Load Specs File

Lori Shepherd and Jonathan Dare

July 10, 2007

Contents

This will give a detailed description of how to create and what should be in a load.spec file. This file must be created by the user in order to use the aCGHplus package

The purpose of the load.specs file is to give information on how to read a sample's image analysis flat file, as well as what data to store from that flat file. This file may be created by any text editor such as emacs, excel, etc. It should consist of two columns of data. The columns MUST only be seperated by a comma (therefore when completing the file in programs such as excel the user must save as a comma delimited text file). There should be no header line(s); information should begin on the first line of the file.

There are a few standard pieces of infomration that MUST appear in the first column of the file:

- numfile
- cy.suffix (one file) or cy3.suffix, cy5.suffix (two file)
- spot.ID
- Column
- Row

The following are piece of information that should appear in the first column of the file if applicable:

- cy.sep
- cy.quote
- cy.skip
- cy.comment
- Meta.Row
- Meta.Column

Numfile represents the number of files associated with a single sample. Most softwares output a single file, however there are some like Imagene that output two files for each sample. Currently only one and two are acceptable values for numfile. This numeric value should be placed in the second column of the row in which numfile appears.

cy.suffix, cy3.suffix, and cy5.suffix represents the extenstion that may be found on the image analysis flat files. If numfile was equal to 1, cy.suffix should be in the first column. If numfile was equal to 2, cy3.suffix and cy5.suffix should appear in seperate rows of the first column. On all flat files appears a common suffix, this suffix is what should be listed as the value in the second column. If this suffix is removed from the flat file name, the remaining name should be analgous to the sample's image.name in the inventory file. For example, Imagene files commonly have a _532.txt and _635.txt suffix extensions, these values would be placed in the second column for cy3.suffix and cy5.suffix respectively.Remember when figuring this out from the image analysis flat files, this suffix should be the same on ALL files (or one of the two for numfile=2).

spot.ID is the name of the column in an image analysis flat file that stores the spot identification names (BAC IDs, Probe ID,ProbeName, Gene ID). These should match the names of the spot.IDs in the mapping file (see build map for more detail). These identifiers represent what was spotted down on the microarray chip. The name of the column within the flat file that has this information should go in the second column of the row for spot.ID. Note on header names in R: when R reads in a table it can sometimes change header names by removing or replacing characters. A space for example is replaced with a period. I honestly do not know all of the specifics but am working on a list of these types of conversions. This may be the cause of some errors when specifing fields.

We will discuss Row and Column momentarily when discussing Meta.Row and Meta.Column.

cy.sep, cy.quote, cy.skip, and cy.comment are optional to include in the first column but are strongly recommended. Standard defaults are set if they are not located in the load.specs file. These represent arguements to the read.table call that will read in the image analysis flat file. cy.sep is the seperation character for the data. This refers to how columns of the main section of data are seperated. The most common cy.sep is the tab, therefore if cy.sep is not specified in the first column of the load.specs file the program defaults to a tab delimited text. If another seperation character is used, it should be placed in double quotation marks in the second column of the row for cy.sep (for example a comma ","). cy.quote will specifiy a quotation character for the flat file. This will indicate how strings are represented. The default is the standard double quote. If a different quotation character is used, it should be placed in the second column or the row for cy.quote in double quotes. Be cautious when specifing a quotation character; if this character apppears it will treat anything following as part of the string until another quotation character appears. If it is not used or specified properly in the files, data may be truncated. cy.comment indicates the comment character in the image analysis flat file. The standard R comment is the pound

sign (#), however, this is often used for data and labeling in files. We have therefore changed the default to not have any comment character. Be cautious when including a comment character; any data remaining on a line where a comment character occurs is not read in as data.

To fully understand cy.skip the user must understand a little about how our package reads in data. When reading the image analysis flat files, the program will identify the area with the largest number of matching line lengths (entries per line determined by the seperation character). This area is taken to be the lines of raw data. In some flat files there may be information preceding or following this area; this information is disregarded. Once this region is identified cy.skip determines how many lines of data to skip before the raw data occurs. Some output, like from feature extraction, will have a line of the column type (INT, FLOAT, DOUBLE), a line for column names, and then the data. In this case we would want to skip a single line, excluding the column type but keeping the header names and data. cy.skip therefore is the number of lines to skip of the region of the most common number of entries. The user should keep column headers and data. The default is not to skip any lines.

As stated above we will now explain what we mean by Row, Column, Meta.Row, and Meta.Column, and how are program uses such information. We expect one of three situations to occur:

1. Meta.Row, Meta.Column, Row and Column occur in the file

2. Row and Column occur in the file and represent what we consider master row and master column

3. Row and Column occur in the file and represent the array spotter pin dimensions. This requires the following additional fields also be specified in the first column of the load.specs file: mapBlock, Block, and dimen.

Microarray chips are spotted using an arrayer. This arrayer has a group of pins that spot the chip. This creates a gird within a grid environment. The inner grid represents the pins; it will be the dimension of the pin group. The outter grid represents the group of pins as a whole, and where the arrayer spots the chip. If we say the pin group as a whole is a 'Block', and that Block spotted the chip 24 times, it could be in a 12 x 2 pattern, 4 x 6 pattern, etc. We will use a 4 x 6 grid. The chip would have a pattern like the following:

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

Each one of these 'Block' locations is a grid. The inner grid is the dimensions of the pin group used to spot. If the arrayer used a 18 x 12 pin group, each block (1 thru 24) has 216 spots arranged in an 18 X 12 pattern. Pick a block in the above table, 1,9, 22, they all have the following pattern as the inner grid:

	1	2	3	...	11	12
1	1	2	3	...	11	12
2	13	14	15	...	23	24
3	25	26		...	35	36
. . .				...		
17	193	194	195	...	203	204
18	205	206	207	...	215	216

If these were the dimensions of a chip, 6 x 4 and 18 x 12, there would be a total of 5184 spots per chip. The dimensions of all spots over the entire chip would be a 72 x 72 [(18 x 4) x (12 x 6)].

Let us return to the first case where Row, Column, Meta.Row and Meta.Col are included in the image analysis output files. In this case, Meta.Row and Meta.Column indicate the Block location. Using the above case of 24 blocks in a 6 x 4 matrix, Meta.Row would be a number one to four while Meta.Column would be a number from one to six. These tell which block the spot is located. The names of the columns that store this information in the image analysis flat files should be the value of the second column in the load.specs file for Meta.Row and Meta.Column rows. Row and Column should indicate the location of the pin within the pin group. Using the above case of 216 spots in an 18 x 12 matrix, Row would be a number one to 18 while Column would be a number one to twelve. The name of the columns in the image analysis flat file that stores these values should be given in the second column of the load.specs file for Row and Column IF THE FIRST CASE IS TRUE AND META.ROW AND META.COLUMN ARE ALSO GIVEN IN THE LOAD.SPECS FILE. The program will use Meta.Row, Meta.Col, Row and Column to calculate the mstr.row and mstr.column. This master row and column is the overall chip row and column number; In the above example these would be numbers from one to 72.

In the second case where only Row and Column are specified in the load.specs file, Row and Column should be the master row and master column. In the above case, these columns would have numbers from one to 72. The name of the column in the image analysis flat file that stores these values would be the value for the second column of the load.specs file.

In the third case Row, Column occur in the first column of the load.specs file. Row and Column represent the dimensions of the pin group or 'Block'. In our example case, the Row column would have numbers from one to 18 and the Column column would have numbers from one to 12. The value in the second column of the load.specs file would be the name of the column storing these values. If this case is true, there must be a mapBlock, Block, and dimen field in the first column of the load.specs file. The second column of the load.specs file should be T, or True, for the mapBlock field. The Block column in the image analysis flat file should indicate the block number representing when the chip was spotted using the arrayer. In our example above, this column would contain numbers from one to 24. The value in the second column of the load.specs file

would be the name of the column in the image analysis flat file that stores this information. Dimen represents the dimensions of how the pin group or 'Block' is spotted. Our examples above is 6 x 4. The value for the dimen field would therefore be "c(4,6)". It is mandatory for the value to be in double quotation and in the syntax of c(row,column). A field that should be specified by has a default if it is not is major. major should be in the first column of the load.specs file if this case is true. This field gives information on whether the arrays spotted arrays by row (row major) or by column(column major). Our program currently does not support an arrayer that spotted the chips it any random fashion. In the above example the arrayer would have spotted by row major, and therefore the value in the second column of the load.specs file is row. If the arrayer spotted by column major, the value in the second column would be column and the chip would appear as the following:

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

If major is excluded as an arguement, it is assumed to be spotted by row major. NOTE: If the user is using software that exports files with this scenerio or downloads data with this scenerio, there may be occassions where a group of experiments have different dimensions (different pin group used) or may have been spotted differently (row vs column major). The load.specs file will carry only one option. Each scenerio, in this case, should have a different load.specs file. This file will be specified in the inventory file.

If these cases do not exist in the image analysis flat files, it is the users responsiblity to alter the flat files to allow one of the cases to be true.

Looking at our load.specs file thus far we might have something like the following:

```
numfile,2
cy3.suffix,"_532.txt"
cy5.suffix,"_635.txt"
cy.sep,"\t"
cy.quote,""
cy.skip,0
cy.comment,""
spot.ID,Gene.ID
Meta.Column,Meta.Column
Meta.Row,Meta.Row
Column,Column
Row,Row


or
```

```
numfile,1
cy.suffix,.txt
cy.sep,"\t"
cy.quote,""
cy.skip,1
cy.comment,""
spot.ID,ProbeName
Column,Col
Row,Row
```

or

```
numfile,1
cy.suffix,".txt"
cy.sep,"\t"
cy.skip,0
cy.comment,""
spot.ID, Name
Row,Row
Column,Column
mapBlock,T
dimen,"c(12,6)"
major,row
Block,Block
```

The remaining arguments in the load.specs file are optional. They will tell what columns of data should be stored for future use/analysis. The first column in the load.specs file will set the path/name of where the data should be stored in a sample's array.image object. Each sample has an array.image object created during the load-in process that stores useful indicies and data about the sample. The second column of the load.specs file now will be the name of the column in the image analysis flat file which stores desired data for the variable in the first column. Our recommendation is to create a matrix object that will seperate and store cy3 and cy5 data. In this case, the first column would have values such as:

```
matrix$cy3$X
matrix$cy5$X
```

X could be variables such as Signal.Mean, StandardDeviation, Signal.Median, etc.

The utility of using matrix is the program has built in features based on the mstr.row and mstr.col calculation to place values where they were located on the chip. This allows chip heatmaps to be generated for different statistics.

The second column for all these variables in the first column should be a single character value representing the column name where the values of this variable are located. The Signal.Mean for example could be Signal.Mean, cy3.mean, g.mean, etc. We recommend saving Background Mean, Background Standard Deviation, Mean, Standard Deviation, and, if available, Flags (representing spot quality flags) and log ratios.

The following are examples of load.specs files:

```
For Imagene BAC arrays:

numfile,2
cy3.suffix,"_532.txt"
cy5.suffix,"_635.txt"
cy.sep,"\t"
cy.quote,""
cy.skip,0
cy.comment,""
spot.ID,Gene.ID
Meta.Column,Meta.Column
Meta.Row,Meta.Row
Column,Column
Row,Row
matrix$cy3$Flag,Flag
matrix$cy3$Background.Mean,Background.Mean
matrix$cy3$Background.Stdev,Background.Stdev
matrix$cy3$Signal.Mean,Signal.Mean
matrix$cy3$Signal.Stdev,Signal.Stdev
matrix$cy5$Flag,Flag
matrix$cy5$Background.Mean,Background.Mean
matrix$cy5$Background.Stdev,Background.Stdev
matrix$cy5$Signal.Mean,Signal.Mean
matrix$cy5$Signal.Stdev,Signal.Stdev


For Feature Extraction BAC array:

numfile,1
cy.suffix,".txt"
cy.sep,"\t"
cy.skip,0
cy.comment,""
spot.ID, Name
Row,Row
Column,Column
mapBlock,T
dimen,"c(12,2)"
```

```
major,row
Block,Block
matrix$cy3$Flag,Flags
matrix$cy3$Background.Mean,B532.Mean
matrix$cy3$Background.Stdev,B532.SD
matrix$cy3$Signal.Mean,F532.Mean
matrix$cy3$Signal.Stdev,F532.SD
matrix$cy5$Flag,Flags
matrix$cy5$Background.Mean,B635.Mean
matrix$cy5$Background.Stdev,B635.SD
matrix$cy5$Signal.Mean,F635.Mean
matrix$cy5$Signal.Stdev,F635.SD
matrix$LogRatio,Log.Ratio..532.635.
```

For Featured Extraction Agilent array:

```
numfile,1
cy.suffix,.txt
cy.sep,"\t"
cy.quote,""
cy.skip,1
cy.comment,""
spot.ID,ProbeName
Column,Col
Row,Row
matrix$cy3$Background.Mean,gBGMeanSignal
matrix$cy3$Background.Stdev,gBGPixSDev
matrix$cy3$Signal.Mean,gProcessedSignal
matrix$cy3$Signal.Stdev,gProcessedSigError
matrix$cy3$SpotExtendX,SpotExtentX
matrix$cy3$SpotExtendY,SpotExtentY
matrix$cy3$IsManualFlag,IsManualFlag
matrix$cy3$gIsFound,gIsFound
matrix$cy3$gIsFeatNonUnifOL,gIsFeatNonUnifOL
matrix$cy3$gIsFeatPopnOL,gIsFeatPopnOL
matrix$cy3$gIsPosAndSignif,gIsPosAndSignif
matrix$cy3$gIsWellAboveBG,gIsWellAboveBG
matrix$cy3$gIsSaturated,gIsSaturated
matrix$cy3$gIsInNegCtrlRange,gIsInNegCtrlRange
matrix$cy3$gIsBGNonUnifOL,gIsBGNonUnifOL
matrix$cy3$gIsBGPopnOL,gIsBGPopnOL
matrix$cy5$Background.Mean,rBGMeanSignal
matrix$cy5$Background.Stdev,rBGPixSDev
matrix$cy5$Signal.Mean,rProcessedSignal
```

```
matrix$cy5$Signal.Stdev,rProcessedSigError
matrix$cy5$SpotExtendX,SpotExtentX
matrix$cy5$SpotExtendY,SpotExtentY
matrix$cy5$IsManualFlag,IsManualFlag
matrix$cy5$rIsFound,rIsFound
matrix$cy5$rIsFeatNonUnifOL,rIsFeatNonUnifOL
matrix$cy5$rIsFeatPopnOL,rIsFeatPopnOL
matrix$cy5$rIsPosAndSignif,rIsPosAndSignif
matrix$cy5$rIsWellAboveBG,rIsWellAboveBG
matrix$cy5$rIsSaturated,rIsSaturated
matrix$cy5$rIsInNegCtrlRange,rIsInNegCtrlRange
matrix$cy5$rIsBGNonUnifOL,rIsBGNonUnifOL
matrix$cy5$rIsBGPopnOL,rIsBGPopnOL
matrix$LogRatio,LogRatio
```

As you can see, Feature Extraction for Agilent has many more columns we view as significant. Most of these columns are quality control flags. We recommend saving any quality control flags; if there is not an overall flag for quality control, it is highly recommended the user create a flag based on given knowledge of multiple quality control flags.