

This is a short discription to how to use the listed programs for the preparation and analysis of T-RFLP data:

Pre Steps:

Previous to using the programs provided the user will have to have access to perl, R, and SAS. perl and R are free software the can be downloaded from the internet at the. The URL for these programs is as follows:

For perl: <http://www.perl.com/>

For R: <http://www.r-project.org/>

SAS is a commercial software, and hence a license must be purchased if one wants to use it. More information about SAS can be found at: <http://www.sas.com/>.

Preparing the raw data:

Before using the listed programs make sure that the raw data is prepared according to our requirements which are as follows:

- 1) Make sure that the cutoff you use is as low as possible. In many cases it should be 5 fu. The aim is to have as much background noise as possible. This background noise helps in separating the peaks from the noise using a progressive filtering approach.
- 2) Incase you can't set the cutoff level very low, you can still use the programs though you might want to adjust the filtering level in order not to lose large peaks in the data this is described below.

Using the programs:

- 1) Preparing the files that hold the raw data:

The raw data files will have to text files (.txt). The format of these files should follow the ABI's format with 6 columns. The first column holds the color and fragment number, the second is time, the third is the estimated fragment length, the fourth is the peak height, the fifth is the peak area, and the last is the point number. The columns shouldn't have a header. An example of how the file should look like is below:

B,1	11.34	202.69	55	514	4251
B,2	11.4	204.85	87	730	4275
B,3	21.41	593.82	764	9412	8029
G,1	7.82	88.57	69	1809	2932
G,2	10.86	186.56	169	1607	4073
G,3	10.93	188.91	304	2241	4099
G,4	10.96	189.91	175	1536	4110
G,5	13.78	286.84	899	7930	5166
G,6	15	331.42	242	2123	5626
G,7	21.41	593.94	104	1423	8030
Y,1	7.04	65.66	25	169	2641
Y,2	7.57	81.22	25	180	2840
Y,3	7.58	81.54	25	80	2844
Y,4	7.72	85.6	25	210	2895

Y,5	7.86	89.78	25	210	2947
Y,6	8.07	96.3	25	157	3027

The resulting text files should be numbered consecutively. For example, one might have 10 files with the following names: Sample1.txt, Sample2.txt, Sample3.txt, Sample4.txt, ..., and Sample10.txt. This is important to facilitate reading these files into R described in the next step.

2) Using perl program to prepare for data entry into R

The file "AutomaticProgR.txt" contains a perl program that helps in reading the data into R and programming the analysis for you. This program will tell you what to do step by step. To run this program you need to use command line. You have to be in the "Command Prompt" window if using windows. You also have to be in the same directory as the program. You have to type "perl AutomaticProgR.txt" and hit enter.

The program will ask you to do some things pertaining to the following:

- a) Determining the location of the raw data. The message printed is:
 "Please type in the directory structure from where to read the input files:"
 "Example: C://Main Storage//Research//"

You have to type in where the files containing the raw data are located. Note that you have the have forward slash "/" or forward double slash "/" to describe the directory structure as in the example.

- b) Entering the name of the general name of the raw data files (the part before the number):
 "Please enter the general name you want for a file:"
 "Example: MspI"

In our example here this name is "Sample".

- c) The color to process:
 "Please enter the first letter (capital) of the color to be processed (B/G/Y):"

This is the color of the dye you want to process. Colors are processed separately and then combined at the end. This program supports three colors "B"lue, "G"reen, and "Y"ellow.

- d) Entering the number of files to processes.
 "Please enter the number of files you want to process"

In our example this will be 10 files. This is why it is important to number the raw data files consecutively as these numbers are used to create the R code to read these files into R and it is much easier to do so if they are numbered consecutively.

- e) Providing the name of the output file:
 "Please enter a name of an output file to send the R program to:"

This will tell perl where to save the R program that is generating. Lets call it “B.txt” for our example.

The result of this step will be a file (or a number of files if you are processing multiple colors) that holds an R program that you will use to process your data.

3) Data-entry and processing through R.

Before starting this step!!! You have to open the program files that you’ve created in the previous step and cut the last 4 lines:

```
B.list <- list(Sample1.B,Sample2.B,Sample3.B,...,Sample10.B)
B.Data <- Gfiltering.ftn(B.list, 3)
B.ClusBin <- GclustBin.ftn(B.Data[[1]], 1)
write.table(t(B.ClusBin), file = "C://Main Storage//Research//B.txt",sep="\t",row.names=F,col.names=F)
```

Open a new text file and paste these lines in.

If you have multiple files reflecting multiple colors the new file will look as follows:

```
B.list <- list(Sample1.B,Sample2.B,Sample3.B,...,Sample10.B)
B.Data <- Gfiltering.ftn(B.list, 3)
B.ClusBin <- GclustBin.ftn(B.Data[[1]], 1)
write.table(t(B.ClusBin), file = "C://Main Storage//Research//B.txt",sep="\t",row.names=F,col.names=F)
```

```
G.list <- list(Sample1.G,Sample2.G,Sample3.G,...,Sample10.G)
G.Data <- Gfiltering.ftn(G.list, 1)
G.ClusBin <- GclustBin.ftn(G.Data[[1]], 1)
write.table(t(G.ClusBin), file = "C://Main Storage//Research//G.txt",sep="\t",row.names=F,col.names=F)
```

```
Y.list <- list(Sample1.Y,Sample2.Y,Sample3.Y,...,Sample10.Y)
Y.Data <- Gfiltering.ftn(Y.list, 3)
Y.ClusBin <- GclustBin.ftn(Y.Data[[1]], 1)
write.table(t(Y.ClusBin), file = "C://Main Storage//Research//Y.txt",sep="\t",row.names=F,col.names=F)
```

Remove the “write.table” statement from all. The resulting file should look:

```
B.list <- list(Sample1.B,Sample2.B,Sample3.B,...,Sample10.B)
B.Data <- Gfiltering.ftn(B.list, 3)
B.ClusBin <- GclustBin.ftn(B.Data[[1]], 1)
```

```
G.list <- list(Sample1.G,Sample2.G,Sample3.G,...,Sample10.G)
G.Data <- Gfiltering.ftn(G.list, 3)
G.ClusBin <- GclustBin.ftn(G.Data[[1]], 1)
```

```
Y.list <- list(Sample1.Y,Sample2.Y,Sample3.Y,...,Sample10.Y)
Y.Data <- Gfiltering.ftn(Y.list, 3)
Y.ClusBin <- GclustBin.ftn(Y.Data[[1]], 1)
```

Add the following two lines to the end.

```
ClusBin <- rbind(B.ClusBin,G.ClusBin,Y.ClusBin)
```

```
write.table(t(ClustBin), file = "C://Main Storage//Research//ClusBinMatrix.txt",sep="\t",
row.names=c("length","Sample1","Sample2","Sample3",..., "Sample10"),col.names=F)
```

Note that “row.names” can be set by the user to be anything, though the first name should always be “length”! Save this file in “All.txt” (or any name you want).

Another important note, Gfiltering.ftn is a function that will pick the peaks for you assuming that your cutoff is as low as possible. The 3 in that statement indicates that any area value that is 3 standard deviations higher than the zero (calculated using the background noise) is considered to be a peak. If your cutoff is not low enough you might consider reducing this value to capture the correct peaks. This very important because if you cutoff at 50fu, for example, you will lose some of the large peaks which will result in discrepancies in your results.

Once the data is filtered the fragment lengths need to be aligned (binned) which is the function of “GclustBin.ftn”. The number 1 in the body of this function indicates that any two fragment lengths that are within 1bp from one another will be considered to have the same fragment length. If your machine is more or less precise than the sequencers that we were using you will want to have this number decreased or increased, respectively. DO NOT use a very small value that separates any two peaks as that will create discrepancies in your data analysis as well.

Now, you have to follow the following steps to process the data:

- a) Start R
- b) Open the files you created in using perl.
- c) Copy the contents of these files and paste them into R.

This will result in the data being read into R.

- d) Open the text file containing the R-functions that will be used to process the data “FilteringandBinning.txt”.
- e) Copy the contents of this file and paste them into R.
- f) Close “FilteringandBinning.txt”.

This results in reading the functions that you want to use

- g) Copy the contents of the file

This will result in processing your data and will cause your aligned peaks to be outputted to file “ClusBinMatrix.txt” in the directory you specified.

4) SAS macro

Now the data is ready to perform clustering on it. The SAS macro contained in file “ClusteringMacroSingleuse.sas” will do that for you. Follow the following steps:

- a) start SAS.
- b) Open the “ClusteringMacroSingleuse.sas”

- c) Specify where the input file is and where you want the resulting output to go. This done by locating the following statements and changing the directories and file names in them.

```
%let source = C:\Main Storage\Research\ClusBinMatrix.txt;  
%let dist1 = C:\ Main Storage\Research\Statistics.txt;  
%let dist2 = C:\Main Storage\Research\Clusters.txt;
```

The file “Statistics.txt” will contain the CCC, Pseudo-F, and Pseudo-T2 statistics that will give you an idea about how the optimal number of clusters were chosen (refer to our paper). The “Clusters.txt” file will list have two columns: the first includes the names of your samples (“Sample1”, “Sample2”, ...) and the second column will indicate the group number that a sample belong to (1, stands for cluster number 1, ...). The largest group number is the optimum number of clusters chosen.

- d) Now run the program in SAS.
e) The results will also be printed on screen along with a dendrogram.
f) You have to save the dendrogram by “Exporting” (A command you can click, found under “File”) it as an image.

Now you are ready to do further analysis.