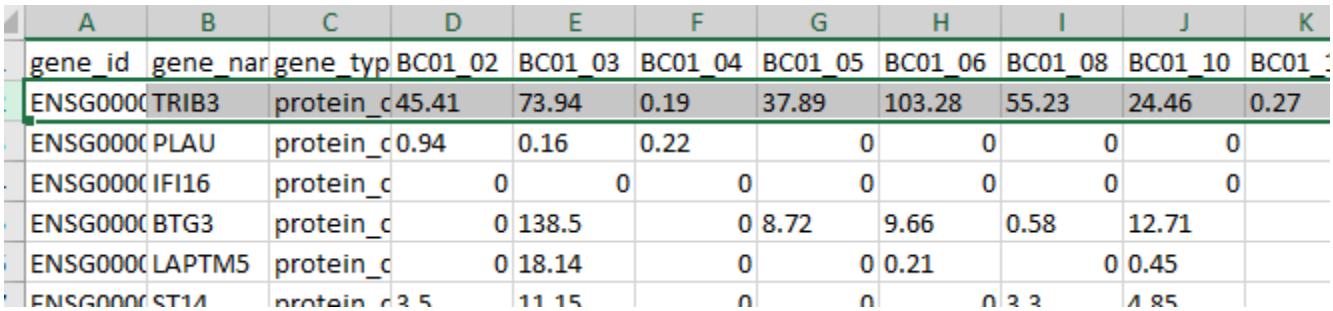
EXECUTIVE GUIDE TO THE METHOD USED IN THE PAPER *DATA-DRIVEN MODELING OF BREAST CANCER TUMORS USING BOOLEAN NETWORKS*

STEP 1

The first operation is to enter the excel sheet with the Single-cell RNA-seq data (SUPPLEMENTARY_TABLE_1), select the first row relating to the TRIB3 gene and save it in a text file (see the TRIB.3.txt text file in the repository)



	A	B	C	D	E	F	G	H	I	J	K
	gene_id	gene_nar	gene_typ	BC01_02	BC01_03	BC01_04	BC01_05	BC01_06	BC01_08	BC01_10	BC01_1
	ENSG0000	TRIB3	protein_c	45.41	73.94	0.19	37.89	103.28	55.23	24.46	0.27
	ENSG0000	PLAU	protein_c	0.94	0.16	0.22	0	0	0	0	
	ENSG0000	IFI16	protein_c	0	0	0	0	0	0	0	
	ENSG0000	BTG3	protein_c	0	138.5	0	8.72	9.66	0.58	12.71	
	ENSG0000	LAPTM5	protein_c	0	18.14	0	0	0.21	0	0.45	
	ENSG0000	ST14	protein_c	3.5	11.15	0	0	0	3.3	4.85	

STEP 2

We now proceed with the binarization of the RNA-seq values detected for TRIB3, a gene present in every cell. To do this, we use the script R "Binarize", specifying in line 6 (yellow highlight) the name of the file to be processed (TRIB3.txt). Make sure the location of the file in the filesystem matches the script workspace.

```
#Package installation of "Binarize"
install.packages("Binarize")
#Package loading
library("Binarize")
#Loading the data file
dati=read.table("TRIB3.txt", header=T)
str(dati)
#Setting binarization method
result=binarize.BASC(dati[,"TRIB3"], method="B", tau=0.1)
print(result)
```

The result produced by the script above is the following:

```
Method: BASC B

Threshold: 95.2

Binarized vector: [ 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 1
 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 ]

p value: 0.001
```

The user should save the result in a line of an excel spreadsheet. Then, by carrying out the operations described in STEP 1 and STEP 2 for all the rows (which represent the genes of the network) present in SUPPLEMENTARY_TABLE_1, we obtain as a final result the sheet SUPPLEMENTARY_TABLE_6

gene_nam	BC01_02	BC01_03	BC01_04	BC01_05	BC01_06	BC01_08	BC01_10	BC01_
TRIB3	0	0	0	0	1	0	0	
PLAU	0	0	0	0	0	0	0	
IFI16	0	0	0	0	0	0	0	
BTG3	0	1	0	0	0	0	0	
LAPTM5	0	0	0	0	0	0	0	
ST14	0	0	0	0	0	0	0	

After this procedure, SUPPLEMENTARY_TABLE_6 contains, in binarized form, all the gene expression values of the genes present in the network (rows) for each cell present in the repository used (columns).

