```
## Supplementary File S2 For "SNAIL driven by a Feed Forward Loop
Motif Promotes TGFβ Induced Epithelial to Mesenchymal Transition"

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

import sys
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

EXTGFB = False
ENTGFB = False
SNAIL = False
MIR200 = True
SMAD7 = True
RSCYT = False
RSNUC = False
RSPCYT = False
RSPNUC = False
COCYT = True
CONUC = False
RSPCYTCOCYT = False
RSPNUCCONUC = False
SMURF = True
PPM1A = True
TCF1 = True
TCF2 = True
MDM2 = False
DNADAMAGE = True
P53 = True
MIR34A = True
ZEB = False
EC = True
NC =  False
TBR = False
TBRP = False
EMT = False

# UPDATING RULES
ENTGFB* = SNAIL and not MIR200
```

```
TBR* = EXTGFB or (TBRP and SMAD7) or ENTGFB
TBRP* = (TBR and ENTGFB) or (TBR and EXTGFB)
RSCYT* = TBRP or RSNUC
RSPCYT* = (TBRP and RSCYT ) or RSPNUC or (MDM2 and RSCYT) or
RSPCYTCOCYT
RSPCYTCOCYT* = RSPCYT and COCYT
RSNUC* = RSCYT or (RSPNUC and PPM1A)
RSPNUC* = RSPNUCCONUC or RSPCYT and not SMURF
RSPNUCCONUC* = RSPCYTCOCYT  or (RSPNUC and CONUC)
CONUC* = RSPNUCCONUC  or COCYT
MDM2* = (TCF1 and RSPNUCCONUC) or P53
P53* = DNADAMAGE and not MDM2
SNAIL* = (TCF2 and RSPNUCCONUC) or MDM2 and not MIR34A
MIR34A* = P53 and not SNAIL and not ZEB
ZEB* = SNAIL and not MIR200
MIR200* = P53 and not SNAIL and not ZEB
EC* = not SNAIL and not ZEB
NC* = SNAIL and ZEB
EMT* = NC and not EC
"""

on = []
off = []
text = boolean2.modify_states(text = text , turnon=on, turnoff=off)

coll  = util.Collector()
for i in range(500):
        model = Model( text=text, mode='async')
        model.initialize()
        model.iterate( steps = 5000 )



# this is a helper function that reports the cycle lengths
# and the  index at wich the cycle started
        model.report_cycles()

# the same thing as above but
# will not print only return the two parameters
#
#print(model.detect_cycles())

#collecting states data for heatmap and putting it in CSV

        nodes = model.nodes
        coll.collect( states=model.states, nodes=nodes )
averages = coll.get_averages( normalize=True )
df = pd.DataFrame(averages)
csv_data = df.to_csv('P53_DD1_Ex0_T10_T20.csv')
```

```
#plotting a heat map by calling the csv file gathered above

data = pd.read_csv('P53_DD1_Ex0_T10_T20.csv')
data = data.drop(data.columns[0], axis=1)
data = data.drop(['SMAD7', 'SMURF','PPM1A'], axis=1)
data =
data[["EXTGFB","ENTGFB",'TBR',"TBRP",'RSCYT','RSPCYT','COCYT','RSPCYTC
OCYT','RSNUC','RSPNUC','CONUC','RSPNUCCONUC','DNADAMAGE','TCF1','TCF2'
,'MDM2','P53','SNAIL','MIR34A','ZEB','MIR200','EC','NC','EMT']]
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(0,100)
#ax.invert_yaxis()
ax.set_title('Node Activity')
ax.set_xlabel('Time Steps', fontsize = 12)
manager = plt.get_current_fig_manager()
#manager.resize(*manager.window.maxsize())
#manager.frame.Maximize(True)
figure = plt.gcf()  # get current figure
figure.set_size_inches(24,10) # set figure's size manually to your
full screen (32x18)
plt.savefig('P53_HM_DD1_Ex0_T10_T20.jpeg', bbox_inches='tight', dpi =
500)


#sns.heatmap(np.rot90 (a, k = 1))
#plt.show()
#plt.savefig('TGFb_SNAIL_on_HM_sync.png', bbox_inches = "tight", dpi =
500)


#Graphic representation of the data other than heat map
# this is how one plots the values, delete this below
# if matplotlib is not installed

fig, axs = plt.subplots(2,2)

p1 = axs[0,0].plot( averages["EXTGFB"], alpha = 0.5, linewidth = 0.7,
marker = "o", color ='b' )
p2 = axs[0,0].plot( averages["ENTGFB"], alpha = 0.5, linewidth = 0.7,
marker = "v", color ='m')
p3 = axs[0,0].plot( averages["TBR"], alpha = 0.5, linewidth = 0.7,
marker = "^" , color ='r')
p4 = axs[0,0].plot( averages["TBRP"], alpha = 0.5, linewidth = 0.7,
marker = "s" , color ='k')
```

```
a1 = axs[0,1].plot( averages["RSCYT"], alpha = 0.5, linewidth = 0.7,
marker = "o", color ='y')
a2 = axs[0,1].plot( averages["RSPCYT"], alpha = 0.5, linewidth = 0.7,
marker = "v", color ='b')
a3 = axs[0,1].plot( averages["RSNUC"], alpha = 0.5, linewidth = 0.7,
marker = "*", color ='k')
a4 = axs[0,1].plot( averages["RSPNUC"] ,alpha = 0.5, linewidth = 0.7,
marker = "^" , color ='r')
a5 = axs[0,1].plot( averages["COCYT"], alpha = 0.5, linewidth = 0.7,
marker = "+", color ='m')
a6 = axs[0,1].plot( averages["CONUC"], alpha = 0.5, linewidth = 0.7,
marker = "s", color ='c')
a7 = axs[0,1].plot( averages["RSPNUCCONUC"], alpha = 0.5, linewidth =
0.7, marker = "s", color ='g')


b1 = axs[1,1].plot( averages["MDM2"], alpha = 0.5, linewidth = 0.7,
marker = "o", color = 'b' )
b2 = axs[1,1].plot( averages["SNAIL"],  alpha = 0.5, linewidth = 0.7,
marker = "v", color = 'm')
b3 = axs[1,1].plot( averages["ZEB"], alpha = 0.5, linewidth = 0.7,
marker = "^" , color ='r')
b4 = axs[1,1].plot( averages["NC"], alpha = 0.5, linewidth = 0.7,
marker = "P", color ='y')
b5 = axs[1,1].plot( averages["EMT"], alpha = 0.5, linewidth = 0.7,
marker = "s", color ='c')


c1 = axs[1,0].plot( averages["MIR34A"], alpha = 0.5, linewidth = 0.7,
marker = "v", color ='m')
c2 = axs[1,0].plot( averages["MIR200"], alpha = 0.5, linewidth = 0.7,
marker = "^", color ='r')
c3 = axs[1,0].plot( averages["P53"] , alpha = 0.5, linewidth = 0.7,
marker = "o" , color ='b')
c4 = axs[1,0].plot( averages["EC"], alpha = 0.5, linewidth = 0.7,
marker = "p", color = 'y' )

for ax in axs.flat:
        axs[0,0].legend(["exTGFb","enTGFb","TbR","TbRP"], prop =
{"size":12})
        axs[0,1].legend(["RScyt",
"RSPcyt","RSnuc","RSPnuc","Cocyt","Conuc","RSPNUCCONUC"], prop =
{"size":12})
        axs[1,0].legend(["miR34","miR200", "p53", "EC"], prop =
{"size":12})
        axs[1,1].legend(["MDM2","SNAIL","ZEB","NC","EMT"], prop =
{"size":12})
        axs[0,0].set_xlim(0,100)
        axs[0,0].set_ylim(0,1)
```

```
        axs[0,1].set_xlim(0,100)
        axs[0,1].set_ylim(0,1)
        axs[1,0].set_xlim(0,100)
        axs[1,0].set_ylim(0,1)
        axs[1,1].set_xlim(0,100)
        axs[1,1].set_ylim(0,1)

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(24, 16) # set figure's size manually to your
full screen (32x18)

plt.savefig('P53_dyn_DD1_Ex0_T10_T20.jpeg', bbox_inches='tight', dpi =
500)
plt.show()
```