Getting Started with NuclearFRAP (Version 0.0.34)

Alan Bush

January 31, 2014

1 Introduction

`NuclearFRAP` provides functions to load, visualize, run model fits and profile likelihood analysis on nuclear FRAP experiments. To install the package follow the install vignette.

```
> vignette("Install")
```

If you already have the package installed, load it with

```
> library(NuclearFRAP)
```

2 Load a nuclear FRAP experiment to R

After finishing all the manual analysis of an experiment, you should end with a directory containing the following files:

levData (or posData): a tab delimited text file, with the description of each position. This file should have columns 'lev' (or 'pos') with the position number, 'pb' with the photobleaching order, and other columns describing the strain, cytokinesis state, etc.

OIF-date.txt: text file with the image capture date of each OIF file. This can be created by running

```
sfk filter -ls+"ImageCaputre" -file .txt > OIF-date.txt
```

lev??-time-series.xls: tab delimited text file created by ImageJ's plugin Time Series Analyzer v2

lev??-tif-fname.txt: text file with the file names of all the tif files of the position

lev??.stk.jpg: reference image, with the cell and the quantified regions drawn on it.

To load all this data into R, use the `loadFrapPairs` function of `NuclearFRAP`. This function returns a cell.data object that can be manipulated with `Rcell`'s functions.

```
> d<-loadFrapPairs("path/to/my/data")
```

If you have the original tif images of the experiment in a different folder than the files described above, you can specify a *img.path* parameter to `loadFrapPairs`.

```
> d<-loadFrapPairs("path/to/my/data", img.path="path/to/my/images")
```

In some experiments, because of the way the were analyzed, there is one cell per position. This means that the mother might be in pos=1 and the daughter in pos=2. In order to deal with all datasets in a transparent manner, you can restructure these datasets. In the restructured data mother and daughter will have the same position number. In order to do the restructuring, the 'levData.txt' file must have a 'pairs' columns, with the pair number.

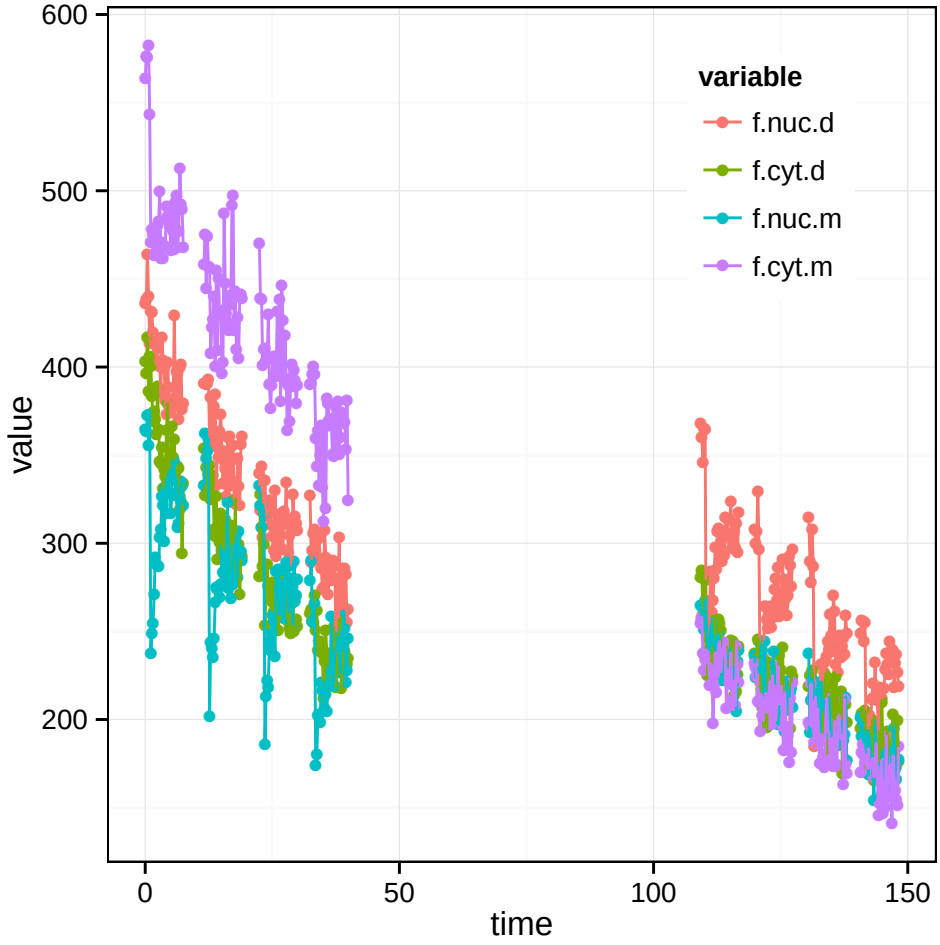


Figure 1: Variables vs time for pos==1

```
> d<-restructure.pairs(d)
```

If you haven't got a dataset to load, you can still go through this vignette using the example dataset.

```
> data(YFP)
```

3 Raw data visualization

Once loaded (and restructured) you can use Rcell's functions. For example to plot the time-course of all variables of pos 1 (Figure 1).

```
> cplot(d, c(f.cyt.d,f.cyt.m,f.nuc.d,f.nuc.m)~time, pos==1, geom=c("point","line")
+         , group=interaction(FRAP,variable)) + theme(legend.position=c(0.8,0.8))
```

You can also use Rcell to create a image montage (Figure 2)

```
> cimage(d, type + FRAP ~ t.index, channel="YFP", subset=pos==1 & t.index%%5==0, box.size=128/2-1
+          , contained.box=TRUE, normalize.group="FRAP")
```

4 Start a Matlab Server

This package uses Matlab's optimization functions, called through the package `R.matlab`. In order for the package's functions to work, there must be an open connection to a Matlab Server named *matlab*. Make sure you have Matlab and `R.matlab` installed. Follow the instructions in the ?Matlab help page of the `R.matlab` package to get things running. Here is an example code of how to initialize the Matlab Server.

```
> library(R.matlab)
> Matlab$startServer()
> matlab <- Matlab()
> open(matlab)
> print(matlab)
```

If everything went OK, the last line should return "... The connection to the MATLAB server is opened."

Matlab allows parallel processing and the functions of this package are design to take advantage of this. In order to use parallel processing, you have to initialize the *workers*. To do so, run the following code from R.

```
> evaluate(matlab, "if matlabpool('size') == 0 matlabpool('local'); end")
```

This line initializes the pool of workers, if they are not already initialized. Note that since Matlab R2013b *matlabpool* has been replaced by *parpool*. If you are planing to leave the package running in the background, then it's probably a good idea to leave some cores free. For example, in my Intel i5 quad-core, I call `matlabpool('local',3)` to initialize three workers, that use three cores, leaving one free to check Facebook. If you don't initialize the pool, the package should still work, but it won't use parallel processing.

The package expects to find all the required Matlab function files (.m) in the current directory. To copy them from the package's folder to the current directory use

```
> copyMatlabFunctions()
```

5 Fitting the model

To fit the model to the data, you can use the function `runModelFit`, which is a wrapper over modelFit.m. As most of `NuclearFRAP`'s functions, there is also a 'pair' version of this function, that calls modelFit.m for both cells of a pair. This function is called `runModelFitPair`.

```
> param <- runModelFitPair(d[[pos==1,]],p=d$parameters)
```

The first argument for `runModelFitPair` is a data.frame, with the data for the pair. This can be easily done with `d[[pos==1,]]`, that returns a data.frame for pos 1. The second parameter is a list containing a whole bunch of parameters for the fit. Check the help page for `?FRAP.options`. This list (with default values) is saved in the `parameters` slot of the cell.data object. The function returns a data.frame with columns *type* (indicating if it is the mother or daughter cell), *fit.index* (the index of the fit, for each cell), the cost of the fit and the values of all the parameters of the model.

Once we have the set of optimal parameters, we'll probably want to see how well these parameters (and model structure) fit the data. For this we can use the function `getSimDataPair`.

```
> sim.db<-getSimDataPair(d[[pos==1,]],param,p=d$parameters)
> sim.db<-restructureSimDataPair(sim.db)
```

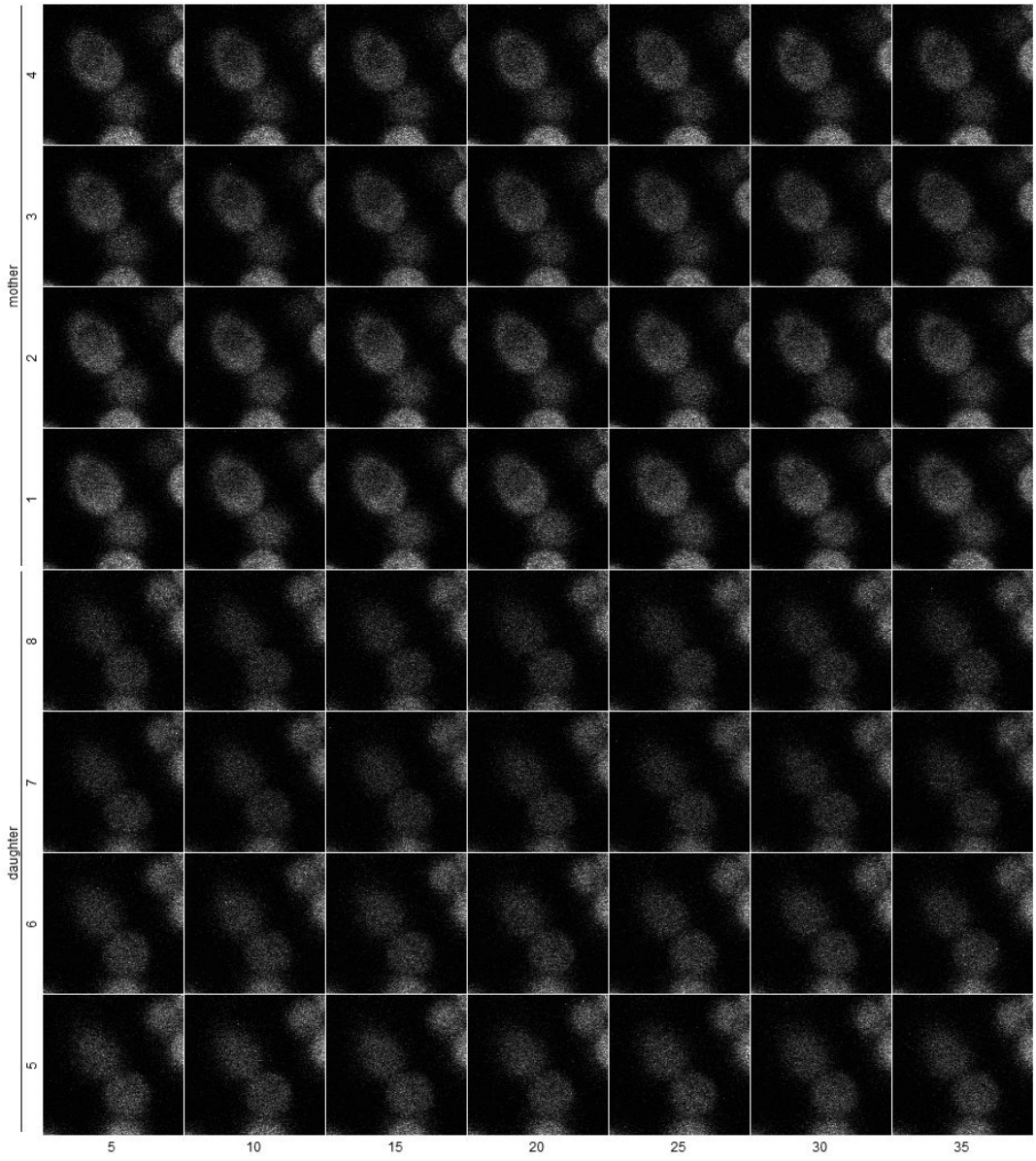


Figure 2: Image montage for pos 1

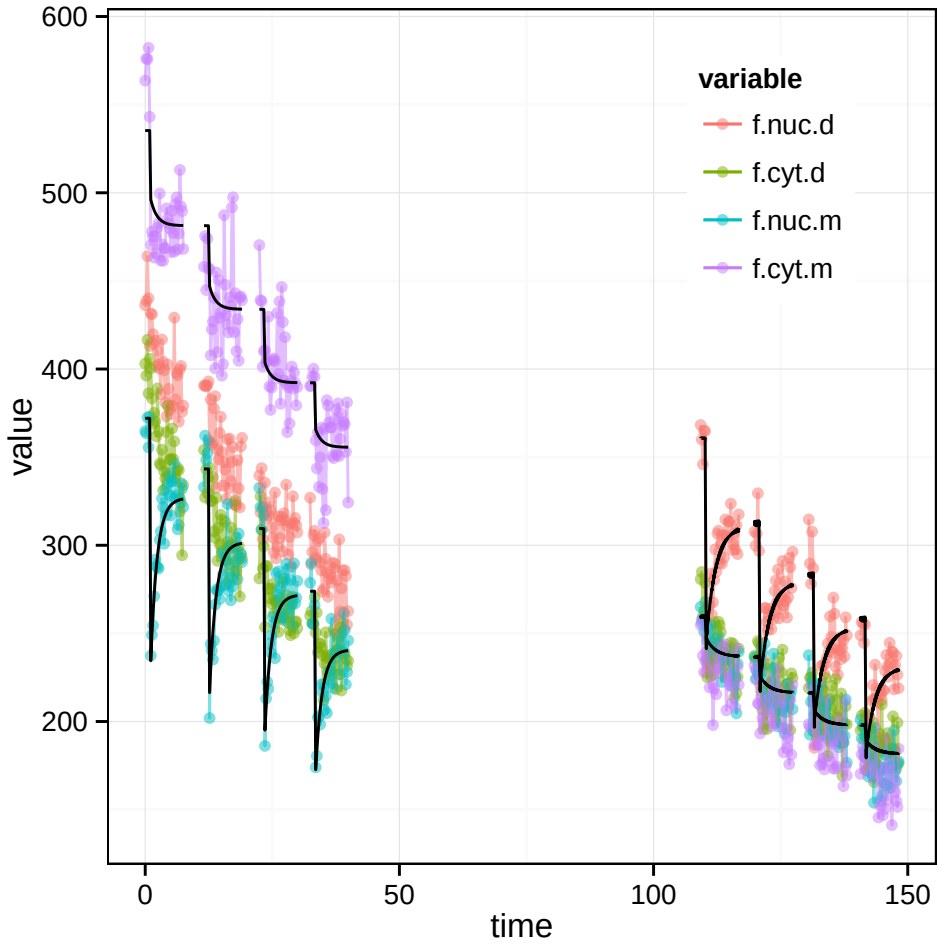


Figure 3: Raw data and overlay fit, for pos 1

The `getSimDataPair` function returns a data.frame with the nuclear and cytoplasmic fluorescence variables for the mother and daughter cell (and the expected standard deviation), for each time point. Because of the way the returned data.frame is organized, there are a lot of NAs. You can easily restructure this data.frame with the function `restructureSimDataPair`, to a format more convenient for plotting. To do an overlay of the data and the model fit, you can use the `cplot` function from `Rcell` (Figure 3).

```
> cplot(d, c(f.cyt.d,f.cyt.m,f.nuc.d,f.nuc.m)~time, pos==1, geom=c("point","line")
+         , group=interaction(FRAP,variable), alpha=0.5) +
+ clayer(sim.db, value~time, geom=c("line"), group=interaction(FRAP,variable), alpha=1, width=1.5)
```

6 Profile Likelihood Analysis

To get an idea of the uncertainty in the parameter estimation, we can do a profile likelihood analysis (Raue et al (2009)). The profile likelihood can be done in one dimension, fixing the value of a single parameter and leaving the others free to minimize the cost function. More interestingly for us, it can be done

(and easily visualized) in two dimensions, fixing the value of two parameters, and leaving the others free. We are particularly interest in the kinetic parameters kI and kEVfrac (see model NuclearFrapModelParameterization.pdf). The region of uncertainty will be the region where the profile likelihood cost is equal to or lower than the minimal cost, plus a threshold. If we use a chi-square cost functions, and the residuals are normally distributed, we can get this threshold from the chi-square distribution with an alpha = 0.05 and the degrees of freedom equal to the total number of free parameters of the model (Raue et al (2009)).

`NuclearFRAP` has two functions to calculate the profile likelihood uncertainty region. The first one is `runProfileLikelihoodContour` (and `runProfileLikelihoodContourPair`). This function works only for 2D profile likelihood analysis. Starting from the best fit point, it draws 'rays' in different directions (*theta*) and calculates where the increase profile likelihood cost equals the threshold (*boundaryCost*). It returns a data.frame with one point per direction. These points lie in the boundary of the uncertainty region. This method works well if the region is sort of **O** shaped, but fails if it is **U** shaped.

The second method is implemented in `runProfileLikelihoodGrid`. This method evaluates a profile likelihood cost in a regular grid in the parameter space. It works for both, 1D and 2D profile likelihood analysis. If the regular grid is not passed as an argument, `runProfileLikelihoodGrid` calls `runProfileLikelihoodContour` to get an estimation of the region it should sample. From the regular grid, the 2D confidence region can be calculated. This method can deal with **U** shaped regions, but is more computationally intense. The following command can take several minutes to complete.

```
> PL.db<-runProfileLikelihoodGridPair(d[[pos==2,]],param,p=d$parameters,scan=c("kEVfrac","kI")
+         ,boundaryCost=qchisq(p=c(0.05), df=9))
```

You can plot the 2D profile likelihood region with the `plotPL2D` function.

```
> plotPL2D(PL.db,x="kEVfrac",y="kI",param=param,signifCost=c(qchisq(p=c(0.05), df=9)))
```

For each cell, two contours are plotted. The solid line corresponds to the contour calculated from the regular grid of points, calculated by `runProfileLikelihoodGrid`. The dashed line is the contour as calculated by `runProfileLikelihoodContour`. The points indicate the best fit for the mother 'm1' or daughter 'd1'. If several local minimum are found they are also shown in this plot as 'm2', 'd3', etc. The chi-square cost of each point is indicated.

7 Analyze FRAP pair

Some 'high level' functions are included in the package. One of these is `analyzeFrapPair`. This function calculates the fit and profile likelihood analysis on kI and kEVfrac, for a pair of cells.

```
> fp<-analyzeFrapPair(d[[pos==1,]],d$parameters)
```

`analyzeFrapPair` returns a *frapPair* object, that contains all the data from the analysis. The package also provides a `plot` method for this class (Figure 5).

```
> plot(fp)
```

The create figure has several panels. The top left panels have the traces (points and thin lines) for the nuclear and cytoplasmic fluorescence, for mother and daughter. The panel title specifies on which cell the FRAP was done. These panels also have the model predictions, as solid lines of the color of the correspondent variable. When several local minima are find, the different predictions are plotted on these same panels. The gray shaded region corresponds to the model prediction of the standard deviation of the data. See the section on the determination of sigma for more details. The dashed black lines are the *swapped predictions*.

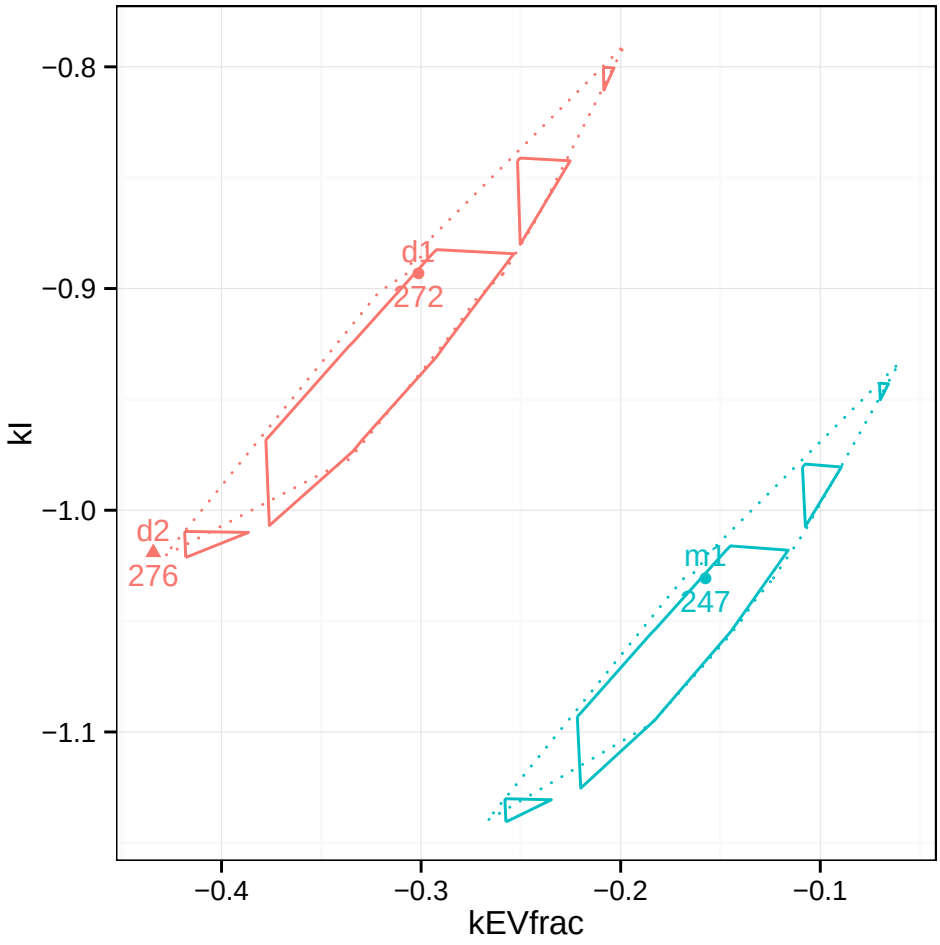
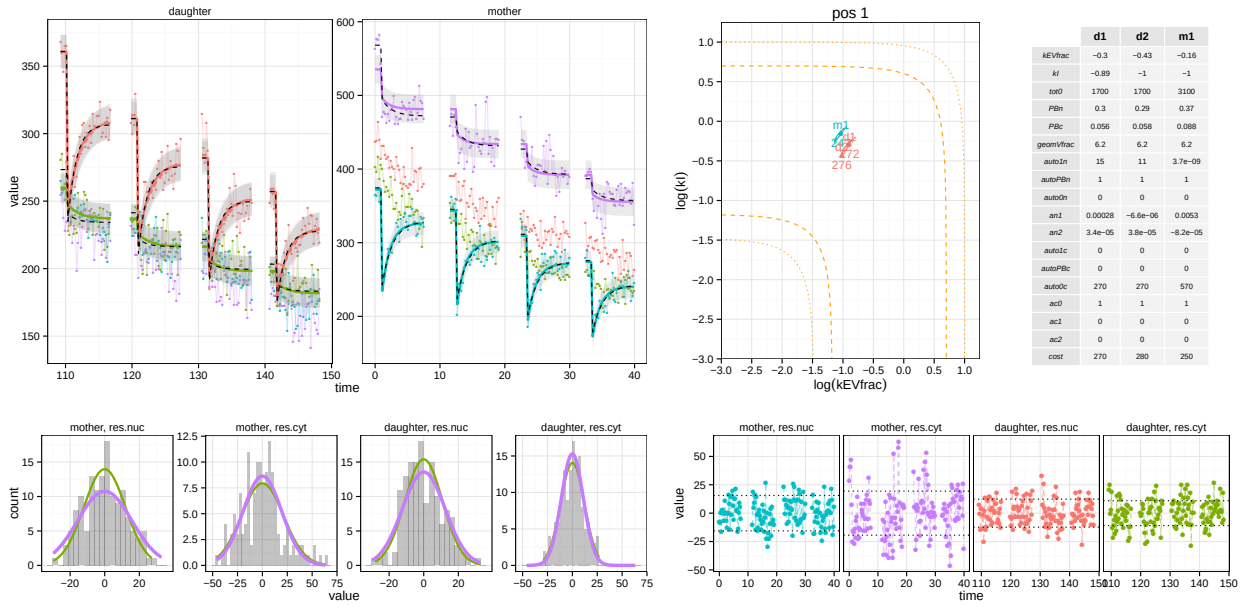


Figure 4: Profile likelihood contour for kEVfrac and kI, for pos 1.



	d1	d2	m1
kEVfrac	−0.3	−0.43	−0.16
kI	−0.89	−1	−1
tot0	1700	1700	3100
PBn	0.3	0.29	0.37
PBc	0.056	0.058	0.088
geomVfrac	6.2	6.2	6.2
auto1n	15	11	3.7e−09
autoPBn	1	1	1
auto0n	0	0	0
an1	0.00028	−6.6e−06	0.0053
an2	3.4e−05	3.8e−05	−8.2e−05
auto1c	0	0	0
autoPBc	0	0	0
auto0c	270	270	570
ac0	1	1	1
ac1	0	0	0
ac2	0	0	0
cost	270	280	250

Figure 5: Analysis of FRAP pair for pos 1

This means that the model is fitted to the daughters data, using the optimal values of kI and kEVfrac for the mother (and leaving all other parameters free). If this line looks like a good fit to the data, this strongly suggest that the daughter's kinetic parameters are NOT significantly different to the mother's. The same analysis is done for the mother, using the fixing the kinetic parameters to the daughter's optimal value.

The following panel to the right is the 2D profile likelihood analysis, for kI and kEVfrac. The local minima for the mother and daughter, and the profile likelihood regions are represented as explained before (see the Profile Likelihood Analysis section). The orange curves represent the *instrumental boundaries* for the kinetic parameters. The dashed line in the upper right are the points that satisfy tau = Tsampling (where tau=1/(kI + kEVfrac), and Tsampling is the time gap between consecutive data points). The dashed curve in the lower left corner is that satisfy tau = Texperiemnt (where Texperiement is the length of the FRAP experiment). The dotted curves are analogous, but using Tsampling/2 and 2*Texperiemnt as boundaries.

The table in the higher right panel gives the values of all parameters for the different local minima found.

The histograms in the lower left panels are the distribution of the residuals of the model (region shaded in gray), and the normal distribution using the mean and standard deviation of the residuals (green curve). The violet curve is the normal distribution, using mean zero and the standard deviation predicted by the model (see the section on the determination of sigma for more details). If the prediction of the sigma prediction is working, then the violet and green curve should coincide. These panels can also be used to determine if the residuals are normally distributed.

In the lower right corner the time course for the residuals are shown, for each variable. The black dotted lines are the predicted standard deviation for the data. These panels can be used to see if there is any tendency in the residuals, indicative that the fit is not good.

8 Estimating Sigma

The package has to methods to estimate sigma (the expected standard deviation of the data). The first one called "SIGMA.FUN", uses a user defined function to calculate sigma from the fluorescent level of the data. To use this method, modify the `paramters` slot of the cell.data object.

```
> d$parameters$SIGMA.METHOD <- "SIGMA.FUN"
> d$parameters$SIGMA.FUN <- function(fl) 25 + 32 * fl
```

An alternative method, and the one used by default, is to estimate the noise from the "unused" data. When doing the FRAP on the daughter, the mother variables are being acquired, but they are not used for the model fitting. So the idea is to use them for the estimation of sigma. Because we want the residuals, we need some "prediction" to calculate them. The first 5 points of each FRAP are assumed to be constant and the remaining points are fitted to a loess curve. From this coarse model, the residual distribution is calculated. The sigma level is assumed to be constant for each cell.

For the second cell to be photobleached, this method works well. For the first cell to be photobleached, it tends to under estimate the standard deviation, probably because after the photobleaching there is a lower signal and consequently a lower noise level. To correct for this, the package assumes the noise is proportional to the fluorescence level and corrects the sigma by the ratio of the fluorescence levels. This correction is only applied to the first cell to be photobleached. To use this method set SIGMA.METHOD to "SINGLE.CELL.PAIR".

```
> d$parameters$SIGMA.METHOD <- "SINGLE.CELL.PAIR"
```

9 Profile Likelihood Matrix

A way to check if all parameters are identifiable is to do a *profile likelihood matrix*, i.e. a matrix of plots where in each panel a 2D profile likelihood contour is shown, for the respective variables. In the diagonal, 1D profile likelihood cost curves are shown. To calculate all these profile likelihood analysis, you can use the high level function `profileLikelihoodFrapPair` and the correspondent plot method (Figure 6). Beware this is a very computationally expensive thing to do. It takes 12hs to complete a single experiment (with 10 pairs) in my Intel i5 quad-core.

```
> PL <- profileLikelihoodFrapPair(d[[pos==1,]], d$parameters,
+         vars=c("kEVfrac","kI","totO","PBn","PBc","auto1n"), pairwise = TRUE)
> plot(PL)
```

10 Closing the matlab connection

When finished, run the following commands to close the matlab server and the connection.

```
> evaluate(matlab, "matlabpool close force local")
> close(matlab)
```

References

NA Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood *BIOINFORMATICS* 25(15)1923-1929

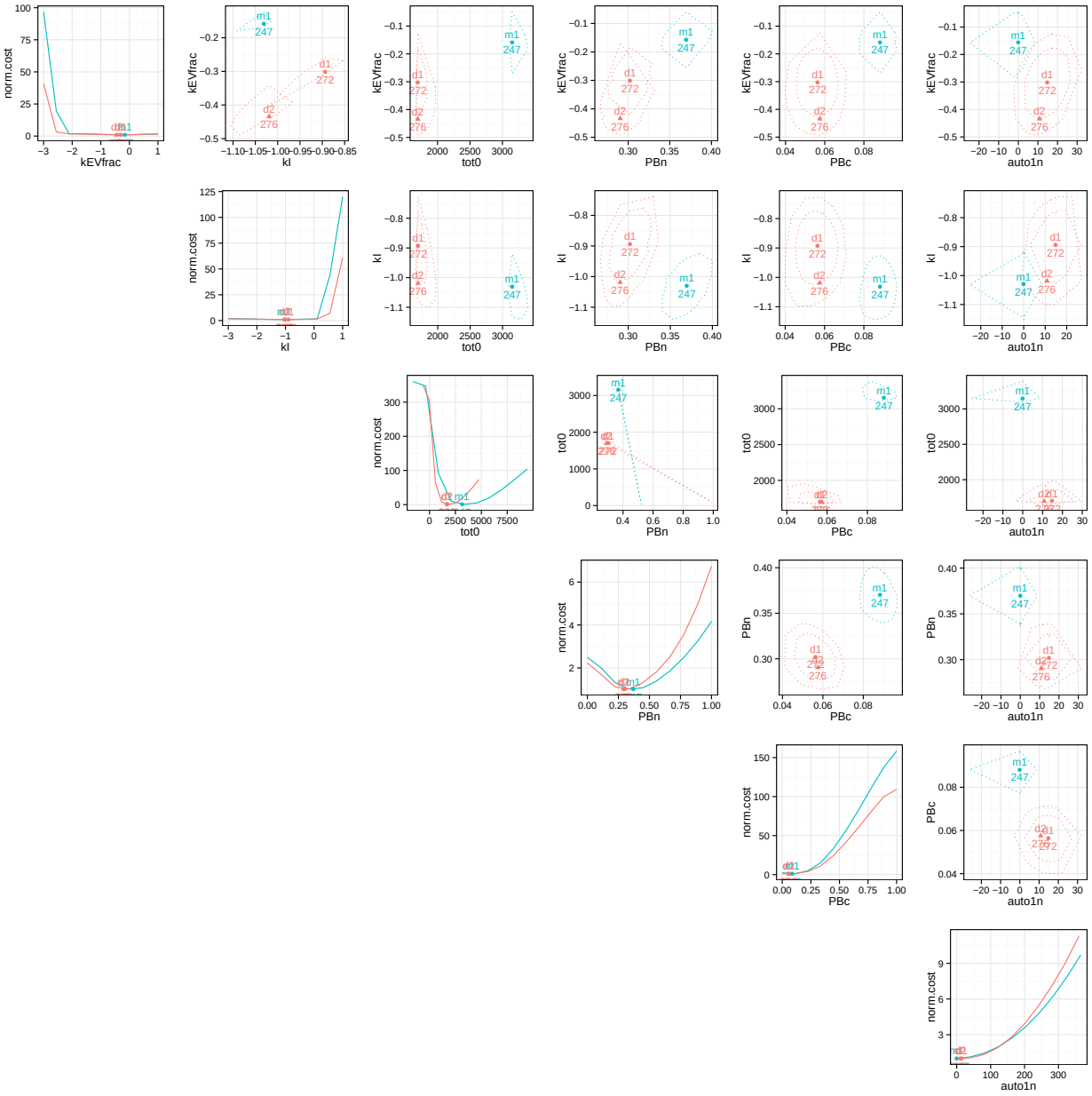


Figure 6: Profile Likelihood Matrix, for pos 1