In [2]:

```
import re
import pandas as pd
```

In [3]:

```
with open('ath.fasta') as fget:
    contents = fget.read().split('>')
```

In [4]:

```
d = {}
for record in contents[1:]:
    tem = record.split('\n')
    try:
        match_object = re.match(r'.*\|geneid=(.*?)\|', tem[0])
        geneid = match_object.group(1)
    except AttributeError:
        pass
    seq = ''.join(tem[1:])
    d[geneid] = d.get(geneid, '') + seq
```

In [5]:

```
df_ref = pd.read_table('../to_rf.tsv', sep='\t', index_col = 'Probe Set ID')
columns = list(df_ref.columns)[4:]
columns.remove('class')
del df_ref
```

In [6]:

```
for geneid, seq in d.items():
    cache = re.match('(TGTC.{6,12}CTGT)', seq)
    if cache:
        print(geneid, cache.groups())
    '''
    for ele in columns:
        pattern = '.{7,9}'.join(ele.split('_'))
        print(re.match(pattern, seq))
    break
    '''
```

```
828892 ('TGTCTCGTTCTAGCTGT',)
28720531 ('TGTCGTTCAGCCTGT',)
```

In [7]:

```
for geneid, seq in d.items():
    cache = re.match('(TGTC.{6,14}GACA)', seq)
    if cache:
        print(geneid, cache.groups())
    '''
    for ele in columns:
        pattern = '.{7,9}'.join(ele.split('_'))
        print(re.match(pattern, seq))
    break
    '''
```

```
841453 ('TGTCATGATCACTTGACA',)
819060 ('TGTCACAGCAGTCGACA',)
28717530 ('TGTCGAGGGTGTGACA',)
28719438 ('TGTCGATAGCGACA',)
```

In [8]:

```
result = {}
for geneid, seq in d.items():
    cache = re.match('(TGTC.{2})', seq)
    if cache:
        result[geneid] = cache.groups()
    '''
    for ele in columns:
        pattern = '.{7,9}'.join(ele.split('_'))
        print(re.match(pattern, seq))
    break
    '''
```

In [10]:

```
flat_result = {}
for key, value in result.items():
    for gid in key.split(','):
        flat_result[gid] = value[0]
```

In [13]:

```
df_anno = pd.read_csv('/home/yijia/Desktop/DEG_analysis/NAA_analysis/Arabidopsis
 thaliana/ATH1-121501.na36.annot.csv')
```

In [33]:

```
list(df_anno[df_anno['Entrez Gene'] == '843750']['Probe Set ID'])[0]
```

Out[33]:

```
'260387_at'
```

In [34]:

```
PSID_result = {}
for key, value in flat_result.items():
    try:
        PSID = list(df_anno[df_anno['Entrez Gene'] == key]['Probe Set ID'])[0]
        PSID_result[PSID] = value
    except Exception as e:
        print(e)
```

```
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
list index out of range
```

In [38]:

```
df_element = pd.DataFrame(list(PSID_result.items()))
```

In [39]:

```
df_element
```

Out[39]:

	0	1
0	264171_at	TGTCAA
1	262121_at	TGTCAG
2	264610_at	TGTCGG
3	261809_at	TGTCAT
4	264507_at	TGTCAT
5	259398_at	TGTCAT
6	245641_at	TGTCGT
7	265162_at	TGTCGC
8	262463_at	TGTCAT
9	262255_at	TGTCAT
10	263003_at	TGTCTA
11	264998_at	TGTCTA
12	260357_at	TGTCAC
13	260419_at	TGTCCA
14	259744_at	TGTCCA
15	260387_at	TGTCTG
16	263853_at	TGTCTT
17	263837_at	TGTCGT
18	264106_at	TGTCAC
19	265933_at	TGTCTA
20	266827_at	TGTCCA
21	265995_at	TGTCAT
22	263541_at	TGTCTA
23	267632_at	TGTCTT
24	267370_at	TGTCAC
25	266465_at	TGTCTC
26	266484_at	TGTCCA
27	259331_at	TGTCAA
28	259210_at	TGTCTG
29	258653_at	TGTCCA
...	...	...
36	252181_at	TGTCTT
37	251803_at	TGTCCC
38	251714_at	TGTCAA