STEP 3

We now proceed to search for any attractors produced by the dynamics of the gene regulation network for each cell present in the repository. To achieve this, we use the following python script (attractor.py). This script uses the BooleanNet library (https://github.com/ialbert/booleannet).

In [ ]:

```
import boolean2 as b2
```

```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import pylab
import pandas as pd


list=[]
while True:
    try:
        #insert name and extension file excel with binary values
        Table=raw_input("Please enter the excel file name: ")

        #get a dataframe from the excel sheet
        df=pd.read_excel(Table)

        try:
            #indicate the cell in which to look for attractors
            cellname=raw_input("Please enter the cell name: ")

            #tranforms the values of the columns to be processed from "1"
and"0"
            #to "True" and "False"
            df['TF']=df[cellname].apply(lambda x : 'True' if x == 1 else
'False')
        except IOError:
            print("The cell name is incorrect")

    except IOError:
         print("The file name is incorrect")
         continue
    break

#create a new dataframe with only two columns,the name of the genes and
#their Boolean value "True" or "False"
mod_df=df[['gene_name','TF']]

#create a list with values in columns of the dataframe "mod_df"
#interspersed with the symbol "="
for i in mod_df.index:
    list.append(mod_df["gene_name"][i] + " = " + mod_df["TF"][i]+"\n")

#tranformation of the list format into a string required for processing
model_definition1='  '.join(list)
```

```
#print(model_definition1)



# Update rule
model_definition2= '''
TRIB3* = not TP53
PLAU* = (TP53 or NFKB1 or HIF1A or ATF2 or ETS1 or PLAUR or ST14 or RELA) and
not HDAC1
IFI16* = TP53
BTG3* = TP53
LAPTM5* = TP53
ST14* = (TP53 or PPARD or STAT1 or ST14) and not RELA
EPHA2* = (TP53 or AKT2 or ETS1 or CREB1) and not(MTA1 or RELA)
PML* = (TP53 or IRF9 or STAT5A or STAT1 or RB1) and not CSNK2B
STAT5A* = (TP53 or HSP90AB1 or RELA or STAT5A) and not(BRCA1 or UBE2D1)
CD82* = TP53 or HIF1A
TP53* = (TP53 or CREB1 or NFKB1 or E2F1 or HIF1A or STAT1 or BRCA1 or
HSP90AB1 or ETS1 or PML or YWHAB or PLK2 or IRF9 or IFI16 or RELA) and
not(HDAC1 or STAT3 or CSNK2B or PARP1 or EZH2 or ARRB1 or PSMD4 or PLAU or
BCL2L1 or BCL6 or SETDB1 or EEF1G or UBE2D1 or WWP1 or TRIM29 or MTA1)
PLK2* = TP53
CTSD* = TP53 or CTSD
KRT15* = TP53
DDR1* = TP53 or E2F1 or DDR1 or TM4SF1
SERPINA3* = TP53 or NFKB1 or STAT3 or STAT1 or RELA
EZH2* = (E2F1 or STAT3 or REL or HSP90AB1 or STAT5A) and not(TP53 or RB1)
RELA* = (E2F1 or ATF2 or CSNK2B or BRCA1 or PARP1 or EZH2 or ARRB1 or AKT2 or
HIF1A or CREB1 or FOS) and not(TP53 or STAT1 or PML or SQSTM1 or TRIB3 or
HDAC1)
ETS1* = (FOS or HIF1A or ARNT) and not(TP53 or RB1 or RELA)
MCM7* = (E2F1 or CCND1) and not(TP53 or RB1)
ACTB* = (HSP90AB1 or SYNPO) and not TP53
VIM* = (NFKB1 or E2F1 or STAT3 or FOS or HIF1A or ATF2 or STAT1 or PARP1 or
HSP90AB1 or EEF1G or ARRB1 or STAT5A or SQSTM1 or HSPB1 or ITGB4 or RELA) and
not(CREB1 or ANXA1 or TP53)
MTA1* = RELA and not(TP53 or PARP1)
ITGB4* = (NFKB1 or ERBB2 or AKT2) and not(TP53 or HDAC1)
CAV1* = (STAT3 or ETS1 or EGFR or ARNT or PLAUR or CD82 or RELA) and not(TP53
or SQSTM1)
SETDB1* = AKT2 and not TP53
KRT17* = (FOS or STAT1) and not(TP53 or BRCA1 or EZH2)
```

