

# Package ‘aCGH.Spline’

January 2, 2012

**Type** Package

**Title** Robust spline interpolation for dual color array comparative genomic hybridisation data

**Version** 2.2

**Date** 2009-06-14

**Author** Tomas William Fitzgerald

**Maintainer** Tom Fitzgerald <tf2@sanger.ac.uk>

**Depends** rJava

**Description** Spline fitting and interpolation

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2010-01-18 10:45:28

## R topics documented:

aCGH.Spline-package . . . . .	2
batch.spline . . . . .	2
dLRs . . . . .	4
f.Noise . . . . .	5
fe.extract . . . . .	6
fe.read . . . . .	7
fe.write . . . . .	8
GenomePlot . . . . .	9
Jspline . . . . .	10
MakePDF . . . . .	12
MAPlot1 . . . . .	13
MAPlot2 . . . . .	14
nim.read . . . . .	15
nim.write . . . . .	17
ProPlot . . . . .	18

run.spline . . . . .	19
segN . . . . .	20
SetJ . . . . .	21
WriteGFF . . . . .	22
WriteQC . . . . .	23

<b>Index</b>	<b>25</b>
--------------	-----------

---

aCGH.Spline-package	<i>Package for robust spline interpolation normalisation on dual color aCGH data.</i>
---------------------	---

---

### Description

Carries out a robust spline interpolation method on aCGH dual color microarray data.

Contains some functions for reading and writing of standard aCGH data formats along with the generation QC statistics.

### Details

Package:	aCGH.Spline
Type:	Package
Version:	1.0
Date:	2009-05-14
License:	Open Source
LazyLoad:	yes

### Author(s)

Tomas William Fitzgerald

Maintainer: tf2@sanger.ac.uk

---

batch.spline	<i>Function for batch processing.</i>
--------------	---------------------------------------

---

### Description

Function for batch processing aCGH data though the "Jspline" method.

## Usage

```
batch.spline(dir, format = "FE", raw=TRUE, robust = TRUE, offset=5, knots=1000,  
ntyp = "percentile", p = 0.68, fact = 4.5, segN = FALSE,  
sn = 0.75, writeFE = FALSE, QC = FALSE, GFF = FALSE, PDF = FALSE)
```

## Arguments

dir	- the directory to process.
format	- the type of file format - c("FE", "NIM").
raw	- to use raw data or not (TRUE or FALSE).
robust	- make the method robust to outliers (TRUE or FALSE).
offset	- parameters to set the number of time the knot point will be offset.
knots	- the number of knot points to use for spline fitting.
ntyp	- type of noise calculation to perform - c("percentile", "derivative", "combined").
p	- value between 0 and 1 (the percentile value).
fact	- numeric value (the factor by which the noise value will be rised).
segN	- option for robust noise calculation (recommended when data contains many outliers e.g. cancer sample) TRUE or FALSE.
sn	- value between 0 and 1 - threshold for segmentation (segN).
writeFE	- Output a Feature Extraction formatted file (only to be used when processing Agilent FE files).
QC	- produce a QC report TRUE or FALSE.
GFF	- produce a GFF formatted file TRUE or FALSE.
PDF	- produce a PDF containing normalisation figures TRUE or FALSE.

## Details

Running this function using default parameters (just providing the directory argument) will perform consistently across a wide range of data qualities.

If method optimisation is required, the number of knot points and offsets can be adjusted along with the various noise estimation parameters.

## Value

As default this function outputs one ".temp" formatted file per file processed.

The ".temp" format consists of seven columns:

- Cy5 intensity values.
- Cy3 intensity values.
- Chromosome number.

- Start position.
- Stop position.
- An index.
- A flag.

**Note**

This function assumes that all "\*.txt" files within the target directory are in the specified format. (Agilent "FeatureExtraction.txt" or Nimblegen "SEG\_MNT.txt" files)

**Author(s)**

Tomas William Fitzgerald

**Examples**

```
# batch.spline(".")  
# This would run batch processing on the current directory using default settings.
```

---

dLRs

*Calculate the DLR Spread.*

---

**Description**

Calculates the "DLR spread" - derivative log<sub>2</sub> ratio spread.

**Usage**

dLRs(x)

**Arguments**

x                   - vector of log ratio values.

**Details**

Calculates the probe-to-probe log ratio difference of an array. This is a noise estimation which is robust to outliers.

**Value**

The derivative log<sub>2</sub> ratio spread.

**Note**

A value used to compute the minimum log ratio difference needed to detect an outlier.

**Author(s)**

Tomas William Fitzgerald

**References**

Agilent

**See Also**

<http://www.home.agilent.com/agilent/home.jspx>

**Examples**

```
r1 = rnorm(100, 0, 3)
dLRs(r1)
r2 = rnorm(100, 0, 5)
dLRs(r2)
```

---

f.Noise

*Noise calculating function.*

---

**Description**

Noise estimation function.

**Usage**

```
f.Noise(r1, fact = 4.5, p = 0.68, typ = "percentile")
```

**Arguments**

r1	- numeric vector of log ratio values.
fact	- numeric value (factor by which to rise the noise value).
p	- value from 0 to 1, the percentile to evaluate.
typ	- noise estimation type c("percentile", "difference", "combined").

**Value**

An estimation of noise.

**Author(s)**

Tomas William Fitzgerald

## Examples

```
r = rnorm(100, 0, 3)
f.Noise(r, fact=4.5, p=0.68, typ="percentile")
f.Noise(r, fact=4.5, typ="derivative")
```

---

fe.extract

*Function to extract Feature Extraction data.*

---

## Description

Read Agilent Feature extraction data into R.

## Usage

```
fe.extract(file, list)
```

## Arguments

**file** - a feature extraction file to read.  
**list** - a list of column names to extract e.g. c("rBGSubSignal", "gBGSubSignal", "SystematicName").

## Details

Function to extract data from a Feature extraction formatted file.

## Value

Returns a dataframe containing all the values specified in the list argument.

## Note

This function is capable of extracting any data contained within a Feature extraction file.

## Author(s)

Tomas William Fitzgerald

## See Also

<http://www.chem.agilent.com/en-us/products/instruments/dnamicroarrays/featureextractionsoftware/pages/default.aspx>

## Examples

```
# data <- fe.extract("FEfile.txt", list = c("rBGSubSignal", "gBGSubSignal", "SystematicName"))
# This would return and data frame containing background subtracted intensity values and positional information.
```

---

fe.read	<i>Function to read Feature Extraction files.</i>
---------	---

---

**Description**

Recommended feature extraction file reading function.

**Usage**

```
fe.read(file, Raw=TRUE)
```

**Arguments**

file	- the feature extraction file to read.
Raw	- the intensity values to extract (TRUE = background subtracted values, FALSE = Agilent normalised values).

**Details**

This function is capable of reading a FE file containing 1 million rows of data within 2 minutes.

Its also flags data points based on Feature extraction exculsion criteria and low intensity probes ( $cy5 + cy3 < 100$ ).

**Value**

A dataframe containing eight columns (".temp" format):

- Log2 ratio values.
- Cy5 intensity values.
- Cy3 intensity values.
- Chromosome number.
- Start position.
- Stop position.
- An index.
- A flag.

**Note**

If the flag is equal to 1 the probe should be removed from further analysis.

The index can be used as a unique identifier, it is the exact row where the data was taken from the FE file.

The function draws on methods contained within the "Jspline" class for increased speed performance.

**Author(s)**

Tomas William Fitzgerald

**Examples**

```
# data <- FeRead("FEfile.txt")
```

---

fe.write

*Function to output a Feature extraction formatted file.*

---

**Description**

Function used to replace intensity values, log ratio values and flags within a feature extraction input file.

**Usage**

```
fe.write(file, x)
```

**Arguments**

file	- a feature extraction input file to use as a template (ideally this should be the FE file from which the data came from).
x	- a ".temp" formatted dataframe containing the intensity values and flags to be written.

**Details**

Using a combination of "fe.read" and "fe.write" it is easy to modifying FE data and output results back to the original format.

**Value**

This outputs a new FE file having exactly the same name as the input FE file except its filename will start with "S".

The new FE file will contain the intensity values, log ratio and flags from the ".temp" format input stucture(x).

NB. If the input FE filename starts with an "S" the file will be overwritten.

**Note**

This function draws on methods contained within the "Jspline" class.

**Author(s)**

Tomas William Fitzgerald

**Examples**

```
# Read "FEfile.txt", flag all data points and write out in FE format.  
# data <- fe.read("FEfile.txt", raw=TRUE)  
# data[,8] = 1  
# fe.write("FEfile.txt", data)
```

---

GenomePlot

*Produces a genome plot.*

---

**Description**

Create a plot of the genome - plots each chromosome separately on the same figure.

**Usage**

```
GenomePlot(x, ylim=c(-3,3), axes=FALSE, pch=46, col="black")
```

**Arguments**

x	- a ".temp" formatted data structure.
ylim	- the y-axis limits defaults to c(-3,3).
axes	- should axes be draw around each plot (TRUE or FALSE).
pch	- the point character to use defaults to 46.
col	- the point color defaults to "black".

**Details**

Plots as many chromosome as are contained within the ".temp" formatted data structure on the same figure.

**Value**

A plot showing each chromosome separatly.

**Author(s)**

Tomas William Fitzgerald

**Examples**

```

# Create a vector of chromosome numbers
chr = rnorm(100,1,0)
for (x in 2:24) {
chr = c(chr, rnorm(100,x,0))
}

# Create a ".temp" data structure
mat = matrix(ncol=8, nrow=length(chr))
mat[,1] = log2(rnorm(length(chr),1000,150)/rnorm(length(chr),800,180))
mat[,2] = rnorm(length(chr),1000,150)
mat[,3] = rnorm(length(chr),800,180)
mat[,4] = chr
mat[,5] = seq(1,length(chr), 1)
mat[,6] = seq(1,length(chr), 1)
mat[,7] = 0
mat[,8] = 0

# Plot the data
GenomePlot(mat)

```

Jspline

*Spline fitting and interpolation function.***Description**

Method to carry out robust spline fitting and interpolation.

**Usage**

```
Jspline(x, offset=5, knots=1000, ntyp="percentile", p=0.68, fact=4.5,
robust=TRUE, segN=FALSE, sn=0.75)
```

**Arguments**

x	- ".temp" formatted data structure.
offset	- numeric value 1 or greater dictating how many times to offset the knot points.
knots	- the number of knot points to use in spline fitting.
ntyp	- the type of noise calculation to use, c("percentile", "derivative", "combined"), defaults to "percentile".
p	- numeric value between 0 and 1, the quantile to use.
fact	- numeric value, the factor by which the noise estimation will be rised.
robust	- make robust (exclude points from spline) TRUE or FALSE.
segN	- use segmentation prior to noise estimation TRUE or FALSE (see segN).
sn	- the segmentation threshold.

**Details**

This method carries out natural cubic spline fitting and interpolation on aCGH dual color microarray data.

**Value**

Input data structure (x) is returned having had cy5 and cy3 intensity data adjusted.

**Note**

Jspline performs consistently over a large range of data qualities and array formats but adds most benefit to noisy, highly rearranged data.

The number of points included in the spline fit can be adjusted (this ensures that the dye bias is assessed on reliable data points).

This method is written in java and contained within the "Jspline" class.

**Author(s)**

Tomas William Fitzgerald

**Examples**

```
## Set up noisy data with a bias
v = seq(1,100000,0.5)
d = sin(2*pi/500 * v)
nd = d + rnorm(length(d),0,100) + 1000
dd = sin(2*pi/1000 * v)
ndd = dd + rnorm(length(dd),0,120) + 1000

## Create ".temp" data structure
mat = matrix(ncol=8,nrow=length(dd))
mat[,1] = log2(nd/ndd)
mat[,2] = nd
mat[,3] = ndd
mat[,4] = 0
mat[,5] = 0
mat[,6] = 0
mat[,7] = 0
mat[,8] = 0

## Add a few flags
mat[100:150,8] = 1

## Plot the data before and after Jspline
par(mfrow=c(2,1),mar=c(1,1,1,1))
MAPlot1(mat,ylim=c(-1,1),xlim=c(9.4,10.4))
aa = Jspline(mat)
```

```
MAPlot1(aa,ylim=c(-1,1),xlim=c(9.4,10.4))
```

---

MakePDF

*Write a PDF containing some normalisation plots.*

---

## Description

This function creates a PDF containing various figures to show the quality of normalisation.

## Usage

```
MakePDF(filename, x, raw, ntyp="percentile", p=0.68, fact=4.5, segN=FALSE, sn=0.75)
```

## Arguments

filename	- character argument, the output file name.
x	- the ".temp" formatted data structure after normalisation.
raw	- the ".temp" formatted data structure before normalisation.
ntyp	- the type of noise calculation to use, c("percentile", "derivative", "combined").
p	- numeric value between 0 and 1, the quantile used.
fact	- numeric value, the factor by which the noise estimation was risen.
segN	- was segN used TRUE or FALSE.
sn	- the segmentation threshold.

## Details

Produces a summary ".pdf" files containing various figures.

## Value

PDF containing a "ProPlot", "GenomePlot" and two MAplots (MAplot1 and MAplot2).

## Note

MA-plots a useful way of assessing the quality of the normalisation. Most points should be centered around zero and the outliers should have little effect on the overall fit (this is the key point of this normalisation method).

## Author(s)

Tomas William Fitzgerald

## Examples

```
# NA
```

---

`MAPlot1`*Produce a MAplot showing points retained vs. points excluded.*

---

**Description**

Produce a MA-plot showing points retained and points excluded.

**Usage**

```
MAPlot1(x, pch=46, Retcol="black", Excol="red", ylim=c(-5,5), xlim=c(0,20), xlab="log2(mean(cy5, cy3))")
```

**Arguments**

<code>x</code>	- ".temp" formatted data structure.
<code>pch</code>	- the point character to use defaults to 46.
<code>Retcol</code>	- the point color to use for the points retained after normalisation, defaults to "black".
<code>Excol</code>	- the point color to use for the points excluded after normalisation, defaults to "red".
<code>xlim</code>	- the x-axis limits defaults to c(0,20).
<code>ylim</code>	- the y-axis limits defaults to c(-5,5).
<code>xlab</code>	- the label for the x-axis.
<code>ylab</code>	- the label for the y-axis.
<code>main</code>	- the main title for the plot.

**Value**

Figure showing the  $\log_2(\text{mean}(\text{red}, \text{green}))$  vs. the  $\log_2(\text{red}/\text{green})$ .

**Author(s)**

Tomas William Fitzgerald

**Examples**

```
## Set up noisy data with a bias
v = seq(1,100000,0.5)
d = sin(2*pi/500 * v)
nd = d + rnorm(length(d),0,100) + 1000
dd = sin(2*pi/1000 * v)
ndd = dd + rnorm(length(dd),0,120) + 1000

## Create ".temp" data structure
mat = matrix(ncol=8,nrow=length(dd))
mat[,1] = log2(nd/ndd)
```

```

mat[,2] = nd
mat[,3] = ndd
mat[,4] = 0
mat[,5] = 0
mat[,6] = 0
mat[,7] = 0
mat[,8] = 0

## Add a few flags
mat[100:150,7] = 1

## Plot the data
par(mfrow=c(2,1),mar=c(1,1,1,1))
MAPlot1(mat,ylim=c(-1,1),xlim=c(9.4,10.4))

```

---

MAPlot2

---

*Produce a MAplot showing points fitted vs. points interpolated.*


---

### Description

Produce a MA-plot showing points fitted and point interpolated.

### Usage

```
MAPlot2(x, raw, ntyp="percentile", p=0.68, fact=4.5, segN=FALSE, sn=0.75, pch=46, Fitcol="green", Intcol="red", xlim=c(0,20), ylim=c(-5,5), xlab, ylab, main)
```

### Arguments

x	- the ".temp" formatted data structure after normalisation.
raw	- the ".temp" formatted data structure before normalisation.
ntyp	- the type of noise calculation to use, c("percentile", "derivative", "combined"), defaults to "percentile".
p	- numeric value between 0 and 1, the quantile to use.
fact	- numeric value, the factor by which the noise estimation will be risen.
segN	- use segmentation prior to noise estimation TRUE or FALSE (see segN).
sn	- the segmentation threshold.
pch	- the point character to use defaults to 46.
Fitcol	- the point color to use for the points used during spline fitting, defaults to "green".
Intcol	- the point color to use for the points excluded from spline fitting, defaults to "red".
xlim	- the x-axis limits defaults to c(0,20).
ylim	- the y-axis limits defaults to c(-5,5).
xlab	- the label for the x-axis.
ylab	- the label for the y-axis.
main	- the main title for the plot.

**Value**

MAplot showing relative number of points considered to be outside the normal distribution.

**Note**

Points outside the interpolation threshold are not used during the spline fitting. We believe that these points are highly unreliable for assessing the dye bias.

**Author(s)**

Tomas William Fitzgerald

**Examples**

```
# Create some noisy data with a bias
v = seq(1,100000,0.5)
d = sin(2*pi/500 * v)
nd = d + rnorm(length(d),0,100) + 1000
dd = sin(2*pi/1000 * v)
nnd = dd + rnorm(length(dd),0,120) + 1000

## Create ".temp" data structure
mat = matrix(ncol=8,nrow=length(dd))
mat[,1] = log2(nd/nnd)
mat[,2] = nd
mat[,3] = nnd
mat[,4] = 0
mat[,5] = 0
mat[,6] = 0
mat[,7] = 0
mat[,8] = 0

# Add a few outliers
mat[100:150,1] = mat[100:150,1] * 1.5
mat[100:150,2] = mat[100:150,2] / 1.5

# Run Jspline and plot
aa = Jspline(mat)

MAPlot2(aa, mat, ylim=c(-1.5,1.5),xlim=c(9.4,10.4))
```

---

nim.read

*Function to read a Nimblegen "SEG\_MNT.txt" file.*

---

**Description**

Function to read a Nimblegen "SEG\_MNT.txt" file.

**Usage**

```
nim.read(file, Raw)
```

**Arguments**

file           - Nimblegen "SEG\_MNT.txt" file to read.  
Raw            - to use raw data or not (TRUE or FALSE).

**Details**

This function reads the contents of a Nimblegen "SEG\_MNT.txt" file into a ".temp" formatted dataframe and flags low intensity probes.

**Value**

The ".temp" dataframe contains eight columns:

- Log2 ratio values.
- Cy5 intensity values.
- Cy3 intensity values.
- Chromosome number.
- Start position.
- Stop position.
- An index.
- A flag.

**Note**

This function reads spatially normalised intensity values from the Nimblegen input file. There is currently no option to output a "SEG\_MNT.txt" format, however, signal-map gff format is supported.

**Author(s)**

Tomas William Fitzgerald

**See Also**

<http://www.nimblegen.com/>

**Examples**

```
# data <- nim.read("Nimblegen_SEG_MNT.txt")
```

---

nim.write	<i>Function to output a Nimblegen formatted file.</i>
-----------	---

---

### Description

Function used to replace intensity values and log ratio values within a Nimblegen input file.

### Usage

```
nim.write(file, x)
```

### Arguments

file	- a Nimblegen input file to use as a template (ideally this should be the Nimblegen file from which the data came from).
x	- a ".temp" formatted dataframe containing the intensity values and log <sub>2</sub> ratio to be written.

### Details

Using a combination of "nim.read" and "nim.write" it is easy to modifying Nimblegen data and output results back to the original format.

### Value

This outputs a new Nimblegen file having exactly the same filename except it will start with "S".

The new FE file will contain the intensity values, log ratio and flags from the ".temp" format input stucture(x).

NB. If the input filename starts with an "S" the file will be overwritten.

### Note

This function draws on methods contained within the "Jspline" class.

### Author(s)

Tomas William Fitzgerald

### Examples

```
# Read "Nimfile.txt", flag all data points and write out in Nimblgen format.  
# data <- nim.read("FEfile.txt", raw=TRUE)  
# data[,8] = 1  
# nim.write("FEfile.txt", data)
```

---

ProPlot

*Produce a simple overview plot of the data.*

---

### Description

Produce a figure showing the log2 ratio sorted by genomic location.

### Usage

```
ProPlot(x, pch=46, col="black", ylim=c(-3,3), xlim=c(0,length(x[,1])), xlab="Index", ylab="log2 ratio")
```

### Arguments

x	- ".temp" formatted data structure.
pch	- the point character to use defaults to 46.
col	- the point color to use, defaults to "black".
xlim	- the x-axis limits.
ylim	- the y-axis limits defaults to c(-3,3).
xlab	- the label for the x-axis.
ylab	- the label for the y-axis.
main	- the main title for the plot.

### Details

Provides an overall picture of array data.

### Value

Figure showing log2 ratio sorted by genomic location.

### Author(s)

Tomas William Fitzgerald

### Examples

```
chr = rnorm(1000,1,0)

for (x in 2:24) {
  chr = c(chr, rnorm(1000,x,0))
}

mat = matrix(ncol=7, nrow=length(chr))
mat[,1] = rnorm(length(chr),1000,150)
mat[,2] = rnorm(length(chr),800,180)
```

```

mat[,3] = chr
mat[,4] = seq(1,length(chr), 1)
mat[,5] = seq(1,length(chr), 1)
mat[,6] = 0
mat[,7] = 0

ProPlot(mat)

```

---

run.spline

*Function to process a single file.*


---

### Description

Function to process a single file.

### Usage

```

run.spline(file, format = "FE", raw=TRUE, robust = TRUE, offset=5, knots=1000, ntyp = "percentile",
p = 0.68, fact = 4.5, segN = FALSE, sn = 0.75, writeFE = FALSE, QC = FALSE, GFF = FALSE, PDF = FALSE)

```

### Arguments

file	- the file to process.
format	- the type of file format - c("FE", "NIM").
raw	- to use raw data or not (TRUE or FALSE).
robust	- make the method robust to outliers (TRUE or FALSE).
offset	- parameters to set the number of time the knot point will be offset.
knots	- the number of knot points to use for spline fitting.
ntyp	- type of noise calculation to perform - c("percentile", "derivative", "combined").
p	- value between 0 and 1 (the percentile value).
fact	- numeric value (the factor by which the noise value will be rised).
segN	- option for robust noise calculation (recommended when data contains many outliers e.g. cancer sample) TRUE or FALSE.
sn	- value between 0 and 1 - threshold for segmentation (segN).
writeFE	- Output a Feature Extraction formatted file (only to be used when processing Agilent FE files).
QC	- produce a QC report TRUE or FALSE.
GFF	- produce a GFF formatted file TRUE or FALSE.
PDF	- produce a PDF containing normalisation figures TRUE or FALSE.

**Details**

In most cases running this function using default parameters (i.e. just providing the file input argument) will perform consistently across a wide range of data qualities. If method optimisation is required the number of knot point and offsets can be adjusted along with the various noise parameters.

**Value**

Output file containing seven columns (".temp" format):

- Cy5 intensity values.
- Cy3 intensity values.
- Chromosome number.
- Start position.
- Stop position.
- An index.
- A flag.

**Author(s)**

Tomas William Fitzgerald

**Examples**

```
# norm <- run.spline("InputFile")  
# This would run the method using default settings.
```

---

segN

*Segmentation method.*

---

**Description**

Java method that performs segmentation using a 'RandomWalk' algorithm.

**Usage**

```
segN(ddd)
```

**Arguments**

ddd - vector of ratio values (sorted by genomic position).

**Details**

This method segments and assesses the difference between consecutive data points using a 'RandomWalk' approach.

**Value**

A vector of length (ddd) containing the segment medians.

**Note**

This method is static and has no parameters available for the user.  
NB. The full method will be available as an R package soon.

**Author(s)**

Tomas William Fitzgerald

**Examples**

```
v = seq(1,100000,0.5)
d = sin(2*pi/500 * v)
red = d + rnorm(length(d),0,100) + 1000
dd = sin(2*pi/1000 * v)
green = dd + rnorm(length(dd),0,120) + 1000
rat = log2(red / green) - median(log2(red / green), na.rm=TRUE)
rat[20000:30000] = abs(rat[20000:30000] * 2)
rat[60000:70000] = -abs(rat[60000:70000] * 2)
seg = segN(rat)
par(mfrow=c(2,1))
plot(rat, pch=46, ylim=c(-2,2), main="Before_segN")
plot(rat, pch=46, ylim=c(-2,2), main="After_segN")
matplot(seg, pch=46, ylim=c(-2,2), col='red', add=TRUE)
legend("bottomright", bty = "n", pch=20, c("Original data values", "Segment medians"), col=c("black", "red"))
```

---

SetJ

*Function to set the java class path and heap size.*

---

**Description**

Sets the java class path and adjusts the heap size.

**Usage**

```
SetJ(JavaHeapSize = "-Xmx1000m")
```

**Arguments**

JavaHeapSize - argument to adjust the heap size - defaults to 1000Mb.

**Note**

R seems to be limited at -Xmx1600, this allows FE files containing 1 million rows to be read (much more may start to cause problems).

This function needs to be run first to adjust the heap (heap size can only be set once within an R session).

NB. if either "fe.read", "fe.write" or "Jspline" are run before "SetJ()" the heap will be set to default (1000Mb).

**Author(s)**

Tomas William Fitzgerald

**Examples**

```
SetJ()
```

---

WriteGFF

*Function to write a GFF formatted file.*

---

**Description**

Output a GFF formatted file for use in the Signal-Map software (Nimblegen software).

**Usage**

```
WriteGFF(filename, x)
```

**Arguments**

filename - the name for the output file.  
x - a ".temp" formatted data structure.

**Details**

This will output a GFF formatted file containing the log2 ratio for all probes which were not flagged.

**Value**

GFF file for input into the Signal-Map software.

**Author(s)**

Tomas William Fitzgerald

**See Also**

<http://www.nimblegen.com/products/lit/signalmap1.9usersguide.pdf>

**Examples**

```
chr = rnorm(1000,1,0)

for (x in 2:24) {
  chr = c(chr, rnorm(1000,x,0))
}

mat = matrix(ncol=8, nrow=length(chr))
mat[,1] = log2(rnorm(length(chr),1000,150) / rnorm(length(chr),800,180))
mat[,2] = rnorm(length(chr),1000,150)
mat[,3] = rnorm(length(chr),800,180)
mat[,4] = chr
mat[,5] = seq(1,length(chr), 1)
mat[,6] = seq(1,length(chr), 1)
mat[,7] = 0
mat[,8] = 0

WriteGFF("test.gff", mat)
```

---

WriteQC

*Function to produce a QC report.*

---

**Description**

Write a QC report containing some summery statistics

**Usage**

```
WriteQC(filename, batch, x)
```

**Arguments**

filename       - the name for the output file.  
batch           - TRUE or FALSE (is it being run in batch mode).  
x               - a ".temp" formatted data stucture.

**Value**

Produces a table containing the filename, 68th percentile, dLRs and chromosme X median.

**Author(s)**

Tomas William Fitzgerald

**Examples**

```
chr = rnorm(1000,1,0)

for (x in 2:24) {
chr = c(chr, rnorm(1000,x,0))
}

mat = matrix(ncol=7, nrow=length(chr))
mat[,1] = rnorm(length(chr),1000,150)
mat[,2] = rnorm(length(chr),800,180)
mat[,3] = chr
mat[,4] = seq(1,length(chr), 1)
mat[,5] = seq(1,length(chr), 1)
mat[,6] = 0
mat[,7] = 0

WriteQC("test", batch=FALSE, mat)
```

# Index

## \*Topic **IO**

- batch.spline, 2
- fe.extract, 6
- fe.read, 7
- fe.write, 8
- MakePDF, 12
- nim.read, 15
- nim.write, 17
- run.spline, 19
- WriteGFF, 22
- WriteQC, 23

## \*Topic **aplot**

- GenomePlot, 9
- MAPlot1, 13
- MAPlot2, 14
- ProPlot, 18

## \*Topic **environment**

- SetJ, 21

## \*Topic **interface**

- batch.spline, 2
- fe.read, 7
- fe.write, 8
- Jspline, 10
- nim.write, 17
- run.spline, 19

## \*Topic **manip**

- batch.spline, 2
- Jspline, 10
- run.spline, 19
- segN, 20

## \*Topic **math**

- dLRs, 4

## \*Topic **optimize**

- f.Noise, 5

## \*Topic **package**

- aCGH.Spline-package, 2

aCGH.Spline (aCGH.Spline-package), 2

aCGH.Spline-package, 2

batch.spline, 2

dLRs, 4

f.Noise, 5

fe.extract, 6

fe.read, 7

fe.write, 8

GenomePlot, 9

Jspline, 10

MakePDF, 12

MAPlot1, 13

MAPlot2, 14

nim.read, 15

nim.write, 17

ProPlot, 18

run.spline, 19

segN, 20

SetJ, 21

WriteGFF, 22

WriteQC, 23