Training of boolean logic models of signalling networks to time course data with *CNORdt*

Aidan MacNamara

March 16, 2012

Contents

1 Background

This software is written in the R language, so in order to use it you will need to have R installed on your computer. For more information, please refer to `http://www.r-project.org/`. For more information about how to install R packages, please refer to `http://cran.r-project.org/doc/manuals/R-admin.html#Installing-packages`.

CNORdt is an add-on to *CellNOptR* [1], a software package that trains logic models to data [2]. More details and the package itself can be found using the following command to install:

```
> install.packages("path_to_CellNOptR/CellNOptR_1.0.0.tar.gz", repos=NULL)
```

CNORdt can be installed by:

```
> install.packages("path_to_CNORdt/CNORdt_1.0.0.tar.gz", repos=NULL)
```

Alternatively, if you have the source code, you can install the packages in a R session by typing:

```
> install.packages("CellNOptR_1.0.0.tar.gz", repos=NULL)
> install.packages("CNORdt_1.0.0.tar.gz", repos=NULL)
```

2 CNORdt

CNORdt introduces a scaling parameter that defines the time scale of the Boolean synchronous simulation. Where each 'tick' (t) (or simulation step) is the synchronous updating of all nodes in the model according to their inputs at $t-1$, the scaling parameter defines the tick frequency relative to the time scale of the real

data. Although this is a crude approach (i.e. it implies a single rate across all reactions), it allows us to fit a synchronous Boolean simulation to data. Hence, all data points can be fitted to the model and hyperedges that cause feedback in the model can be included, which allows the model to reveal more complex dynamics such as oscillations.

3 Load Data

The first step of the analysis with *CNORdt* is to load the necessary libraries and the data:

```
> library(CellNOptR)
> library(CNORdt)
```

The model and data are then loaded. These are taken from MacNamara et al. [3] and consist of a biologically realistic toy model based on the EGFR signaling pathway and *in silico*-generated data:

```
> data(CNOlistPB, package="CNORdt")
> data(ToyModelPB, package="CNORdt")
```

4 Preprocessing

The full details of preprocessing the model can be found in the *CellNOptR* package (the vignette gives a comprehensive explanation). The following steps are taken:

```
> # processing the model (indexing, compression and expansion)
> # index the stimuli, readouts and inhibitors
> indexOrig <- indexFinder(CNOlist=CNOlist, Model=Model, verbose=T)

[1] "The following species are measured: raf1, erk, ap1, gsk3, p38, nfkb"
[1] "The following species are stimulated: egf, tnfa"
[1] "The following species are inhibited: pi3k, raf1"

> # find the indexes of the non-observables and the non-controllable species
> indexNONC <- findNONC(Model=Model, indexes=indexOrig, verbose=T)

[1] "The following species are not observable and/or not controllable: p90rsk, creb"

> # cut the data according to 'indexNONC'
> ModelCut <- cutNONC(Model=Model, NONCindexes=indexNONC)
> # find the indexes again as model may have changed
> indexNONCcut <- indexFinder(CNOlist=CNOlist, Model=ModelCut)
> # compress the model
> ModelCutCompress <- compressModel(Model=ModelCut, indexes=indexNONCcut)
> # find the indexes again
> indexNONCcutComp <- indexFinder(CNOlist=CNOlist, Model=ModelCutCompress)
> # expand
> ModelCutCompressExpand <- expandGates(Model=ModelCutCompress)
> # extract information for simulation
> fields4Sim <- prep4Sim(Model=ModelCutCompressExpand)
> initBstring <- rep(1, length(ModelCutCompressExpand$reacID))
```

5 Optimization

This is where the difference between CNORdt and its parent package CellNOptR becomes visible. CellNOptR fits the model to data at steady state i.e. the simulation runs until all species are static. This gives a robust overview of the model behaviour but this formalism cannot model more complex dynamics such as oscillations. CNORdt uses full time course data by scaling the boolean simulation. Hence the difference betweeen model and data can be calculated and optimized across all data points at not just a single one or two.

An additional feature from CellNOptR can also be implemented in CNORdt: the model fitting can be carried out in 2 stages if the user has sufficient knowledge that the data represents early and late reactions (for example, an early phosphoprotein activation followed by later deactivation). This feature is controlled by supplying the division time ('divTime') - the experimental time point where the late reactions start to occur. If no such value is supplied, divTime is set to NULL and the model fitting is not divided into early and late. In the case of this model below, a division time is set at 10 (minutes) as there is a delayed deactivation of phosphoproteins in the data (see MacNamara et al. [3] for more details).

The other data that currently needs to be supplied in addition to the optimization parameteres descrived are 'boolUpdates' and the upper and lower bounds for the optimization of the scaling factor ('upperB' and 'lowerB'). The variable 'boolUpdates' controls the number of simulation steps ('ticks') and is either 1 or 2 values depending whether 'divTime' has been set. This will be set to an intelligent default value in future releases (it can be viewed as a function of model size and experimental time) but it is currently a manual entry. The upper and lower bounds control the range of values for the optimization of the scaling factor.

```
> # the 'switch' time between the 2 time phases needs to be known
> opt1 <- gaBinaryTimeScale(CNOlist=CNOlist, Model=ModelCutCompressExpand,
+ SimList=fields4Sim, indexList=indexNONCcutComp, initBstring=initBstring,
+ verbose=TRUE, boolUpdates=c(10,20), divTime=10, MaxTime=100, lowerB=0.8, upperB=10)
```

Here you can visualize the 'early' fit of the data:

```
> cutAndPlotResultsTimeScale(Model=ModelCutCompressExpand, bString=opt1$bString, SimList=fields4Sim,
+ CNOlist=CNOlist, indexList=indexNONCcutComp, boolUpdates=c(10,20), divTime=10)
```

Simulate the above model to get the starting point for second steady state:

```
> dataStartPoint = cutModel(Model=ModelCutCompressExpand,
+ SimList=fields4Sim, bitString=opt1$bString)
> simT1 = simulatorTimeScale(CNOlist=CNOlist,
+ Model=dataStartPoint[[1]], SimList=dataStartPoint[[2]],
+ indexList=indexNONCcutComp, boolUpdates=10)
```

Optimize the 'late' network, then stitch the results together:

```
> opt2 <- gaBinaryTimeScaleT2(CNOlist=CNOlist,
+ Model=ModelCutCompressExpand, SimList=fields4Sim,
+ indexList=indexNONCcutComp, bStringT1=opt1$bString,
+ SimResT1=simT1, verbose=TRUE, boolUpdates=c(10,20),
+ divTime=10, lowerB=0.8, upperB=10)
```

Visualize total result:

```
> cutAndPlotResultsTimeScaleT2(Model=ModelCutCompressExpand,
+ bStringT1=opt1$bString, bStringT2=opt2$bString, SimList=fields4Sim,
+ CNOlist=CNOlist, indexList=indexNONCcutComp, boolUpdates=c(10,20),
+ divTime=10, lowerB=0.8, upperB=10)
```

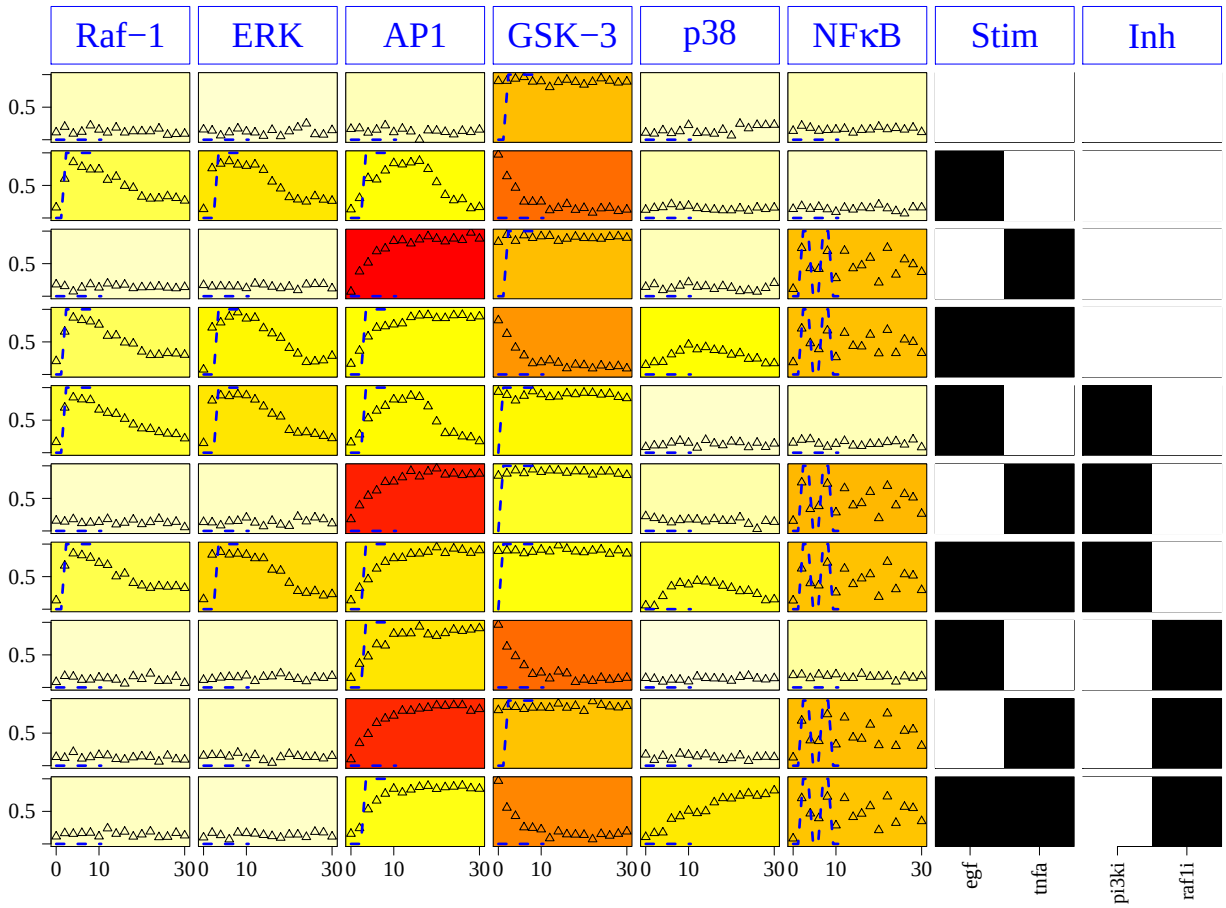

Figure 1:

6 Post-Optimization

References

[1] C. Terfve. CellNOptR: R version of CellNOpt, boolean features only. em R package version 0.99.14, (2012) http://www.bioconductor.org/packages/release/bioc/html/CellNOptR.html

[2] J. Saez-Rodriguez, L. Alexopoulos, J. Epperlein, R. Samaga, D. Lauffenburger, S. Klamt and P.K. Sorger. Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Molecular Systems Biology*, 5:331, 2009.

[3] A. MacNamara, C. Terfve, D. Henriques, B. Peñalver Bernabé and J. Saez-Rodriguez State-time spectrum of signal transduction logic models. *Physical Biology*, submitted, 2012.

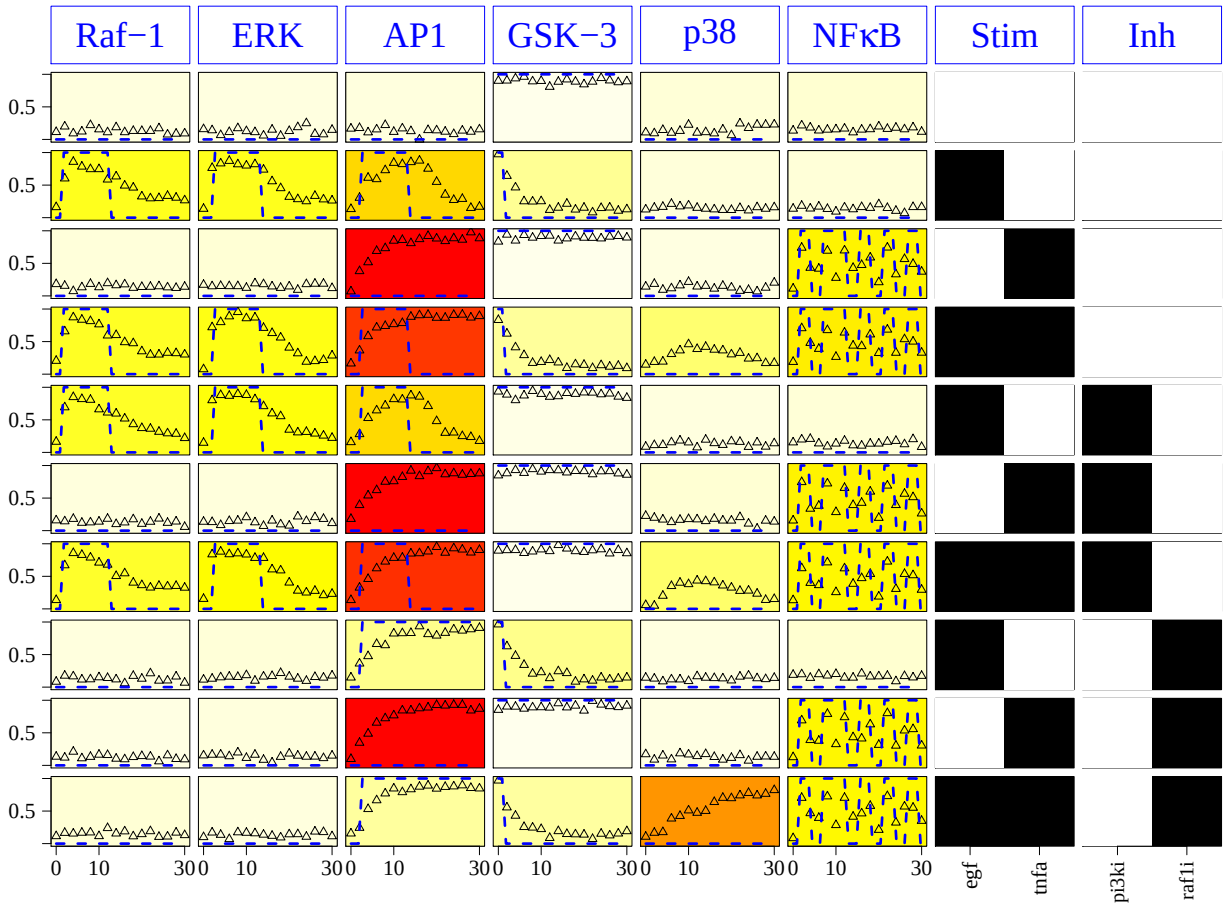
Raf−1
ERK
AP1
GSK−3
p38
NFκB
Stim
Inh
0.5
0
10
30
egf
tnfa
pi3ki
raf1i

Figure 2: