```
## Supplementary File S3 For "SNAIL driven by a Feed Forward Loop
Motif Promotes TGFβ Induced Epithelial to Mesenchymal Transition"

# -*- coding: utf-8 -*-

import sys
import random
import csv
import boolean2
import pylab
from boolean2 import util, Model, network, state
import networkx
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

text = """
#initial values
Sx = True
Sy = True
SMAD = False
MDM2 = False
SNAIL = False
#updating rules
Sx* = Sx
Sy* = Sy
SMAD* = Sx
MDM2* = Sy or SMAD
SNAIL* = MDM2 and SMAD
"""


def set_value(state, name, value, p):
        "Custom value setter"
        global s
        if name == 'Sx' or name == 'Sy':
                s = s+1
                #print s
        #if (s >= 9 and s <=10.5) or (s >= 44):
        if (s > 8 and s<21):# or (s >= 44):
                if (name == 'Sy') or (name == 'Sx'):
                        value = True
                setattr(state, name, value)
                return value
        else:
                if (name == 'Sx') or (name == 'Sy'):
                        value = False
                setattr(state, name, value)
                return value
```

```
coll=util.Collector()
for i in range(500):
        s = 0
        #print s
        model = Model( text=text, mode='sync')
        model.parser.RULE_SETVALUE = set_value
        model.initialize()
        model.iterate(steps = 5000)

        #model.report_cycles()
        #nodes = model.nodes
        coll.collect(states=model.states, nodes=model.nodes )
#count = 0
#for index in model.states:
#       count = count+1
        #print(count)
averages = coll.get_averages( normalize=True )
df = pd.DataFrame(averages)
csv_data = df.to_csv('FFL_PD_i_or1.csv')

data = pd.read_csv('FFL_PD_i_or1.csv')
data = data.drop(data.columns[0], axis=1)
data = data[['Sx','Sy','SMAD','MDM2','SNAIL']]
data = data.transpose()
fig, ax = plt.subplots()
sns.heatmap(data, center=0, cmap='Reds', xticklabels=True,
yticklabels=True)
#sns.set(font_scale = 12)
#ax.set(xticklabels=data['header'])
#ax.set_ylim(0,50)
ax.set_xlim(0,15)
#ax.invert_yaxis()
#ax.set_title('Node Activity', fontweight='bold')
ax.set_xlabel('Time Steps', fontsize = 12, fontweight='bold')
plt.xticks(fontsize=11)
plt.yticks(fontsize=10)
#manager = plt.get_current_fig_manager()
#manager.resize(*manager.window.maxsize())
figure = plt.gcf()  # get current figure
#figure.set_size_inches(20, 10) # set figure's size manually to your
full screen (32x18)
plt.savefig('FFL_HM_AND_1.jpeg', bbox_inches='tight') # bbox_inches
removes extra white spaces


fig, axs = plt.subplots(5, sharex = True)
p1 = axs[2].plot( averages["SMAD"], alpha = 0.7, linewidth = 1.0,
marker = "o", color ='k' )
a1 = axs[3].plot( averages["MDM2"], alpha = 0.7, linewidth = 1.0,
marker = "o", color ='k')
```

```
b1 = axs[4].plot( averages["SNAIL"], alpha = 0.7, linewidth = 1.0,
marker = "o", color = 'k' )
c1 = axs[0].plot( averages["Sx"], alpha = 0.7, linewidth = 1.0, marker
= "o", color = 'k' )
d1 = axs[1].plot( averages["Sy"], alpha = 0.7, linewidth = 1.0, marker
= "o", color = 'k' )

for ax in axs.flat:
        #axs[0].set_xlim(0,10)
        #axs[0].set_ylim(0,1)
        #axs[0].set_xlabel('Time', fontsize = 10)
        axs[0].set_ylabel('Sx', fontsize = 10, fontweight='bold')
        axs[0].set_yticks([0.0, 0.5, 1.0])
        axs[0].tick_params(axis="y", labelsize=10)
        #axs[1].set_xlim(0,10)
        #axs[1].set_ylim(0,1)
        #axs[1].set_xlabel('Time', fontsize = 10)
        axs[1].set_ylabel('Sy', fontsize = 10, fontweight='bold')
        axs[1].set_yticks([0.0, 0.5, 1.0])
        axs[1].tick_params(axis="y", labelsize=10)
        #axs[2].set_xlim(0,10)
        #axs[0].set_xlabel('Time', fontsize = 10)
        axs[2].set_ylabel('SMAD', fontsize = 10, fontweight='bold')
        axs[2].tick_params(axis="y", labelsize=10)
        #axs[3].set_xlim(0,10)
        #axs[1].set_xlabelTime', fontsize = 10)
        axs[3].set_ylabel('MDM2', fontsize = 10, fontweight='bold')
        axs[3].tick_params(axis="y", labelsize=10)
        axs[4].set_xlim(0,15)
        #axs[2].set_xlabel('Time', fontsize = 10)
        axs[4].set_ylabel('SNAIL', fontsize = 10, fontweight='bold')
        axs[4].tick_params(axis="y", labelsize=10)

plt.setp(ax.get_xticklabels(), fontsize=14)

#fig.add_subplot(111, frame_on=False)
fig.text(0.08, 0.4, 'Node Activity', ha='center', rotation='vertical',
fontsize = 16, fontweight='bold')
fig.text(0.55,0.03, 'Time Steps', ha='right', rotation='horizontal',
fontsize = 16, fontweight='bold')
#plt.tick_params(labelcolor="none", bottom=False, left=False)
#plt.xlabel('Time Steps', fontsize = 16, fontweight='bold')
#plt.ylabel('Node Activity', fontsize = 16)

manager = plt.get_current_fig_manager()
#manager.resize(*manager.window.maxsize())
#fig.savefig('TGFb_MDM2_on_sync.png')
figure = plt.gcf()  # get current figure
figure.set_size_inches(10,6) # set figure's size manually to your full
screen (32x18)
```

```
plt.savefig('FFL_AND_1.jpeg') # bbox_inches removes extra white spaces
plt.show()
```