CellNOpt tutorial

A. Gabor & A. V. Ponce

1/16/2020

Contents

Introduction

The goal of this tutorial is to introduce the CellNOpt framework (Terfve et al 2012) and give some basics in network motifs.

CellNOptR

CellNOpt is a software

- creates logic-based models of signal transduction networks using logic formalisms (Boolean, Fuzzy, or differential equations)
- combines prior knowledge and perturbation experiments
- builds an executable model to provide predictions or a testable hypothesis.

These specific, contextualised models can be used, for example, to understand the different signaling patterns among cell-lines or patients or to predict drug response.

Dependencies

```
# installs devtools package if not already installed
if(!require("devtools")) install.packages('devtools')

# installs CellNOptR and CNORode from GitHub:
if(!require("CellNOptR")) devtools::install_github('saezlab/CellNOptR')
if(!require("CNORode")) devtools::install_github('saezlab/CNORode')
```

If you dont have `devtools` and cannot install it, then

1. please visit the https://github.com/saezlab/CellNOptR and https://github.com/saezlab/CNORode websites,
2. download the toolboxes by clicking "Clone or download" then "Download Zip"
3. Unzip the files
4. In RStudio run:
   `install.packages("../CellNOptR-master", repos = NULL, type = "source")`
   `install.packages("../CNORode-master", repos = NULL, type = "source")`

Make sure to import the libraries

```
library(CellNOptR)
library(CNORode)
```

PART I

The goals of part I are

- get familiar with basic inputs of CellNOpt
- check small network motifs to
  - get use to the visualisation schemes
  - understand the dynamic logic models

Input Data

CellNOpt uses a prior knowledge network stated as an interaction file to build a Boolean logic model.

TASK 1: check the format of the SIF file, in `data/tutorial_1_network.sif`: You can open it in RStudio or running:

```
writeLines(readLines("./data/tutorial_1_network.sif"))
```

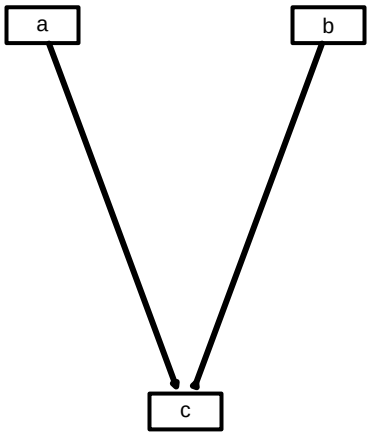
```
## a 1 c
## b 1 c
```

We find 2 lines that describes 2 interactions between nodes a, b and c. Both node a and node b can activate node c.

Large networks are complicated to check, therefore we can plot the network as a graph.

TASK 2 Visualise the SIF file in CellNOptR:

```
model <- readSIF("./data/tutorial_1_network.sif")
plotModel(model)
```



The graph shows the 2 interactions as expected.

The network is already converted to a network object:

```
print(model)
```

```
## $reacID
## [1] "a=c" "b=c"
##
## $namesSpecies
## [1] "a" "b" "c"
##
## $interMat
##   a=c b=c
## a  -1   0
## b   0  -1
## c   1   1
##
## $notMat
##   a=c b=c
## a   0   0
## b   0   0
## c   0   0
```

- *reacID* enumerates the edges of the network.
- *nameSpecies*: contains the nodes
- *interMat*: is an interaction matrix between nodes and edges
- *notMat*: shows inhibitor edges (none in this model)

TASK 3: check the format of the MIDAS (*Minimum Information for DataAnalysis in Systems Biology*) file, in `data/tutorial_1_data.csv` (best in Excel):

```
writeLines(readLines("./data/tutorial_1_data.csv"))
```

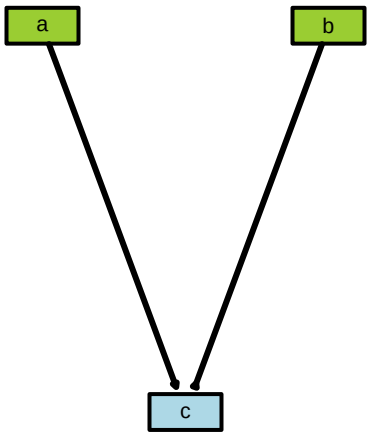
```
## TR:CL:CellLine,TR:a,TR:b,DA:c,DV:c
## 1,0,0,0,0
## 1,0,1,0,0
## 1,1,0,0,0
## 1,1,1,0,0
## 1,0,0,30,0
## 1,0,1,30,0
## 1,1,0,30,0
## 1,1,1,30,1
```

Each row of the MIDAS file encodes a measurement. Column notations:

- *TR*: treatment
- *DA*: time of data acquisition
- *DV*: measured value of the node

TASK 4: Create a CNOlist object from the MIDAS data file and annotate the network

```
cnodata <- CNOlist("./data/tutorial_1_data.csv")
plotModel(model = model, CNOlist = cnodata)
```



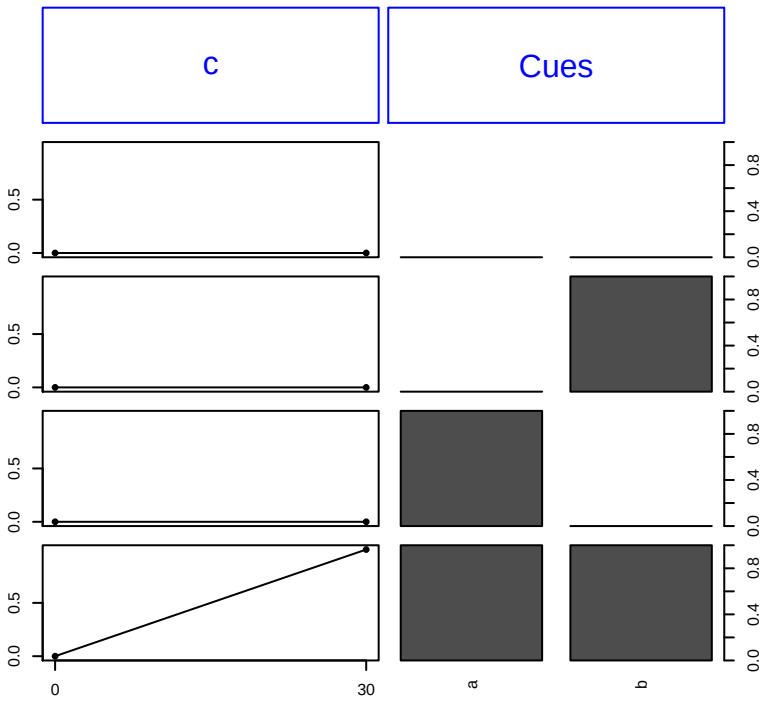
Inputs (a and b) are highlighted with green, measured nodes are with blue.

TASK 5: Print and visualise the data object

```
print(cnodata)
```

```
## class: CNOlist
## cues: a b
## inhibitors:
## stimuli: a b
## timepoints: 0 30
## signals: c
## variances: c
## --
## To see the values of any data contained in this instance, just use the
## appropriate getter method (e.g., getCues(cnolist), getSignals(cnolist), ...
```

```
plot(cnodata)
```



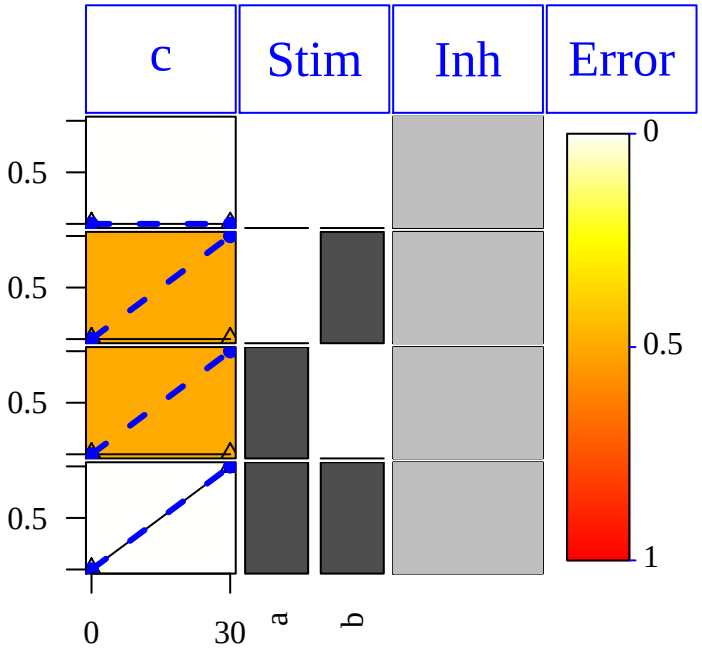
The figure shows an experiment in each line.
Perturbations/Cues (a and b) are 1 (on) or 0 (off). Node C is activated only in the last condition, where both A and B are activated.

Model building

TASK 6: Simulate the model and compare it to the experimental data:

```
edges = c("a=c" = 1,
          "b=c" = 1)

sim_res <- cutAndPlot(cnodata, model, list(edges))
```



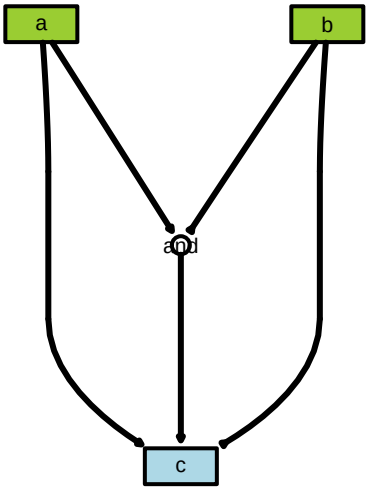
The model predicts an increase of node C if any of A or B increased, i.e. both A and B can activate C.

How do we fix it?

```
prep_model = preprocessing(data = cnodata,
                           model = model,
                           cutNONC = TRUE,  # cut non-controllable subnetwork
                           compression = TRUE, # compress if possible
                           expansion = TRUE, # expand OR gates
                           verbose = TRUE)
```

```
## [1] "The following species are measured: c"
## [1] "The following species are stimulated: a, b"
## [1] "The following species are inhibited: "
## [1] "The following species are not observable and/or not controllable: "
```

```
plotModel(prep_model,cnodata)
```



The preprocessing steps included an **AND** gate between the inputs:

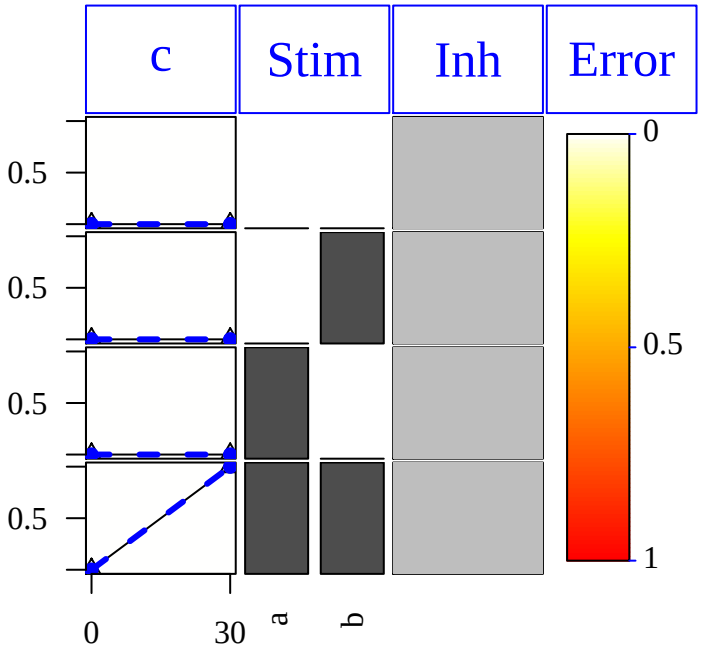
```
print(prep_model$reacID)
```

```
## [1] "a=c"   "b=c"   "a+b=c"
```

Let's fix the model to match the measured data:

```
# we turn off the a=c and b=c edges:
edges = c("a=c" = 0,
          "b=c" = 0,
          "a+b=c" = 1)

sim_res_and <- cutAndPlot(cnodata, prep_model, list(edges))
```



The model with a single AND gate between the 2 edges is now in agreement with the data.

PART II: Small, but realistic boolean model

The goals of PART II is to

- check a small but more realistic model
- optimise the model to data
- do predictions to other condition –> experimental design.

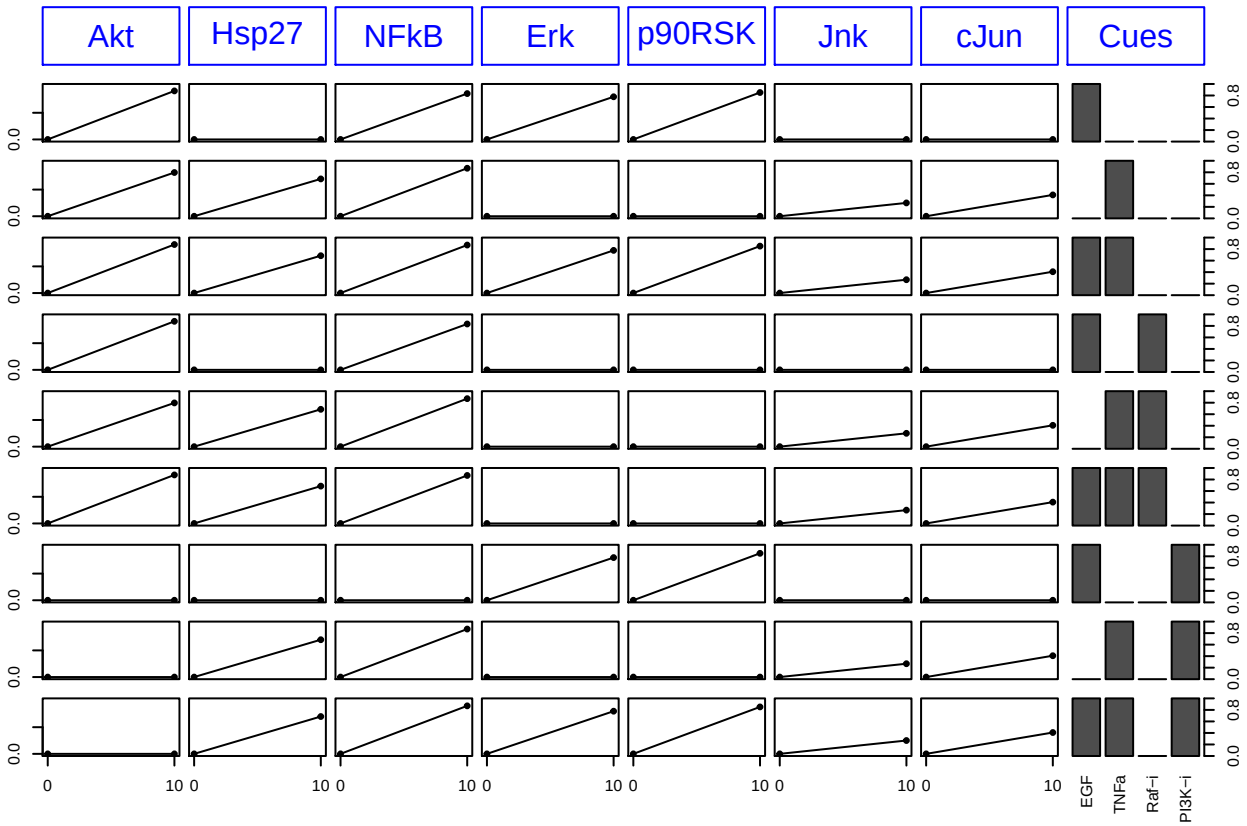
Input data

Task : import the data and network file, visualise them separately.

```
dataToy <- CNOlist("data/tutorial_2_data.csv")
networkToy <- readSIF("data/tutorial_2_network.sif")
```

The preprocessed experimental data:

```
plot(dataToy)
```



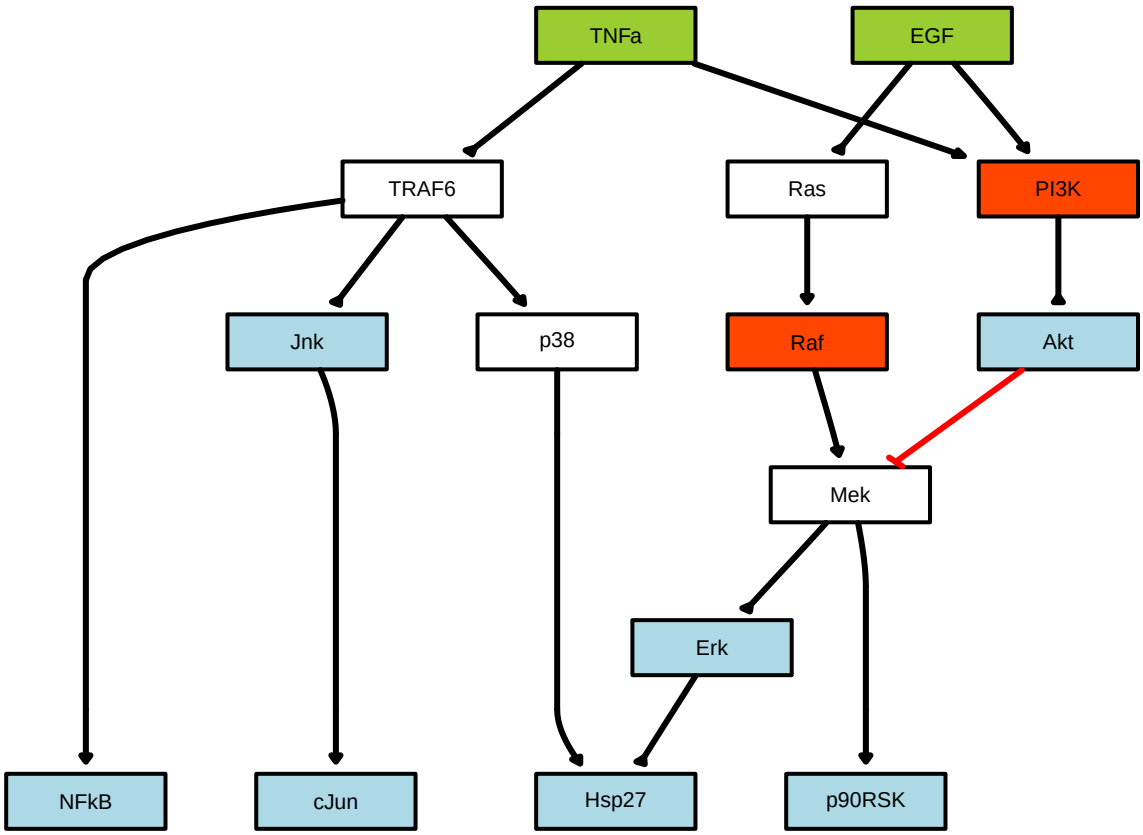
This is a complex dataset, where the cells were treated with EGF and TNFa stimuli in combination with RAF and PI3K inhibitors.

Can we answer questions, like: 1. what ligand activates AKT? 2. Is Hsp27 responding to EGF stimulation?

Now visualise the prior knowledge network: (it is very handy that we have one and we dont have to read dozens of papers. . .)

This prior knowledge network contains all the edges among these nodes that we think is relevant to the problem.

```
plotModel(networkToy,CNOlist = dataToy)
```



There are red and white nodes:

- red box indicates nodes that are inhibited in certain conditions,
- white boxes are nodes that are not measured

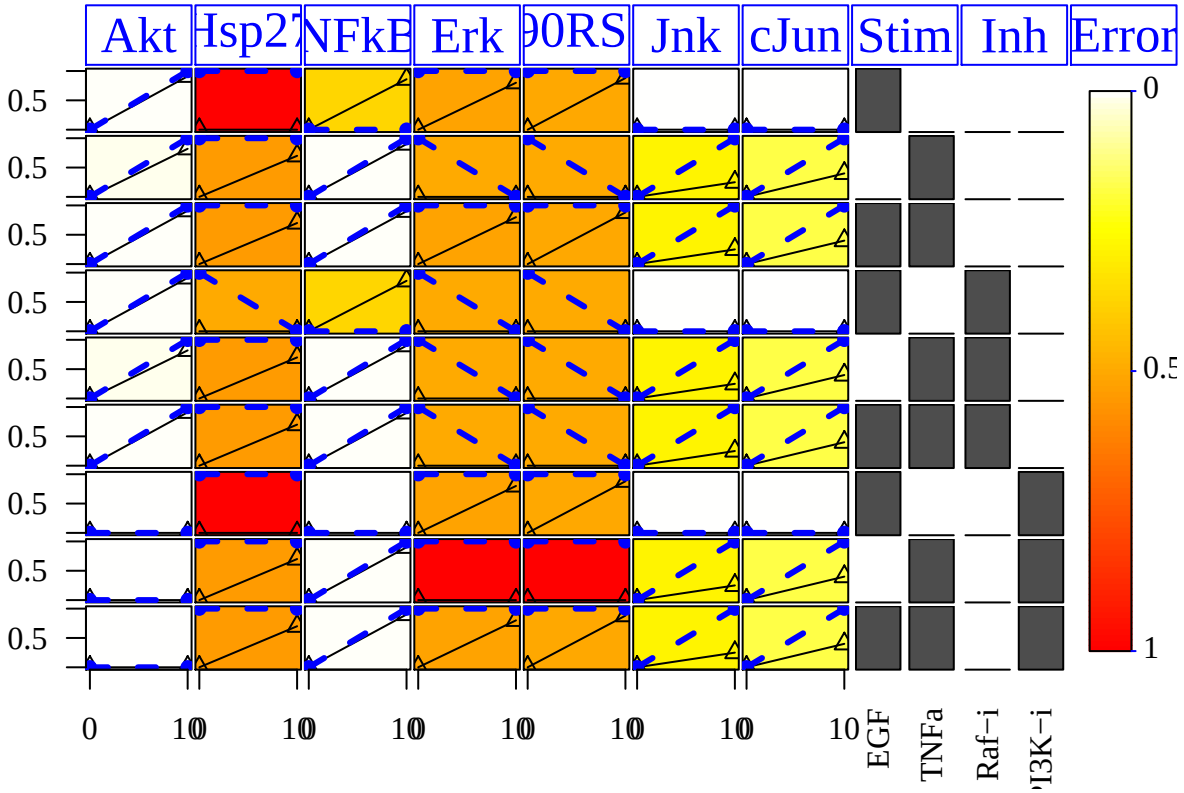
Red T-shape arrows indicate inhibitory interactions.

Is this prior knowledge network in agreement with the above data?
According to the prior knowledge, should Hsp27 respond to EGF?

Simulate the model

Let's simulate the prior knowledge and compare it with the data:

```
sim_PKN <- cutAndPlot(CNOlist = dataToy,
           model = networkToy,
           bStrings = list(rep(1,length(networkToy$reacID))))
```



Apparently the prior knowledge is not in agreement with the data. This happens, because prior knowledge is not specific to the measured cell-line. It assessed by us and we included many interactions that we belived is needed.

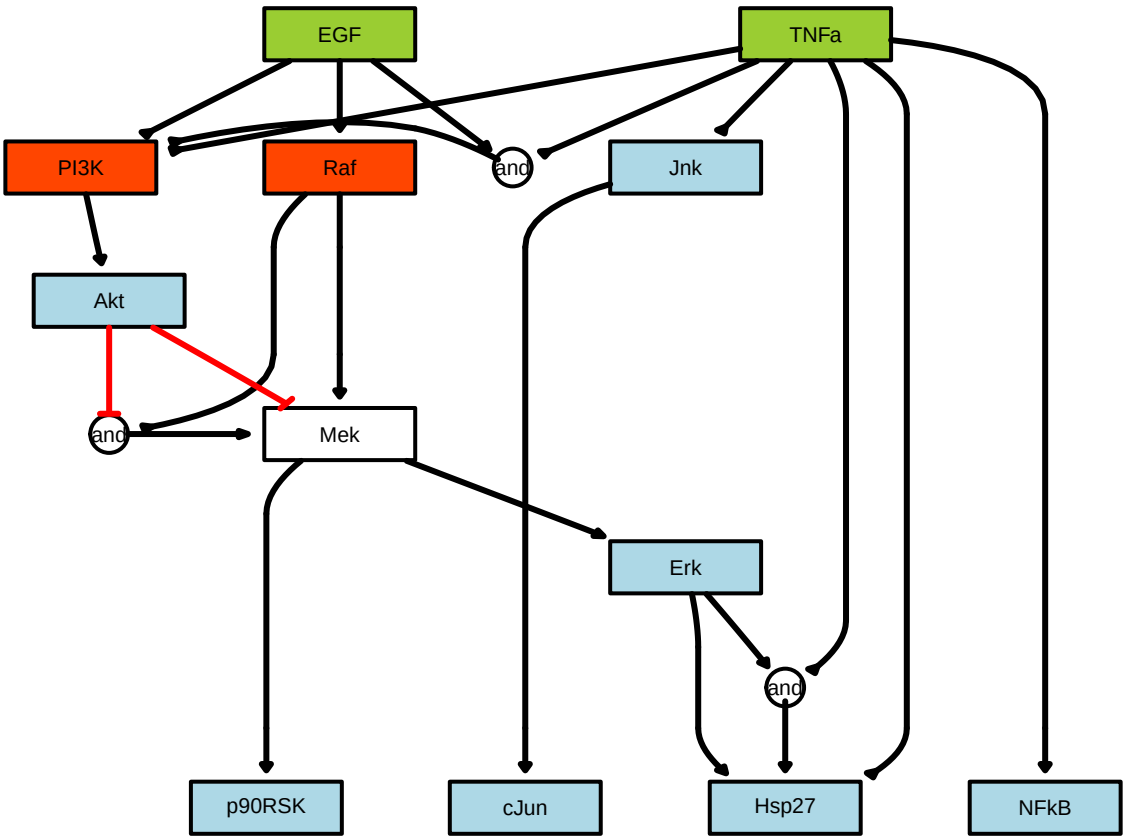
Network model optimisation

To find a *subset* of edges that makes the model to fit the data we use optimisation.

```
prep_network_Toy <- preprocessing(dataToy, networkToy,
                                  expansion=TRUE,
                                  compression=TRUE,
                                  cutNONC=TRUE,
                                  verbose=TRUE)
```

```
## [1] "The following species are measured: Akt, Hsp27, NFkB, Erk, p90RSK, Jnk, cJun"
## [1] "The following species are stimulated: EGF, TNFa"
## [1] "The following species are inhibited: Raf, PI3K"
## [1] "The following species are not observable and/or not controllable: "
```

```
plotModel(prep_network_Toy ,dataToy)
```



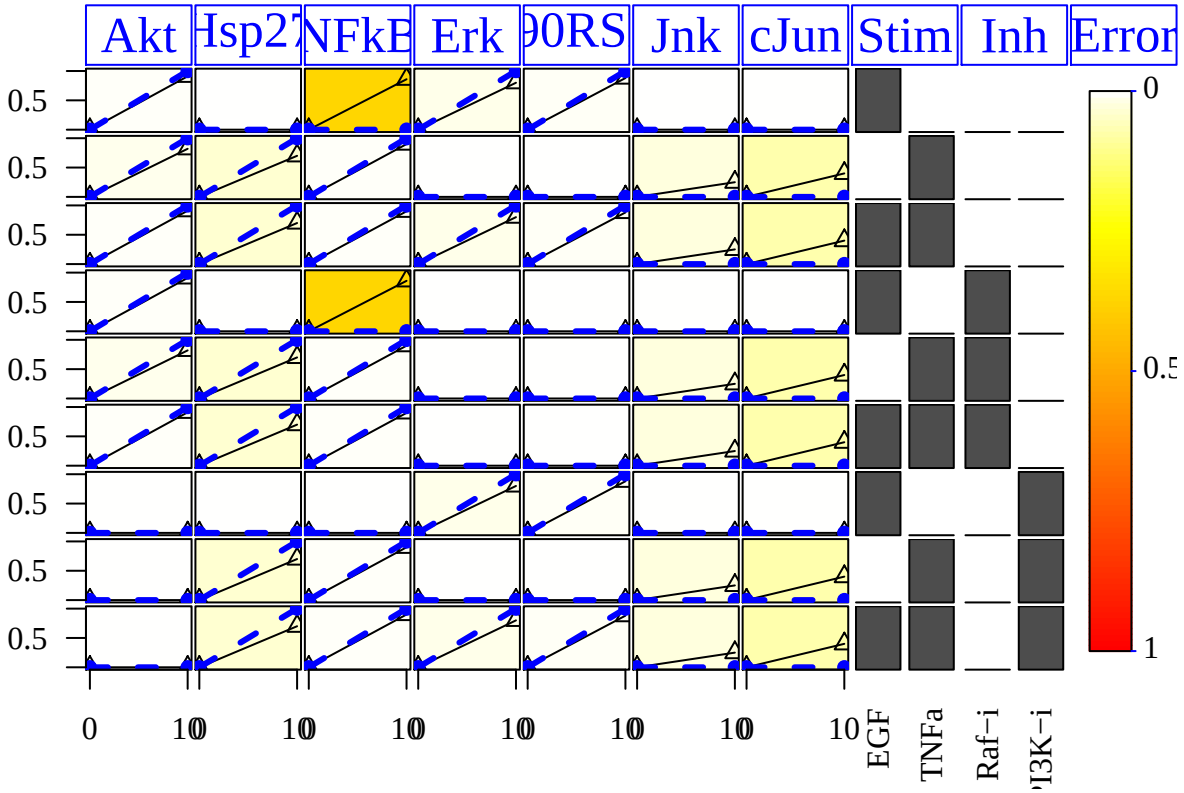
Note the AND gates and missing white boxes. They disappeared beacause of compression.

We use genetic algorithm to find an optimal set of edges.

```
ToyT1opt <- gaBinaryT1(CNOlist=dataToy,
                       model=prep_network_Toy,
                       verbose=FALSE)
```

Plot the optimised network's prediction:

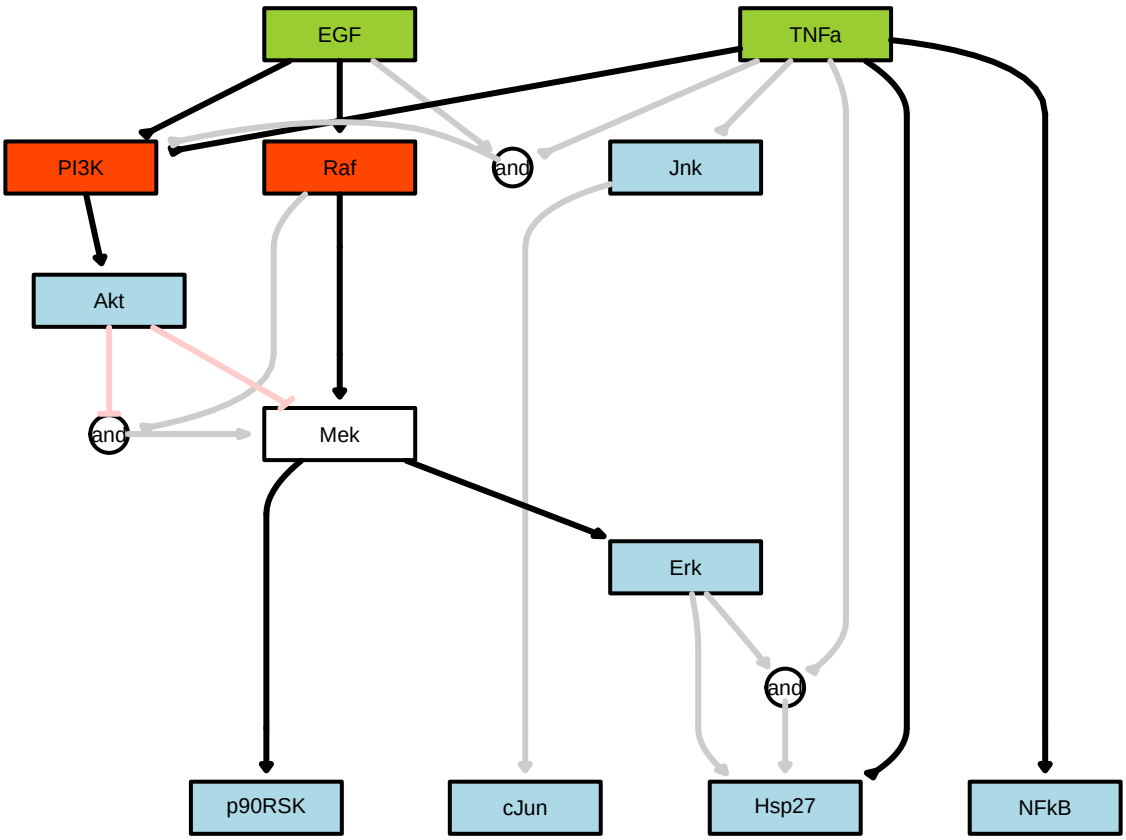
```
sim_opt_PKN <- cutAndPlot(CNOlist = dataToy,
           model = prep_network_Toy,
           bStrings = list(ToyT1opt$bString))
```



show the optimised network:

```
plotModel(prep_network_Toy,dataToy,bString = ToyT1opt$bString, removeEmptyAnds = F)
```

```
## [1] "showing red link even if empty"
## [1] "showing red link even if empty"
```



Here the edge color encodes if the edge is still included in the optimised subnetwork. Black edge means that the edge is still active in the model, and grey links are removed by the optimiser.

References

C Terfve, T Cokelaer, A MacNamara, D Henriques, E Goncalves, MK Morris, M van Iersel, DA Lauffenburger, J Saez-Rodriguez. CellNOptR: a flexible toolkit to train protein signaling networks to data using multiple logic formalisms. *BMC Systems Biology*, 2012, 6:133