

# Untitled

June 22, 2017

```
In [1]: import numpy as np
import pandas as pd
import sklearn
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt

from IPython.display import set_matplotlib_formats
set_matplotlib_formats('png', 'pdf')
```

```
In [2]: r1 = pd.read_csv('tps.ISB.ratios.tab', sep=' ')
print(r1.shape)
r1.head()
```

(12099, 67)

/Users/dreiss/miniconda3/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2683: DtypeWarning: Columns (interactivity=interactivity, compiler=compiler, result=result)

	nrm_diel_Day1.Dk	nrm_diel_Day1.Lt	nrm_diel_Day2.Dk	nrm_diel_Day2.Lt	\
1000	-0.54922	0.58181	0.10346	0.42690	
10002	-1.24427	1.19105	-1.00189	0.86039	
10006	-0.36077	-0.04452	-0.10789	-0.09317	
10008	-1.96042	-0.57802	-0.87794	-2.17830	
10010	0.02945	-1.14524	-0.97718	1.06226	

	nrm_diel_Day3.Dk	nrm_diel_Day3.Lt	nrm_diel_Day4.Dk	nrm_diel_Day4.Lt	\
1000	0.39565	0.45545	-0.24153	-0.84529	
10002	-1.04884	0.83065	-0.83815	0.29552	
10006	0.46013	-0.13913	0.37880	-0.27903	
10008	0.93808	1.19216	1.55504	0.56126	
10010	-0.16960	-0.50967	0.59087	-0.91881	

	nrm_diel_Day5.Dk	nrm_diel_Day5.Lt	...	\
1000	-0.39492	0.42528	...	
10002	0.30941	1.45448	...	
10006	0.58274	0.20985	...	
10008	2.55721	2.44418	...	
10010	-0.33980	3.03597	...	

	nrmlight.B.nrm.exp.3	nrmlight.B.nrm.sta.1	nrmlight.B.nrm.sta.2	\
1000	-0.02569	-0.60172	-0.21575	
10002	0.27507	-0.48276	-0.05589	
10006	0.45859	0.51178	0.88394	
10008	-0.68742	0.04835	1.35192	
10010	-0.73207	-0.31512	-0.08387	

	nrmlight.B.nrm.sta.3	nrmlight.B.elev.exp.1	nrmlight.B.elev.exp.2	\
1000	-0.32312	-0.29853	-0.24846	
10002	-0.37492	0.44073	0.60202	
10006	0.54188	0.66271	1.68050	
10008	0.53908	-0.12631	-0.86967	
10010	-0.14934	-0.52712	-0.73413	

	nrmlight.B.elev.exp.3	nrmlight.B.elev.sta.1	nrmlight.B.elev.sta.2	\
1000	0.20443	-0.96469	-0.73780	
10002	0.42461	-0.77719	-0.16530	
10006	0.48628	0.86174	0.62019	
10008	1.06930	0.28077	0.18448	
10010	-0.06333	-0.24718	-0.07036	

	nrmlight.B.elev.sta.3
1000	-0.29041
10002	0.20979
10006	1.07254
10008	0.91407
10010	-0.39420

[5 rows x 67 columns]

```
In [3]: import re
```

```
def grepl(pattern, strings, values=False):
    out = np.array([bool(re.search(pattern, i)) for i in strings])
    if values:
        out = strings[out]
    return out

#def grepl(pattern, strings):
#    pd.Series(strings).str.contains(pattern).tolist()
```

```
In [4]: colnames = r1.columns.values
print(colnames)
print(grepl('Dk', colnames))
colnames[grepl('Dk', colnames)]
```

```
['nrm_diel_Day1.Dk' 'nrm_diel_Day1.Lt' 'nrm_diel_Day2.Dk'
'nrm_diel_Day2.Lt' 'nrm_diel_Day3.Dk' 'nrm_diel_Day3.Lt'
'nrm_diel_Day4.Dk' 'nrm_diel_Day4.Lt' 'nrm_diel_Day5.Dk'
'nrm_diel_Day5.Lt' 'elev_diel_Day1.Dk' 'elev_diel_Day1.Lt'
'elev_diel_Day2.Dk' 'elev_diel_Day2.Lt' 'elev_diel_Day3.Dk'
'elev_diel_Day3.Lt' 'elev_diel_Day4.Dk' 'elev_diel_Day4.Lt'
'elev_diel_Day5.Dk' 'highlight.A.nrm.exp.1' 'highlight.A.nrm.exp.2'
'highlight.A.nrm.exp.3' 'highlight.A.nrm.sta.1' 'highlight.A.nrm.sta.2'
'highlight.A.nrm.sta.3' 'highlight.A.elev.exp.1' 'highlight.A.elev.exp.2'
'highlight.A.elev.exp.3' 'highlight.A.elev.sta.1' 'highlight.A.elev.sta.2'
'highlight.A.elev.sta.3' 'highlight.B.nrm.exp.1' 'highlight.B.nrm.exp.2'
'highlight.B.nrm.exp.3' 'highlight.B.nrm.sta.1' 'highlight.B.nrm.sta.2'
'highlight.B.nrm.sta.3' 'highlight.B.elev.exp.1' 'highlight.B.elev.exp.2'
'highlight.B.elev.exp.3' 'highlight.B.elev.sta.1' 'highlight.B.elev.sta.2'
'highlight.B.elev.sta.3' 'nrmlight.A.nrm.exp.1' 'nrmlight.A.nrm.exp.2'
'nrmlight.A.nrm.exp.3' 'nrmlight.A.nrm.sta.1' 'nrmlight.A.nrm.sta.2'
'nrmlight.A.nrm.sta.3' 'nrmlight.A.elev.exp.1' 'nrmlight.A.elev.exp.2'
'nrmlight.A.elev.exp.3' 'nrmlight.A.elev.sta.1' 'nrmlight.A.elev.sta.2'
'nrmlight.A.elev.sta.3' 'nrmlight.B.nrm.exp.1' 'nrmlight.B.nrm.exp.2'
'nrmlight.B.nrm.exp.3' 'nrmlight.B.nrm.sta.1' 'nrmlight.B.nrm.sta.2'
'nrmlight.B.nrm.sta.3' 'nrmlight.B.elev.exp.1' 'nrmlight.B.elev.exp.2']
```

```
'nrmlight.B.elev.exp.3' 'nrmlight.B.elev.sta.1' 'nrmlight.B.elev.sta.2'
'nrmlight.B.elev.sta.3']
[ True False  True False  True False  True False  True False  True False
  True False  True False  True False  True False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False]
```

```
Out[4]: array(['nrm_diel_Day1.Dk', 'nrm_diel_Day2.Dk', 'nrm_diel_Day3.Dk',
              'nrm_diel_Day4.Dk', 'nrm_diel_Day5.Dk', 'elev_diel_Day1.Dk',
              'elev_diel_Day2.Dk', 'elev_diel_Day3.Dk', 'elev_diel_Day4.Dk',
              'elev_diel_Day5.Dk'], dtype=object)
```

From Justin (it's in R):

```
condclasses = list(
  ~ dark12hr = function(x){ grepl('Dk',x) },
  ~ light12hr = function(x){ grepl('Lt',x) },
  ~ exponential_diel = function(x){ grepl('diel_Day[12]',x) | grepl('diel_Day3.Dk',x) },
  ~ stationary_diel = function(x){ grepl('diel_Day[45]',x) | grepl('diel_Day3.Lt',x) },
  ~ exponential_24light = function(x){ grepl('nrmlight.+exp',x) },
  ~ stationary_24light = function(x){ grepl('nrmlight.+sta',x) },
  ~ exponential_24highlight = function(x){ grepl('highlight.+exp',x) },
  ~ stationary_24highlight = function(x){ grepl('highlight.+sta',x) },
  ~ highlight_arrays = function(x){ grepl('highlight',x) },
  ~ normallight_arrays = function(x){ grepl('nrmlight',x) },
  ~ co2_elevated_arrays = function(x){ grepl('elev',x) & !grepl('diel',x) },
  ~ co2_elevated_expo_arrays = function(x){ grepl('elev',x) & grepl('\\.exp',x) & !grepl('diel',x) },
  ~ co2_elevated_stat_arrays = function(x){ grepl('elev',x) & grepl('\\.sta',x) & !grepl('diel',x) },
  ~ co2_moderate_arrays = function(x){ grepl('nrm',x) & !grepl('elev',x) & !grepl('diel',x) },
  ~ co2_moderate_expo_arrays = function(x){ grepl('nrm',x) & grepl('\\.exp',x) & !grepl('elev',x) & !grepl('diel',x) },
  ~ co2_moderate_stat_arrays = function(x){ grepl('nrm',x) & grepl('\\.sta',x) & !grepl('elev',x) & !grepl('diel',x) }
)
```