HSP90AB1* = (HDAC1 or STAT3 or CSNK2B or STAT1 or HSPB1) and not TP53
CCND1* = (CREB1 or NFKB1 or E2F1 or FOS or ATF2 or STAT1 or PARP1 or REL or EZH2 or ETS1 or STAT5A or MTA1 or EGFR or RELA) and not(HDAC1 or TP53 or PML or CAV1 or PSMD4 or HINT1 or TNRC6A or HIF1A or NR3C1)
E2F1* = (CREB1 or E2F1 or ARRB1) and not(TP53 or HDAC1 or PARP1 or IFI16 or BTG3 or RELA or NFKB1 or HDAC1 or SETDB1 or RB1 or E2F4)
BCL2L1* = (CREB1 or NFKB1 or STAT3 or FOS or HIF1A or ATF2 or STAT1 or REL or ETS1 or STAT5A or NFE2L2 or GABP or RELA) and not(TP53 or HDAC1 or ERBB2 or BCL2L1)
BRCA1* = (CREB1 or PARP1 or NFE2L2 or AKT2 or GABP) and not(TP53 or HDAC1 or RB1 or MTA1 or PML)
HIF1A* = (CREB1 or NFKB1 or STAT3 or BRCA1 or REL or HSP90AB1 or ARRB1 or MTA1 or ARNT or RELA or ATF2 or E2F1 or HIF1A or PARP1 or HDAC1 or SAT1) and not(TP53 or PSMD4 or MCM7 or SQSTM1 or STAT1)
PLAUR* = (NFKB1 or FOS or HIF1A or ATF2 or PLAU or RELA or CAV1) and not(TP53 or HDAC1)
STAT1* = (CREB1 or BRCA1) and not(ARRB1 or PML or HDAC1)
STAT3* = (CREB1 or STAT3 or FOS or ATF2 or EZH2 or MTA1 or HIF1A or NFKB1 or SETDB1 or CD44 or RELA) and not(HDAC1 or RB1 or PML or CAV1 or CCND1)
TGFB1* = (CREB1 or NFKB1 or STAT3 or HIF1A or FOS or ATF2 or PPARD or RELA or REL) and not(HSP90AB1 or NFE2L2)
RB1* = (CREB1 or E2F1 or ATF2 or BRCA1 or GABP or PML) and not(HDAC1 or BCL6)
FOS* = (CREB1 or E2F1 or STAT3 or FOS or ATF2 or STAT1 or PARP1 or STAT5A or NFE2L2 or IKBKG) and not HDAC1
CREB1* = CREB1 or ARRB1 or AKT2
PLAT* = CREB1 or ATF2 or RELA or FOS
SQSTM1* = NFKB1 or RELA or NFE2L2 or FOS
REL* = HIF1A and not NFKB1
NFKB1* = (FOS or BRCA1 or PARP1 or PSMD4 or RELA or HIF1A or ETS1) and not(E2F1 or HDAC1 or ARRB1)
CD44* = (FOS or NFKB1 or ETS1 or RELA) and not E2F1
TK1* = E2F1
PARP1* = ETS1 and not(AKT2 or PARP1 or HDAC1)
SOD1* = (NFE2L2 or RELA) and not(HDAC1 or CAV1 or PSMD4)
HDAC1* = (CSNK2B or STAT3 or EZH2 or RB1 or STAT5A or MTA1 or CCND1 or RELA) and not HDAC1
LDHB* = STAT3 or PPARD
CSNK2B* = (STAT3 or CSNK2B or HSP90AB1 or ETS1 or PLAU) and not ACTB
KRT5* = FOS and not BRCA1
TGFBR2* = FOS or TGFB1
ARNT* = (HIF1A or RELA or BRCA1) and not STAT5A
YWHAB* = (PPARD or AKT2) and not SOD1
ARAF* = CSNK2B

```
NFE2L2* = (CSNK2B or BRCA1 or NFE2L2 or AKT2) and not(EZH2 or PML or CAV1 or
RELA)
S100A10* = STAT1
PSMD4* = PARP1
HMGCR* = PARP1
AKT2* = (ARRB1 or HSP90AB1 or CAV1) and not(BRCA1 or EZH2 or TRIB3)
TNRC6A* = HSP90AB1
ERBB2* = HSP90AB1 or EGFR or HSPB1 or ITGB4 or ETV1
IKBKG* = HSP90AB1 or ANXA1 or MAP3K7 or SQSTM1
ATF2* = RB1 or BRCA1
PPARD* = not HSP90AB1 and not RELA
SLC3A2* = NFE2L2 or RELA
UBQLN1* = GABP
SYNPO* = YWHAB
BAX* = not BCL2L1
EGFR* = EGFR or ERBB2
ETV1* = ERBB2
MAP3K7* = MAP3K7 or TGFBR1
TGFBR1* = (TGFB1 or TGFBR2 or YWHAB) and not CAV1
SMAD4* = not MAP3K7
MAP3K4* = GADD45B
INSR* = PPARG
A2M* = PPARG
PAX6* = PPARG
CYP3A4* = NR3C1
NEDD9* = NR3C1
GADD45B* = REL or TP53 or E2F1 or MYC
PPARG* = (NCOA4 or AKT2) and not SMAD3
NR3C1* = AR
SMAD3* = PPARG
RELA* = PCGF5
'''
#union of the two defined strings
model_definition = model_definition1 + model_definition2

#Refers to the text containing in the model definition and the model update
model = b2.Model(text=model_definition, mode='sync')
model.initialize()
#Number of iteration
model.iterate(steps=number)

for node in model.data:
    print node, model.data[node]
#Cheking for fixed states
```

```
model.report_cycles()

#Save the result
model.save_states('attractors.xls')
```

For attractor search, the user should insert in the appropriate spaces the name of the excel file created in steps 1 and 2 (SUPPLEMENTARY_TABLE_6).The line of code where to insert the file name is: `Table=raw_input("Please enter the excel file name: ")` **. The second step is to insert the name of the cell in which you want to search for the presence of attractors. For example, we can insert the name "BC04-54" (present on the first line of the sheet), which indicates cell 54 of the patient BC04 in the line of the code** `cellname=raw_input("Please enter the cell name: ")`**, set the number of iterations that the program will perform before reaching the desired result** `model.iterate(steps=number)` **and indicate the name of the file where the results will be saved** `model.save_states('attractors.xls'`**. The following figure illustrates the result obtained for the BC04_54 cell:**

	A	B	C	D	E	F	G	H	I	J	K	L	
9	2360	False	False	False	False	False	False	False	True	True	True	False	Fals
10	2361	False	False	False	False	False	False	False	True	True	True	False	Fals
11	2362	False	True	False	False	False	True	False	True	False	True	False	Fals
12	2363	False	True	False	False	False	True	False	True	True	True	False	Fals
13	2364	False	True	False	False	False	False	False	True	False	True	False	Fals
14	2365	False	False	False	False	False	False	False	True	True	True	False	Fals
15	2366	False	True	False	False	False	False	True	True	False	True	False	Fals
16	2367	False	False	False	False	False	True	False	True	True	True	False	Fals
17	2368	False	True	False	False	False	False	False	True	True	True	False	Fals
18	2376	False	True	False	False	False	True	False	True	False	True	False	Fals
19	2377	False	True	False	False	False	False	False	True	True	True	False	Fals
20	2378	False	True	False	False	False	False	False	True	False	True	False	Fals
21	2379	False	False	False	False	False	False	False	True	True	True	False	Fals
22	2380	False	True	False	False	False	False	False	True	False	True	False	Fals
23	2381	False	False	False	False	False	True	False	True	True	True	False	Fals
24	2382	False	True	False	False	False	False	False	True	False	True	False	Fals
25	2383	False	False	False	False	False	True	False	True	True	True	False	Fals
26	2384	False	True	False	False	False	False	True	True	False	True	False	Fals
27	2385	False	False	False	False	False	True	False	True	True	True	False	Fals
28	2386	False	True	False	False	False	False	True	True	True	True	False	Fals
29	2387	False	True	False	False	False	True	False	True	False	True	False	Fals
30	2377	False	True	False	False	False	False	False	True	True	True	False	Fals
31	2378	False	True	False	False	False	False	False	True	False	True	False	Fals
32	2379	False	False	False	False	False	False	False	True	True	True	False	Fals
33	2380	False	True	False	False	False	False	False	True	False	True	False	Fals
34	2381	False	False	False	False	False	True	False	True	True	True	False	Fals
35	2382	False	True	False	False	False	False	False	True	False	True	False	Fals
36	2383	False	False	False	False	False	True	False	True	True	True	False	Fals
37	2384	False	True	False	False	False	False	True	True	False	True	False	Fals
38	2385	False	False	False	False	False	True	False	True	True	True	False	Fals
39	2386	False	True	False	False	False	False	True	True	True	True	False	Fals
40	2387	False	True	False	False	False	True	False	True	False	True	False	Fals

In the above figure, iteration 19 indicates the beginning of a cycle (highlighted in yellow), which repeats itself infinitely. Thus, the set of highlighted lines represents a cyclic attractor of the analyzed cell.

STEP 4

From the result (file) obtained from the previous step, it is necessary to exclude all the lines that do not belong to the attractor (the lines not highlighted in yellow). One example of this procedure is the BC04_54.xlsx file present in the repository. After that, we include this file name inside the following python script (attractor_2.py) in the line:

```
wb = openpyxl.load_workbook('BC04_54.xlsx')
```

In [ ]:

```
import openpyxl

#takes the filename as arguments and returns a workbook value
wb = openpyxl.load_workbook('BC04_54.xlsx')
sheet = wb.get_active_sheet()
#define the boundaries of the sheet
col = sheet.max_column
row = sheet.max_row

b=0

#nested for loop to establish the possible regularity
#of the analyzed values
for i in range(1, col+1):
    a1=sheet.cell(row=1, column=i)

    for j in range(1, row+1 ):
      if sheet.cell(row=j, column=i).value != a1.value:
        b=1
        break
      else:
        b=0
    if b == 0:
        print(a1.value)
    else:
        print('x')
```

This script detects a constant state of gene activation, inhibition, or an alternation between all cycle states. The output of the above script indicates with "True" the state of continuous activation, "False" the state of constant inhibition, and "X" in case of alternation of the two states within the cycle.

The user can then insert the result in a spreadsheet row (SUPPLEMENTARY_TABLE_7, available in the repository), summarizing all attractors' information. This spreadsheet makes it possible to obtain all the information relating to the attractors of the different cells belonging to all patients.

TUMOR CELLS												
CELL	A2M	ACTB	AKT2	ANXA1	AR	ARAF	ARNT	ARRB1	ATF2	BAX	BCL2L1	BCl
BC01_02	False	x	False	False	False	x	False	False	x	True	False	Fal
BC01_03	False	True	False	False	False	False	False	True	x	True	False	Fal
BC01_04	x	True	False	False	False	False	False	False	x	True	False	Fal
BC01_05	False	x	False	False	False	x	False	False	x	True	False	Fal
BC01_06	False	True	False	False	False	False	False	False	x	True	False	Fal
BC01_08	False	x	False	False	False	x	x	True	x	True	False	Fal
BC01_10	x	x	False	False	False	x	False	False	x	True	False	Fal
BC01_33	False	x	False	False	False	x	False	False	x	True	False	Fal
BC01_34	x	x	False	False	False	x	x	True	x	True	False	Fal
BC01_53	False	x	False	False	False	x	x	True	x	True	False	Fal
BC01_55	False	x	False	False	False	x	x	True	x	True	False	Fal
BC01_57	False	True	False	False	False	False	False	True	x	True	False	Fal
BC01_66	False	x	False	False	False	x	False	False	x	True	False	Fal
BC01_69	x	True	False	False	False	False	False	True	x	True	False	Fal
BC01_70	False	x	False	False	False	x	x	True	x	True	False	Fal
BC01_72	False	x	False	False	False	x	False	False	x	True	False	Fal
BC01_77	False	x	False	False	False	x	False	False	x	True	False	Fal
BC01_87	False	True	False	False	False	False	False	True	x	True	False	Fal
BC01_95	x	True	False	False	True	False	False	True	x	True	False	Fal
CELL	A2M	ACTB	AKT2	ANXA1	AR	ARAF	ARNT	ARRB1	ATF2	BAX	BCL2L1	BCl
BC02_02	False	True	False	False	False	False	x	False	False	True	False	Fal
BC02_08	False	True	False	False	False	False	False	False	x	True	False	Fal
BC02_09	False	x	False	False	False	x	False	False	x	True	False	Fal
BC02_10	False	x	False	False	True	x	x	False	x	True	False	Fal
BC02_11	False	x	False	False	False	False	False	False	x	True	False	Fal
BC02_12	False	x	False	False	False	False	False	False	x	True	False	Fal
BC02_13	x	x	False	False	True	x	False	False	x	True	False	Fal