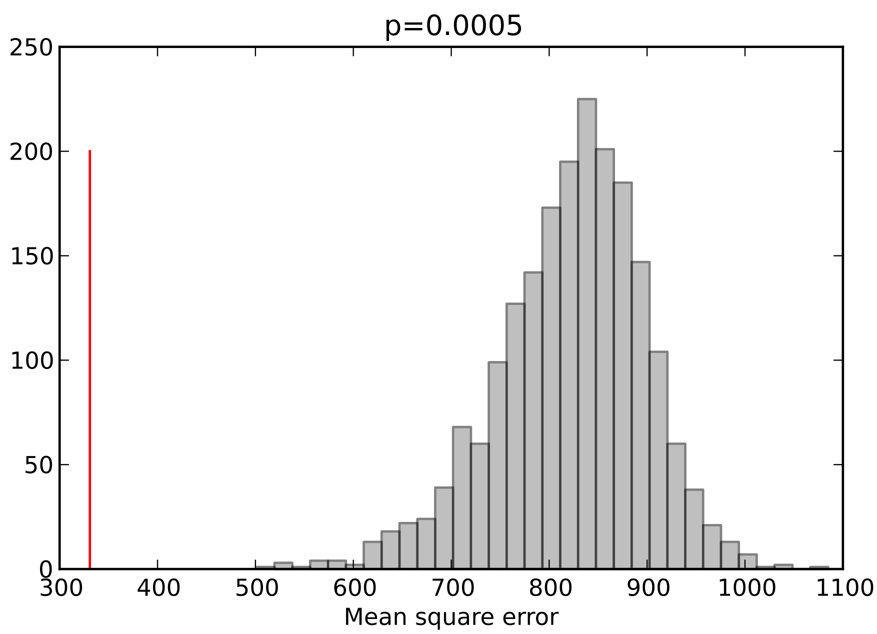
In [25]:
```
np.corrcoef(result.T)
```

Out[25]:
```
array([[ 1.        ,  0.72534911],
       [ 0.72534911,  1.        ]])
```

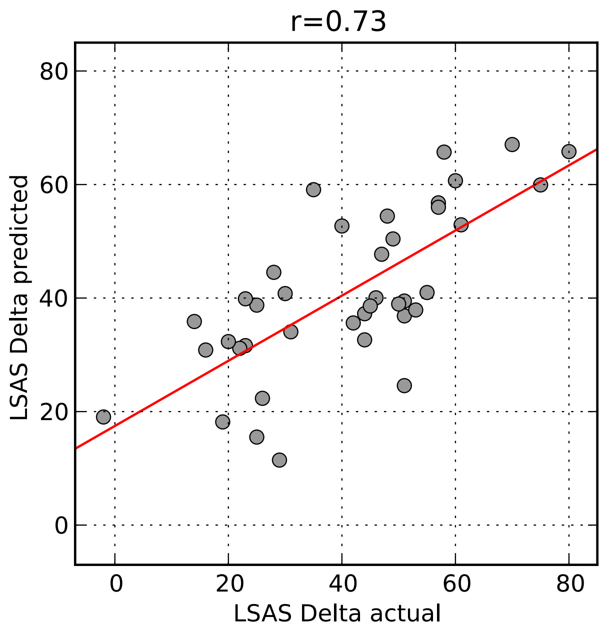
In [26]:
```
value, distribution, pvalue = cv.permutation_test_score(LinearRegression(), Xnew, y,
                                                        score_func=skm.mean_square_error,
                                                        cv=cv.StratifiedKFold(pdata.classtype,
                                                        n_permutations=2000,
                                                        )
```

In [27]:
```
pvalue = min(pvalue, 1-1./2000)
hist(distribution, 32, alpha=0.5, color='gray')
plot([value, value], [0,200], 'r')
title('p=%.4f' % (1-pvalue))
xlabel('Mean square error')
```

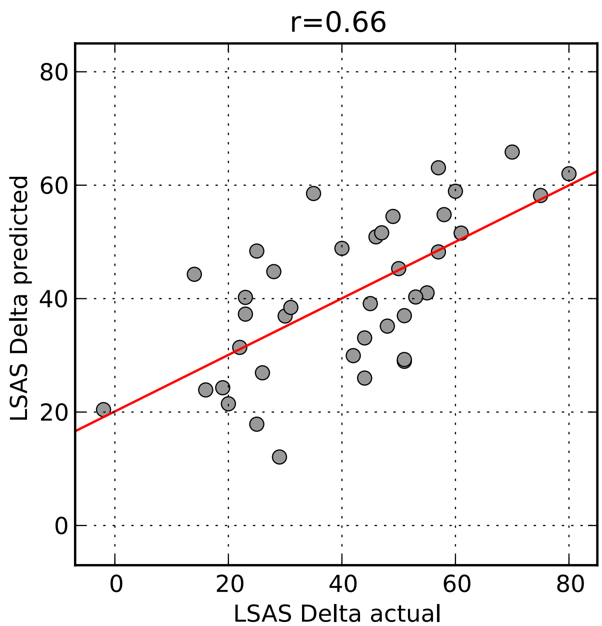
Out[27]:
```
<matplotlib.text.Text at 0x7fd86f1fe490>
```



In [28]:
```
plot(result[:,0], result[:,1], 'o', color=[0.6,0.6,0.6])
xlabel('LSAS Delta actual')
ylabel('LSAS Delta predicted')
minv = np.min(result)-5
maxv = np.max(result)+5
plot_regression_line(result[:,0], result[:,1], [minv-1, maxv+1], color='r')
axis('scaled')
ylim([minv, maxv])
xlim([minv, maxv])
grid()
title('r=%.2f' % np.corrcoef(result.T)[0,1])
savefig('figures/loo_group_cluster.svg')
savefig('figures/loo_group_cluster.png', dpi=600)
```



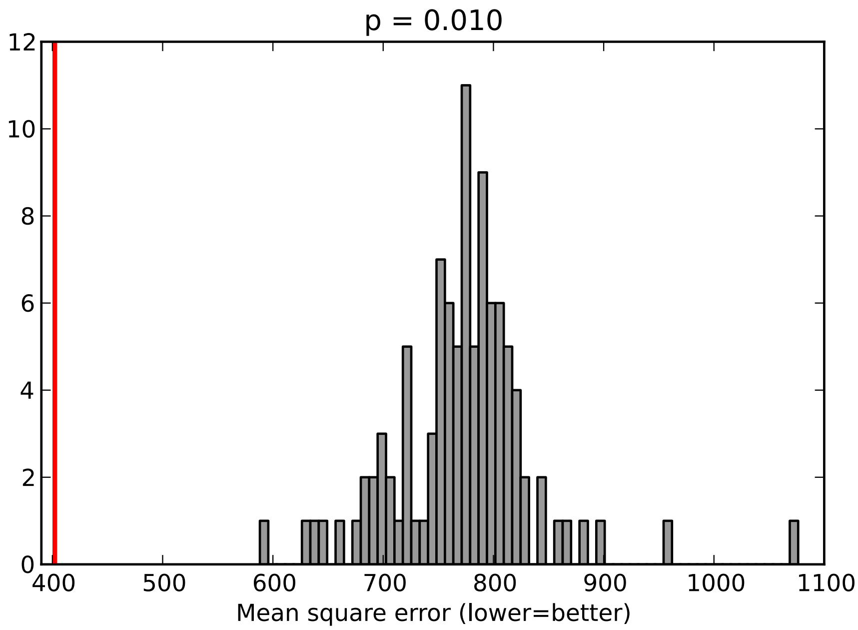
In [29]:
```
cvres = np.load('result_cv.npz')
minv = np.min(cvres['aout'])-5
maxv = np.max(cvres['aout'])+5
plot(cvres['aout'][:,0], cvres['aout'][:,1], 'o', color=[0.6,0.6,0.6])
plot_regression_line(cvres['aout'][:,0], cvres['aout'][:,1], [minv-1, maxv+1], color='r')
xlabel('LSAS Delta actual')
ylabel('LSAS Delta predicted')
axis('scaled')
ylim([minv, maxv])
xlim([minv, maxv])
grid()
title('r=%.2f' % np.corrcoef(cvres['aout'].T)[0,1])
savefig('figures/fullcv_results.svg')
savefig('figures/fullcv_results.png', dpi=600)
```



In [30]:
```
skm.explained_variance_score(cvres['aout'][:,0], cvres['aout'][:,1])
Rmodel(cvres['aout'][:,0], cvres['aout'][:,1])
```

```
Call:
lm(formula = "y_true~y_pred")

Residuals:
    Min      1Q  Median      3Q     Max
-30.217  -8.168   3.147  11.093  20.483

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.9900     6.9811   0.858    0.396
y_pred        0.8631     0.1633   5.285 5.83e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.85 on 37 degrees of freedom
Multiple R-squared: 0.4302,     Adjusted R-squared: 0.4148
F-statistic: 27.93 on 1 and 37 DF,  p-value: 5.829e-06
```

In [31]:
```
permdata = np.load('100iter.npz')
hist(permdata['distribution'], 64, color=[0.6,0.6,0.6])
plot([permdata['value'], permdata['value']], [0, 12], color='r', linewidth=2)
title('p = %.3f' % max(1./100, (1-permdata['pvalue'])))
xlim([390, 1100])
xlabel('Mean square error (lower=better)')
savefig("figures/permtest_hist.svg")
savefig("figures/permtest_hist.png", dpi=600)
```

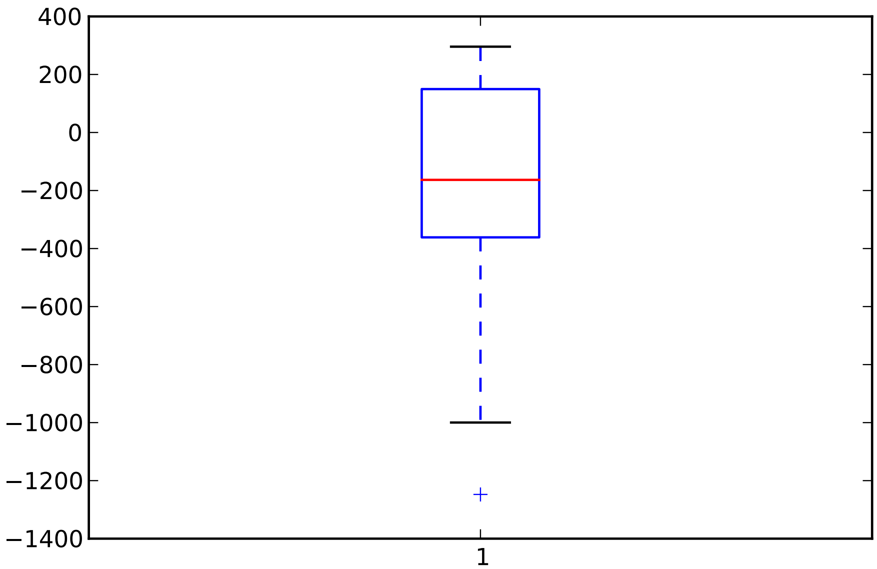


In [32]:
```
msedata = []
for idx, res in enumerate(result_lsas):
    msedata.append((skm.mean_square_error(res[0], res[1]),
                    skm.mean_square_error(cvres['result'][idx][0],
                                          cvres['result'][idx][1])))
```