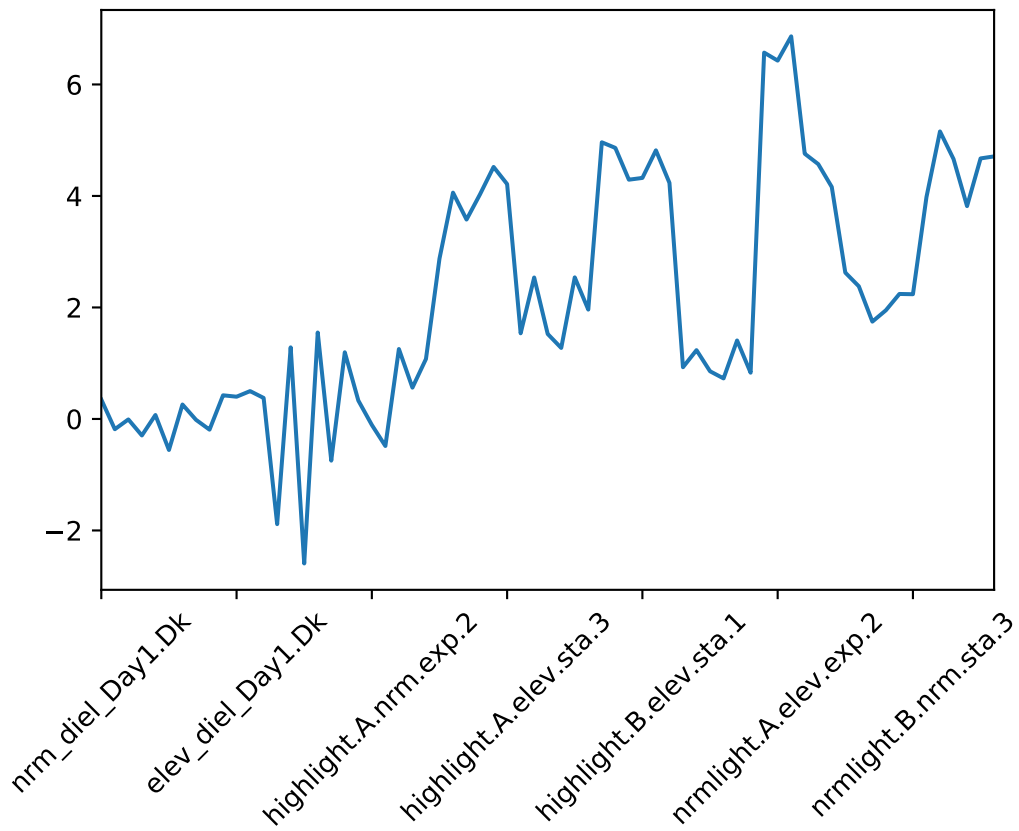
In [5]: *# Convert to python:*

```
condclasses = {
  'dark12hr': lambda x: grepl('Dk',x),
  'light12hr': lambda x: grepl('Lt',x),
  'exponential_diel': lambda x: grepl('diel_Day[12]',x) | grepl('diel_Day3.Dk',x),
  'stationary_diel': lambda x: grepl('diel_Day[45]',x) | grepl('diel_Day3.Lt',x),
  'exponential_24light': lambda x: grepl('nrmlight.+exp',x),
  'stationary_24light': lambda x: grepl('nrmlight.+sta',x),
  'exponential_24highlight': lambda x: grepl('highlight.+exp',x),
  'stationary_24highlight': lambda x: grepl('highlight.+sta',x),
  'highlight_arrays': lambda x: grepl('highlight',x),
  'normallight_arrays': lambda x: grepl('nrmlight',x),
  'co2_elevated_arrays': lambda x: grepl('elev',x), # & ~grepl('diel',x),
  'co2_elevated_expo_arrays': lambda x: grepl('elev',x) & grepl('\\.exp',x), # & ~grepl('diel',x),
  'co2_elevated_stat_arrays': lambda x: grepl('elev',x) & grepl('\\.sta',x), # & ~grepl('diel',x),
  'co2_moderate_arrays': lambda x: grepl('nrm',x) & ~grepl('elev',x), # & ~grepl('diel',x),
  'co2_moderate_expo_arrays': lambda x: grepl('nrm',x) & grepl('\\.exp',x), # & ~grepl('elev',x) & ~grepl('diel',x),
  'co2_moderate_stat_arrays': lambda x: grepl('nrm',x) & grepl('\\.sta',x), # & ~grepl('elev',x) & ~grepl('diel',x)
}
```

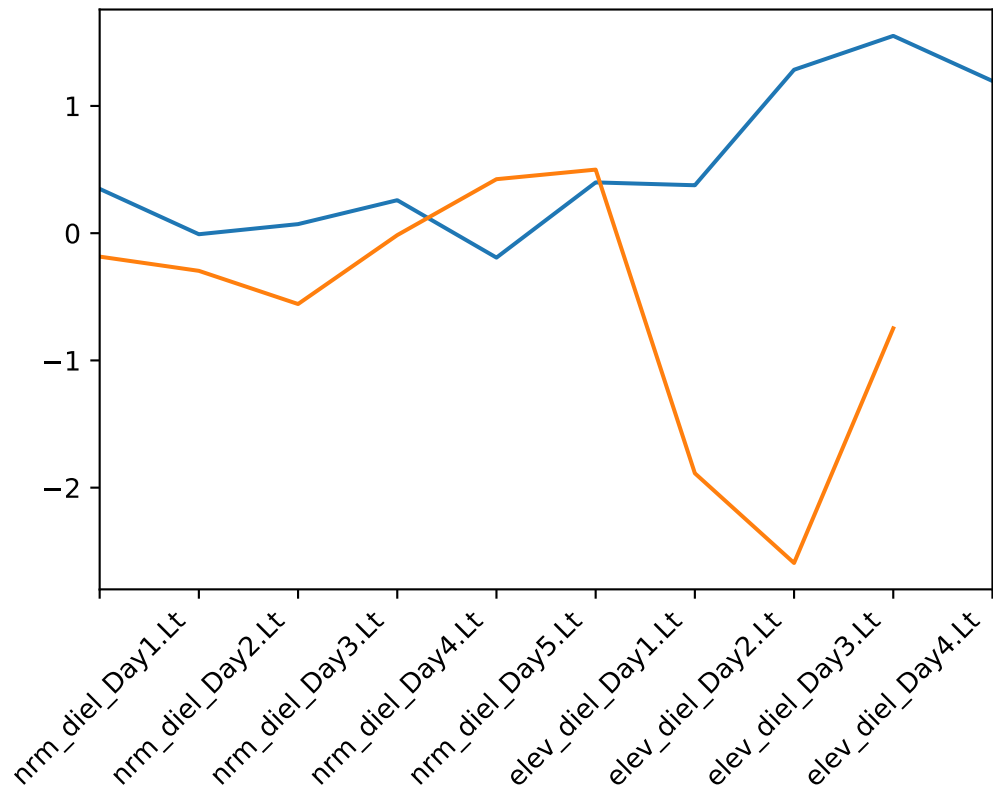
In [6]: colnames[condclasses['dark12hr'](colnames)]

```
Out[6]: array(['nrm_diel_Day1.Dk', 'nrm_diel_Day2.Dk', 'nrm_diel_Day3.Dk',
              'nrm_diel_Day4.Dk', 'nrm_diel_Day5.Dk', 'elev_diel_Day1.Dk',
              'elev_diel_Day2.Dk', 'elev_diel_Day3.Dk', 'elev_diel_Day4.Dk',
              'elev_diel_Day5.Dk'], dtype=object)
```

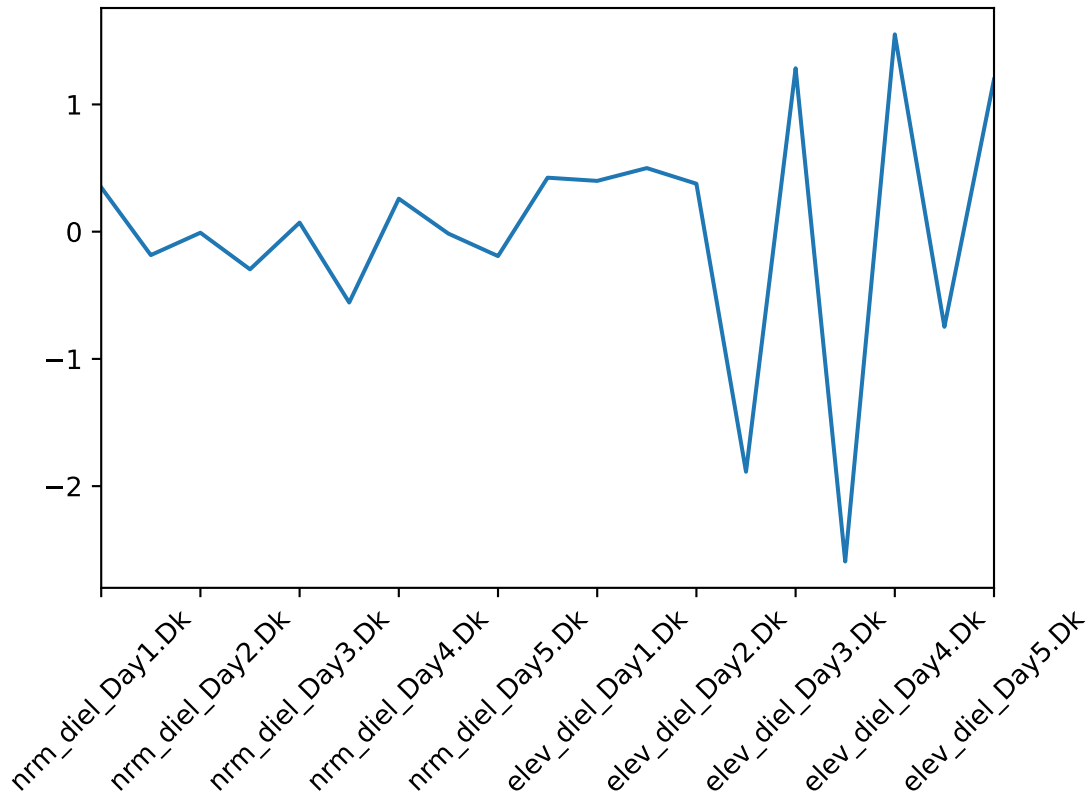
```
In [7]: the_gene = 262258
ax = r1.loc[the_gene].plot()
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
```



```
In [8]: isdark = condclasses['dark12hr'](colnames)
islight = condclasses['light12hr'](colnames)
ax = r1.loc[the_gene, colnames[isdark]].plot()
ax = r1.loc[the_gene, colnames[islight]].plot()
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
```



```
In [9]: ax = r1.loc[the_gene, colnames[isdark | islight]].plot()
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
```



```

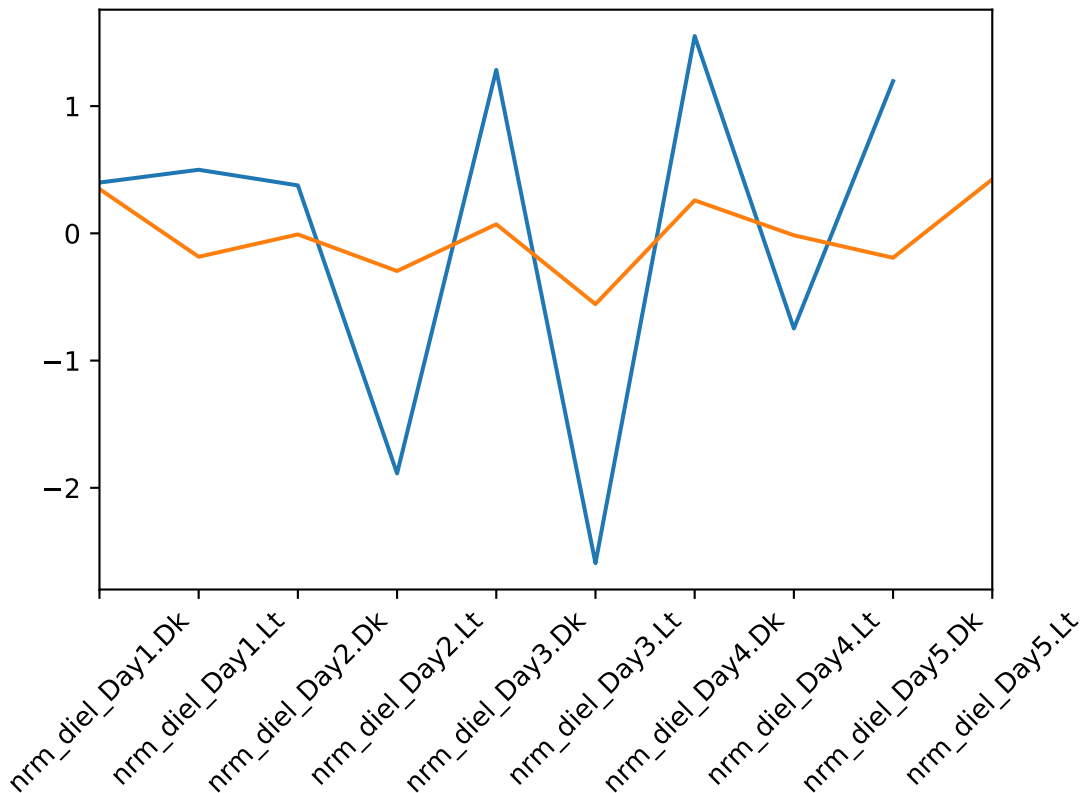
In [10]: #print(colnames[dark])
         #print(colnames[light])
         isco2_high = condclasses['co2_elevated_arrays'](colnames)
         isco2_low = condclasses['co2_moderate_arrays'](colnames)
         #print(colnames[co2_high])

In [11]: print(colnames[(isdark | islight) & isco2_high])
         print(colnames[(isdark | islight) & isco2_low])

['elev_diel_Day1.Dk' 'elev_diel_Day1.Lt' 'elev_diel_Day2.Dk'
 'elev_diel_Day2.Lt' 'elev_diel_Day3.Dk' 'elev_diel_Day3.Lt'
 'elev_diel_Day4.Dk' 'elev_diel_Day4.Lt' 'elev_diel_Day5.Dk']
['nrm_diel_Day1.Dk' 'nrm_diel_Day1.Lt' 'nrm_diel_Day2.Dk'
 'nrm_diel_Day2.Lt' 'nrm_diel_Day3.Dk' 'nrm_diel_Day3.Lt'
 'nrm_diel_Day4.Dk' 'nrm_diel_Day4.Lt' 'nrm_diel_Day5.Dk'
 'nrm_diel_Day5.Lt']

In [12]: ax = r1.loc[the_gene, colnames[(isdark | islight) & isco2_high]].plot()
         ax = r1.loc[the_gene, colnames[(isdark | islight) & isco2_low]].plot()
         ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);

```



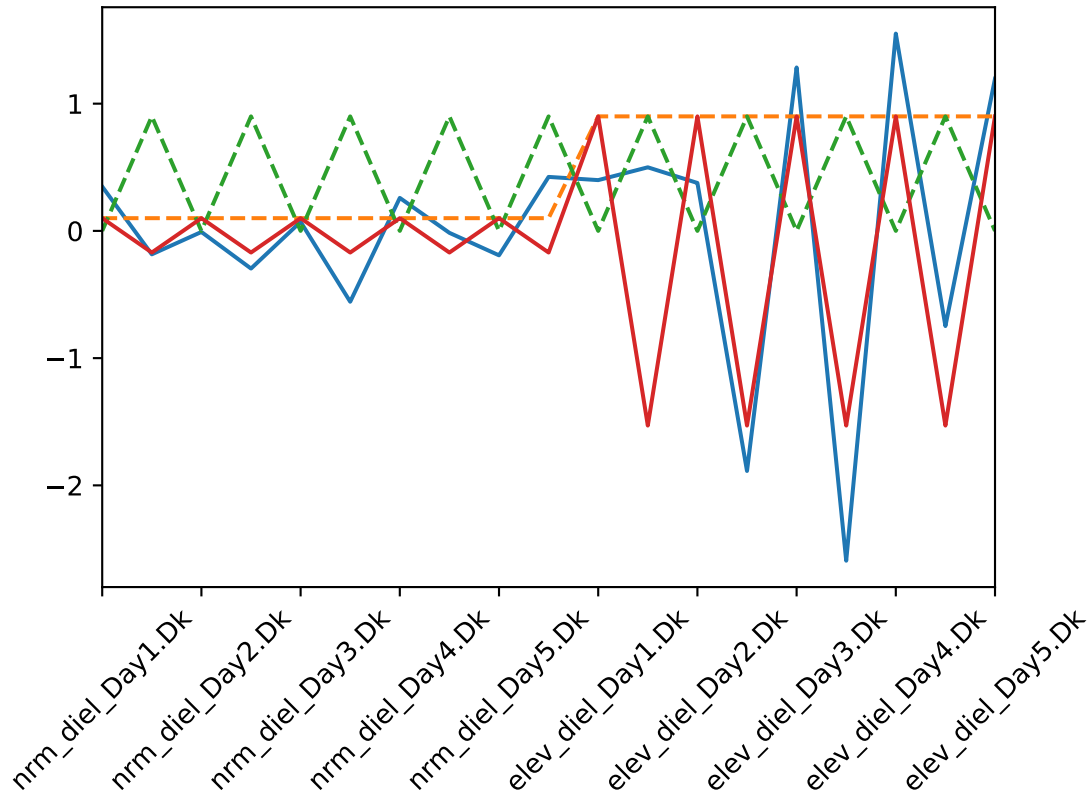
Note here I call it 'isdark' but it's actually tracing the light since it is set to 0.1 for dark and 0.9 for light:

```

In [13]: co2 = np.where(isco2_high, 0.9, 0.1)
         dark = np.where(isdark, 0.0, 0.9)
         ax = r1.loc[the_gene, colnames[(isdark | islight)]] .plot()
         ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
         ax.plot(co2[isdark | islight], ls='dashed')
         ax.plot(dark[isdark | islight], ls='dashed')
         ax.plot(-3. * (co2*dark)[isdark | islight] + 1. * co2[isdark | islight])

```

Out[13]: [<matplotlib.lines.Line2D at 0x11cfcb860>]



In [14]: `from sklearn import linear_model`

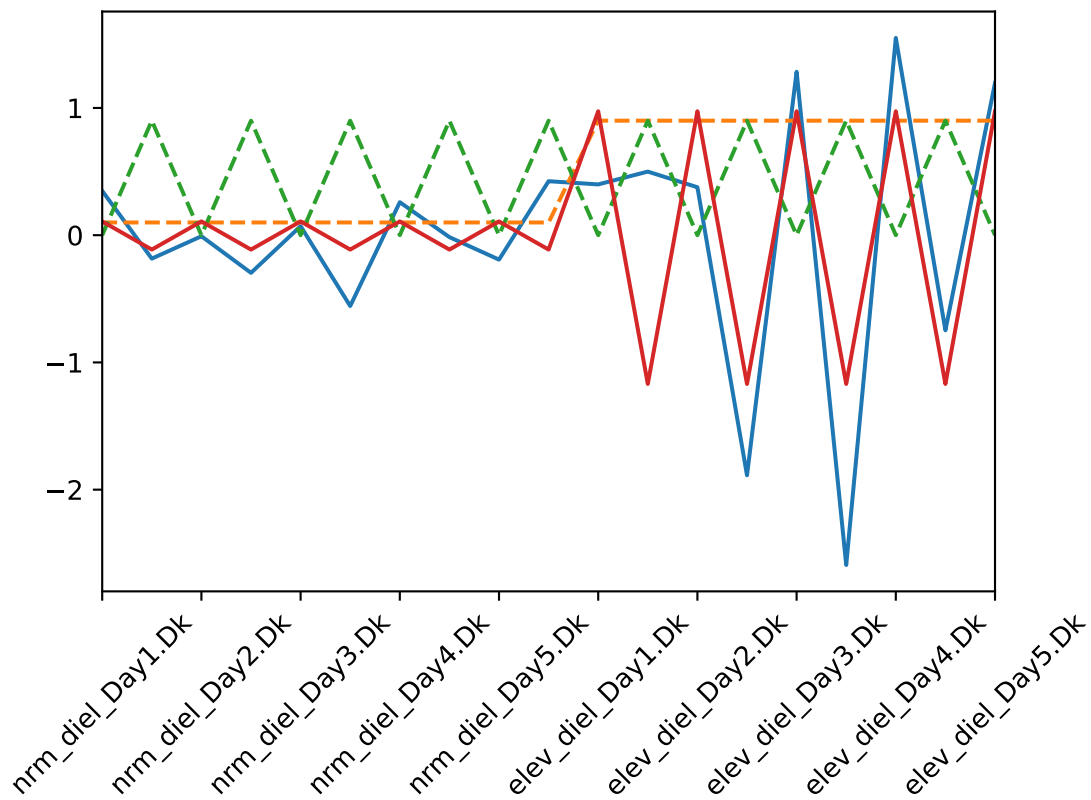
```
y = r1.loc[the_gene, colnames[(isdark | islight)]]
x = np.vstack((np.ones_like(y),
               co2[isdark | islight],
               dark[isdark | islight],
               (co2*dark)[isdark | islight]))
print(x.shape, y.shape)
reg = linear_model.LinearRegression()
reg.fit(x.T, y) #, sample_weight=np.abs(y))
print(reg.score(x.T, y)) # R^2
reg.coef_
```

(4, 19) (19,)
0.583197557061

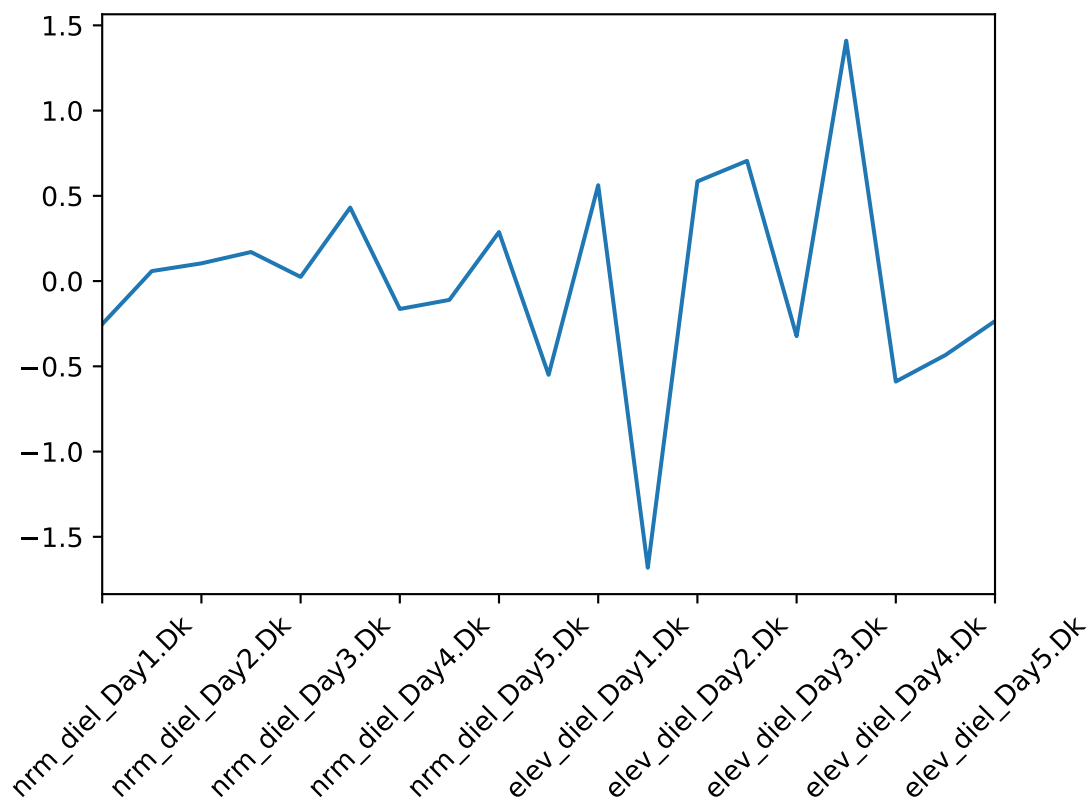
Out[14]: `array([ 0. , 1.08214 , 0.02106882, -2.66937708])`

```
In [15]: ax = r1.loc[the_gene, colnames[(isdark | islight)]] .plot()
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
ax.plot(co2[isdark | islight], ls='dashed')
ax.plot(dark[isdark | islight], ls='dashed')
ax.plot(reg.coef_[3] * (co2*dark)[isdark | islight] +
        reg.coef_[1] * co2[isdark | islight] +
        reg.coef_[2] * dark[isdark | islight])
```

Out[15]: [<matplotlib.lines.Line2D at 0x11cb194e0>]



```
In [16]: resids = reg.predict(x.T) - y
ax = resids.plot()
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
```





Lets use `statsmodels` to get errors on coefficients.

```
In [17]: import statsmodels.api as sm
         results = sm.OLS(y, x.T).fit()
         results.summary()
```

```
/Users/dreiss/miniconda3/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarning: The pandas.core
from pandas.core import datetools
```

```
/Users/dreiss/miniconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1334: UserWarning: kurtosistest only vali
"anyway, n=%i" % int(n))
```

```
Out [17]: <class 'statsmodels.iolib.summary.Summary'>
        """
```

```

                        OLS Regression Results
=====
Dep. Variable:          262258      R-squared:          0.583
Model:                  OLS      Adj. R-squared:        0.500
Method:                 Least Squares      F-statistic:        6.996
Date:                   Thu, 22 Jun 2017      Prob (F-statistic):    0.00363
Time:                   15:07:46      Log-Likelihood:       -17.982
No. Observations:       19      AIC:                43.96
Df Residuals:           15      BIC:                47.74
Df Model:                3
Covariance Type:        nonrobust
=====
               coef      std err          t      P>|t|      [0.025      0.975]
-----
const          -0.0125      0.355      -0.035      0.972      -0.770      0.745
x1              1.0821      0.555       1.951      0.070      -0.100      2.264
x2              0.0211      0.559       0.038      0.970      -1.169      1.212
x3             -2.6694      0.898      -2.971      0.010      -4.584     -0.754
=====
Omnibus:              4.009      Durbin-Watson:          2.851
Prob(Omnibus):         0.135      Jarque-Bera (JB):        2.024
Skew:                  0.354      Prob(JB):              0.363
Kurtosis:              4.433      Cond. No.              8.56
=====
```

Warnings:

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
"""
```

Try Lasso...

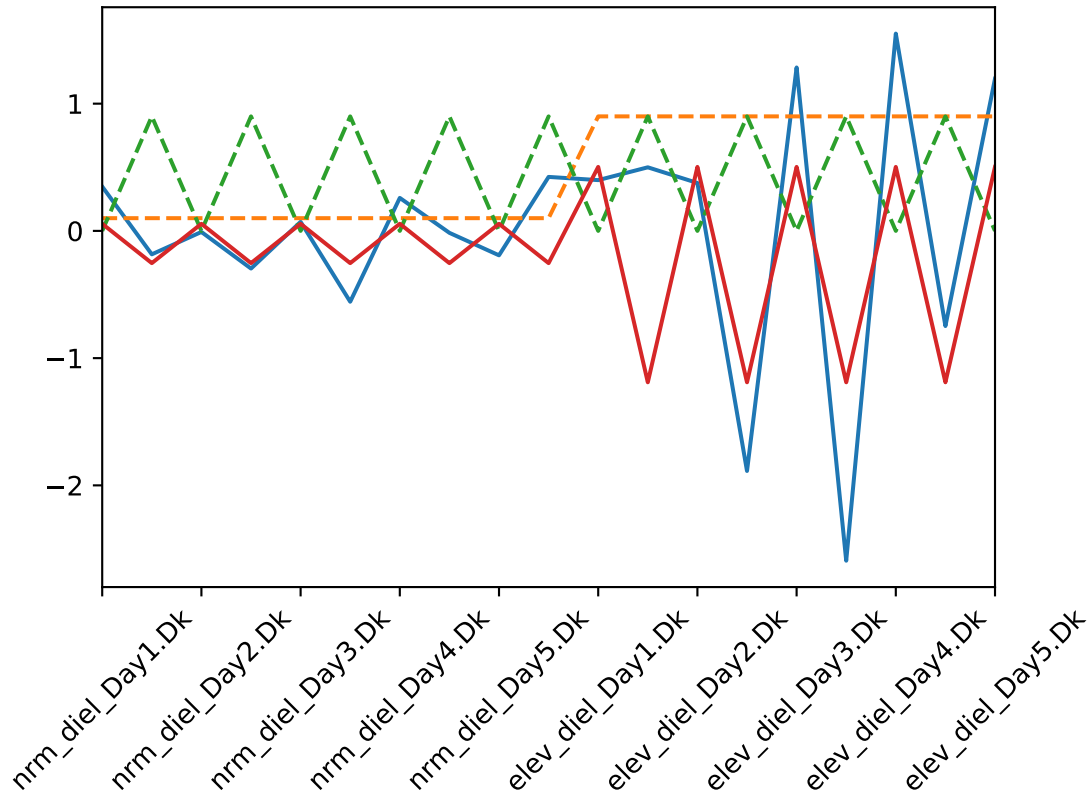
```
In [18]: reg = linear_model.LassoLars(alpha=0.02)
         reg.fit(x.T, y) #, sample_weight=np.abs(y))
         print(reg.score(x.T, y)) # R^2
         print(reg.coef_)

ax = r1.loc[the_gene, colnames[(isdark | islight)]].plot()
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
ax.plot(co2[isdark | islight], ls='dashed')
ax.plot(dark[isdark | islight], ls='dashed')
ax.plot(reg.coef_[3] * (co2*dark)[isdark | islight] +
         reg.coef_[1] * co2[isdark | islight] +
         reg.coef_[2] * dark[isdark | islight])
```

```
0.548514155569
```

```
[ 0.          0.55873282 -0.15137112 -1.92173963]
```

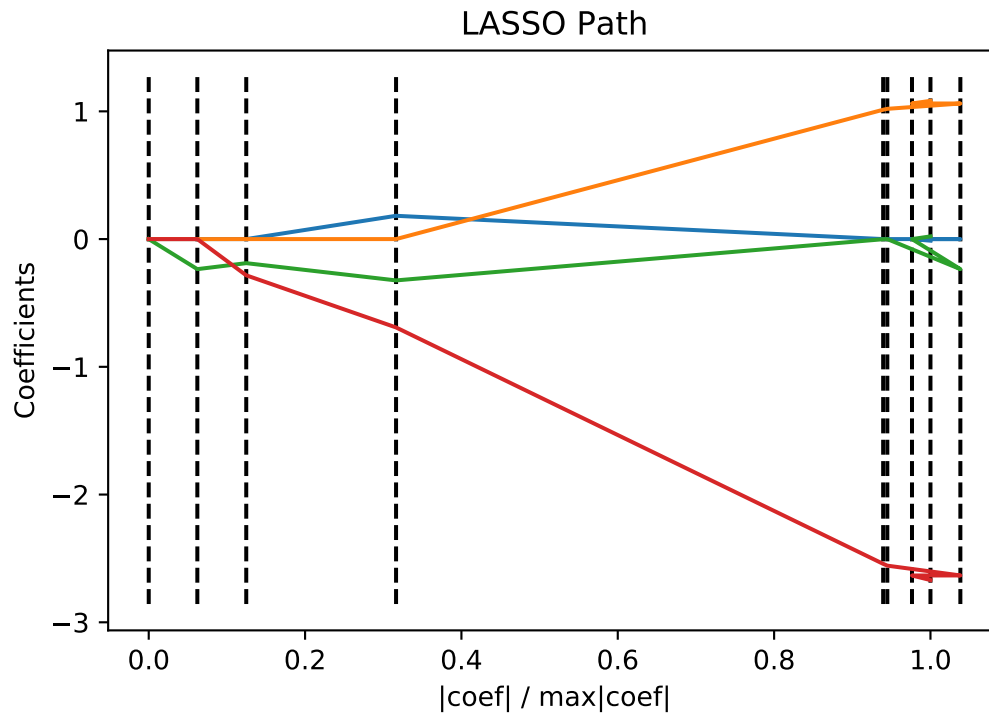
Out[18]: [matplotlib.lines.Line2D at 0x120eb8ef0]



```
In [19]: alphas, _, coefs = linear_model.lars_path(x.T, y, method='lasso', verbose=True)
```

```
xx = np.sum(np.abs(coefs.T), axis=1)
xx /= xx[-1]

plt.plot(xx, coefs.T)
ymin, ymax = plt.ylim()
plt.vlines(xx, ymin, ymax, linestyle='dashed')
plt.xlabel('|coef| / max|coef|')
plt.ylabel('Coefficients')
plt.title('LASSO Path')
plt.axis('tight')
plt.show()
```



Try k-means to get a cluster containing this super awesome gene...

```
In [20]: from sklearn.preprocessing import scale
print(r1.shape)
print(np.mean(np.isnan(r1.values)))
r1norm = r1.dropna(axis=0, how='all', thresh=None, subset=None, inplace=False)
print(r1norm.shape)
r1norm = r1norm.loc[:, colnames[(isdark | islight)]]
#print(np.mean(np.isnan(r1norm)))
r1norm.fillna(0., inplace=True)
r1norm.values[:, :] = scale(r1norm.values)
print(r1norm.shape)
```

```
(12099, 67)
0.0190554788665
(12099, 67)
(12099, 19)
```

```
In [21]: from sklearn.cluster import KMeans
np.random.seed(4)
km = KMeans(n_clusters=1200, init='k-means++', n_init=10, max_iter=20,
            tol=0.0001, precompute_distances='auto', verbose=0, random_state=None,
            copy_x=True, n_jobs=1, algorithm='auto')
km.fit(r1norm)
labels = km.labels_
print(len(labels))
labels
```

```
12099
```

```
Out[21]: array([713, 618, 878, ..., 226, 226, 654], dtype=int32)
```

```
In [22]: index = labels[np.where(r1.index.values == the_gene)]
print(index)
```

```

genes = r1.index.values[np.where(labels == index)]
print(len(genes))
print(the_gene in genes)
print(genes)

ax = r1norm.loc[genes].T.plot(legend=False)
plt.plot(r1norm.loc[the_gene, :].values, ls='dotted', lw=3)
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);

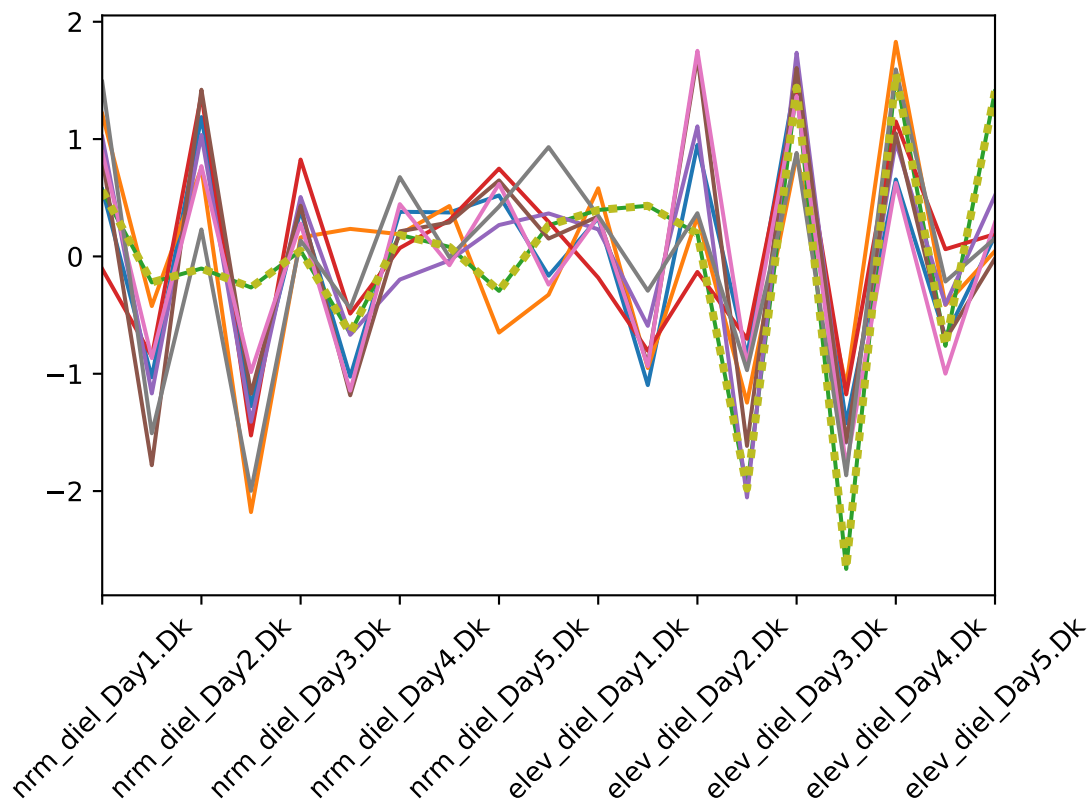
```

[601]

8

True

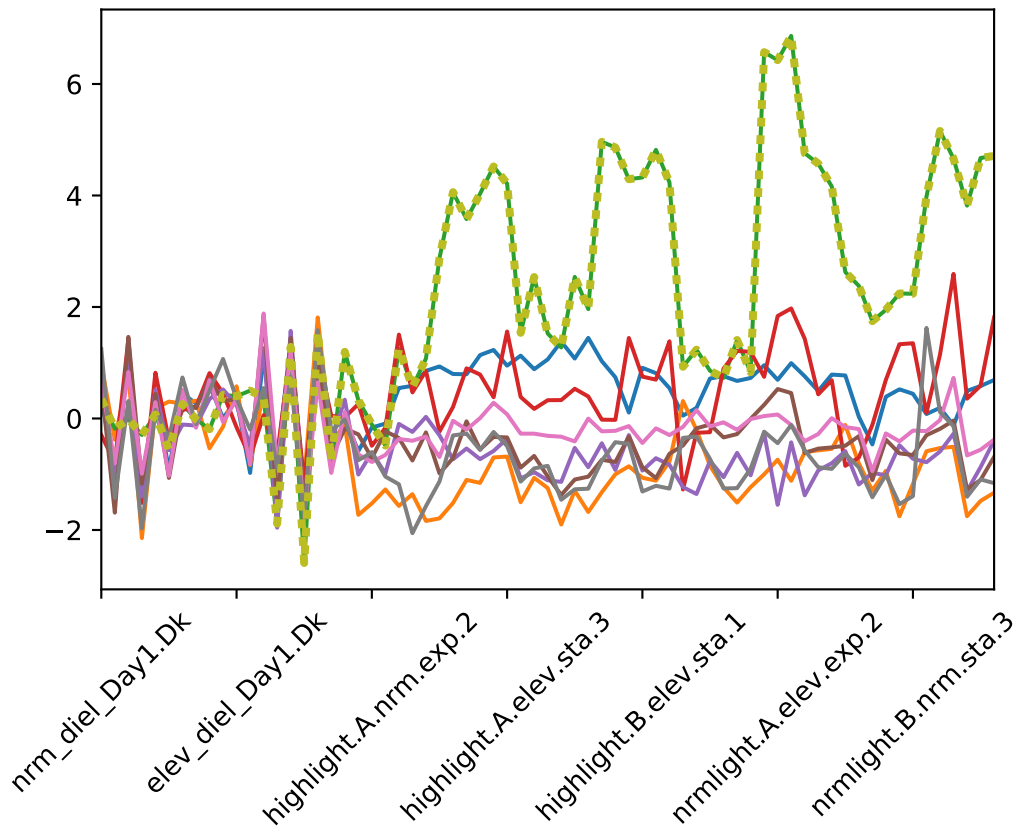
[14096 25744 262258 264821 34170 '6010' '9135' '9976']



