Preparing Affymetrix Data

GENEVA Coordinating Center
Department of Biostatistics
University of Washington

November 1, 2022

1 Overview

This vignette describes how to prepare Affymetrix data for use in GWASTools. After following the examples here for preparing the data, the analysis procedure is the same as presented in the GWAS Data Cleaning vignette. For a description of the file formats (NetCDF and GDS), see the Data Formats vignette.

2 Creating the SNP Annotation Data Object

All of the functions in GWASTools require a minimum set of variables in the SNP annotation data object. The minimum required variables are

- `snpID`, a unique integer identifier for each SNP
- `chromosome`, an integer mapping for each chromosome, with values 1-27, mapped in order from 1-22, 23=X, 24=XY (the pseudoautosomal region), 25=Y, 26=M (the mitochondrial probes), and 27=U (probes with unknown positions)
- `position`, the base position of each SNP on the chromosome.

We create the integer chromosome mapping for a few reasons. The chromosome is stored as an integer in the NetCDF or GDS files, so in order to link the SNP annotation with the NetCDF/GDS file, we use the integer values in the annotation as well. For convenience when using GWASTools functions, the chromosome variable is most times assumed to be an integer value. Thus, for the sex chromosomes, we can simply use the `chromosome` values. For presentation of results, it is important to have the mapping of the integer values back to the standard designations for the chromosome names, thus the `getChromosome()` functions in the GWASTools objects have a `char=TRUE` option to return the characters 1-22, X, XY, Y, M, U. The position variable should hold all numeric values of the physical position of a probe. *The SNP annotation file is assumed to list the probes in order of chromosome and position within chromosome.*

Note that for Affymetrix data, the rs ID is often not unique, as there may be multiple probes for a given SNP. The probe ID is usually the unique SNP identifier. Also, Affymetrix annotation generally has a separate column or columns to indicate pseudoautosomal regions; some manipulation is usually required to include this information within the chromosome column itself.

```
> library(GWASTools)
> library(GWASdata)
> # Load the SNP annotation (simple data frame)
> data(affy_snp_annot)
> # Create a SnpAnnotationDataFrame
> snpAnnot <- SnpAnnotationDataFrame(affy_snp_annot)
> # names of columns
> varLabels(snpAnnot)

[1] "snpID"      "chromosome" "position"   "rsID"       "probeID"

> # data
> head(pData(snpAnnot))

   snpID chromosome position      rsID       probeID
1 869828         21 13733610 rs3132407 SNP_A-8340403
2 869844         21 13852569 rs2775671 SNP_A-8340413
3 869864         21 14038583 rs2775018 SNP_A-8340427
4 869889         21 14136579 rs3115511 SNP_A-8340440
5 869922         21 14396024 rs2822404 SNP_A-8340775
6 869925         21 14404476 rs1556276 SNP_A-1968967

> # Add metadata to describe the columns
> meta <- varMetadata(snpAnnot)
> meta[c("snpID", "chromosome", "position", "rsID", "probeID"),
+   "labelDescription"] <- c("unique integer ID for SNPs",
+   paste("integer code for chromosome: 1:22=autosomes,",
+    "23=X, 24=pseudoautosomal, 25=Y, 26=Mitochondrial, 27=Unknown"),
+   "base pair position on chromosome (build 36)",
+   "RS identifier",
+   "unique ID from Affymetrix")
> varMetadata(snpAnnot) <- meta
```

3 Creating the Scan Annotation Data Object

The scan annotation file holds attributes for each genotyping scan that are relevant to genotypic data cleaning. These data include processing variables such as tissue type, DNA extraction method, and genotype processing batch. They also include individual characteristics such as gender and race. The initial sample annotation file is created from the raw data supplied by the genotyping center and/or study investigator, providing a mapping from the raw data file(s) for each sample scan to other sample information such as sex, coded as `M` and `F`, ethnicity, unique scan identifier, called `scanID`, and unique subject identifier. Since a single subject may have been genotyped multiple times as a quality control measure, it is important to distinguish between the scanID (unique genotyping instance) and subjectID (person providing a DNA sample).

```
> # Load the scan annotation (simple data frame)
> data(affy_scan_annot)
```

```
> # Create a ScanAnnotationDataFrame
> scanAnnot <- ScanAnnotationDataFrame(affy_scan_annot)
> # names of columns
> varLabels(scanAnnot)

 [1] "scanID"     "subjectID"  "family"     "father"     "mother"
 [6] "CoriellID"  "race"       "sex"        "status"     "genoRunID"
[11] "plate"      "alleleFile" "chpFile"

> # data
> head(pData(scanAnnot))

  scanID subjectID family father mother CoriellID race sex status
1      3 200150062     28      0      0   NA18912  YRI   F      0
2      5 200122600   1341      0      0   NA07034  CEU   M      1
3     14 200122151     58      0      0   NA19222  YRI   F      0
4     15 200033736      9      0      0   NA18508  YRI   F      1
5     17 200116780   1344      0      0   NA12056  CEU   M      1
6     28 200003216     28      0      0   NA18913  YRI   M      0
                                          genoRunID              plate
1 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_A03_31250 GAINmixHapMapAffy2
2 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_A05_31282 GAINmixHapMapAffy2
3 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_B02_31236 GAINmixHapMapAffy2
4 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_B03_31252 GAINmixHapMapAffy2
5 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_B05_31284 GAINmixHapMapAffy2
6 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_C04_31270 GAINmixHapMapAffy2
                                                                 alleleFile
1 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_A03_31250.BIRDSEED.ALLELE_SUMMARY.TXT
2 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_A05_31282.BIRDSEED.ALLELE_SUMMARY.TXT
3 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_B02_31236.BIRDSEED.ALLELE_SUMMARY.TXT
4 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_B03_31252.BIRDSEED.ALLELE_SUMMARY.TXT
5 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_B05_31284.BIRDSEED.ALLELE_SUMMARY.TXT
6 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_C04_31270.BIRDSEED.ALLELE_SUMMARY.TXT
                                                            chpFile
1 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_A03_31250.BIRDSEED.CHP.TXT
2 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_A05_31282.BIRDSEED.CHP.TXT
3 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_B02_31236.BIRDSEED.CHP.TXT
4 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_B03_31252.BIRDSEED.CHP.TXT
5 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_B05_31284.BIRDSEED.CHP.TXT
6 GIGAS_g_GAINmixHapMapAffy2_GenomeWideEx_6_C04_31270.BIRDSEED.CHP.TXT

> # Add metadata to describe the columns
> meta <- varMetadata(scanAnnot)
> meta[c("scanID", "subjectID", "family", "father", "mother",
+   "CoriellID", "race", "sex", "status", "genoRunID", "plate",
+   "alleleFile", "chpFile"), "labelDescription"] <-
```

```
+    c("unique integer ID for scans",
+   "subject identifier (may have multiple scans)",
+   "family identifier",
+   "father identifier as subjectID",
+   "mother identifier as subjectID",
+   "Coriell subject identifier",
+   "HapMap population group",
+   "sex coded as M=male and F=female",
+   "simulated case/control status" ,
+   "genotyping instance identifier",
+   "plate containing samples processed together for genotyping chemistry",
+   "data file with intensities",
+   "data file with genotypes and quality scores")
> varMetadata(scanAnnot) <- meta
```

4 Creating the Data Files

The data for genotype calls, allelic intensities and other variables such as BAlleleFrequency are stored as NetCDF or GDS files.

Genotype Files

The genotype files store genotypic data in 0, 1, 2 format indicating the number of "A" alleles in the genotype (i.e. AA=2, AB=1, BB=0 and missing=-1). The conversion from AB format and forward strand (or other) allele formats can be stored in the SNP annotation file.

Creating the Genotype file

```
> geno.file <- "tmp.geno.gds"
> # first 3 samples only
> scan.annotation <- affy_scan_annot[1:3, c("scanID", "genoRunID", "chpFile")]
> names(scan.annotation) <- c("scanID", "scanName", "file")
> # indicate which column of SNP annotation is referenced in data files
> snp.annotation <- affy_snp_annot[,c("snpID", "probeID", "chromosome", "position")]
> names(snp.annotation)[2] <- "snpName"
> # col.nums is an integer vector indicating which columns of the raw text file
> # contain variables for input
> col.nums <- as.integer(c(2,3))
> names(col.nums) <- c("snp", "geno")
> # Define a path to the raw data CHP text files
> path <- system.file("extdata", "affy_raw_data", package="GWASdata")
> diag.geno.file <- "diag.geno.RData"
> diag.geno <- createDataFile(path = path,
+   filename = geno.file,
+   file.type = "gds",
+   variables = "genotype",
```

```
+   snp.annotation = snp.annotation,
+   scan.annotation = scan.annotation,
+   sep.type = "\t",
+   skip.num = 1,
+   col.total = 6,
+   col.nums = col.nums,
+   scan.name.in.file = -1,
+   diagnostics.filename = diag.geno.file,
+   verbose = FALSE)
> # Look at the values included in the "diag.geno" object which holds
> #   all output from the function call
> names(diag.geno)

[1] "read.file"    "row.num"      "samples"      "sample.match" "missg"
[6] "snp.chk"      "chk"

> # `read.file' is a vector indicating whether (1) or not (0) each file
> #   specified in the `files' argument was read successfully
> table(diag.geno$read.file)

1
3

> # `row.num' is a vector of the number of rows read from each file
> table(diag.geno$row.num)

3300
   3

> # `sample.match' is a vector indicating whether (1) or not (0)
> #   the sample name inside the raw text file matches that in the
> #   sample annotation data.frame
> table(diag.geno$sample.match)

1
3

> # `snp.chk' is a vector indicating whether (1) or not (0)
> #   the raw text file has the expected set of SNP names
> table(diag.geno$snp.chk)

1
3

> # `chk' is a vector indicating whether (1) or not (0) all previous
> #   checks were successful and the data were written to the file
> table(diag.geno$chk)
```

```
1
3

> # open the file we just created
> (genofile <- GdsGenotypeReader(geno.file))

File: /private/var/folders/db/4tvgx8jx4z3fm1gzlnlzw9rc0000gq/T/RtmpygOePQ/Rbuild1c4f4979204a/GWA
+    [  ]
|--+ sample.id   { Int32 3 LZMA_ra(683.3%), 89B }
|--+ snp.id   { Int32 3300 LZMA_ra(25.0%), 3.2K }
|--+ snp.chromosome   { UInt8 3300 LZMA_ra(3.45%), 121B }
|--+ snp.position   { Int32 3300 LZMA_ra(68.0%), 8.8K }
|--+ snp.rs.id   { Str8 3300 LZMA_ra(20.5%), 9.3K }
\--+ genotype   { Bit2 3300x3, 2.4K } *

> # Take out genotype data for the first 3 samples and
> #   the first 5 SNPs
> (genos <- getGenotype(genofile, snp=c(1,5), scan=c(1,3)))

     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    2    2    2
[3,]    2    2    2
[4,]    1    0    2
[5,]    0    2    1

> close(genofile)
> # check that the data file matches the raw text files
> check.geno.file <- "check.geno.RData"
> check.geno <- checkGenotypeFile(path = path,
+   filename = geno.file,
+   file.type = "gds",
+   snp.annotation = snp.annotation,
+   scan.annotation = scan.annotation,
+   sep.type = "\t",
+   skip.num = 1,
+   col.total = 6,
+   col.nums = col.nums,
+   scan.name.in.file = -1,
+   check.scan.index = 1:3,
+   n.scans.loaded = 3,
+   diagnostics.filename = check.geno.file,
+   verbose = FALSE)
> # Look at the values included in the "check.geno" object which holds
> #   all output from the function call
> names(check.geno)
```

```
[1] "read.file"    "row.num"      "sample.names" "sample.match" "missg"
[6] "snp.chk"      "chk"          "geno.chk"
```

```
> # 'geno.chk' is a vector indicating whether (1) or not (0) the genotypes
> #   match the text file
> table(check.geno$geno.chk)
```

```
1
3
```

Intensity Files

The intensity files store quality scores and allelic intensity data for each SNP. Affymetrix data are provided in two files per genotyping scan. The CHP file holds the genotype calls, used to create the genotype file, as well as the confidence score, which is written to the quality file. The normalized X and Y intensity data are stored in the `allele_summary` files in the format of two rows per SNP, one for each allelic probe. These are written to the intensity file.

Creating the quality file

```
> qual.file <- "tmp.qual.gds"
> scan.annotation <- affy_scan_annot[1:3, c("scanID", "genoRunID", "chpFile")]
> names(scan.annotation) <- c("scanID", "scanName", "file")
> snp.annotation <- affy_snp_annot[,c("snpID", "probeID", "chromosome", "position")]
> names(snp.annotation)[2] <- "snpName"
> col.nums <- as.integer(c(2,4))
> names(col.nums) <- c("snp", "quality")
> path <- system.file("extdata", "affy_raw_data", package="GWASdata")
> diag.qual.file <- "diag.qual.RData"
> diag.qual <- createDataFile(path = path,
+   filename = qual.file,
+   file.type = "gds",
+   variables = "quality",
+   snp.annotation = snp.annotation,
+   scan.annotation = scan.annotation,
+   sep.type = "\t",
+   skip.num = 1,
+   col.total = 6,
+   col.nums = col.nums,
+   scan.name.in.file = -1,
+   diagnostics.filename = diag.qual.file,
+   verbose = FALSE)
> # Open the GDS file we just created
> (qualfile <- GdsIntensityReader(qual.file))
```

```
File: /private/var/folders/db/4tvgx8jx4z3fm1gzlnlzw9rc0000gq/T/RtmpygOePQ/Rbuild1c4f4979204a/GWA
+    [  ]
```

```
|--+ sample.id   { Int32 3 LZMA_ra(683.3%), 89B }
|--+ snp.id   { Int32 3300 LZMA_ra(25.0%), 3.2K }
|--+ snp.chromosome   { UInt8 3300 LZMA_ra(3.45%), 121B }
|--+ snp.position   { Int32 3300 LZMA_ra(68.0%), 8.8K }
|--+ snp.rs.id   { Str8 3300 LZMA_ra(20.5%), 9.3K }
\--+ quality   { Float32 3300x3 LZMA_ra(87.0%), 33.7K }
```

```
> # Take out the quality scores for the first
> #     5 SNPs for the first 3 samples
> (qual <- getQuality(qualfile, snp=c(1,5), scan=c(1,3)))
```

```
            [,1]        [,2]        [,3]
[1,] 0.014382409 0.002228203 0.004184095
[2,] 0.005246114 0.003796505 0.005314134
[3,] 0.002767287 0.017275169 0.005179304
[4,] 0.002980622 0.005857171 0.005547870
[5,] 0.006078269 0.005080181 0.002624712
```

```
> close(qualfile)
> # check
> check.qual.file <- "check.qual.RData"
> check.qual <- checkIntensityFile(path = path,
+   filename = qual.file,
+   file.type = "gds",
+   snp.annotation = snp.annotation,
+   scan.annotation = scan.annotation,
+   sep.type = "\t",
+   skip.num = 1,
+   col.total = 6,
+   col.nums = col.nums,
+   scan.name.in.file = -1,
+   check.scan.index = 1:3,
+   n.scans.loaded = 3,
+   affy.inten = FALSE,
+   diagnostics.filename = check.qual.file,
+   verbose = FALSE)
>
```

Creating the Intensity file

```
> xy.file <- "tmp.xy.gds"
> # we need the allele files instead of CHP files
> scan.annotation <- affy_scan_annot[1:3, c("scanID", "genoRunID", "alleleFile")]
> names(scan.annotation) <- c("scanID", "scanName", "file")
> snp.annotation <- affy_snp_annot[,c("snpID", "probeID", "chromosome", "position")]
> names(snp.annotation)[2] <- "snpName"
```

```
> diag.xy.file <- "diag.xy.RData"
> diag.xy <- createAffyIntensityFile(path = path,
+   filename = xy.file,
+   file.type = "gds",
+   snp.annotation = snp.annotation,
+   scan.annotation = scan.annotation,
+   diagnostics.filename = diag.xy.file,
+   verbose = FALSE)
> # Open the file we just created
> (intenfile <- GdsIntensityReader(xy.file))

File: /private/var/folders/db/4tvgx8jx4z3fm1gzlnlzw9rc0000gq/T/RtmpygOePQ/Rbuild1c4f4979204a/GWA
+    [  ]
|--+ sample.id   { Int32 3 LZMA_ra(683.3%), 89B }
|--+ snp.id   { Int32 3300 LZMA_ra(25.0%), 3.2K }
|--+ snp.chromosome   { UInt8 3300 LZMA_ra(3.45%), 121B }
|--+ snp.position   { Int32 3300 LZMA_ra(68.0%), 8.8K }
|--+ snp.rs.id   { Str8 3300 LZMA_ra(20.5%), 9.3K }
|--+ X   { Float32 3300x3 LZMA_ra(85.2%), 32.9K }
\--+ Y   { Float32 3300x3 LZMA_ra(83.8%), 32.4K }

> # Take out the normalized X intensity values for the first
> #     5 SNPs for the first 3 samples
> (xinten <- getX(intenfile, snp=c(1,5), scan=c(1,3)))

          [,1]      [,2]      [,3]
[1,]  501.2622  385.5622  356.8760
[2,]  614.1541  651.9782  710.6095
[3,] 1968.4933 2550.7141 2265.3674
[4,] 1607.2856  293.1671 2906.5942
[5,]  398.2835 1902.5592 1355.5342

> close(intenfile)
> # check the intensity files - note affy.inten=TRUE
> check.xy.file <- "check.xy.RData"
> check.xy <- checkIntensityFile(path = path,
+   filename = xy.file,
+   file.type = "gds",
+   snp.annotation = snp.annotation,
+   scan.annotation = scan.annotation,
+   sep.type = "\t",
+   skip.num = 1,
+   col.total = 2,
+   col.nums = setNames(as.integer(c(1,2,2)), c("snp", "X", "Y")),
+   scan.name.in.file = -1,
+   check.scan.index = 1:3,
```

```
+   n.scans.loaded = 3,
+   affy.inten = TRUE,
+   diagnostics.filename = check.xy.file,
+   verbose = FALSE)
```

BAlleleFrequency and LogRRatio Files

The BAlleleFrequency and LogRRatio file stores these values for every sample by SNP. For Affymetrix data, these values must be calculated by the user. For a thorough explanation and presentation of an application of these values, please refer to Peiffer, Daniel A., et al. (2006).[1]

For a given sample and SNP, R and θ are calculated using the X and Y intensities, where

$$R = X + Y \tag{1}$$

$$\theta = \frac{2\arctan(Y/X)}{\pi}$$

θ corresponds to the polar coordinate angle and R is the sum of the normalized X and Y intensities (not, as one might assume, the magnitude of the polar coordinate vector). It is from these values that we calculate the LogRRatio and BAlleleFrequency.

The LogRRatio is given below. The expected value of R is derived from a plot of θ versus R for a given SNP. It is the predicted value of R derived from a line connecting the centers of the two nearest genotype clusters.

$$\text{LogRRatio} = \log\left(\frac{R_{\text{observed values}}}{R_{\text{expected values}}}\right) \tag{2}$$

Variation in the LogRRatio across a single chromosome indicates possible duplication or deletion, and is an indication of overall sample quality.

The BAlleleFrequency is the frequency of the B allele in the population of cells from which the DNA is extracted. Each sample and SNP combination has a BAlleleFrequency value. Note the BAlleleFrequency values vary for a subject with each DNA extraction and tissue used. After all SNPs have been read and all samples have been clustered for a probe, the mean θ "cluster" value is calculated for each probe, for each of the three genotype clusters, resulting in θ_{AA}, θ_{AB} and θ_{BB} for every probe. Then the θ value for each sample, call it θ_n, is compared to θ_{AA}, θ_{AB} and θ_{BB}. The BAlleleFrequency is calculated

$$\text{BAlleleFrequency} = \begin{cases} 0 & \text{if } \theta_n < \theta_{AA} \\ \dfrac{(1/2)(\theta_n - \theta_{AA})}{\theta_{AB} - \theta_{AA}} & \text{if } \theta_{AA} \le \theta_n < \theta_{AB} \\ \dfrac{1}{2} + \dfrac{(1/2)(\theta_n - \theta_{AB})}{\theta_{BB} - \theta_{AB}} & \text{if } \theta_{AB} \le \theta_n < \theta_{BB} \\ 1 & \text{if } \theta_n \ge \theta_{BB} \end{cases}$$

[1] Peiffer, Daniel A., et al. High-resolution genomic profiling of chromosomal aberrations using Infinium whole-genome genotyping. *Genome Research* **16**, 1136-1148 (September 2006).

A θ_n value of 0 or 1 corresponds to a homozygote genotype for sample n at that particular probe, and a θ_n value of $1/2$ indicates a heterozygote genotype. Thus, $BAlleleFrequency \in [0, 1]$ for each probe. Across a chromosome, three bands are expected, one hovering around 0, one around 1 and one around 0.5, and any deviation from this is considered aberrant.

We use the BAlleleFrequency and LogRRatio values to detect mixed samples or samples of low quality, as well as chromosomal duplications and deletions. Samples that have a significantly large (partial or full chromosome) aberration for a particular chromosome as detected from the BAlleleFrequency values are recommended to be filtered out, for the genotype data are not reliable in these situations. Because of these applications, the BAlleleFrequency and LogRRatio values are a salient part of the data cleaning steps.

Because we have already completed the creation of both the genotype and intensity files, we simply use those files to access the data. The BAlleleFrequency and LogRRatio values are calculated in subsets for efficiency and written to the corresponding subset indices in the file.

Creating the BAlleleFrequency and LogRRatio file

We calculate the BAlleleFrequency and LogRRatio values for each sample by SNP and write these values to a data file by calling the function `BAFfromGenotypes`. We will also select "by.study" as the call method, so all 3 samples have their genotype clusters called together. In normal usage, we recommend calling Affymetrix genotypes "by.plate" (in which case the `plate.name` argument is passed to the function). For more detail regarding the `BAFfromGenotypes` function, please see the function documentation. After the function is complete, we will look at a few values to ensure the file was created successfully.

```
> bl.file <- "tmp.bl.gds"
> xyData <- IntensityData(GdsIntensityReader(xy.file),
+                         snpAnnot=snpAnnot)
> genoData <- GenotypeData(GdsGenotypeReader(geno.file),
+                          snpAnnot=snpAnnot)
> BAFfromGenotypes(xyData, genoData,
+                  filename = bl.file,
+                  file.type = "gds",
+                  min.n.genotypes = 0,
+                  call.method ="by.study",
+                  verbose = FALSE)
> close(xyData)
> close(genoData)
> # Open the file we just created
> (blfile <- GdsIntensityReader(bl.file))

File: /private/var/folders/db/4tvgx8jx4z3fm1gzlnlzw9rc0000gq/T/RtmpygOePQ/Rbuild1c4f4979204a/GWA
+    [  ]
|--+ sample.id   { Int32 3 LZMA_ra(683.3%), 89B }
|--+ snp.id   { Int32 3300 LZMA_ra(25.0%), 3.2K }
|--+ snp.chromosome   { UInt8 3300 LZMA_ra(3.45%), 121B }
|--+ snp.position   { Int32 3300 LZMA_ra(68.0%), 8.8K }
```

```
|--+ BAlleleFreq   { Float32 3300x3, 38.7K }
\--+ LogRRatio   { Float32 3300x3, 38.7K }

> # Look at the BAlleleFrequency values for the first 5 SNPs
> (baf <- getBAlleleFreq(blfile, snp=c(1,5), scan=c(1,3)))

     [,1] [,2] [,3]
[1,]  NaN    1  NaN
[2,]  0.0  NaN  0.0
[3,]  0.0  NaN  NaN
[4,]  0.5    1  0.0
[5,]  1.0    0  0.5

> close(blfile)
```