38	251714_at	TGTCAA

	0	1
39	251522_at	TGTCAA
40	251384_at	TGTCTG
41	254724_at	TGTCTT
42	245604_at	TGTCTC
43	245341_at	TGTCTT
44	254455_at	TGTCCA
45	253833_at	TGTCTC
46	253787_at	TGTCTC
47	253554_at	TGTCTT
48	253312_s_at	TGTCTT
49	253227_at	TGTCTA
50	253070_at	TGTCGT
51	253035_at	TGTCCT
52	252951_at	TGTCTT
53	252878_at	TGTCTC
54	251076_at	TGTCAA
55	250472_at	TGTCCT
56	245943_at	TGTCTG
57	246125_at	TGTCAA
58	249907_at	TGTCAT
59	249875_at	TGTCAC
60	249338_at	TGTCTT
61	248078_at	TGTCTG
62	247912_at	TGTCAT
63	258446_at	TGTCTT
64	259927_at	TGTCGC
65	259956_at	TGTCGC

66 rows × 2 columns

In [40]:

```
%matplotlib inline

import pickle
from scipy import stats
from matplotlib import pyplot as plt
import seaborn as sns
from pprint import pprint

from sklearn import linear_model
from sklearn.feature_selection import SelectFromModel, RFE, RFECV
```

In [42]:

```
df = pd.read_table('../integrated.tsv', sep='\t', index_col = 'Probe Set ID')[[
'NAA_mean', 'IAA_mean','p-value','NAA-IAA']]
```

In [43]:

```
AuxREs = list(set(PSID_result.values()))
```

Out[43]:

```
['TGTCGT',
 'TGTCTC',
 'TGTCGC',
 'TGTCAA',
 'TGTCGG',
 'TGTCAC',
 'TGTCCC',
 'TGTCCT',
 'TGTCTG',
 'TGTCAG',
 'TGTCAT',
 'TGTCTA',
 'TGTCTT',
 'TGTCCA']
```

In [45]:

```
for AuxRE in AuxREs:
    df[AuxRE] = 0
for key, value in PSID_result.items():
    df.loc[key, value] += 1
#df.to_csv('AuxRE_count_all.tsv', sep='\t')
```

In [53]:

```
with_AuxRE = df[df['TGTCTC']+df['TGTCTG']+df['TGTCGG']+df['TGTCCA']+df['TGTCAT']
+df['TGTCAG']+df['TGTCGC']+df['TGTCTA']+df['TGTCAA']+df['TGTCGT']+df['TGTCTT']+d
f['TGTCCC']+df['TGTCCT']+df['TGTCAC'] >= 1]
with_AuxRE.to_csv('promoter_with_AuxRE.tsv', sep='\t')
```

In [54]:

```
to_rf = with_AuxRE.copy()
to_rf['class'] = 0
for i in with_AuxRE.index:
    if to_rf.loc[i, 'NAA-IAA'] >= 1:
        to_rf.loc[i, 'class'] = 0
    elif to_rf.loc[i, 'NAA-IAA'] >= 0.5:
        to_rf.loc[i, 'class'] = 1
    elif to_rf.loc[i, 'NAA-IAA'] >= 0:
        to_rf.loc[i, 'class'] = 2
    elif to_rf.loc[i, 'NAA-IAA'] >= -0.5:
        to_rf.loc[i, 'class'] = 3
    elif to_rf.loc[i, 'NAA-IAA'] >= -1:
        to_rf.loc[i, 'class'] = 4
    else:
        to_rf.loc[i, 'class'] = 5
```

Out[54]:

	NAA_mean	IAA_mean	p-value	NAA-IAA	TGTCGT	TGTCTC	TGTCG
Probe Set ID							
245341_at	-0.195868	0.000823	0.493042	-0.196691	0	0	0
245604_at	0.085949	-0.746957	0.008400	0.832907	0	1	0
245641_at	-0.229192	-0.226666	0.991265	-0.002526	1	0	0
245943_at	-0.228244	0.365762	0.489465	-0.594006	0	0	0
246125_at	-0.028185	-0.236490	0.613994	0.208305	0	0	0
247912_at	-0.656348	-0.518475	0.614504	-0.137873	0	0	0
248078_at	-0.369745	0.229914	0.268882	-0.599659	0	0	0
249338_at	0.046248	0.446648	0.404348	-0.400400	0	0	0
249875_at	-0.761942	0.053716	0.049592	-0.815659	0	0	0
249907_at	-0.369632	-0.072718	0.577987	-0.296914	0	0	0
250472_at	-3.233041	-0.561792	0.012085	-2.671249	0	0	0
251076_at	-0.581004	0.299152	0.142982	-0.880156	0	0	0
251384_at	-0.704694	1.326325	0.010719	-2.031018	0	0	0
251522_at	0.938737	0.041399	0.136652	0.897339	0	0	0
251714_at	0.372800	-0.161851	0.315700	0.534652	0	0	0
251803_at	0.125052	0.285587	0.587628	-0.160535	0	0	0
252181_at	-0.424954	0.147334	0.108595	-0.572288	0	0	0
252878_at	-0.174563	0.139049	0.292544	-0.313611	0	1	0
252951_at	0.577065	-0.527271	0.027104	1.104336	0	0	0
253035_at	-0.426773	0.121441	0.140120	-0.548214	0	0	0
253070_at	0.743471	-0.030598	0.448077	0.774070	1	0	0
253227_at	-1.661551	-0.606973	0.086382	-1.054578	0	0	0
253312_s_at	0.449659	0.284586	0.697446	0.165073	0	0	0
253554_at	1.441347	-0.218896	0.001988	1.660243	0	0	0
253787_at	1.102017	-0.746054	0.066598	1.848071	0	1	0
253833_at	0.792368	0.175003	0.358694	0.617365	0	1	0
254455_at	0.806465	0.044155	0.078614	0.762311	0	0	0
254724_at	-0.139859	-0.189205	0.771945	0.049346	0	0	0
256638_at	-0.040083	-0.282299	0.558142	0.242217	0	0	0
256861_at	-0.061086	-0.080697	0.942257	0.019611	0	0	0
...	...	...	...	...	...	...	...
259210_at	0.511147	0.311613	0.452499	0.199533	0	0	0

	NAA_mean	IAA_mean	p-value	NAA-IAA	TGTCGT	TGTCTC	TGTCG
Probe Set ID							
259331_at	0.452350	0.927621	0.545913	-0.475271	0	0	0
259398_at	-0.776583	-0.060348	0.387982	-0.716235	0	0	0
259744_at	0.264467	0.241427	0.961453	0.023040	0	0	0
259927_at	-0.535844	0.502326	0.037901	-1.038170	0	0	1
259956_at	0.907636	-0.069173	0.125841	0.976809	0	0	1
260357_at	0.751399	-0.236443	0.101614	0.987842	0	0	0
260387_at	-0.888110	0.119853	0.000077	-1.007963	0	0	0
260419_at	-0.088118	0.187186	0.537459	-0.275304	0	0	0
261809_at	-0.398822	-0.308123	0.765067	-0.090699	0	0	0
262121_at	-0.307288	-0.086468	0.716372	-0.220820	0	0	0
262255_at	0.285909	-0.503495	0.057316	0.789404	0	0	0
262463_at	-0.374931	-0.566353	0.659041	0.191422	0	0	0
263003_at	-0.216141	-0.409232	0.670091	0.193090	0	0	0
263541_at	-0.169162	-0.279504	0.654938	0.110342	0	0	0
263837_at	-0.323223	0.004379	0.328299	-0.327602	1	0	0
263853_at	-0.795741	0.794304	0.118355	-1.590046	0	0	0
264106_at	-0.038855	0.256276	0.519494	-0.295130	0	0	0
264171_at	-0.092816	-0.196683	0.850707	0.103867	0	0	0
264507_at	-0.311082	0.479900	0.280229	-0.790982	0	0	0
264610_at	-1.319850	0.085214	0.049572	-1.405064	0	0	0
264998_at	-0.106359	-0.573380	0.137015	0.467020	0	0	0
265162_at	-0.172598	-0.032449	0.623584	-0.140150	0	0	1
265933_at	0.375085	1.076705	0.232502	-0.701620	0	0	0
265995_at	-0.465574	-0.028440	0.451828	-0.437134	0	0	0
266465_at	0.146224	0.783837	0.394795	-0.637613	0	1	0
266484_at	-0.029005	0.351070	0.353448	-0.380074	0	0	0
266827_at	-0.803000	-0.150430	0.312066	-0.652570	0	0	0
267370_at	-0.483082	0.331650	0.001339	-0.814733	0	0	0
267632_at	-0.130554	0.166408	0.224012	-0.296962	0	0	0

66 rows × 19 columns

In [55]:

```
columns = list(to_rf.columns)
columns.remove('NAA_mean')
columns.remove('IAA_mean')
columns.remove('p-value')
columns.remove('NAA-IAA')
columns.remove('class')
```

In [56]:

```
estimator = linear_model.Lasso()
selector = RFE(estimator, 5, step=1)
selector = selector.fit(to_rf[columns], to_rf['NAA-IAA'])
print(selector.support_)
print('====================')
print(selector.ranking_)
```

```
[False False False False False False False False False  True  True
True
  True  True]
====================
[10  9  8  7  6  5  4  3  2  1  1  1  1  1]
```

In [60]:

```
columns[selector.support_[feature for feature, tag in zip(columns, rfecv.get_sup
port()) if tag]]
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-60-8253cb9b6e78> in <module>()
----> 1 columns[selector.support_]

TypeError: only integer scalar arrays can be converted to a scalar i
ndex
```

RFECV

In [59]:

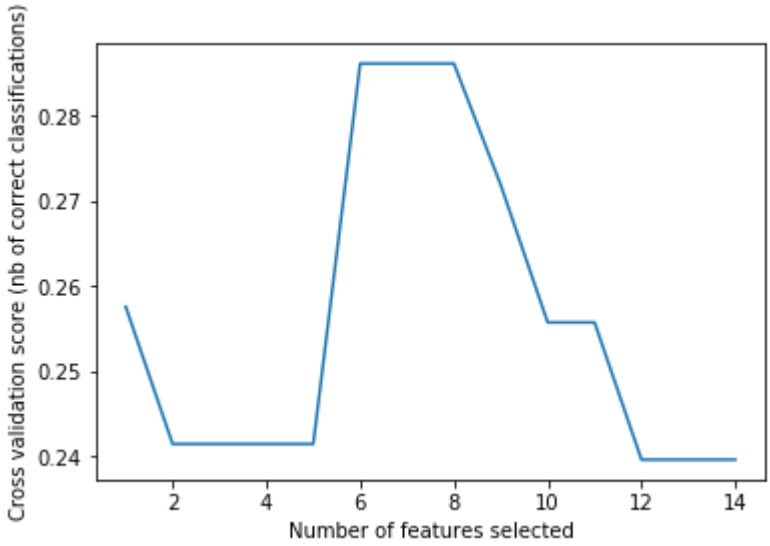
```
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
import numpy as np
# Build a classification task using 3 informative features
#X, y = make_classification(n_samples=3341, n_features=25, n_informative=3,
#                           n_redundant=2, n_repeated=0, n_classes=8,
#                           n_clusters_per_class=1, random_state=0)

# Create the RFE object and compute a cross-validated score.
svc = SVC(kernel="linear")
# The "accuracy" scoring is proportional to the number of correct
# classifications
rfecv = RFECV(estimator=svc, step=1, cv=StratifiedKFold(2),
              scoring='accuracy')
rfecv.fit(to_rf[columns], np.array(to_rf['class'], dtype=int))

print("Optimal number of features : %d" % rfecv.n_features_)

# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()
```

```
Optimal number of features : 8
```



In [62]:

```
l = [feature for feature, tag in zip(columns, rfecv.get_support()) if tag]
l
```

Out[62]:

```
['TGTCTC',
 'TGTCGC',
 'TGTCGG',
 'TGTCAC',
 'TGTCCT',
 'TGTCTG',
 'TGTCTA',
 'TGTCTT']
```

In [85]:

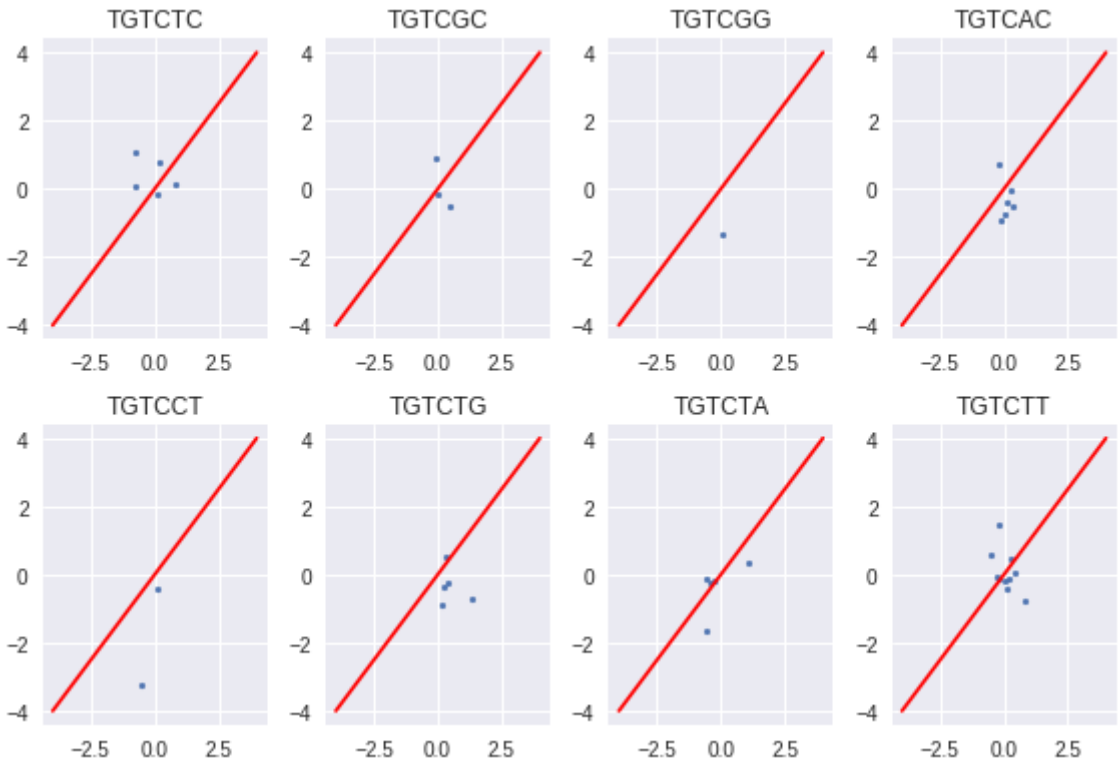
```
', '.join(l)
```

Out[85]:

```
'TGTCTC, TGTCGC, TGTCGG, TGTCAC, TGTCCT, TGTCTG, TGTCTA, TGTCTT'
```

In [65]:

```
plt.style.use(u'seaborn')
fig = plt.figure(1)#, figsize=(10,9))
subplots = [plt.subplot(2, 4, 1+i) for i in range(8)]
for i, AuxRE in zip(range(8), l):
    subplots[i].scatter(x='IAA_mean', y='NAA_mean', data=df[df[AuxRE]==1], s=8)
    subplots[i].plot([-4,4], [-4,4], 'r')
    subplots[i].set_title(AuxRE)
fig.tight_layout()
```



In [69]:

```
with_AuxRE['AuxRE'] = 0
```

```
/home/yijia/anaconda3/lib/python3.6/site-packages/ipykernel_launche
r.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

In [67]:

```
def splited_violinplot(dataf, x, order=None, bw=.2, figsize=(8,7)):
    df_copy = dataf.copy()
    l = len(df_copy)
    df_copy['logFC_mean'] = 0
    df_copy['hue'] = 'Null'
    df_copy = pd.concat([df_copy, df_copy], ignore_index=True)
    for i in range(l):
        df_copy.loc[i, 'logFC_mean'] = df_copy.loc[i, 'IAA_mean']
        df_copy.loc[i, 'hue'] = 'IAA'
        df_copy.loc[i+l, 'logFC_mean'] = df_copy.loc[i, 'NAA_mean']
        df_copy.loc[i+l, 'hue'] = 'NAA'
    plt.figure(figsize=figsize)
    if order is None:
        return sns.violinplot(x=x, y='logFC_mean', hue='hue', data=df_copy, spli
t=True, bw=bw)
    return sns.violinplot(x=x, y='logFC_mean', hue='hue', data=df_copy, split=Tr
ue, order=order, bw=bw)
```

In [82]:

```
df_selected = with_AuxRE[['NAA_mean', 'IAA_mean','p-value','NAA-IAA'] + l + ['Au
xRE']]
df_selected['AuxRE'] = with_AuxRE['TGTCTC'] + with_AuxRE['TGTCGC'] + with_AuxRE[
'TGTCGG'] + with_AuxRE['TGTCAC'] + with_AuxRE['TGTCCT'] + with_AuxRE['TGTCTG'] +
 with_AuxRE['TGTCTA'] + with_AuxRE['TGTCTT']
```

```
/home/yijia/anaconda3/lib/python3.6/site-packages/ipykernel_launche
r.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
```

In [84]:

```
splited_violinplot(df_selected, 'AuxRE', bw=.3)
```

Out[84]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f2df37f69e8>