```

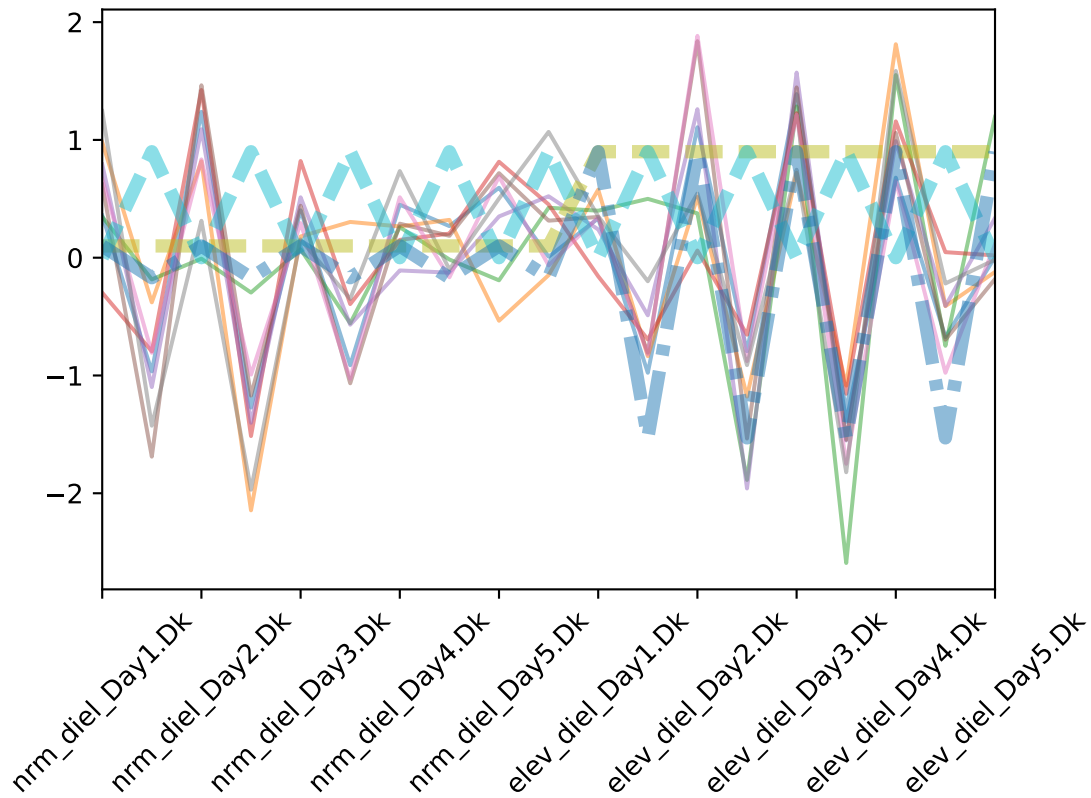
In [23]: ax = r1.loc[genes].T.plot(legend=False)
plt.plot(r1.loc[the_gene, :].values, ls='dotted', lw=3)
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);

```



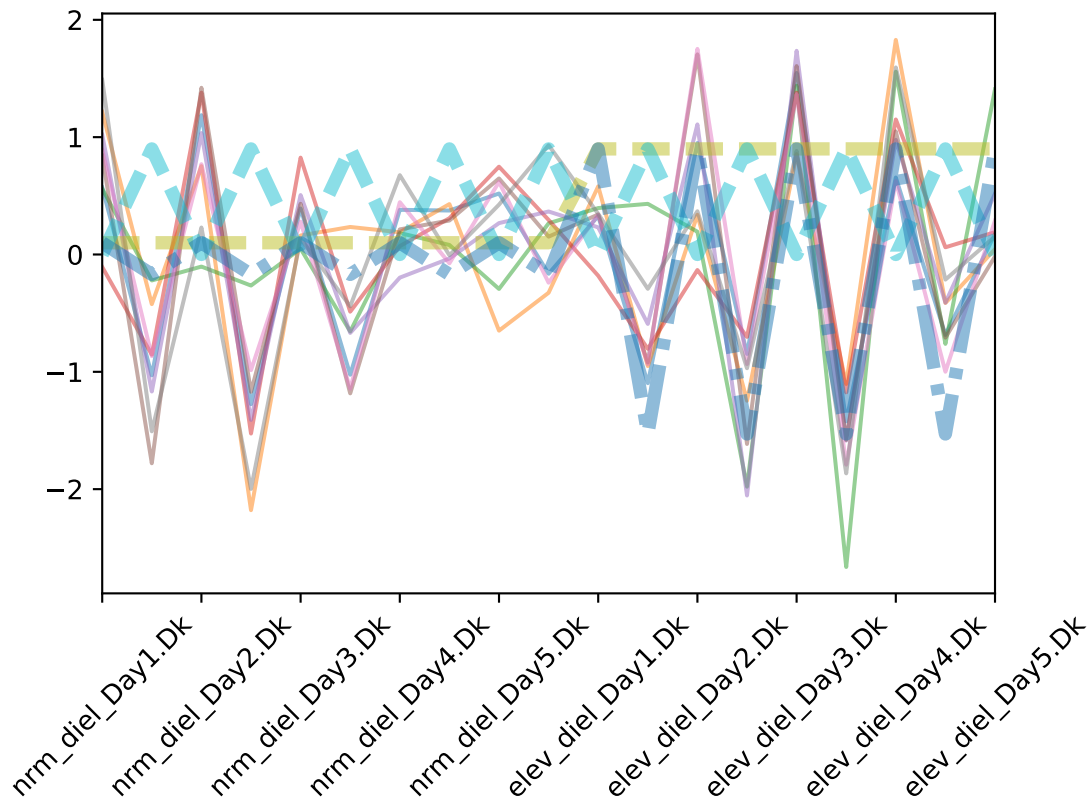
```
In [24]: ax = r1.loc[genes, colnames[(isdark | islight)]] .T.plot(legend=False, alpha=0.5)
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45)
ax.plot(co2[isdark | islight], ls='dashed', lw=5, alpha=0.5)
ax.plot(dark[isdark | islight], ls='dashed', lw=5, alpha=0.5)
ax.plot(-3. * (co2*dark)[isdark | islight] + 1. * co2[isdark | islight], lw=5, alpha=0.5, ls='dashdot')
```

```
Out[24]: [<matplotlib.lines.Line2D at 0x11c5bf978>]
```



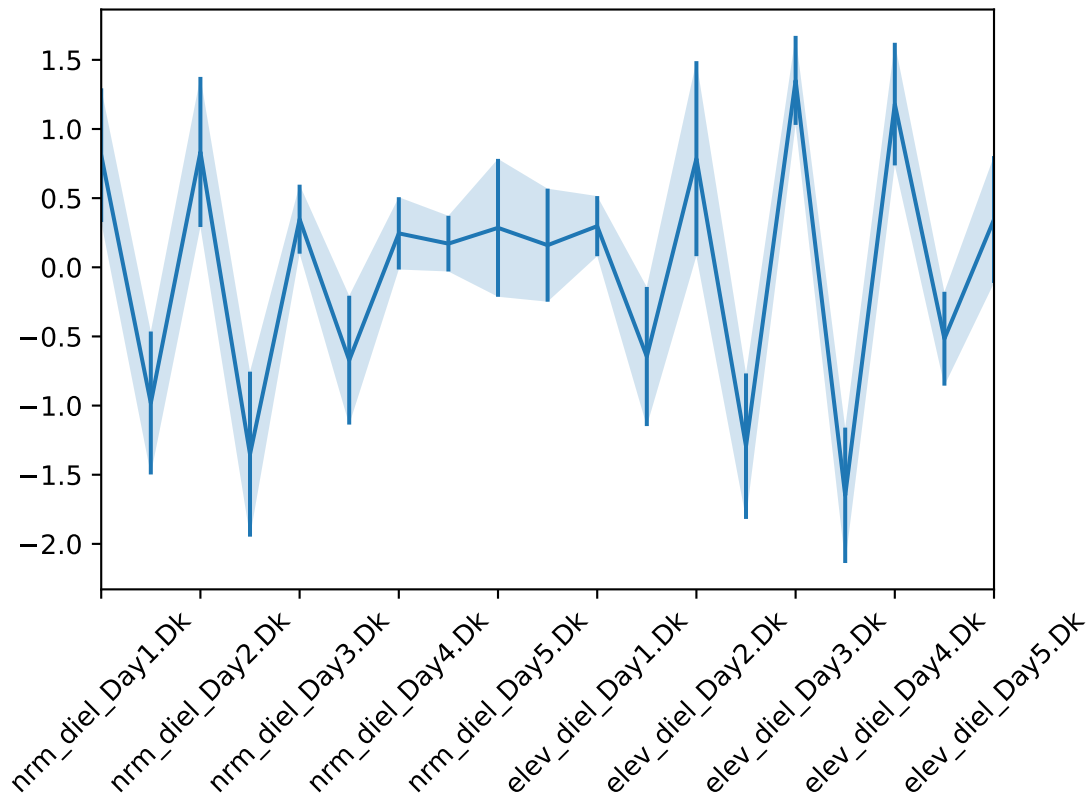
```
In [25]: r1a = r1.copy()
r1a.loc[:, colnames[(isdark | islight)]] = r1norm
ax = r1a.loc[genes, colnames[(isdark | islight)]] .T.plot(legend=False, alpha=0.5)
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45)
ax.plot(co2[isdark | islight], ls='dashed', lw=5, alpha=0.5)
ax.plot(dark[isdark | islight], ls='dashed', lw=5, alpha=0.5)
ax.plot(-3. * (co2*dark)[isdark | islight] + 1. * co2[isdark | islight], lw=5, alpha=0.5, ls='dashdot')
```

```
Out[25]: [<matplotlib.lines.Line2D at 0x11cccd2b0>]
```



```
In [26]: mn = r1a.loc[genes, colnames[(isdark | islight)]].mean()
err = r1a.loc[genes, colnames[(isdark | islight)]].std()
ax = mn.plot(yerr=err)
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
ax.fill_between(range(len(mn)), mn-err, mn+err, alpha=0.2)
```

