Drug prediction tutorial

A. Gabor & A. V. Ponce

1/16/2020

Contents

Introduction

The goal of this tutorial is to show an example where dynamic logic model is used to predict drug response.

The tutorial is based on the paper

> Eduati et al (2017) Drug resistance mechanisms in colorectal cancer dissected with cell type-specific dynamic logic models.

We investigate here the drug-response of colorectal cancer cell lines. For this, we use drug response data and a signaling dataset.

The Genomics of Drug Sensitivity in Cancer (GDSC), https://www.cancerrxgene.org/ offers drug response data for more than a 1000 human cancer cell lines, for houndreds of drugs. A small part of these data is can be found in `"./data/IC50_GDSC.csv"`.

The perturbation dataset contains the short time signaling response of 14 colorectal cancer cell lines, where 14 phosphoproteins are measured under 43 perturbation conditions (5 stimuli, 7 inhibitors).

First, we construct signaling models based on the perturbation data to the cell lines, here we use the CNORode modelling package. In the next step, we will associate model feratures to drug response to see why certain cell lines respond to certain drugs and others do not. Here we use a linear modeling framework.

CNORode

CNORode is a member of the CellNOptR logic based tool family. It can translate the network to ordinary differential equation (ODE) model, fit the model parameters to data and make predictions.

Dependencies

These should be already installed from previous tutorial.

```
# 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(!require("dplyr")) install.packages('dplyr')
if(!require("readr")) install.packages('readr')
if(!require("tidyr")) install.packages('tidyr')
```

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)
library(MEIGOR)
```

```
## Loading required package: Rsolnp
```

```
## Loading required package: snowfall
```

```
## Loading required package: snow
```

```
##
## Attaching package: 'snow'
```

```
## The following objects are masked from 'package:BiocGenerics':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, clusterSplit, parApply, parCapply,
##     parLapply, parRapply, parSapply
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, clusterSplit, makeCluster,
##     parApply, parCapply, parLapply, parRapply, parSapply,
##     splitIndices, stopCluster
```

```
## Loading required package: deSolve
```

```
library(dplyr)
library(tidyr)
library(ggplot2)
```

PART I: DRUG response exploration

```
IC50 <- readr::read_csv("./data/tutorial_3/IC50_GDSC.csv") %>% rename("cell_line" = "X1")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   X1 = col_character()
## )
```

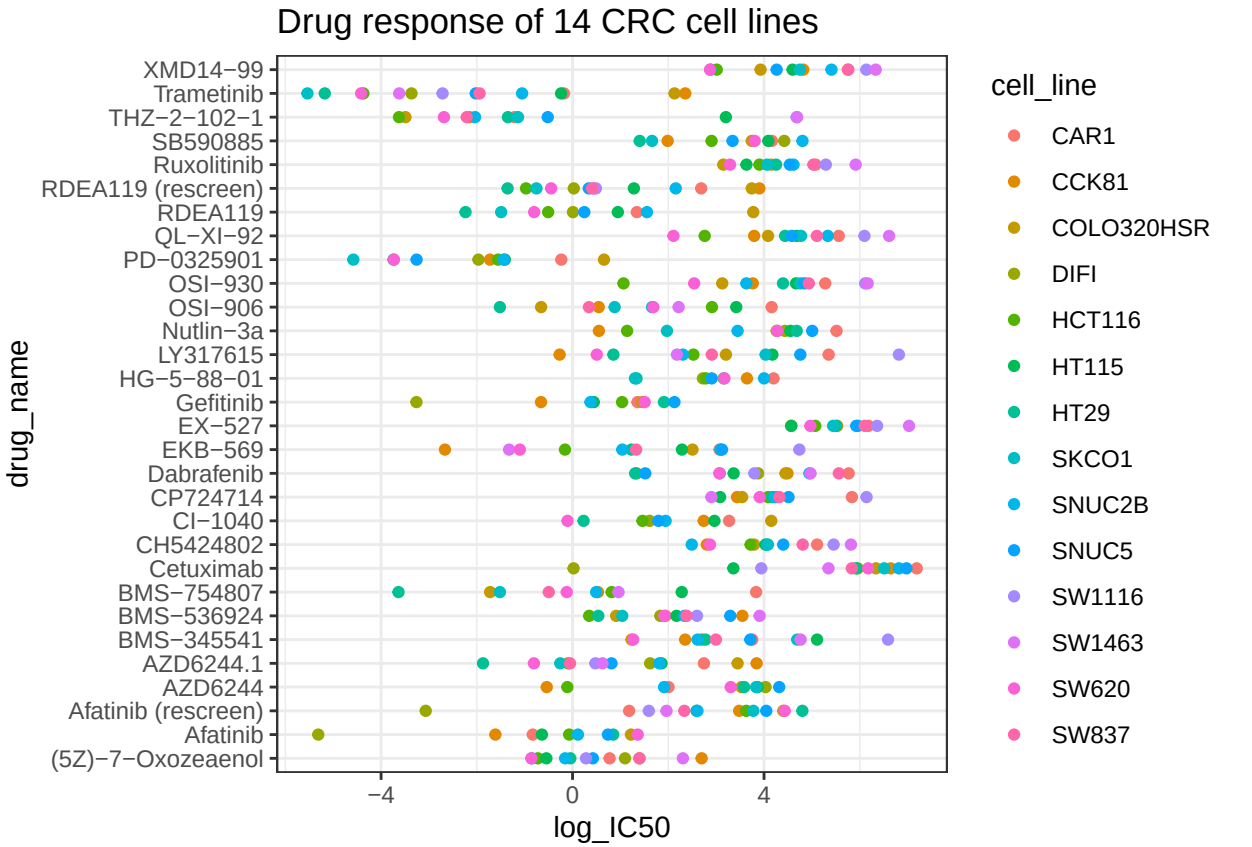
```
## See spec(...) for full column specifications.
```

```
print(IC50)
```

```
## # A tibble: 14 x 31
##    cell_line Gefitinib  RDEA119 `CI-1040` Afatinib `Nutlin-3a` `PD-0325901`
##    <chr>         <dbl>    <dbl>     <dbl>    <dbl>       <dbl>        <dbl>
##  1 CAR1           1.36   1.34        3.27   -0.831        5.51        -0.235
##  2 CCK81         -0.659 NA           2.74   -1.61         0.551       -1.72
##  3 COLO320H~      1.46   3.78        4.15    1.22         4.26         0.658
##  4 DIFI          -3.26   0.00908     1.61   -5.31         4.44        -1.97
##  5 HCT116         1.03  -0.510       1.46   -0.0704       1.14        -1.56
##  6 HT115          0.444  0.946       2.96   -0.640        4.55        -1.41
##  7 HT29           1.91  -2.24        0.230   0.849        4.68        -3.74
##  8 SKCO1         NA     -1.49       NA      NA            1.97        -4.58
##  9 SNUC2B         0.375  1.55        1.94    0.113        3.45        -1.44
## 10 SNUC5          2.13   0.246       1.79    0.739        5.01        -3.26
## 11 SW1116        NA     NA          NA      NA           NA           NA
## 12 SW1463        NA     NA          NA      NA           NA           NA
## 13 SW620          1.51  -0.802      -0.104   1.36         4.28        -3.73
## 14 SW837         NA     NA          NA      NA           NA           NA
## # ... with 24 more variables: SB590885 <dbl>, AZD6244 <dbl>,
## #   `BMS-536924` <dbl>, Cetuximab <dbl>, `HG-5-88-01` <dbl>,
## #   `(5Z)-7-Oxozeaenol` <dbl>, Trametinib <dbl>, Dabrafenib <dbl>,
## #   `Afatinib (rescreen)` <dbl>, AZD6244.1 <dbl>, `RDEA119
## #   (rescreen)` <dbl>, `BMS-754807` <dbl>, `OSI-906` <dbl>,
## #   `BMS-345541` <dbl>, Ruxolitinib <dbl>, LY317615 <dbl>,
## #   `XMD14-99` <dbl>, CP724714 <dbl>, CH5424802 <dbl>, `EKB-569` <dbl>,
## #   `OSI-930` <dbl>, `QL-XI-92` <dbl>, `EX-527` <dbl>, `THZ-2-102-1` <dbl>
```

```
IC50 %>% gather(drug_name, log_IC50, -cell_line) %>%
  ggplot() +
  geom_point(aes(drug_name, log_IC50,col=cell_line)) +
  coord_flip() +
  theme_bw() +
  ggtitle("Drug response of 14 CRC cell lines")
```

```
## Warning: Removed 49 rows containing missing values (geom_point).
```



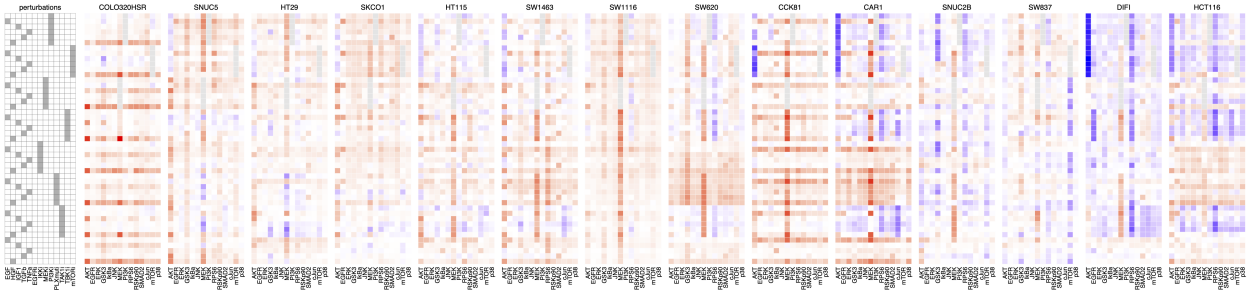
Form the raw IC50 values we can see that there are some drugs that are more effective (Trametinib) than others, like XMD14-99. There are also cell-line differences, for example, DIFI shows stronger sensitivity to Afatinib than any other cell lines. What could be the reason for this?

PART II: cell-line models

The goal of part II is to build a cell-line specific model from the perturbation data

Perturbation data

The following heatmap shows an overview on the perturbation data. The first block outlines the combinations of treatment. Then each other block represents the response of a cell line. Different columns within a block shows the different phosphoprotein markers.



In the turial we make a single model for he first cell line *COLO320HSR*.

Model of a single cell line

```
# load Prior Knowledge Network (PKN)
pknmodel<-readSIF("./data/tutorial_3/PKN.sif")

# load normalised perturbation data
# select MIDAS file for the desired cell line
MIDASfile <- "./data/tutorial_3/processed/MIDAS/MD-COLO320HSR_Ktuned_v1_n4_all_noEGFRi_CNORode.csv"

Mydata<-readMIDAS(MIDASfile=MIDASfile)
```

```
## [1] "Your data set comprises  86 conditions (i.e. combinations of time point and treatment)"
## [1] "Your data comprises only a DA:ALL column;  all readouts are assumed to have been acquired at the
## [1] "Your data set comprises measurements on  25  different species"
## [1] "Your data set comprises  13 stimuli/inhibitors and 1 cell line(s) ( COLO320HSR )"
## [1] "Please be aware that CNO only handles measurements on one cell line at this time."
## [1] "Your data file contained 'NaN'. We have assumed that these were missing values and replaced ther
```

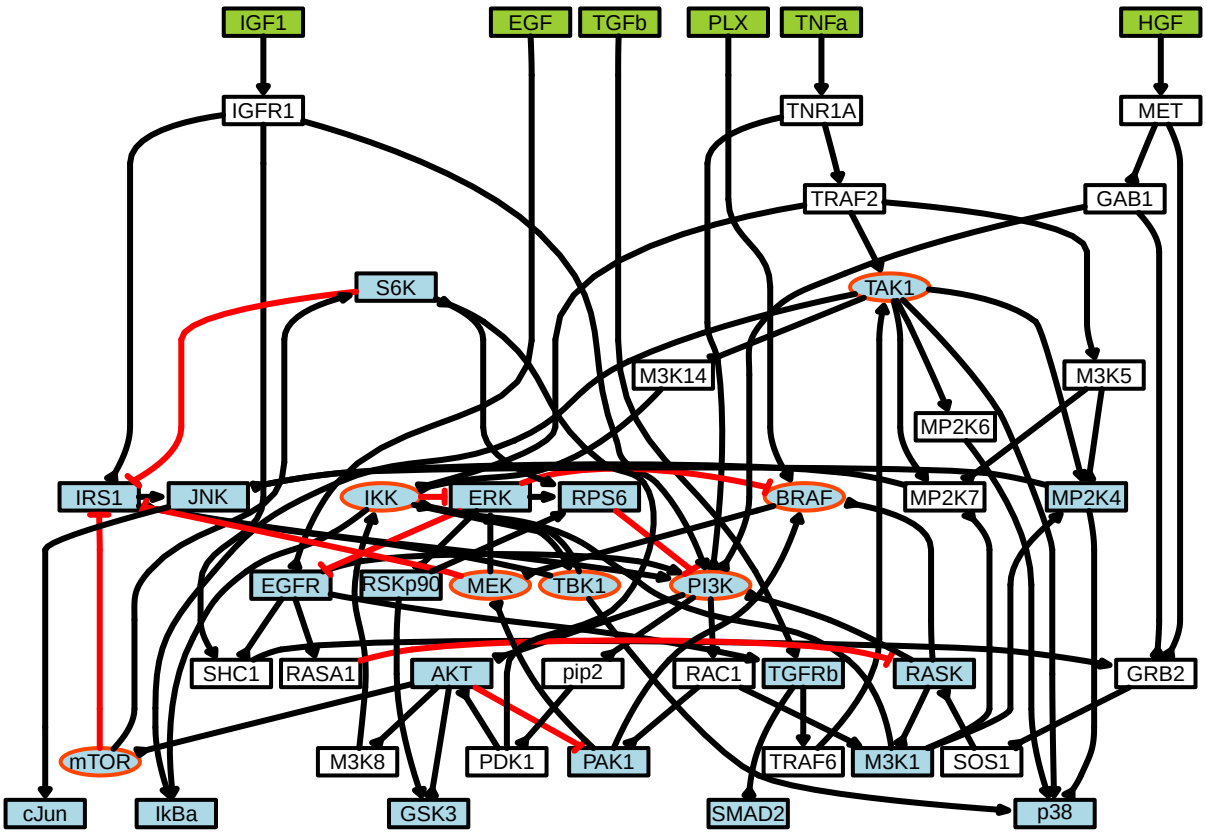
```
cnolist<-makeCNOlist(Mydata, subfield=F)
```

```
## [1] "Please be aware that if you only have some conditions at time zero (e.g.only inhibitor/no inhibi
```

```
cnolist$valueStimuli[cnolist$valueStimuli==0]=0.5
```

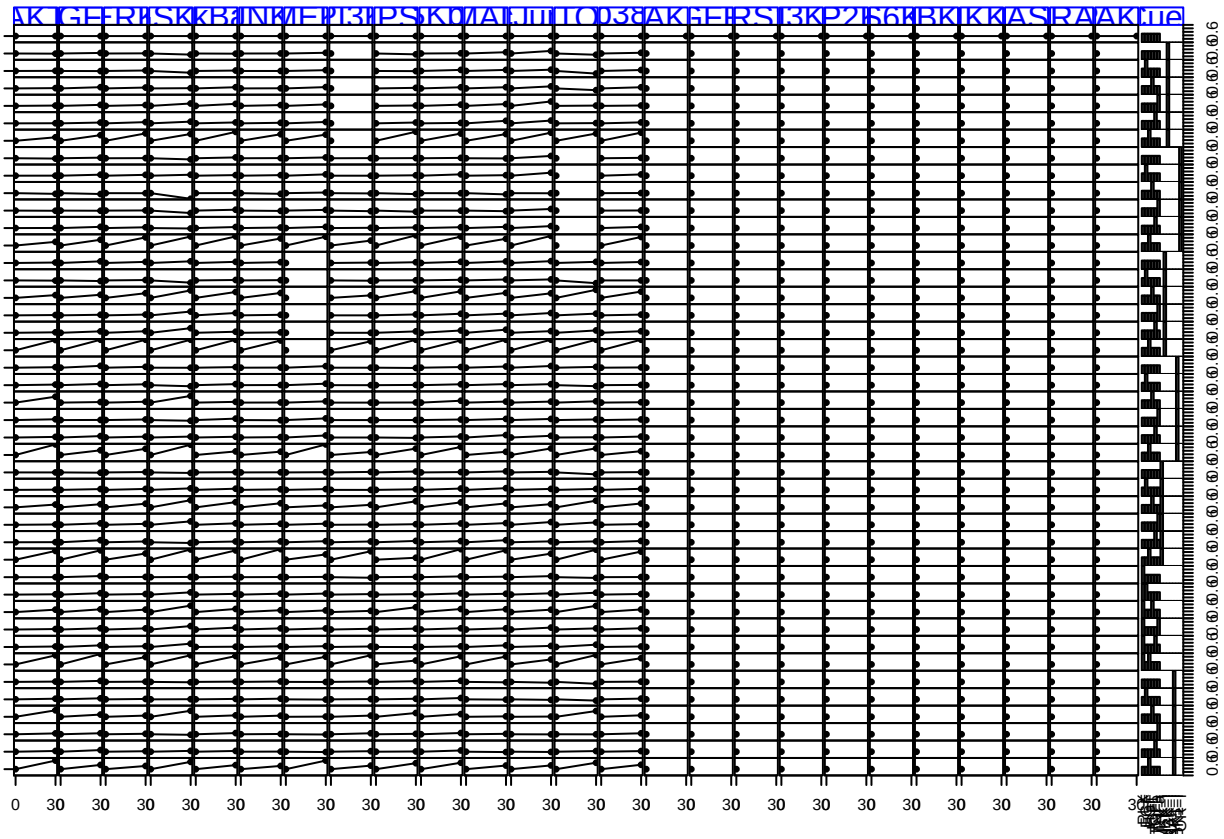
Show the network first

```
plotModel(pknmodel,cnolist)
```



Then the data in CellNOpt format:

```
plotCNOlist(cnolist)
```



```
# compress the network (no expansion, only OR gates are considered)
model<-preprocessing(data=cnolist, model=pknmodel, compression=TRUE, expansion=FALSE)
```

```
## [1] "The following species are measured: AKT, EGFR, ERK, GSK3, IkBa, JNK, MEK, PI3K, RPS6, RSKp90, S
## [1] "The following species are stimulated: PLX, EGF, HGF, IGF1, TGFb, TNFa"
## [1] "The following species are inhibited: IKK, MEK, PI3K, BRAF, TAK1, TBK1, mTOR"
## [1] "The following species are not observable and/or not controllable: "
```

```
# set initial parameters (here parameters 'k' and 'tau' are optimised and 'n' fixed to 3)
ode_parameters <- createLBodeContPars(model,
                                      LB_n = 1, LB_k = 0, LB_tau = 0,
                                      UB_n = 3, UB_k = 1, UB_tau = 1,
                                      default_n = 3,
                                      default_k = 0.5,
                                      default_tau = 0.01,
                                      opt_n = FALSE, opt_k = TRUE, opt_tau = TRUE,
                                      random = TRUE)

# PLX -> BRAF is an artificial regulation used to model paradoxical effect of PLX4720,
# which works as selective BRAF inhibitor in cell-lines where BRAF is mutated in V600E (i.e. HT29 and S
# but induces a paradoxical activation of wild type BRAF cells (modeled as stimulus on those cell lines
ode_parameters$parValues[which(ode_parameters$parNames=="PLX_k_BRAF")]<-0.5
ode_parameters$index_opt_pars<-setdiff(ode_parameters$index_opt_pars, which(ode_parameters$parNames=="P

## Parameter Optimization
# essm
```

```
paramsSSm=defaultParametersSSm()
paramsSSm$local_solver = "DHC"
paramsSSm$maxtime = 30; #36000;
paramsSSm$transfer_function = 4;
```

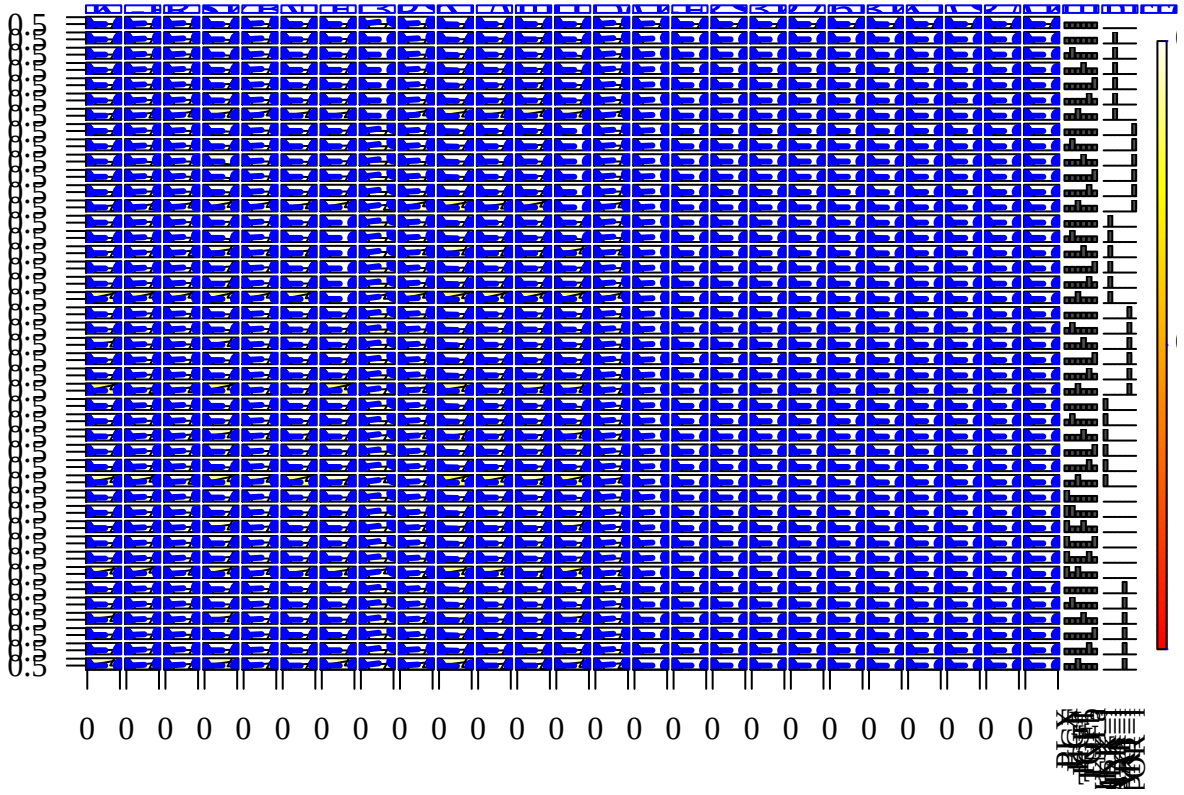
The actual optimisation takes around 10 mins, instead of the 30 sec. So, instead of running it here, we just load the results:

```
opt_pars = parEstimationLBode(cnolist, model, method="essm",
                              ode_parameters=ode_parameters,
                              paramsSSm=paramsSSm)
#write_rds(opt_pars,"data/tutorial_3/opt_pars_30sec.RDS")
```

```
opt_pars <- read_rds("data/tutorial_3/opt_pars_30sec.RDS")
```

Plot the fit of the model:

```
sim_res <- CNORode::plotLBodeFitness(cnolist,model,ode_parameters = opt_pars)
```



The fit is a bit better than a random model, but these optimisations should be run for around 10 hours.

The optimised parameters for this cell line

```
opt_par_values <- opt_pars$parValues
names(opt_par_values) <- opt_pars$parNames
opt_par_values
```

```
##  TGFRb_n_TAK1  TGFRb_k_TAK1   TNFa_n_TAK1   TNFa_k_TAK1      tau_TAK1
##  3.000000e+00  6.290662e-01  3.000000e+00  0.000000e+00  7.787088e-03
##  TGFb_n_TGFRb  TGFb_k_TGFRb  EGFR_n_TGFRb  EGFR_k_TGFRb     tau_TGFRb
##  3.000000e+00  2.802733e-01  3.000000e+00  1.542701e-01  0.000000e+00
```

```
##    EGF_n_EGFR    EGF_k_EGFR    ERK_n_EGFR    ERK_k_EGFR      tau_EGFR
##  3.000000e+00  0.000000e+00  3.000000e+00  2.007754e-07  7.015971e-03
##    S6K_n_IRS1    S6K_k_IRS1   TBK1_n_IRS1   TBK1_k_IRS1   mTOR_n_IRS1
##  3.000000e+00  0.000000e+00  3.000000e+00  0.000000e+00  3.000000e+00
##   mTOR_k_IRS1   IGF1_n_IRS1   IGF1_k_IRS1    MEK_n_IRS1    MEK_k_IRS1
##  0.000000e+00  3.000000e+00  0.000000e+00  3.000000e+00  4.060043e-07
##      tau_IRS1    PI3K_n_AKT    PI3K_k_AKT       tau_AKT   PI3K_n_M3K1
##  0.000000e+00  3.000000e+00  4.100319e-02  0.000000e+00  3.000000e+00
##   PI3K_k_M3K1   RASK_n_M3K1   RASK_k_M3K1      tau_M3K1    ERK_n_RPS6
##  4.432509e-01  3.000000e+00  5.075054e-08  1.568557e-09  3.000000e+00
##    ERK_k_RPS6    S6K_n_RPS6    S6K_k_RPS6 RSKp90_n_RPS6 RSKp90_k_RPS6
##  2.498247e-01  3.000000e+00  0.000000e+00  3.000000e+00  8.162135e-01
##      tau_RPS6     IKK_n_ERK     IKK_k_ERK     MEK_n_ERK     MEK_k_ERK
##  1.027507e-02  3.000000e+00  3.065270e-01  3.000000e+00  1.271568e-01
##       tau_ERK  TAK1_n_MP2K4  TAK1_k_MP2K4  M3K1_n_MP2K4  M3K1_k_MP2K4
##  9.911941e-03  3.000000e+00  1.000000e+00  3.000000e+00  0.000000e+00
##  TNFa_n_MP2K4  TNFa_k_MP2K4     tau_MP2K4    mTOR_n_S6K    mTOR_k_S6K
##  3.000000e+00  0.000000e+00  0.000000e+00  3.000000e+00  5.503381e-01
##    PI3K_n_S6K    PI3K_k_S6K       tau_S6K    IKK_n_TBK1    IKK_k_TBK1
##  3.000000e+00  0.000000e+00  1.462364e-05  3.000000e+00  6.452658e-01
##      tau_TBK1    AKT_n_mTOR    AKT_k_mTOR      tau_mTOR    TAK1_n_IKK
##  9.931725e-03  3.000000e+00  0.000000e+00  0.000000e+00  3.000000e+00
##    TAK1_k_IKK     AKT_n_IKK     AKT_k_IKK    M3K1_n_IKK    M3K1_k_IKK
##  8.246211e-01  3.000000e+00  0.000000e+00  3.000000e+00  0.000000e+00
##    TNFa_n_IKK    TNFa_k_IKK    TBK1_n_IKK    TBK1_k_IKK       tau_IKK
##  3.000000e+00  0.000000e+00  3.000000e+00  8.707054e-05  0.000000e+00
##   EGFR_n_PI3K   EGFR_k_PI3K   IRS1_n_PI3K   IRS1_k_PI3K   RPS6_n_PI3K
##  3.000000e+00  1.599331e-01  3.000000e+00  0.000000e+00  3.000000e+00
##   RPS6_k_PI3K   TNFa_n_PI3K   TNFa_k_PI3K   IGF1_n_PI3K   IGF1_k_PI3K
##  1.000000e+00  3.000000e+00  0.000000e+00  3.000000e+00  0.000000e+00
##    HGF_n_PI3K    HGF_k_PI3K   RASK_n_PI3K   RASK_k_PI3K      tau_PI3K
##  3.000000e+00  4.751120e-01  3.000000e+00  3.009077e-02  3.042961e-02
##   EGFR_n_RASK   EGFR_k_RASK   IGF1_n_RASK   IGF1_k_RASK    HGF_n_RASK
##  3.000000e+00  0.000000e+00  3.000000e+00  0.000000e+00  3.000000e+00
##    HGF_k_RASK      tau_RASK    BRAF_n_MEK    BRAF_k_MEK    PAK1_n_MEK
##  5.156353e-07  0.000000e+00  3.000000e+00  7.505019e-03  3.000000e+00
##    PAK1_k_MEK       tau_MEK    ERK_n_BRAF    ERK_k_BRAF   RASK_n_BRAF
##  0.000000e+00  0.000000e+00  3.000000e+00  5.244096e-02  3.000000e+00
##   RASK_k_BRAF   PAK1_n_BRAF   PAK1_k_BRAF    PLX_n_BRAF    PLX_k_BRAF
##  0.000000e+00  3.000000e+00  9.269049e-02  3.000000e+00  3.603077e-06
##      tau_BRAF  ERK_n_RSKp90  ERK_k_RSKp90    tau_RSKp90    TAK1_n_JNK
##  0.000000e+00  3.000000e+00  9.381897e-01  1.471951e-02  3.000000e+00
##    TAK1_k_JNK    IRS1_n_JNK    IRS1_k_JNK    M3K1_n_JNK    M3K1_k_JNK
##  0.000000e+00  3.000000e+00  3.617738e-05  3.000000e+00  0.000000e+00
##    TNFa_n_JNK    TNFa_k_JNK   MP2K4_n_JNK   MP2K4_k_JNK       tau_JNK
##  3.000000e+00  8.352255e-02  3.000000e+00  0.000000e+00  0.000000e+00
##    AKT_n_PAK1    AKT_k_PAK1   PI3K_n_PAK1   PI3K_k_PAK1      tau_PAK1
##  3.000000e+00  5.496838e-01  3.000000e+00  0.000000e+00  0.000000e+00
##    TAK1_n_p38    TAK1_k_p38   MP2K4_n_p38   MP2K4_k_p38    TBK1_n_p38
##  3.000000e+00  2.582269e-01  3.000000e+00  0.000000e+00  3.000000e+00
##    TBK1_k_p38       tau_p38 TGFRb_n_SMAD2 TGFRb_k_SMAD2     tau_SMAD2
##  8.704423e-01  1.299676e-02  3.000000e+00  0.000000e+00  0.000000e+00
##    AKT_n_GSK3    AKT_k_GSK3 RSKp90_n_GSK3 RSKp90_k_GSK3      tau_GSK3
##  3.000000e+00  0.000000e+00  3.000000e+00  0.000000e+00  0.000000e+00
```

```
##   TAK1_n_IkBa   TAK1_k_IkBa    IKK_n_IkBa    IKK_k_IkBa      tau_IkBa
##  3.000000e+00  4.080694e-01  3.000000e+00  7.350341e-01  1.539245e-02
##    JNK_n_cJun    JNK_k_cJun      tau_cJun
##  3.000000e+00  0.000000e+00  0.000000e+00
```

Similar to the above cell-line, we can build a model for each of the cell lines. This is very time consuming, therefore we just load the optimised parameters from the paper.

```
optimised_parameters <- read_delim("./data/tutorial_3/allModelsParameters.txt",delim = "\t")
```

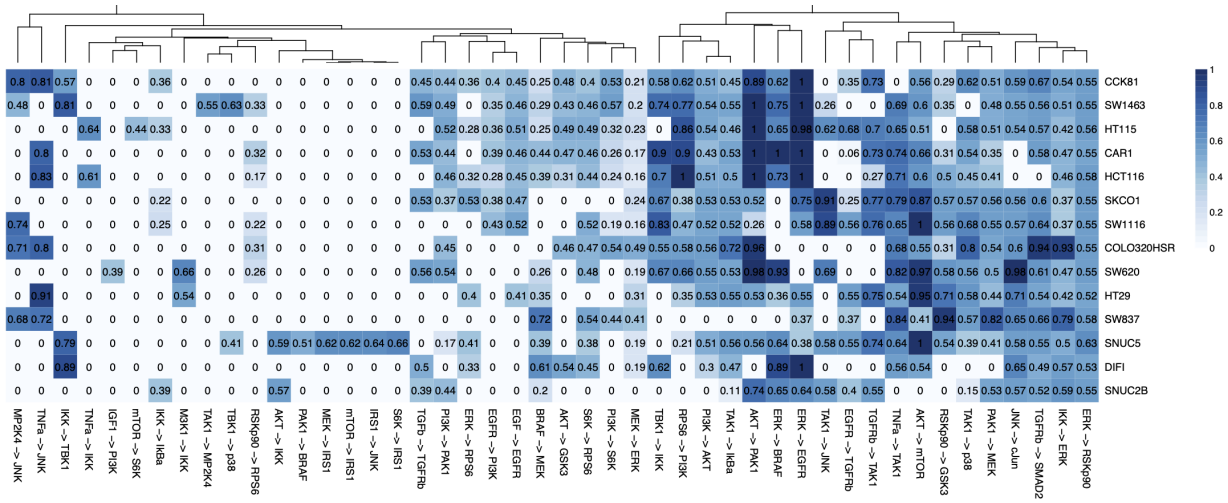
```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Cell_line = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

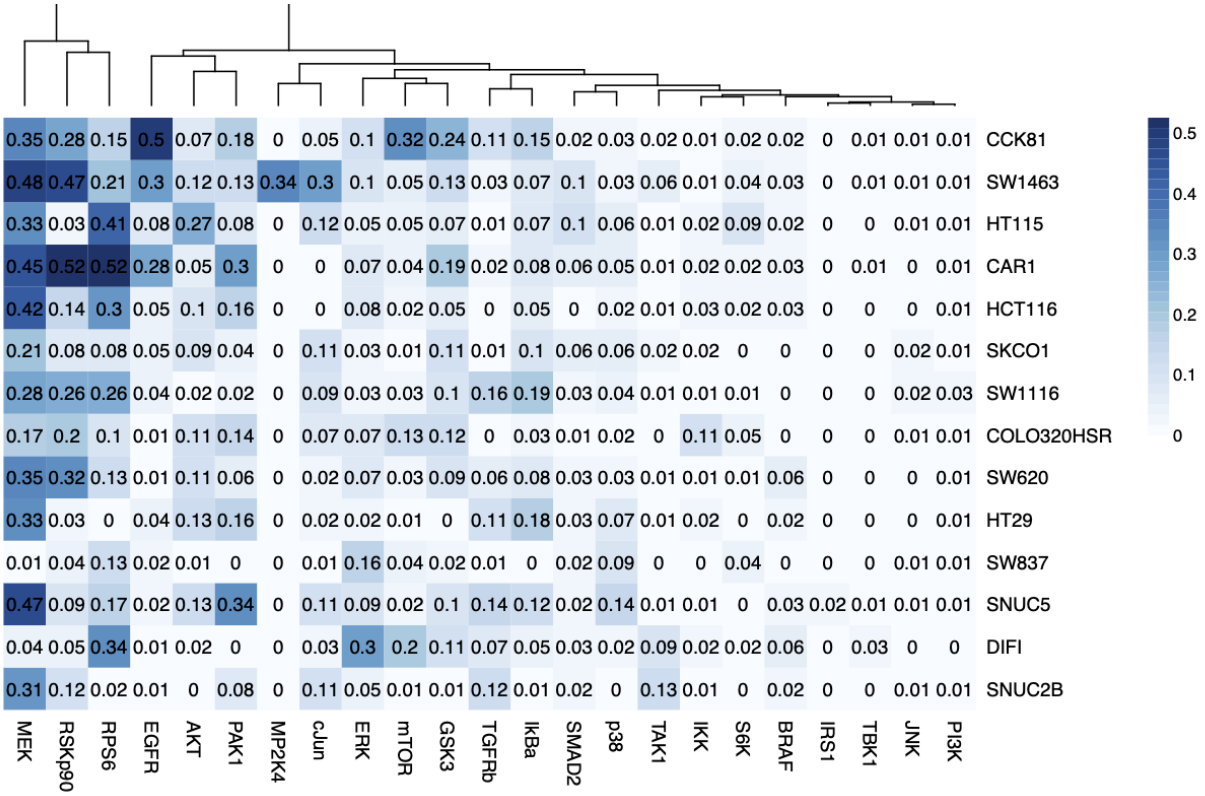
Let's check edge and node parameters.

```
edge_id <- grep("_k_",colnames(optimised_parameters))
edge_parameters_HM <-  optimised_parameters[edge_id]%>% as.matrix()
rownames(edge_parameters_HM) <- optimised_parameters$Cell_line
# heatmap(edge_parameters_HM, main = "Cell line edge parameters")

knitr::include_graphics("./data/tutorial_3/parHeatmap_k.png")
```




```
node_id <- grep("tau_",colnames(optimised_parameters))
node_parameters_HM <-  optimised_parameters[node_id]%>% as.matrix()
rownames(node_parameters_HM) <- optimised_parameters$Cell_line
# heatmap(node_parameters_HM, main = "Cell line node parameters")
knitr::include_graphics("./data/tutorial_3/parHeatmap_tau.png")
```



The level of edge and node parameters differs across the cell-lines.

Associate model parameters and drug response.