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

Aidan MacNamara

October 17, 2016

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* [**?**], a software package that trains logic models to data [**?**]. More details can be found using the following command to install:

```
> source("http//bioconductor.org/biocLite.R")
> biocLite("CNORdt")
```

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. [**?**] and consist of a biologically realistic toy model based on the EGFR signaling pathway and *in silico*-generated data:

```
> data(CNOlistPB, package="CNORdt")
> data(modelPB, 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:

```
> # pre-process model
> model = preprocessing(CNOlistPB, modelPB)
```

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

```
> initBstring <- rep(1, length(model$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.

The other data that currently needs to be supplied in addition to the optimization parameters described 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'). This will be set to an intelligent default value in future releases (it can be considered 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.

```
> opt1 <- gaBinaryDT(CNOlist=CNOlistPB, model=model, initBstring=initBstring,
+ verbose=FALSE, boolUpdates=10, maxTime=30, lowerB=0.8, upperB=10)
```

Visualize the result (figure **??**):

```
> cutAndPlotResultsDT(
+         model=model,
+         CNOlist=CNOlistPB,
+         bString=opt1$bString,
+         plotPDF=FALSE,
+         boolUpdates=10,
+         lowerB=0.8,
+         upperB=10
+ )
```

```
list()
```

6 Post-Optimization

As mentioned above, the difference between *CNORdt* and *CellNOptR* is in the simulation of the data. Hence, it is possible to use the range of post-optimization functions available in *CellNOptR* to view, for example, the evolution of fit during optimization (*plotFit*), or map the optimized network onto the input network or PKN (*writeNetwork*). Please see the *CellNOptR* package for more details about the following:

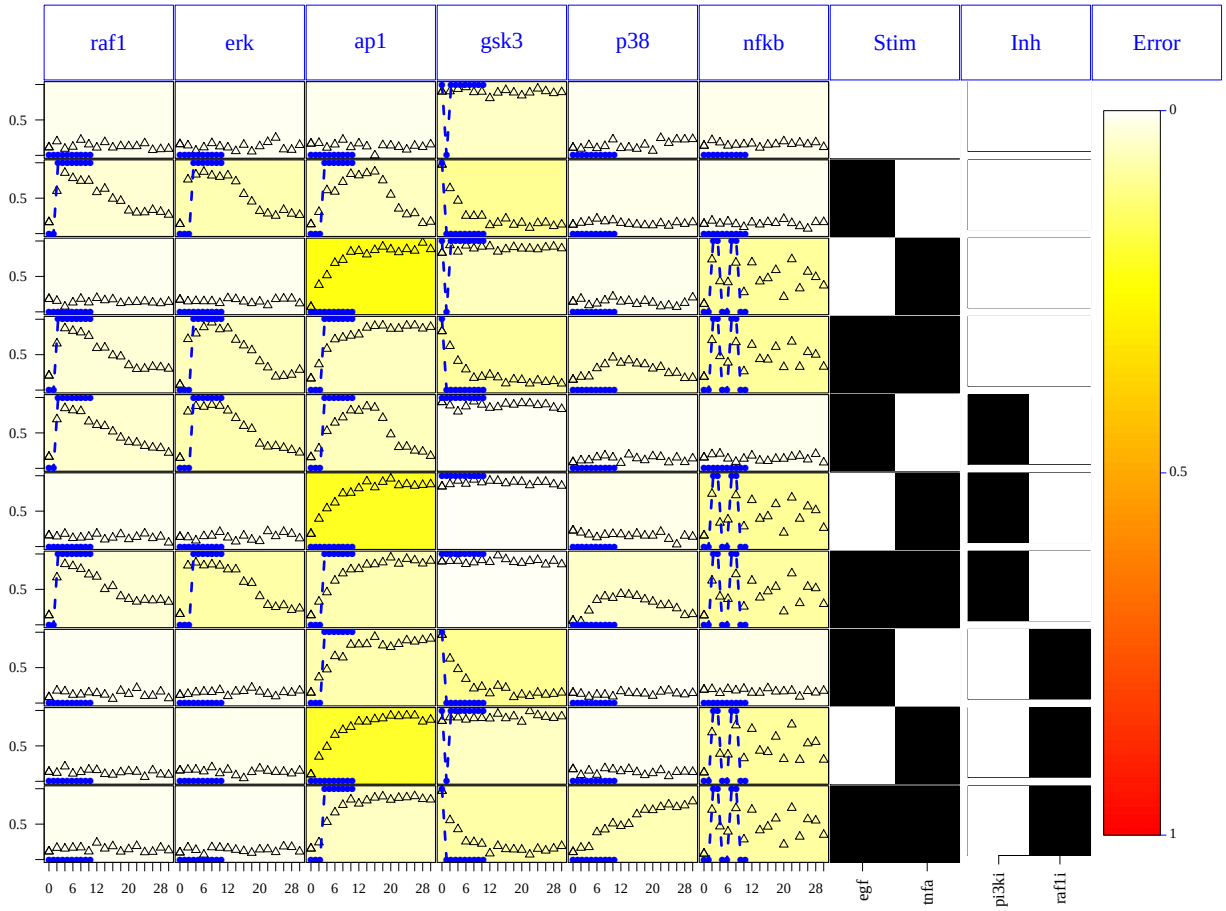

Figure 1: The fit of the model at multiple time points. The training took 30 seconds.

```
> writeScaffold(
+         modelComprExpanded=model,
+         optimResT1=opt1,
+         optimResT2=NA,
+         modelOriginal=modelPB,
+         CNOlist=CNOlistPB
+ )
> writeNetwork(
+         modelOriginal=modelPB,
+         modelComprExpanded=model,
+         optimResT1=opt1,
+         CNOlist=CNOlistPB
+ )
```

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 modeling as a means to link protein signaling 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*, 2012, 9(4), p.045003.