```
Out[26]: <matplotlib.collections.PolyCollection at 0x110b36278>
```



```
In [27]: reg = linear_model.LinearRegression()
reg.fit(x.T, mn, sample_weight=1/err)
print(reg.score(x.T, y, sample_weight=1/err)) # R^2
reg.coef_
```

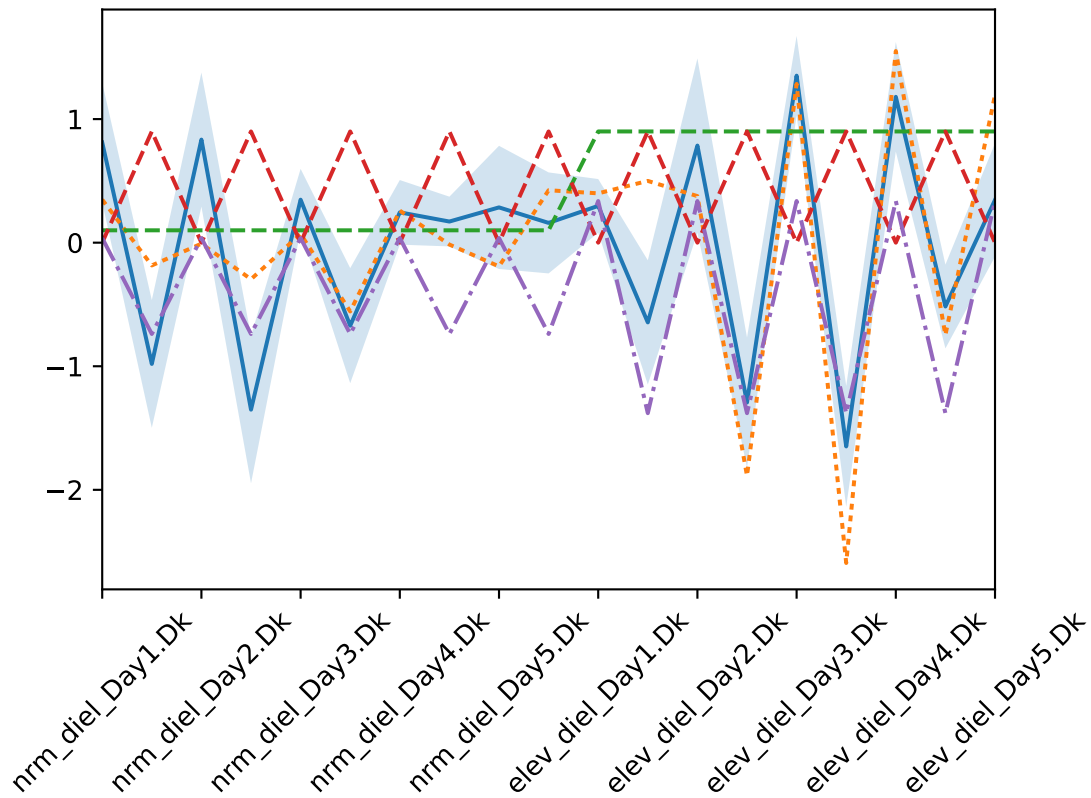
0.514930892128

```
Out [27]: array([ 0.          ,  0.37437859, -0.73216368, -1.30594988])
```

```
In [28]: ax = mn.plot() #yerr=err)
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
ax.fill_between(range(len(mn)), mn-err, mn+err, alpha=0.2)
ax.plot(r1.loc[the_gene, colnames[(isdark | islight)]].values, ls='dotted')
ax.plot(co2[isdark | islight], ls='dashed')
ax.plot(dark[isdark | islight], ls='dashed')
ax.plot(reg.coef_[3] * (co2*dark)[isdark | islight] +
        reg.coef_[1] * co2[isdark | islight] +
        reg.coef_[2] * dark[isdark | islight], ls='dashdot')
```

```
Out [28]: [matplotlib.lines.Line2D at 0x11d9387f0]
```





```
In [29]: results = sm.WLS(mn, x.T, weights=1/err**2.).fit()
         results.summary()
```

```
/Users/dreiss/miniconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1334: UserWarning: kurtosistest only valid
"anyway, n=%i" % int(n))
```

```
Out [29]: <class 'statsmodels.iolib.summary.Summary'>
        """
```

```

                                WLS Regression Results
=====
Dep. Variable:                  y    R-squared:                0.603
Model:                        WLS    Adj. R-squared:           0.523
Method:                    Least Squares    F-statistic:            7.588
Date:                Thu, 22 Jun 2017    Prob (F-statistic):      0.00257
Time:                      15:09:00    Log-Likelihood:         -13.191
No. Observations:                19    AIC:                   34.38
Df Residuals:                    15    BIC:                   38.16
Df Model:                        3
Covariance Type:                nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
const	0.3576	0.231	1.548	0.143	-0.135	0.850
x1	0.3465	0.360	0.962	0.351	-0.421	1.114
x2	-0.4321	0.366	-1.182	0.256	-1.211	0.347
x3	-1.4702	0.639	-2.300	0.036	-2.833	-0.108

```

=====
Omnibus:                1.660    Durbin-Watson:           2.906
Prob(Omnibus):           0.436    Jarque-Bera (JB):        0.931
Skew:                    -0.010    Prob(JB):                0.628

```

Kurtosis: 1.915 Cond. No. 8.22

=====

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
"""

Find clusters that may be anticorrelated with the\_gene.

```
In [30]: def get_cluster_profile(ind):
    genes = r1.index.values[np.where(labels == ind)]
    mn1 = r1a.loc[genes, colnames[(isdark | islight)]].mean()
    err1 = r1a.loc[genes, colnames[(isdark | islight)]].std()
    return mn1, err1

    for lab in range(max(labels)):
        mn1, _ = get_cluster_profile(lab)
        corr = np.corrcoef(mn1, mn)[0, 1]
        if np.abs(corr) > 0.85:
            print(lab, corr)
```

16 0.875917093914  
17 -0.886837101736  
26 0.913622075356  
51 0.947320514238  
58 0.900811857266  
103 -0.878456522929  
112 0.934666079963  
170 -0.893175978822  
174 0.879700896006  
180 0.932091999425  
235 0.902334600112  
264 0.881464889712  
309 0.913380606222  
381 0.870723547997  
384 0.85208112038  
439 0.931438462027  
443 0.934761774506  
464 0.854619922791  
495 -0.852214388152  
528 -0.885249195782  
535 0.918347994424  
552 0.869160951163  
601 1.0  
646 0.876867077841  
650 0.883170107871  
719 0.904411250974  
731 0.880618583152  
749 0.92443441189  
751 0.878472346778  
803 0.900515585077  
917 -0.868159627678  
935 0.928134219879  
956 0.851196891911  
1042 -0.860309777686  
1067 0.923141483006  
1126 0.910982499787  
1146 0.860771335809  
1149 -0.884697663519  
1198 0.8879071125

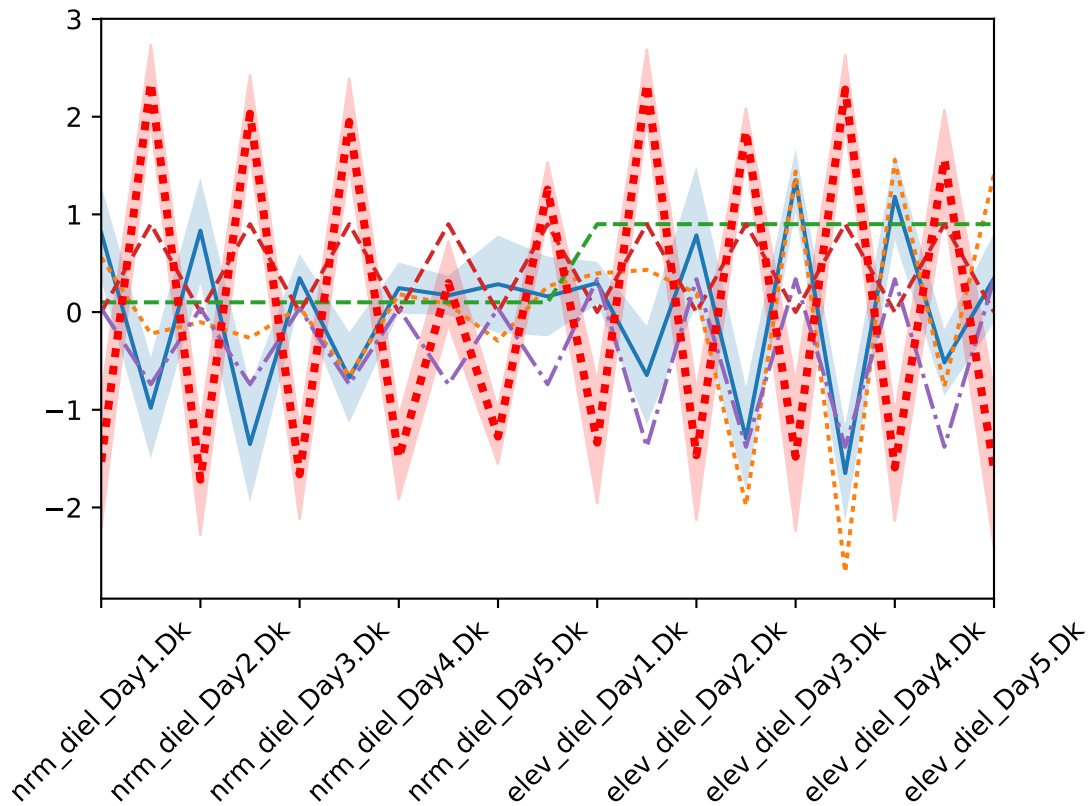
```

In [31]: ax = mn.plot() #yerr=err)
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
ax.fill_between(range(len(mn)), mn-err, mn+err, alpha=0.2)
ax.plot(r1a.loc[the_gene, colnames[(isdark | islight)]] .values, ls='dotted')
ax.plot(co2[isdark | islight], ls='dashed')
ax.plot(dark[isdark | islight], ls='dashed')
ax.plot(reg.coef_[3] * (co2*dark)[isdark | islight] +
        reg.coef_[1] * co2[isdark | islight] +
        reg.coef_[2] * dark[isdark | islight], ls='dashdot')

mn1, err1 = get_cluster_profile(528)
ax.plot(mn1.values, ls='dotted', lw=3, c='r')
ax.fill_between(range(len(mn1)), mn1-err1, mn1+err1, alpha=0.2, color='r')

```

Out[31]: <matplotlib.collections.PolyCollection at 0x110b46390>



```

In [32]: results = sm.WLS(mn1, x.T, weights=1/err1**2.).fit()
results.summary()

```

/Users/dreiss/miniconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1334: UserWarning: kurtosistest only valid for n>=30  
 "anyway, n=%i" % int(n))

Out[32]: <class 'statsmodels.iolib.summary.Summary'>  
 """

```

                                WLS Regression Results
=====
Dep. Variable:                  y    R-squared:                0.933
Model:                            WLS    Adj. R-squared:         0.920
Method:                    Least Squares    F-statistic:             70.00
Date:                Thu, 22 Jun 2017    Prob (F-statistic):       4.75e-09

```

```

Time:                15:09:05   Log-Likelihood:          -12.514
No. Observations:    19       AIC:                33.03
Df Residuals:        15       BIC:                36.80
Df Model:             3
Covariance Type:     nonrobust

```

```

=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
const         -1.4441        0.247      -5.851      0.000      -1.970      -0.918
x1             -0.0572        0.504      -0.114      0.911      -1.131       1.017
x2              3.2510        0.364       8.931      0.000       2.475       4.027
x3              0.7214        0.671       1.076      0.299      -0.708       2.151
=====
Omnibus:                5.871   Durbin-Watson:           2.496
Prob(Omnibus):          0.053   Jarque-Bera (JB):       3.527
Skew:                   -0.660   Prob(JB):               0.171
Kurtosis:               4.646   Cond. No.               10.7
=====

```

Warnings:

```

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
"""

```

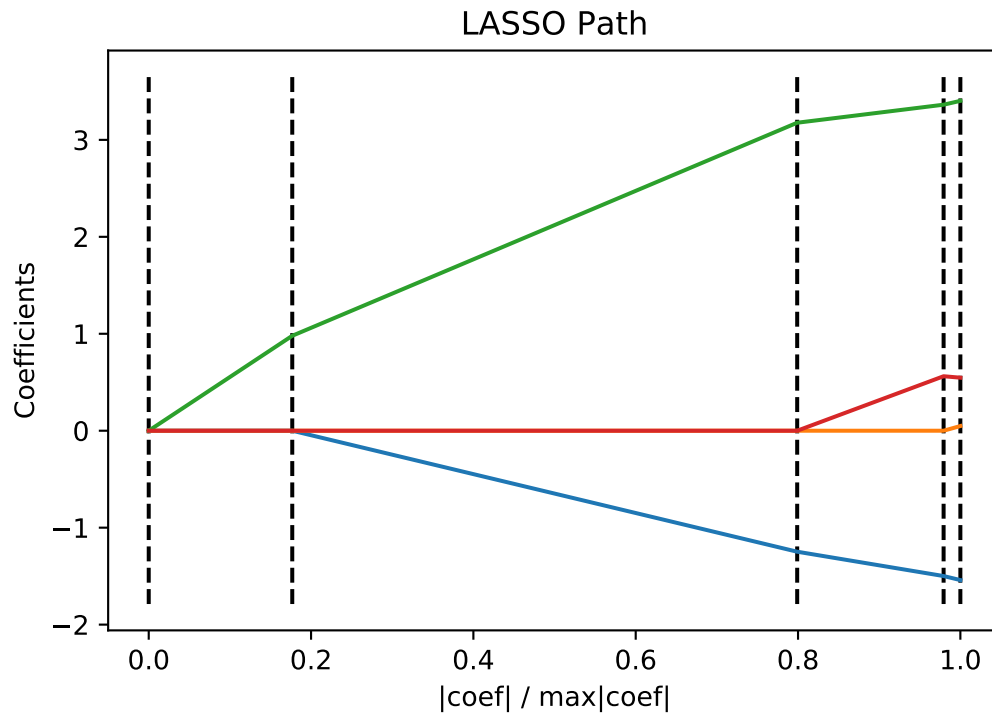
```
In [33]: alphas, _, coefs = linear_model.lars_path(x.T, mn1, method='lasso', verbose=True)
```

```

xx = np.sum(np.abs(coefs.T), axis=1)
xx /= xx[-1]

plt.plot(xx, coefs.T)
ymin, ymax = plt.ylim()
plt.vlines(xx, ymin, ymax, linestyle='dashed')
plt.xlabel('|coef| / max|coef|')
plt.ylabel('Coefficients')
plt.title('LASSO Path')
plt.axis('tight')
plt.show()

```



In [ ]:

Try fitting all clusters, see if we can find different combo's of coefficients. Especially look for ones which only are affected by co2 (param x1, or index 1).

In [34]: `reg = linear_model.LinearRegression()`

```
def fit_cluster(ind):
    mn, std = get_cluster_profile(ind)
    reg.fit(x.T, mn) #, sample_weight=np.abs(y))
    score = reg.score(x.T, mn) # R^2
    return reg.coef_, score
```

```
scores = [fit_cluster(ind) for ind in range(1200)]
```

In [35]: `#co2_coeffs = np.array([sc[0][1] for sc in scores])`  
`#print(np.min(co2_coeffs), np.max(co2_coeffs))`  
`#lab = np.argmin(co2_coeffs)`  
`#print(lab)`  
`#print(scores[lab])`

```
for i in range(1200):
    sc = scores[i]
    if np.abs(sc[0][1]) > np.abs(sc[0][2]) + np.abs(sc[0][3]): # and sc[1] > 0.4:
        print(i, sc)
```

*# Not finding much!*

```
7 (array([ 0.          ,  0.02805658, -0.01060968, -0.01539051]), 0.014527977498730138)
123 (array([ 0.          ,  0.13014645,  0.01530956, -0.06231341]), 0.0053610042689555826)
153 (array([ 0.          , -0.18053608, -0.12957559,  0.02376715]), 0.0024103419863978903)
209 (array([ 0.          ,  0.1339631 ,  0.01378686, -0.05484272]), 0.0027783321125754985)
273 (array([ 0.          ,  0.16378924,  0.10021357, -0.04324233]), 0.015875872136872027)
329 (array([ 0.          ,  0.10394438,  0.0102262 , -0.0160302 ]), 0.0064332734534413571)
```

```

351 (array([ 0.          ,  0.49059658, -0.04136066, -0.18041325]), 0.0033111682453613733)
422 (array([ 0.          , -0.28258216,  0.04348452,  0.10745876]), 0.0054957613118427062)
480 (array([ 0.          , -0.14819229, -0.07165215,  0.01274871]), 0.011380127029799403)
564 (array([ 0.          ,  0.26590792,  0.01590301, -0.13498175]), 0.0087819132474280526)
570 (array([ 0.          , -0.88246099, -0.4512164 ,  0.16506981]), 0.015400235591607081)
822 (array([ 0.          ,  0.10831328,  0.04421293,  0.00537372]), 0.0084467502221186042)
1069 (array([ 0.          , -0.40087296,  0.08233548,  0.26586085]), 0.010932752350386599)

```

Try to look for all cases of high interaction (co2 \* light):

```

In [36]: print(scores[601])

for i in range(1200):
    sc = scores[i]
    if np.abs(sc[0][3]) > 1. and np.abs(sc[0][3]) > 2.*np.abs(sc[0][2]) and sc[1] > 0.4:
        print(i, sc)

(array([ 0.          ,  0.35925953, -1.04670126, -1.08165068]), 0.71631563247766072)
459 (array([ 0.          , -1.15503253,  0.47029975,  2.98318686]), 0.51592706479888784)
755 (array([ 0.          , -1.6113901 , -0.20309454,  4.11958912]), 0.58824159983905022)
1105 (array([ 0.          , -0.63498353,  0.49535103,  1.50899811]), 0.55388881195919604)

```

```

In [37]: mn1, err1 = get_cluster_profile(755)
ax = mn1.plot()
#ax.plot(mn1.values, ls='dotted', lw=3, c='r')
ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45);
ax.fill_between(range(len(mn1)), mn1-err1, mn1+err1, alpha=0.2, color='r')

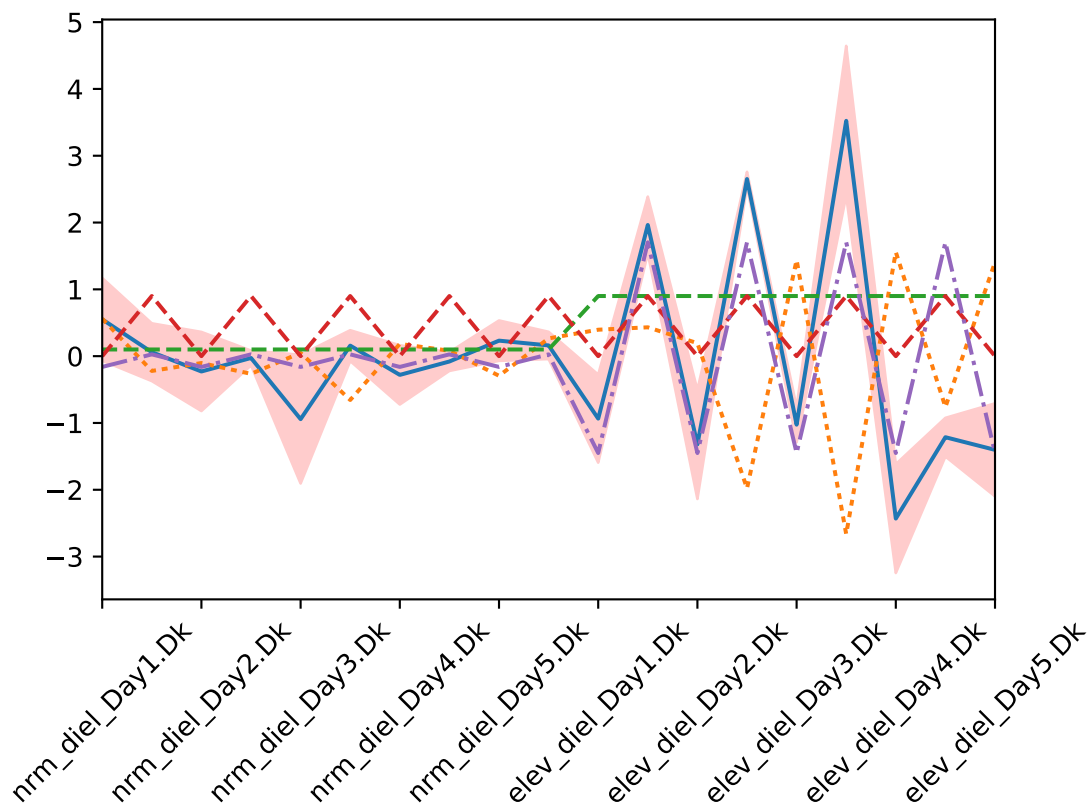
ax.plot(r1a.loc[the_gene, colnames[(isdark | islight)]].values, ls='dotted')
ax.plot(co2[isdark | islight], ls='dashed')
ax.plot(dark[isdark | islight], ls='dashed')
coef = scores[755][0]
ax.plot(coef[3] * (co2*dark)[isdark | islight] +
        coef[1] * co2[isdark | islight] +
        coef[2] * dark[isdark | islight], ls='dashdot')

```

```

Out[37]: [<matplotlib.lines.Line2D at 0x12199aeb8>]

```



In [ ]: