Automated R tools for beads analysis

Version 0.11-2

Jan Stuchlý
Department of Paediatric Haematology and Oncology
2nd Faculty of Medicine
Charles University

Karel Fišer
Department of Paediatric Haematology and Oncology
2nd Faculty of Medicine
Charles University

October 17, 2011

Contents

Chapter 1

Running program

1.1 Defining structure of the array

This part is going to be obsolete as in next versions the structure will be extracted automatically from ”arraylist” and not developed further. To tell to the code how the experiment looks like, one has to create a named list with its properties. In our example, we will use the flowframe

```
flowFrame object 'a8772cc1-c7f5-481b-86e6-56e2a1080203'
with 60333 cells and 14 observables:
           name desc  range minRange maxRange
$P1        Time <NA> 262144     0.00   262143
$P2       FSC-A <NA> 262144     0.00   262143
$P3       SSC-A <NA> 262144     0.00   262143
$P4  B-530/30-A <NA> 262144  -111.00   262143
$P5  B-575/26-A <NA> 262144  -111.00   262143
$P6  B-610/20-A <NA> 262144   -50.22   262143
$P7  B-695/40-A <NA> 262144  -111.00   262143
$P8  B-780/60-A <NA> 262144  -111.00   262143
$P9  V-450/50-A <NA> 262144   -79.10   262143
$P10 V-536/26-A <NA> 262144  -111.00   262143
$P11 R-660/20-A <NA> 262144  -111.00   262143
$P12 R-710/50-A <NA> 262144  -111.00   262143
$P13 R-780/60-A <NA> 262144  -111.00   262143
$P14 L-586/15-A <NA> 262144   -65.32   262143
```

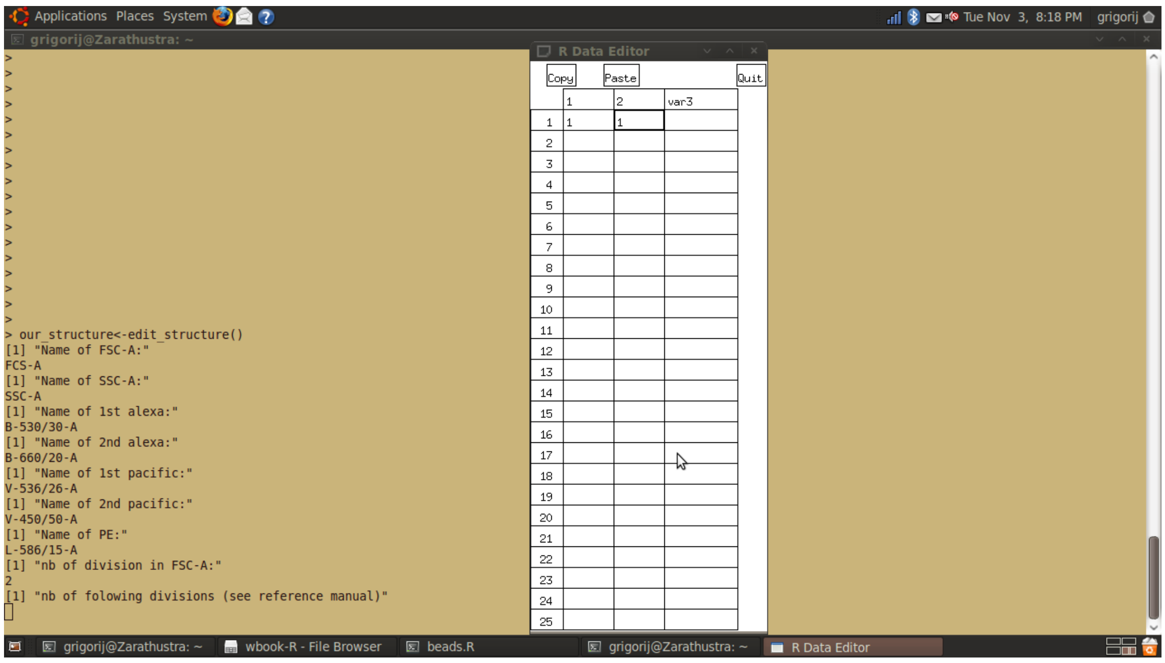
You can do it by typing

```
>our_frame<-edit_structure()
```

The name of the variable (here `our_frame`) is up to you. Do not forget the brackets at the end (it is a function). Now this function start to ask you about the names, number of population etc. You only answer this questions (caution: R is case sensitive). Follow steps on figure 1.1

This is (in my opinion) the most complicated point of structure definition. So far I have indicated only names, but now the code asks about the structure of populations in scatters (i.e. with respect to the size). (This part is rather awkward, but it is given by the algorithm, which clusters events subject to the bead-size). Here I told him divide the events first in two population in forward scatter

```
[1] "nb of division in FSC-A:"
```



```
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
> our_structure<-edit_structure()
[1] "Name of FSC-A:"
FCS-A
[1] "Name of SSC-A:"
SSC-A
[1] "Name of 1st alexa:"
B-530/30-A
[1] "Name of 2nd alexa:"
B-660/20-A
[1] "Name of 1st pacific:"
V-536/26-A
[1] "Name of 2nd pacific:"
V-450/50-A
[1] "Name of PE:"
L-586/15-A
[1] "nb of division in FSC-A:"
2
[1] "nb of folowing divisions (see reference manual)"
```

	1	2	var3
1	1	1	
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			

Figure 1.1:

    2

and then each of this resulting population ”divide” into one population (i.e. let it be) (the window of R Data Editor).
For understanding the rest of this procedure we take a particular example see fig 1.2. In this case we have 5×4 population (I call this basic structure), with missing population $9, 10, 11, 20$ see fig 1.3. And the rest of defining of the structure goes in the same way.
You could see, that I have misspelled the name of one column! In the place of `R-660/20-A` I have typed `B-660/20-A`. When we look on our result

```
> our_frame$gate_frame
$scatters
$scatters$names
[1] "FCS-A" "SSC-A"

$scatters$ddiv1
[1] 2

$scatters$ddiv2
[1] 1 1


$alexas
$alexas$names
[1] "B-530/30-A" "B-660/20-A"
```

```
$alexas$struct
$alexas$struct[[1]]
$alexas$struct[[1]]$base
[1] 5 4

$alexas$struct[[1]]$miss
[1]  9 10 11 20


$alexas$struct[[2]]
$alexas$struct[[2]]$base
[1] 5 5

$alexas$struct[[2]]$miss
[1] 25




$pacific
$pacific$names
[1] "V-536/26-A" "V-450/50-A"

$pacific$struct
$pacific$struct[[1]]
$pacific$struct[[1]]$base
[1] 3 4

$pacific$struct[[1]]$miss
      [,1]
 [1,]    0
 [2,]    0
 [3,]    0
 [4,]    0
 [5,]    0
 [6,]    0
 [7,]    0
 [8,]    0
 [9,]    0
[10,]    0
[11,]    0
[12,]    0
[13,]    0


$pacific$struct[[2]]
$pacific$struct[[2]]$base
[1] 3 4

$pacific$struct[[2]]$miss
```

```
      [,1]
 [1,]    0
 [2,]    0
 [3,]    0
 [4,]    0
 [5,]    0
 [6,]    0
 [7,]    0
 [8,]    0
 [9,]    0
[10,]    0
[11,]    0
[12,]    0
[13,]    0




$pe
[1] "L-586/15-A"

$beadsize
[1] "4u" "6u"
```

it looks... O.K. it is complicated, but using the operators `<-` and `$`, we can really easily correct the error.

```
>our_frame$gate_frame$alexas$names[2]<-"R-660/20-A"
```

The component `our_frame$gate_frame$alexas$names` has two "slots", that is why the "`[2]`".
And **we save** our result

```
>save(our_frame,file="the_name_of_file_of_our_choice")
```

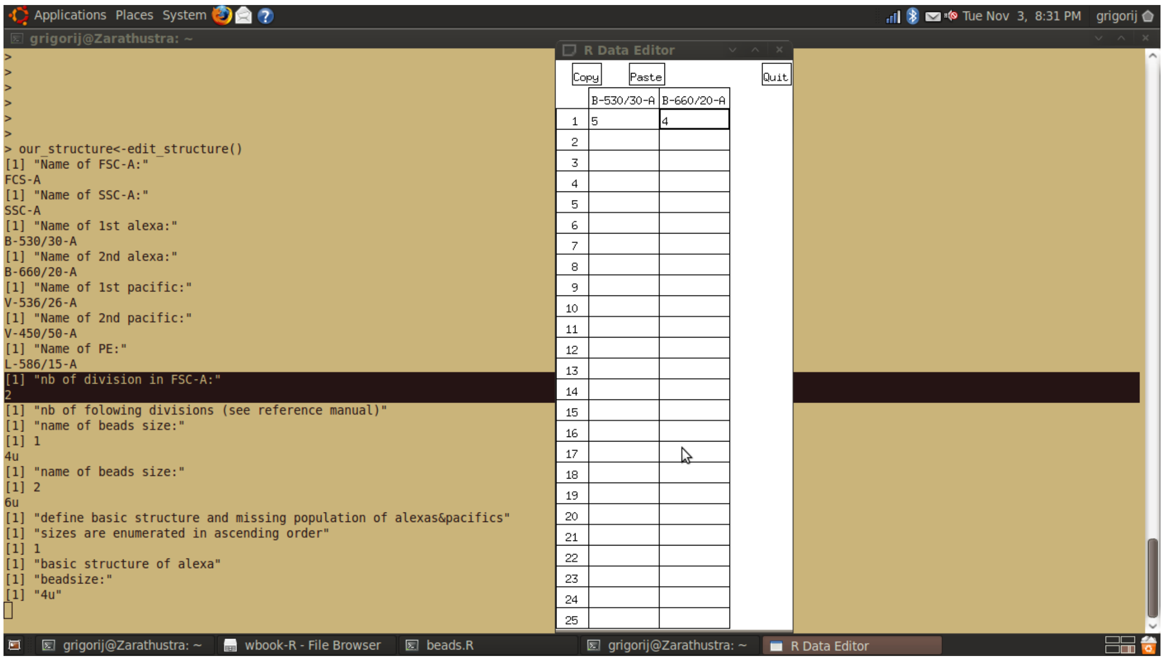
1.2 Creating pre-frame

Now we have created the structure, to add the information about clustering of training file(s) type

```
>our_frame<-create_gate_frame(frame=our_frame,inputfile="the_name_of_fcs_file(s)",
+popdesc="name_of_file")
```

If the parameter file is a single names the program tries to cluster it with respect to the defined structure. In the case the parameter `inputfile` is a vector of names (created via the concatenation function) the program takes it as an a-priori clustering of Ax750/Ax647 (FCS/SSC) populations i.e. takes each file as one population in this projection. The popdesc argument refers to the file which contains the description of population - text file with three columns (region, array, antibody), no empty cells and no spaces are allowed.
That is all. I call this result pre-frame, because it does not contain anything more than a clustering of one particular file.
And **we save** our result

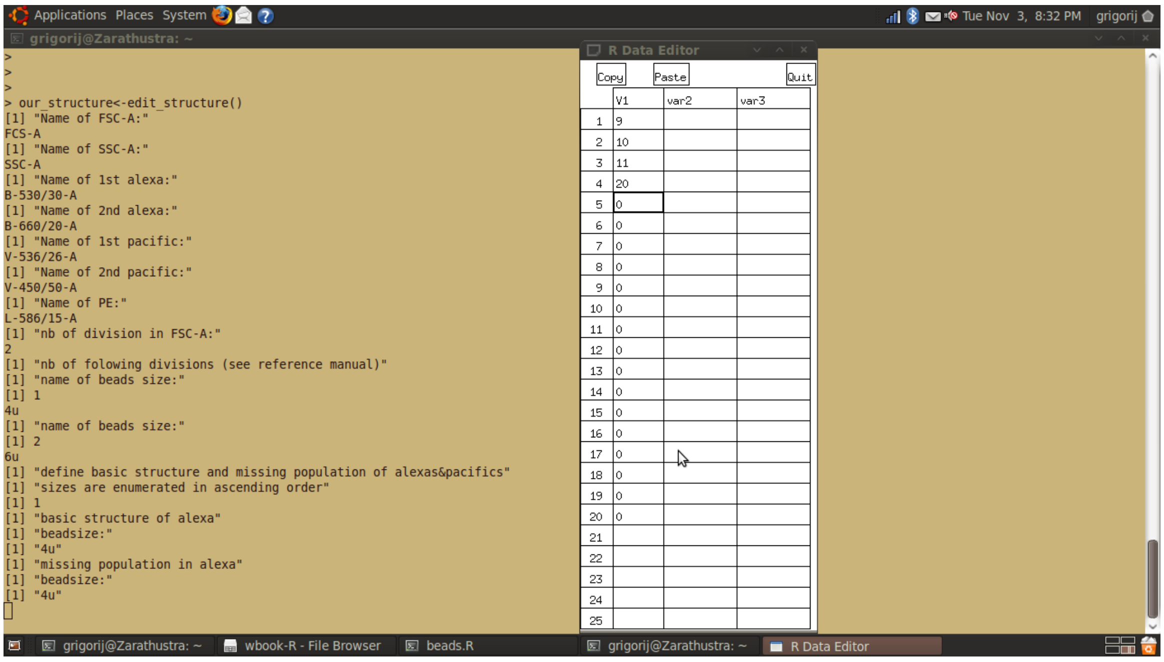
```
>save(our_frame,file="the_name_of_choice")
```



```
>
>
>
>
>
>
> our_structure<-edit_structure()
[1] "Name of FSC-A:"
FCS-A
[1] "Name of SSC-A:"
SSC-A
[1] "Name of 1st alexa:"
B-530/30-A
[1] "Name of 2nd alexa:"
B-660/20-A
[1] "Name of 1st pacific:"
V-536/26-A
[1] "Name of 2nd pacific:"
V-450/50-A
[1] "Name of PE:"
L-586/15-A
[1] "nb of division in FSC-A:"
2
[1] "nb of folowing divisions (see reference manual)"
[1] "name of beads size:"
[1] 1
4u
[1] "name of beads size:"
[1] 2
6u
[1] "define basic structure and missing population of alexas&pacifics"
[1] "sizes are enumerated in ascending order"
[1] 1
[1] "basic structure of alexa"
[1] "beadsize:"
[1] "4u"
```

	B-530/30-A	B-660/20-A
1	5	4
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		

Figure 1.2:



```
>
>
>
> our_structure<-edit_structure()
[1] "Name of FSC-A:"
FCS-A
[1] "Name of SSC-A:"
SSC-A
[1] "Name of 1st alexa:"
B-530/30-A
[1] "Name of 2nd alexa:"
B-660/20-A
[1] "Name of 1st pacific:"
V-536/26-A
[1] "Name of 2nd pacific:"
V-450/50-A
[1] "Name of PE:"
L-586/15-A
[1] "nb of division in FSC-A:"
2
[1] "nb of folowing divisions (see reference manual)"
[1] "name of beads size:"
[1] 1
4u
[1] "name of beads size:"
[1] 2
6u
[1] "define basic structure and missing population of alexas&pacifics"
[1] "sizes are enumerated in ascending order"
[1] 1
[1] "basic structure of alexa"
[1] "beadsize:"
[1] "4u"
[1] "missing population in alexa"
[1] "beadsize:"
[1] "4u"
```

	V1	var2	var3
1	9		
2	10		
3	11		
4	20		
5	0		
6	0		
7	0		
8	0		
9	0		
10	0		
11	0		
12	0		
13	0		
14	0		
15	0		
16	0		
17	0		
18	0		
19	0		
20	0		
21			
22			
23			
24			
25			

Figure 1.3:

1.3 Editing pre-frame & creating final template

Now we have to inspect the results of the code, modify it (if necessary) and create the final template.

```
>our_frame<-inspect_gate_frame(frame=our_frame,inputfile=NULL)
```

The optional `inputfile` argument allows you to check if the frame fits to particuliar file i.e. the frame is used to gate population on this file and shows you in the same way as without this argument the difference is in the set of ploted events - without this argument the events of the file given to `create_gate_frame` function are shown).
The program leads you and proposes possible actions.
Typing "finish" at the end of inspection creates the template for automated gating. This is the last part of preparation of a new array.
We have to note here, that before this action the `our_frame` was more general - now (after using the function `inspect_gate_frame()`), the structure is modified with respect to the missing population. If you use now this frame for the function `create_gate_frame()`, this changes are applied. This is usefull, in the case of changing parameters of the cytometer for example, but for new training for new array, you should use the former version. Therefore we change the name of our frame. And **we save** our result

```
>our_frame2<-our_frame
>save(our_frame2,file="the_name_of_choice")
```

If you want to do some modifications, enter the interactive mode by typing "i" and pressing enter. Now you can choose the action, follow the contextual help.

Manual gating

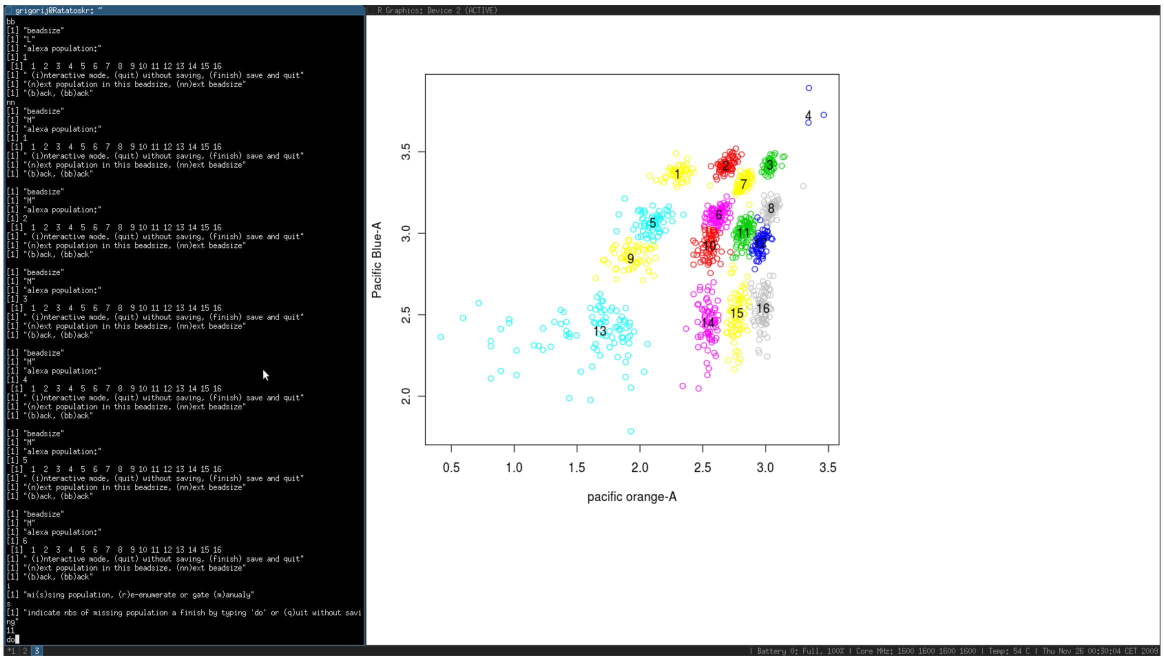
By typing "m" (enter), you launch an interactive ggobi interface for manual gating. Select the events you want to assign to population and write number of this population in R console (and press 'enter').
You can also unassign the events from population. There are two levels of unassignement - 1) by assigning the gated events to the population 0, you only remove them from current clustering, but there will be considered during re-clustering by the Missing population feature. 2) by assigning the gated events to the population -1, you exclude these events from future considerations (this could be useful for removing noise, which harms the effectivity of atomated clustering). This unassignement could be undo simply by assigning all events to 0 and re-clustering them with Missing population feature.
By typing "q" you leave this action without saving the changes, unlike by typing "sq" (enter).

Re-enumerating

In the case you want to change the enumeration of population, type "r" (enter) and make your changes in the pop-up spreadsheet. If you make a mistake, you can undo the changes by following action.

Missing population

If you want to re-cluster the population with different number of population, type "s" (enter) and indicate the numbers of missing population (see fig 1.4) and type "do" (enter). If you type "do" without indicating the numbers of populations, this sheet is re-clustered with respect to basic structure. Using this action, you can undo all changes (with respect to the level of unassignement of unassigned events - see Manual gating description).



```
bb
[1] "beadsize"
[1] "L"
[1] "alexa population:"
[1] 1
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
[1] " (i)nteractive mode, (quit) without saving, (finish) save and quit"
[1] "(n)ext population in this beadsize, (nn)ext beadsize"
[1] "(b)ack, (bb)ack"
nn
[1] "beadsize"
[1] "M"
[1] "alexa population:"
[1] 1
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
[1] " (i)nteractive mode, (quit) without saving, (finish) save and quit"
[1] "(n)ext population in this beadsize, (nn)ext beadsize"
[1] "(b)ack, (bb)ack"

[1] "beadsize"
[1] "M"
[1] "alexa population:"
[1] 2
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
[1] " (i)nteractive mode, (quit) without saving, (finish) save and quit"
[1] "(n)ext population in this beadsize, (nn)ext beadsize"
[1] "(b)ack, (bb)ack"

[1] "beadsize"
[1] "M"
[1] "alexa population:"
[1] 3
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
[1] " (i)nteractive mode, (quit) without saving, (finish) save and quit"
[1] "(n)ext population in this beadsize, (nn)ext beadsize"
[1] "(b)ack, (bb)ack"

[1] "beadsize"
[1] "M"
[1] "alexa population:"
[1] 4
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
[1] " (i)nteractive mode, (quit) without saving, (finish) save and quit"
[1] "(n)ext population in this beadsize, (nn)ext beadsize"
[1] "(b)ack, (bb)ack"

[1] "beadsize"
[1] "M"
[1] "alexa population:"
[1] 5
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
[1] " (i)nteractive mode, (quit) without saving, (finish) save and quit"
[1] "(n)ext population in this beadsize, (nn)ext beadsize"
[1] "(b)ack, (bb)ack"

[1] "beadsize"
[1] "M"
[1] "alexa population:"
[1] 6
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
[1] " (i)nteractive mode, (quit) without saving, (finish) save and quit"
[1] "(n)ext population in this beadsize, (nn)ext beadsize"
[1] "(b)ack, (bb)ack"
i
[1] "mi(s)sing population, (r)e-enumerate or gate (m)anualy"
s
[1] "indicate nbs of missing population a finish by typing 'do' or (q)uit without savi
ng"
11
do
```

Figure 1.4:

1.4 Applying template

The rest of our effort is very simple. We have created the template and the machine will do the rest.

```
>apply_gate_template(frame=our_frame,directory="path_to_the_file_to_analyse",
+ outputfile="path_and_name_of_the_output_file",tube="well")
```

Here I order to take `our_frame` as the template for gating the files in particular directory a write the medians of population in the file `outputfile`. The `tube` argument sets the description slot unique for each tube. (i.e. set `tube="TUBE NAME"` if you use tube name slot). Or if you have a directory which contains fcs files, Minimal_array_info.txt (must have suffix ".txt") and frame (its name must contain string "array"), you can just write

```
>apply_gate_template(directory="/path_to/experiment_source" , experiment_destination=
+"/path/test_output", tube="well")
```

And in the directory `experiment_destination` new directory with the name taken from the file `Minimal_array_info.txt` will be created. In this directory you can find extracted medians (and interpolated medians - if spurious value could be interpolated) (in subdirectory `MEDS`, some quality control information (including problems occured during gating) - subdirectory `QC`.

(See subsection Example)

1.5 Creating lineplots

The package contains a simple function for creating pdf file with lineplots.
Function

```
plot_lineplots(inputfile=NULL,pdfoutput=NULL,nb_fr=24,log=TRUE,fixedlimits=TRUE,
nogrid=FALSE,selection=NULL)
```

has seven parameters - `inputfile` is the name of the file(s) with population medians created by the function `apply_gate_template()`, `pdfoutput` is the name of the pdf file where you want to plot the lineplots. `nb_fr` is number of fractions, it is an optional parameter set to 24 by default. By default the logaritmical scale is used in lineplots, you can change it by setting the parameter `log=FALSE`. By default y-axis is fixed at 4 log span, `fixed=TRUE`. The horizontal grid is added to lineplots by default (you can remove it by setting `nogrid=TRUE`). Argument `selection` allows you to plot selected specimen (the 24 columns) only. For example `selection=c(1,7)` plots first and seventh 24 columns.

Or if you have experiment directory structure (as mentioned at the end of previous step) you can write

```
plot_lineplots(experiment="path_to_experiment_directory_structure")
```

And the lineplots will be created in subdirectory `ANALYSIS`.

(See subsection Example)

1.6 analysing results

This package contains a user-level function for basic analysis. `analyseBeads()` performs background substraction (based on popopulation with no antibody attached), intersample normalisation (based on protein entities stable subjet to experimental condition i.e. housekeeping proteins) and extract the values of proteins levels for futher use.

```
analyseBeads(peaks_list_path, array_list,hskeepers, hskeepers_peaks)
```

`peaks_list_path` - an absolut or relative path to the text file containing the definition of entities. `array_list` - data frame describing the array structure, see example and the web site for its structure. `hskeepers` - vector of character strings - the names of housekeeping proteins used for normalization. `hskeepers_peaks` - entities bounds, integer matrix with nrow(hskeepers_peaks)==length(hskeepers), first column the starting indices (largest fraction) of protein entity on lineplot, second column the ending indices (smallest fraction). After launching this function, user is asked to provide the path to the experiment directory (created by `apply_gate_template`), the results are then stored in the subfolder `ANALYSIS`.

1.7 Example

Now we will do all the work on a testing sample of files (a part of experiment with titration).
Visit site from http://www.bioinformin.net/sample.php.
Create directory - for example /home/work. Create subdirectory /home/work/exp_source and extract the fcs files (and minimal array info file) (sample data 1 - Experiment) in it. Download files archframe, sample data 2 - HIGH, sample data 3- LOW, array6558.txt, ProtArrPeaks_list_6t.txt and array65_58_list.csv to the directory /home/work/.

Launch terminal, enter the directory `~/work` and run R.
Load the R-package

```
library(rgbeads)
```

Now load the prepared frame.

```
load("archframe")
```

This frame is result of the function `edit_structure()` and contains only description of structure of the array. Now we add to this frame information about population distribution in the training file(s).

```
array6558<-create_gate_frame(frame=archframe,inputfile=c("1ST_C1_C01_LOW.fcs",
"1ST_C1_C01_HIGH.fcs"),popdesc="array6558.txt")
```

Now we need to inspect this frame and add to it the spatial template for rapid batch gating.

```
array6558<-inspect_gate_frame(array6558)
```

When you have checked all population, type ”finish” (enter), and wait a while, while template is being created.
The `frame` is now completed, it contains all information (structure of array, missing population, spatial template) and we can save it!

```
save(array6558, file="./exp_source/array6558")
```

And now eventually we can extract the medians!

```
apply_gate_template(directory="./exp_source")
```

Directory structure is created in /home/work/ directory.
Now we can create lineplots

```
plot_lineplots(experiment="./VK_10_L100702_TITRATION")
```

And eventually normalize data and extract entities.

```
# source KF functions:
source("/path_to/beads_functions_kf.R")
# prepare variables::
peaks_list_path <- "ProtArrPeaks_list_6t.txt"
# path to peaks definition file
array_list <- read.csv(file="array65_58_list.csv", header=TRUE)
# array list with antibody IDs and uniprots
hskeepers<- c("b2m_b2m", "HES1_1F5", "VRK1_1F6") # which proteins
hskeepers_peaks <- rbind(c(19, 24), c(15, 21), c(14, 18))
# start and end of peaks
x <- analyseBeads(peaks_list_path=peaks_list_path, array_list=array_list,
hskeepers=hskeepers, hskeepers_peaks=hskeepers_peaks)
# run the function
```

The output is several txt files (data) and pdf files (line plots and qc report). Normalised medians data have ending ”_internorm.txt” and peaks data have name of the peaks_list in file name. Note that here linear scale is used.