In [33]:
```
print wilcoxon(np.diff(msedata, axis=1).ravel())
boxplot(np.diff(msedata, axis=1))
```

```
(44.0, 0.070709320478686236)
```

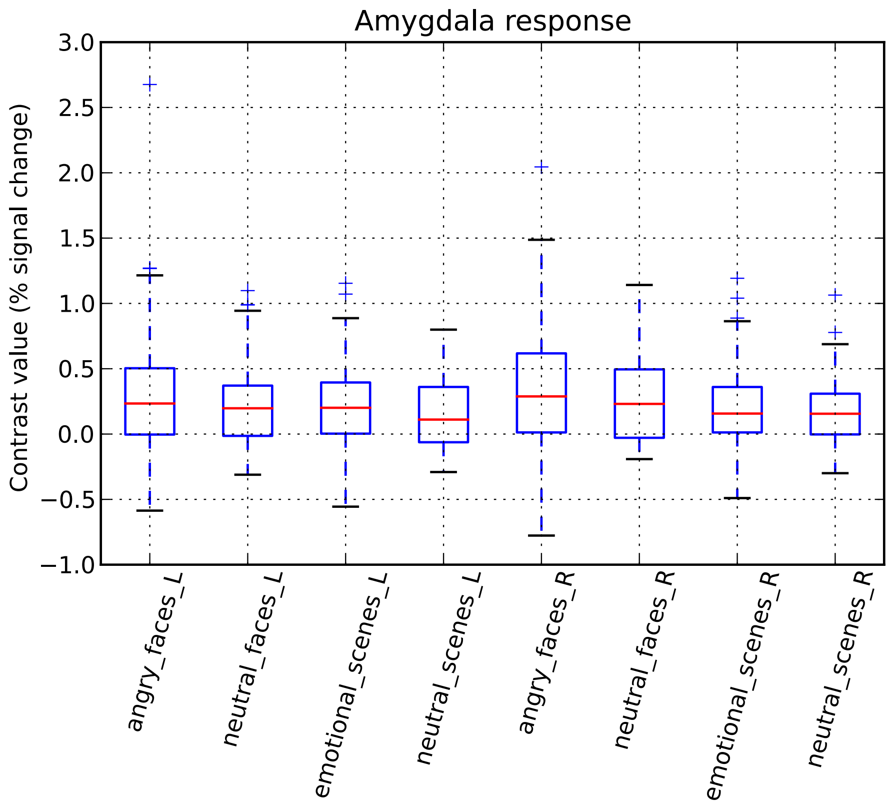
Out[33]:
```
{'boxes': [<matplotlib.lines.Line2D at 0x7fd86d176790>],
 'caps': [<matplotlib.lines.Line2D at 0x7fd86d03a810>,
  <matplotlib.lines.Line2D at 0x7fd86d176110>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fd86d176f10>,
  <matplotlib.lines.Line2D at 0x7fd86f8da310>],
 'medians': [<matplotlib.lines.Line2D at 0x7fd86d176b50>],
 'whiskers': [<matplotlib.lines.Line2D at 0x7fd86d03a7d0>,
  <matplotlib.lines.Line2D at 0x7fd86d03a190>]}
```



Amygdala responses

In [34]:
```
amygdata = recfromcsv('AmygdalaResponses.csv', names=True)
amygX = amygdata.view(np.float64).reshape(39,8)
names = []
for name in amygdata.dtype.names:
    if '_1' in name:
        names.append(name.replace('_1','_R'))
    else:
        names.append(name+'_L')
```

In [35]:
```
bp = boxplot(amygX)
xticks(arange(1,9), names, rotation=75)
ylabel("Contrast value (% signal change)")
grid()
title('Amygdala response')
savefig('figures/amygdala_response.svg', bbox_inches='tight')
savefig('figures/amygdala_response.png', dpi=600, bbox_inches='tight')
```

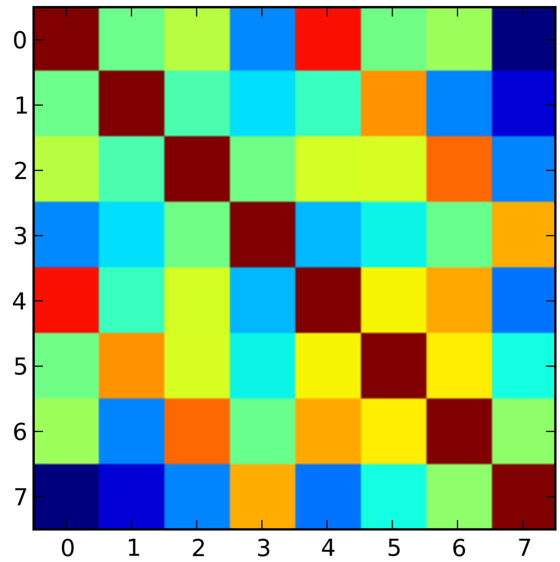


In [36]:
```
robjects.globalenv['y_true'] = robjects.FloatVector(y)
robjects.globalenv['lsaspre'] = robjects.FloatVector(pdata.lsas_pre)
robjects.globalenv['group'] = robjects.IntVector(pdata.classtype-2)
for i,name in enumerate(names):
    robjects.globalenv[name] = robjects.FloatVector(amygX[:,i])
m1str = 'y_true~lsaspre + lsaspre:group + %s + %s' % ('+'.join(names), ':group +'.join(names))
m1 = robjects.r("m1 = lm(%s)" % m1str)
print robjects.r("summary(m1)")
m2 = robjects.r("m2 = lm('y_true~lsaspre + lsaspre:group + angry_faces_R + angry_faces_R:group
print robjects.r("summary(m2)")
m3 = robjects.r("m3 = lm('y_true~lsaspre + lsaspre:group')")
print robjects.r("anova(m3,m1)")
```

```
Call:
lm(formula = y_true ~ lsaspre + lsaspre:group + angry_faces_L +
    neutral_faces_L + emotional_scenes_L + neutral_scenes_L +
    angry_faces_R + neutral_faces_R + emotional_scenes_R + neutral_scenes_R +
    angry_faces_L:group + neutral_faces_L:group + emotional_scenes_L:group +
    neutral_scenes_L:group + angry_faces_R:group + neutral_faces_R:group +
    emotional_scenes_R:group + neutral_scenes_R)

Residuals:
    Min      1Q  Median      3Q     Max
-37.896  -5.868  -1.560   9.892  26.182

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)               10.41843   24.43309   0.426   0.6742
lsaspre                    0.41167    0.31164   1.321   0.2007
angry_faces_L             -7.22086   42.04757  -0.172   0.8653
neutral_faces_L          -65.96567   59.87470  -1.102   0.2830
emotional_scenes_L        29.57672   27.21910   1.087   0.2895
neutral_scenes_L          24.18155   63.63084   0.380   0.7077
angry_faces_R             24.37981   58.96838   0.413   0.6835
neutral_faces_R           60.95377   63.69678   0.957   0.3495
emotional_scenes_R       -75.37781   40.66474  -1.854   0.0779 .
neutral_scenes_R           6.08784   49.26220   0.124   0.9028
lsaspre:group             -0.08846    0.10475  -0.845   0.4079
group:angry_faces_L       -5.13978   51.69948  -0.099   0.9218
group:neutral_faces_L     80.38850   68.62127   1.171   0.2545
group:emotional_scenes_L -38.48076   44.63915  -0.862   0.3984
group:neutral_scenes_L   -11.52532   50.43130  -0.229   0.8214
group:angry_faces_R       -8.67095   70.62746  -0.123   0.9035
group:neutral_faces_R    -95.08737   77.81603  -1.222   0.2353
group:emotional_scenes_R  89.63743   66.07628   1.357   0.1893
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.3 on 21 degrees of freedom
Multiple R-squared: 0.4354,     Adjusted R-squared: -0.02169
F-statistic: 0.9525 on 17 and 21 DF,  p-value: 0.5349



Call:
lm(formula = "y_true~lsaspre + lsaspre:group + angry_faces_R + angry_faces_R:group + neutral_f

Residuals:
    Min      1Q  Median      3Q     Max
-35.533 -11.448  -1.034  13.955  24.262

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           -5.13664   19.33876  -0.266   0.7922
lsaspre                0.61734    0.22797   2.708   0.0108 *
angry_faces_R         -1.41219   10.76728  -0.131   0.8965
neutral_faces_R        6.38532   19.22618   0.332   0.7420
```

In [37]:
```
imshow(corrcoef(amygX.T), interpolation='nearest')
```

Out[37]: <matplotlib.image.AxesImage at 0x7fd86d193ad0>

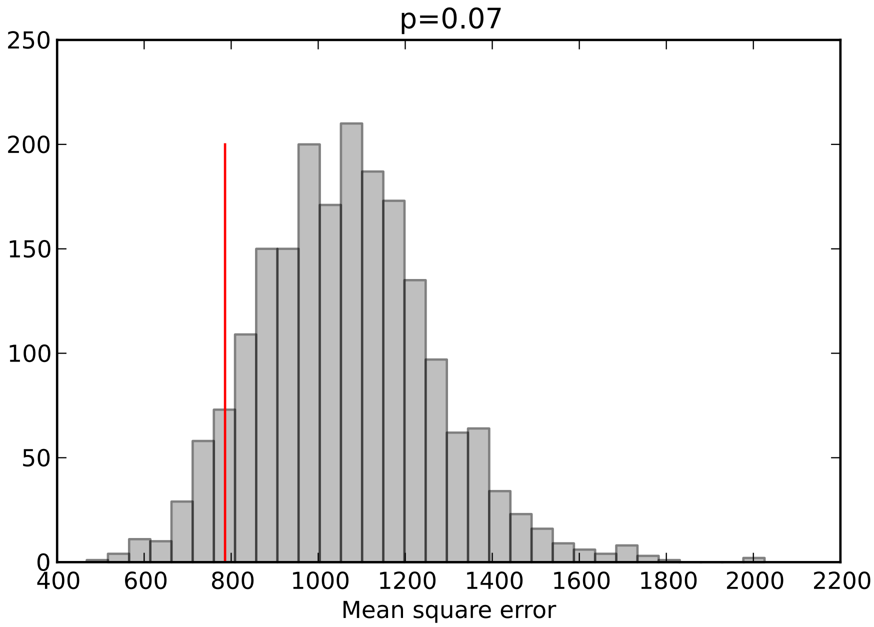


In [38]:
```
result = []
Xnew = np.hstack((np.vstack((pdata.lsas_pre, pdata.lsas_pre*(pdata.classtype-2))).T,
                  amygX))
for train, test in cv.StratifiedKFold(pdata.classtype, 18):
    model = LinearRegression()
    model.fit(Xnew[train], y[train])
    result.append([y[test], model.predict(Xnew[test])])
y_true = []; y_pred = []
for a,b in result:
    y_true.extend(a.tolist())
    y_pred.extend(b.tolist())
result = np.array(np.vstack((y_true, y_pred))).T
```

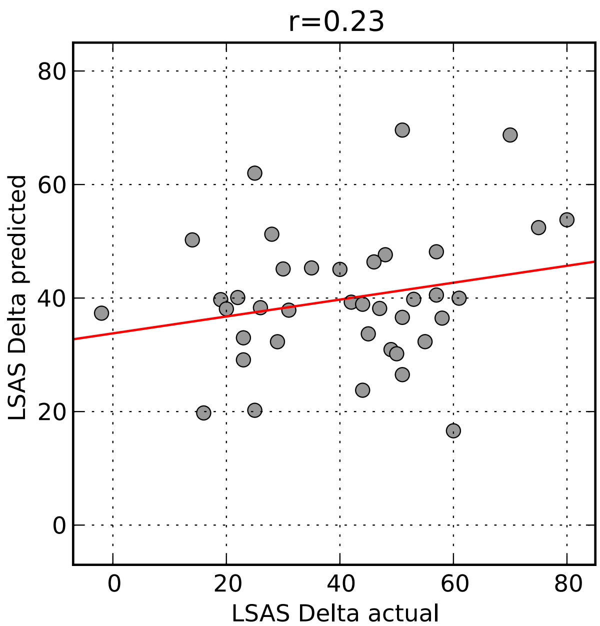
In [39]:
```
value, distribution, pvalue = cv.permutation_test_score(LinearRegression(), Xnew, y,
                                                        score_func=skm.mean_square_error,
                                                        cv=cv.StratifiedKFold(pdata.classtype,
                                                        n_permutations=2000,
                                                        )
```

In [40]:
```
hist(distribution, 32, alpha=0.5, color='gray')
plot([value, value], [0,200], 'r')
title('p=%.2f' % (1-pvalue))
xlabel('Mean square error')
```

Out[40]: <matplotlib.text.Text at 0x7fd86d196050>

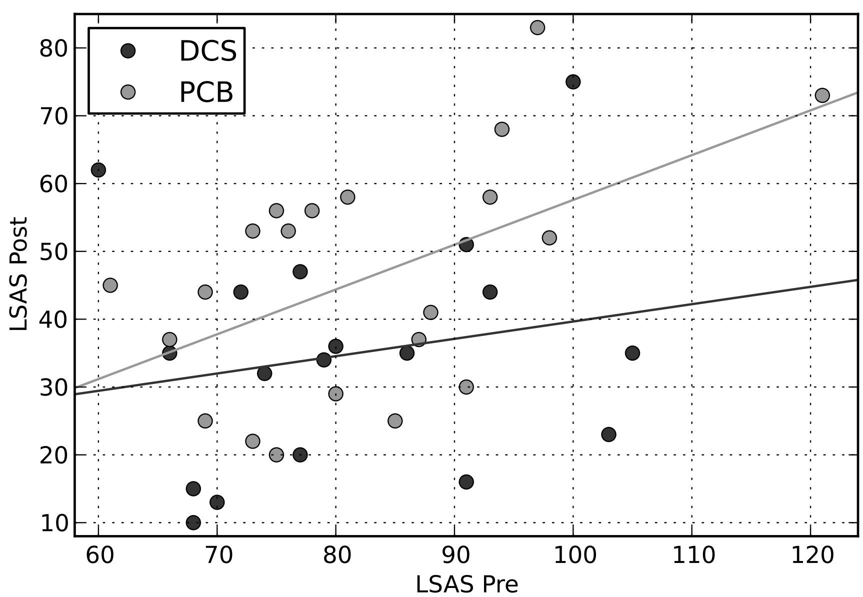


In [41]:
```
plot(result[:,0], result[:,1], 'o', color=[0.6, 0.6, 0.6])
xlabel('LSAS Delta actual')
ylabel('LSAS Delta predicted')
minv = np.min(result)-5
maxv = np.max(result)+5
plot_regression_line(result[:,0], result[:,1], [minv-1, maxv+1], color='r')
axis('scaled')
ylim([minv, maxv])
xlim([minv, maxv])
grid()
title('r=%.2f' % np.corrcoef(result.T)[0,1])
savefig('figures/loo_amygdala.svg')
savefig('figures/loo_amygdala.png', dpi=600)
```



In [42]:
```
print pearsonr(pdata.lsas_pre,pdata.lsas_post)
print pearsonr(pdata.lsas_pre,pdata.lsas_pre-pdata.lsas_post)
print 'DP:', pearsonr(pdata.lsas_pre[dcsidx],pdata.lsas_post[dcsidx])
print 'PP:', pearsonr(pdata.lsas_pre[pcbidx],pdata.lsas_post[pcbidx])
print 'DD:',pearsonr(pdata.lsas_pre[dcsidx],pdata.lsas_pre[dcsidx]-pdata.lsas_post[dcsidx])
print 'PD:',pearsonr(pdata.lsas_pre[pcbidx],pdata.lsas_pre[pcbidx]-pdata.lsas_post[pcbidx])
print spearmanr(pdata.lsas_pre,pdata.lsas_post)
print spearmanr(pdata.lsas_pre,pdata.lsas_pre-pdata.lsas_post)
plot(pdata.lsas_pre[dcsidx], pdata.lsas_post[dcsidx], 'o', color=(0.2,0.2,0.2))
plot(pdata.lsas_pre[pcbidx], pdata.lsas_post[pcbidx], 'o', color=(0.6,0.6,0.6))
legend(['DCS', 'PCB'], 'best', numpoints=1)
plot_regression_line(pdata.lsas_pre[dcsidx], pdata.lsas_post[dcsidx], [55, 125],
                     color=[0.2,0.2,0.2])
plot_regression_line(pdata.lsas_pre[pcbidx], pdata.lsas_post[pcbidx], [55, 125],
                     color=[0.6,0.6,0.6])
xlabel('LSAS Pre')
ylabel('LSAS Post')
xlim([58,124])
ylim([8,85])
grid()
```

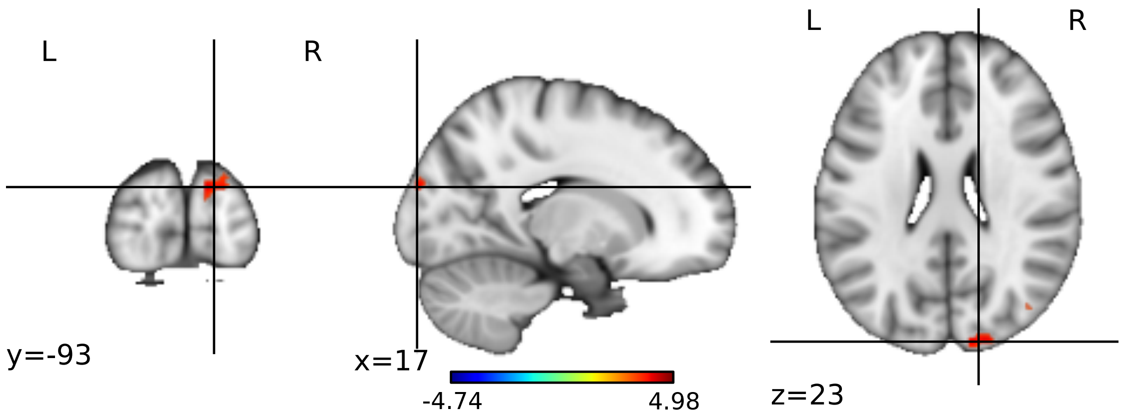
```
(0.36957463542651603, 0.020582499321071621)
(0.37342153991691346, 0.019203555707493748)
DP: (0.19781321306137134, 0.4313860574718511)
PP: (0.51996702206134149, 0.015686563949718763)
DD: (0.50692543723149752, 0.031787597769797171)
PD: (0.29883566124575828, 0.18821010684663192)
(0.30034458449575868, 0.0632007389267807)
(0.27659575035945, 0.088273733122193665)
```

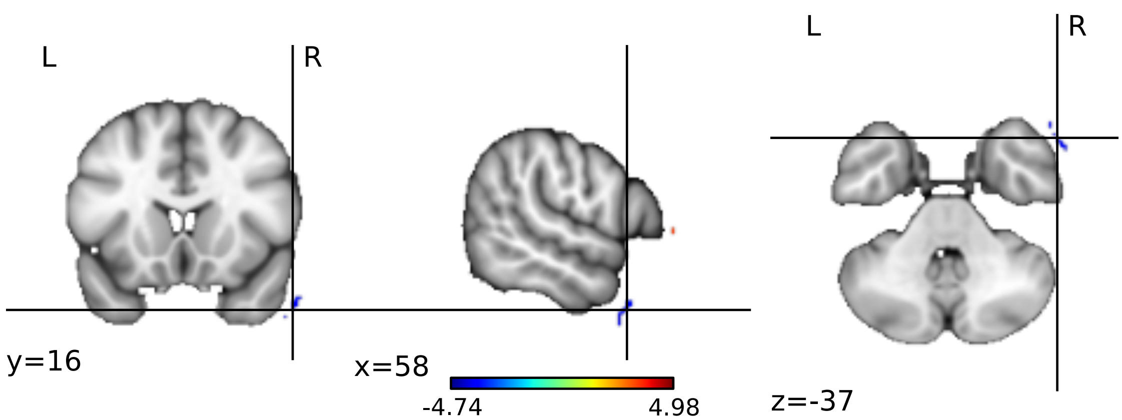


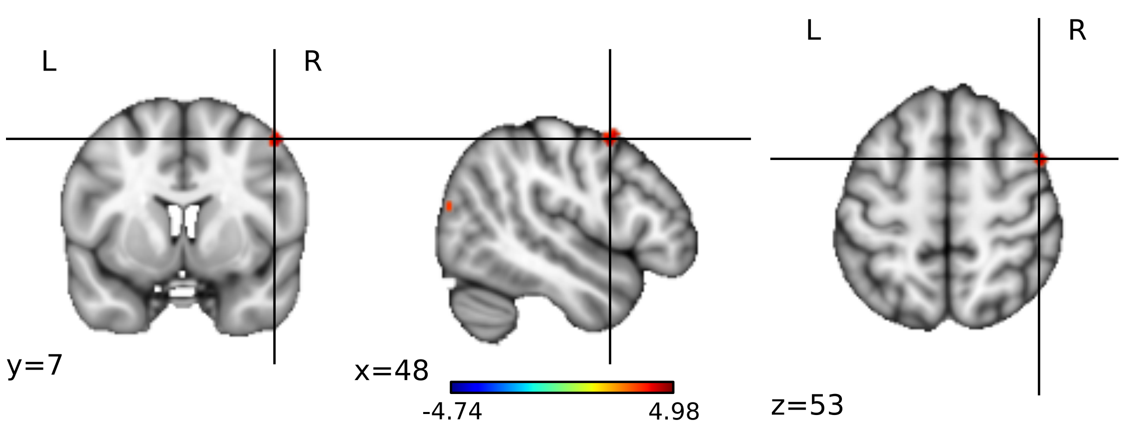
LSAS delta

In [43]:
```python
filename = os.path.join(base_dir, 'lsasdelta_all', 'conest', 'spmT_0001.img')
img=load(filename)
print img.get_header()['descrip']
labels, nlabels = get_labels(abs(img.get_data())>ss.t.ppf(1-0.001,35), 20)
data = img.get_data()
data[labels==0] = 0
#cmeans = get_clustermeans(X, labels, nlabels)
coords = get_coords(labels, img.get_affine())
show_slices(img, coords, threshold=0.5, prefix='uncorrected_lsasdelta', show_colorbar=True,
            cmap=cm.jet)
```

```
SPM{T_[35.0]} - contrast 1: LSAS Delta Response
```



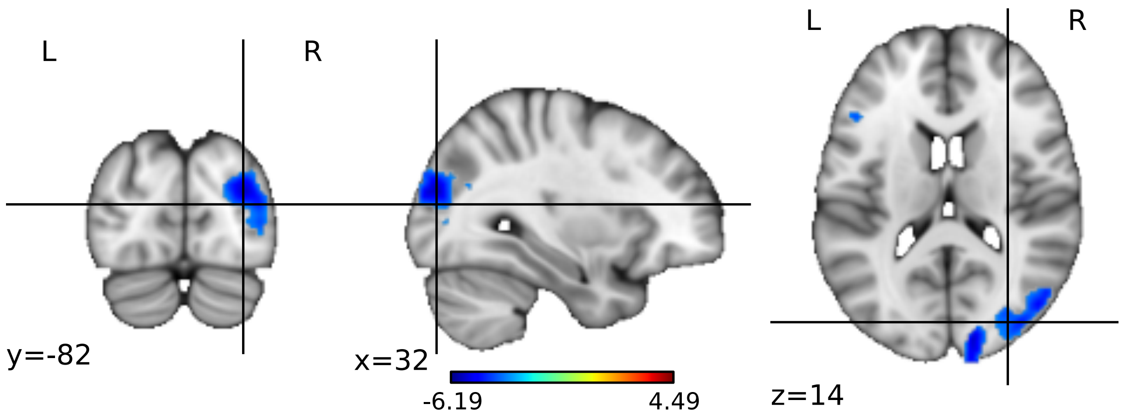


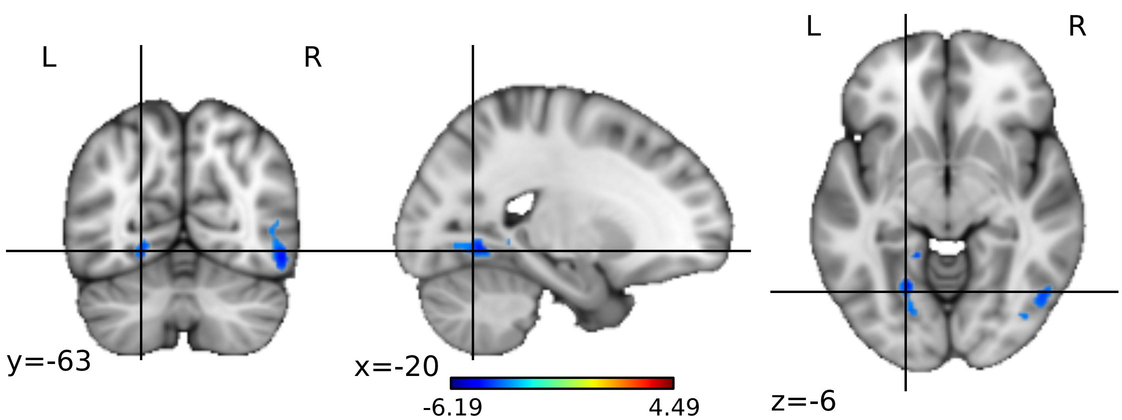


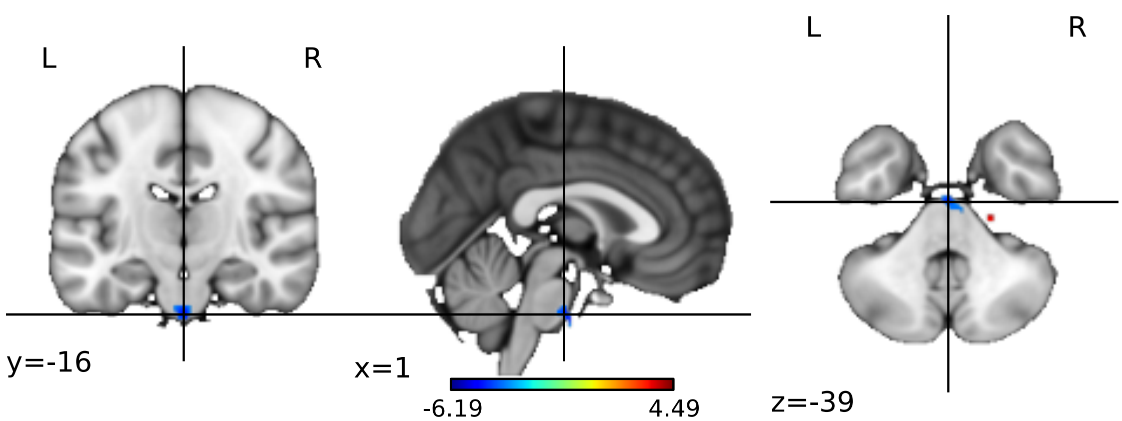
LSAS Post

In [44]:
```
filename = os.path.join(base_dir, 'lsaspost_all', 'conest', 'spmT_0001.img')
img=load(filename)
print img.get_header()['descrip']
labels, nlabels = get_labels(abs(img.get_data())>ss.t.ppf(1-0.001,35), 20)
data = img.get_data()
data[labels==0] = 0
#cmeans = get_clustermeans(X, labels, nlabels)
coords = get_coords(labels, img.get_affine())
show_slices(img, coords, threshold=0.5, prefix='uncorrected_lsaspost', show_colorbar=True,
            cmap=cm.jet)
```

```
SPM{T_[35.0]} - contrast 1: LSAS Delta Response
```



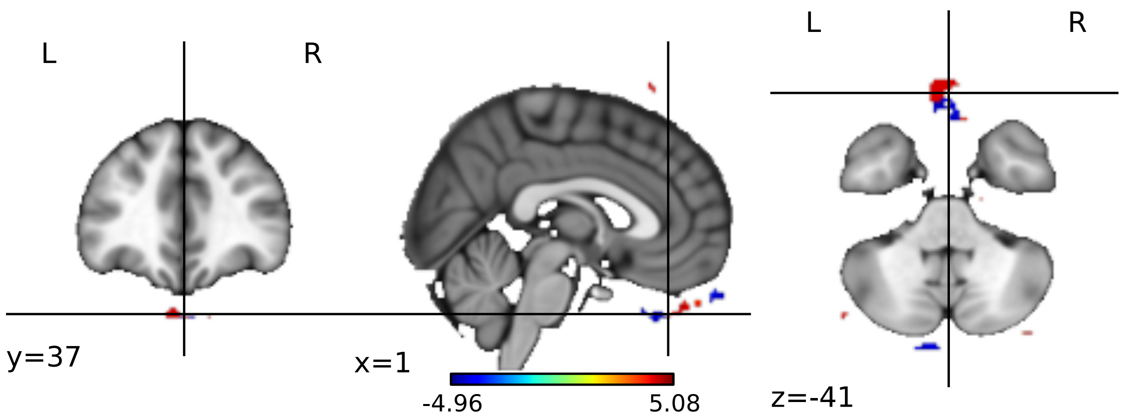


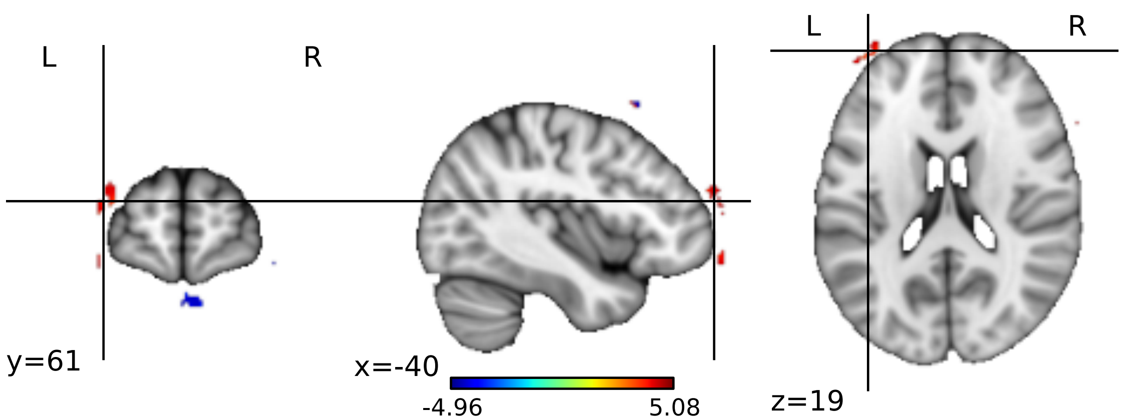


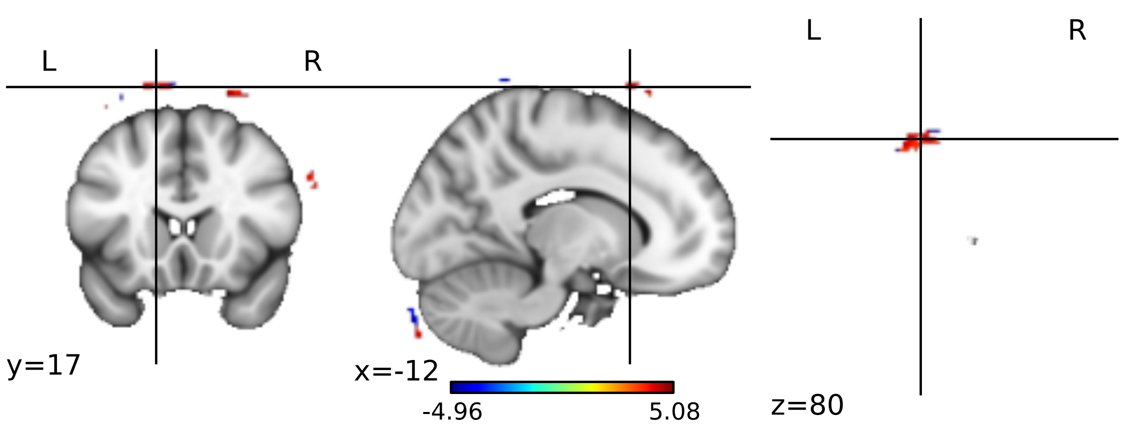
LSAS pre

In [45]:
```
filename = os.path.join(base_dir, 'lsaspre_all', 'conest', 'spmT_0001.img')
img=load(filename)
print img.get_header()['descrip']
labels, nlabels = get_labels(abs(img.get_data())>ss.t.ppf(1-0.001,35), 20)
data = img.get_data()
data[labels==0] = 0
#cmeans = get_clustermeans(X, labels, nlabels)
coords = get_coords(labels, img.get_affine())
show_slices(img, coords, threshold=0.5, prefix='uncorrected_lsaspre', show_colorbar=True,
            cmap=cm.jet)
```

```
SPM{T_[35.0]} - contrast 1: LSAS Delta Response
```







In [45]: