ScriptScriptScriptScriptScriptScriptScriptScriptScriptScriptScript

# Verifying sample relationships using genotypes or inferred genotypes

Maarten van Iterson1\* and Davy Cats1\*\*

1Department of Medical Statistics and Bioinformatics, Section Moleculair Epidemiology, Leiden University Medical Center

\*[mviterson@gmail.com](mailto:mviterson@gmail.com)  
\*\*[davycats.dc@gmail.com](mailto:davycats.dc@gmail.com)

#### 2017-10-29

#### Abstract

Analysis of multiple omics datatypes from the same individuals is becoming increasingly common. For example, several data repositories contain genetic, transcriptomic and epigenetic (DNA methylation) measurements on the same individuals, e.g. TCGA, Geuvadis, BBMRI/BIOS, etc. We have developed a tool that can verify the sample relationships between and across omics types. A small number of misspecified samples can destroy analyses for example assuming an analysis on unrelated individuals whereas parent offspring relations exist.

#### Package

omicsPrint 0.99.20

Here we illustrate the use of the package using artificially generated data and provide an example of usage with experimental data. A few other vignettes are available that show the use of the package on experimental data, i.e. 450k DNA methylation, RNAsequencing data and imputed genotypes.

# 1 Within omics sample relationship verification

## 1.1 Create toy data

Here we generate a single vector with 100 randomly drawn integers from the set; 1, 2, 3, representing 100 SNP calls from a single individual. Three other samples are generated from this one by randomly swapping a certain fraction of the SNPs. For example, swapping only 5 SNPs only introduces some noise but should still reflect the same individual. However, swapping 50% of the SNPs is similar to the difference in genotypes between parents and offspring. Swapping all SNPs will result in a unrelated individual.

swap <- function(x, frac=0.05) {

n <- length(x)

k <- floor(n\*frac)

x1 <- sample(1:n,k)

x2 <- sample(1:n,k) ##could be overlapping

x[x2] <- x[x1]

x

}

x1 <- 1 + rbinom(100, size=2, prob=1/3)

x2 <- swap(x1, 0.05) ##strongly related e.g. replicate

x3 <- swap(x1, 0.5) ##related e.g. parent off spring

x4 <- swap(x1, 1) ##unrelated

x <- cbind(x1, x2, x3, x4)

head(x)

## x1 x2 x3 x4

## [1,] 1 1 1 2

## [2,] 2 2 2 3

## [3,] 1 1 2 2

## [4,] 1 1 1 3

## [5,] 1 1 1 2

## [6,] 2 2 2 1

## 1.2 Running the allelesharing algorithm

We use Identity by State (IBS) for a set of SNPs to infer sample relations. See Abecasis et al. (2001), for the description of this approach applied to genetic data. Briefly, between each sample pair the identity by state vector is calculated, which is a measure of genetic distance between individuals. Next the vector is summarized by its mean and variance. A mean of 2 and variance of 0 indicates that the samples are identical.

library(omicsPrint)

data <- alleleSharing(x)

## Hash relations

## There are 0 SNP dropped because of low call rate!

## There are 0 sample set to NA because too little SNPs called!

## Using 100 polymorphic SNPs to determine allele sharing.

## Running `square` IBS algorithm!

## 5 of 10 (50%) ...

data

## mean var colnames.x colnames.y relation

## 1 2.00 0.00000000 x1 x1 identical

## 2 1.98 0.01979798 x2 x1 unrelated

## 3 1.70 0.23232323 x3 x1 unrelated

## 4 1.34 0.38828283 x4 x1 unrelated

## 5 2.00 0.00000000 x2 x2 identical

## 6 1.68 0.24000000 x3 x2 unrelated

## 7 1.34 0.38828283 x4 x2 unrelated

## 8 2.00 0.00000000 x3 x3 identical

## 9 1.34 0.36808081 x4 x3 unrelated

## 10 2.00 0.00000000 x4 x4 identical

By default no relations are assumed only the self-self relations. Genotype calls can contain NA’s, e.g. not measured in all samples or bad quality. Some pruning is performed to remove really bad cases.

The output is a data.frame containing all pairwise comparisons with the mean and variance of the IBS over the set of SNPs and the reported sample relationship, including the identifiers.

## 1.3 Report mismatches and provide graphical summary

Since, we provided a list of known relations, assuming the majority is correct, we can build a classifier to discover missclassified relations. The current implementation uses linear discriminant analysis and generates a confusion-matrix, graphically representation of the classification boundaries and outputs the missclassified sample pairs.

mismatches <- inferRelations(data)

## Assumed relation

## Predicted relation identical unrelated

## identical 4 1

## unrelated . 5

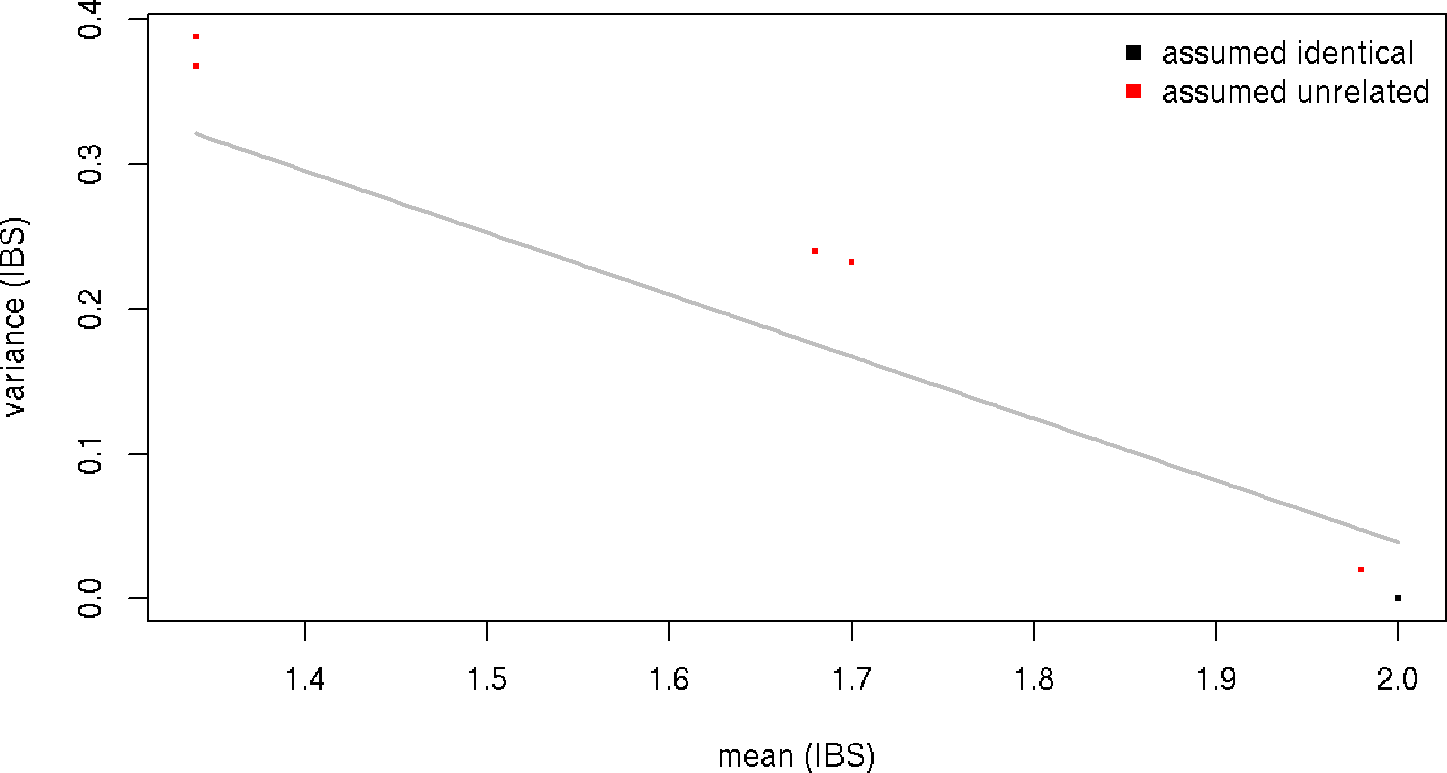


Figure 1: Scatter-plot of IBS mean and variance with classification boundary  
For pairwise comparison between the samples without specifying sample relationships.

mismatches

## mean var colnames.x colnames.y relation predicted

## 2 1.98 0.01979798 x2 x1 unrelated identical

Indeed, there is one misclassified sample, the replicate that we introduced. The true relationship with between x1 and x2 is an identical relation. Furthermore, we see two sample pairs with mean IBS of 1.7 and variance 0.2 which is an indication that also these pairs share a considerable number of alleles. In such, cases we should provide the known relations in a data.frame and specify the relation types.

relations <- expand.grid(idx = colnames(x), idy= colnames(x))

relations$relation\_type <- "unrelated"

relations$relation\_type[relations$idx == relations$idy] <- "identical"

relations$relation\_type[c(2,5)] <- "identical" ##replicate

relations$relation\_type[c(3,7,9,10)] <- "parent offspring"

relations

## idx idy relation\_type

## 1 x1 x1 identical

## 2 x2 x1 identical

## 3 x3 x1 parent offspring

## 4 x4 x1 unrelated

## 5 x1 x2 identical

## 6 x2 x2 identical

## 7 x3 x2 parent offspring

## 8 x4 x2 unrelated

## 9 x1 x3 parent offspring

## 10 x2 x3 parent offspring

## 11 x3 x3 identical

## 12 x4 x3 unrelated

## 13 x1 x4 unrelated

## 14 x2 x4 unrelated

## 15 x3 x4 unrelated

## 16 x4 x4 identical

Rerun the allelesharing algorithm now provided with the known relations.

data <- alleleSharing(x, relations=relations)

## Hash relations

## There are 0 SNP dropped because of low call rate!

## There are 0 sample set to NA because too little SNPs called!

## Using 100 polymorphic SNPs to determine allele sharing.

## Running `square` IBS algorithm!

## 5 of 10 (50%) ...

data

## mean var colnames.x colnames.y relation

## 1 2.00 0.00000000 x1 x1 identical

## 2 1.98 0.01979798 x2 x1 identical

## 3 1.70 0.23232323 x3 x1 parent offspring

## 4 1.34 0.38828283 x4 x1 unrelated

## 5 2.00 0.00000000 x2 x2 identical

## 6 1.68 0.24000000 x3 x2 parent offspring

## 7 1.34 0.38828283 x4 x2 unrelated

## 8 2.00 0.00000000 x3 x3 identical

## 9 1.34 0.36808081 x4 x3 unrelated

## 10 2.00 0.00000000 x4 x4 identical

mismatches <- inferRelations(data)

## Assumed relation

## Predicted relation identical parent offspring unrelated

## identical 5 . .

## parent offspring . 2 .

## unrelated . . 3

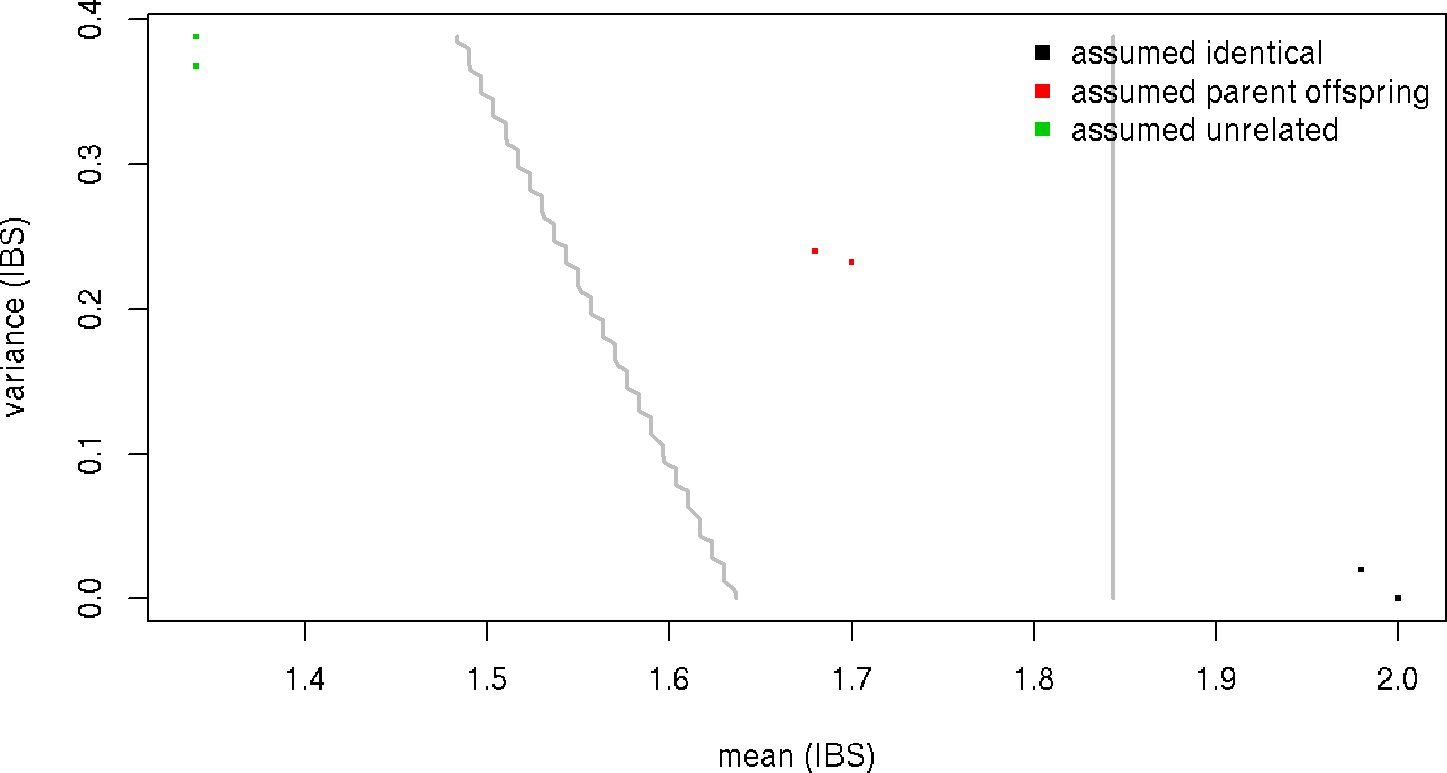


Figure 2: Scatter-plot of IBS mean and variance with classification boundaries  
For pairwise comparison between the samples with specifying sample relationships.

mismatches

## [1] mean var colnames.x colnames.y relation predicted

## <0 rows> (or 0-length row.names)

Now there are no misclassified sample relations.

# 2 Across omics sample relationship verification

The previous example showed how to perform sample relationship verification within an omics data type. If a second set of overlapping SNPs is available, obtained from a different omic type, for the same or related individuals sample relationshi verification can be run between these two.

rownames(x) <- paste0("rs", 1:100)

y <- x[sample(1:100, 80),]

data <- alleleSharing(x, y)

## Hash relations

## There are 0 SNP dropped because of low call rate!

## There are 0 sample set to NA because too little SNPs called!

## There are 0 SNP dropped because of low call rate!

## There are 0 sample set to NA because too little SNPs called!

## Using 80 polymophic SNPs to determine allele sharing.

## Running `rectangular` IBS algorithm!

## 4 of 16 (25%) ...

data

## mean var colnames.x colnames.y relation

## 1 2.0000 0.00000000 x1 x1 identical

## 2 1.9750 0.02468354 x2 x1 unrelated

## 3 1.7125 0.23275316 x3 x1 unrelated

## 4 1.3500 0.38227848 x4 x1 unrelated

## 5 1.9750 0.02468354 x1 x2 unrelated

## 6 2.0000 0.00000000 x2 x2 identical

## 7 1.6875 0.24287975 x3 x2 unrelated

## 8 1.3500 0.38227848 x4 x2 unrelated

## 9 1.7125 0.23275316 x1 x3 unrelated

## 10 1.6875 0.24287975 x2 x3 unrelated

## 11 2.0000 0.00000000 x3 x3 identical

## 12 1.3625 0.36060127 x4 x3 unrelated

## 13 1.3500 0.38227848 x1 x4 unrelated

## 14 1.3500 0.38227848 x2 x4 unrelated

## 15 1.3625 0.36060127 x3 x4 unrelated

## 16 2.0000 0.00000000 x4 x4 identical

mismatches <- inferRelations(data)

## Assumed relation

## Predicted relation identical unrelated

## identical 4 2

## unrelated . 10

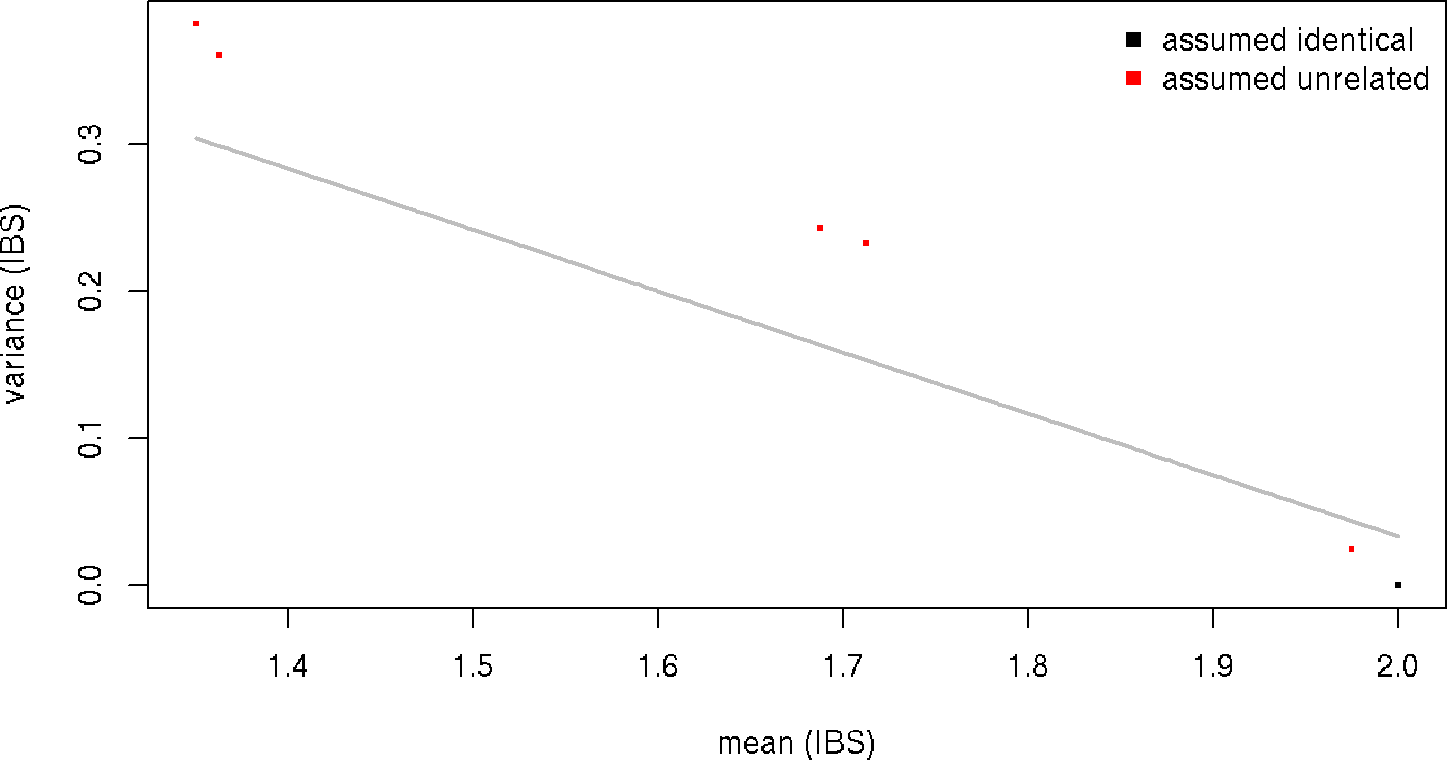


Figure 3: Scatter-plot of IBS mean and variance with classification boundary  
For pairwise comparison between the samples without specifying sample relationships.

mismatches

## mean var colnames.x colnames.y relation predicted

## 2 1.975 0.02468354 x2 x1 unrelated identical

## 5 1.975 0.02468354 x1 x2 unrelated identical

Notice pruning is performed on both data types and automatically the overlapping set of SNPs (80) is used, provided that the rownames between x and y are the same (Note also this holds for providing relations where the relation identifiers idx and idy should match with the columnames of x and y).

data <- alleleSharing(x, y, relations)

## Hash relations

## There are 0 SNP dropped because of low call rate!

## There are 0 sample set to NA because too little SNPs called!

## There are 0 SNP dropped because of low call rate!

## There are 0 sample set to NA because too little SNPs called!

## Using 80 polymophic SNPs to determine allele sharing.

## Running `rectangular` IBS algorithm!

## 4 of 16 (25%) ...

data

## mean var colnames.x colnames.y relation

## 1 2.0000 0.00000000 x1 x1 identical

## 2 1.9750 0.02468354 x2 x1 identical

## 3 1.7125 0.23275316 x3 x1 parent offspring

## 4 1.3500 0.38227848 x4 x1 unrelated

## 5 1.9750 0.02468354 x1 x2 identical

## 6 2.0000 0.00000000 x2 x2 identical

## 7 1.6875 0.24287975 x3 x2 parent offspring

## 8 1.3500 0.38227848 x4 x2 unrelated

## 9 1.7125 0.23275316 x1 x3 parent offspring

## 10 1.6875 0.24287975 x2 x3 parent offspring

## 11 2.0000 0.00000000 x3 x3 identical

## 12 1.3625 0.36060127 x4 x3 unrelated

## 13 1.3500 0.38227848 x1 x4 unrelated

## 14 1.3500 0.38227848 x2 x4 unrelated

## 15 1.3625 0.36060127 x3 x4 unrelated

## 16 2.0000 0.00000000 x4 x4 identical

mismatches <- inferRelations(data)

## Assumed relation

## Predicted relation identical parent offspring unrelated

## identical 6 . .

## parent offspring . 4 .

## unrelated . . 6

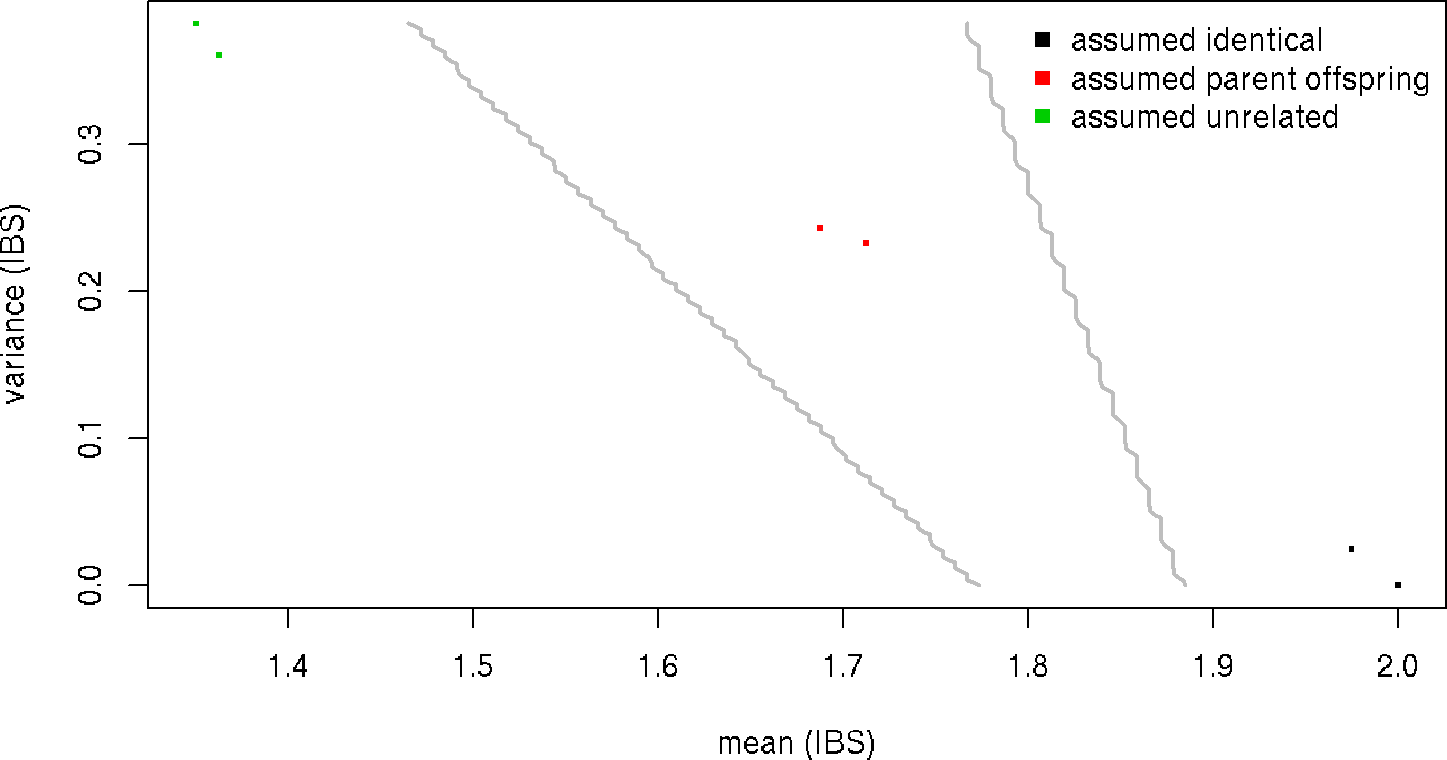


Figure 4: Scatter-plot of IBS mean and variance with classification boundaries  
For pairwise comparison between the samples with specifying sample relationships.

mismatches

## [1] mean var colnames.x colnames.y relation predicted

## <0 rows> (or 0-length row.names)

Again providing the known and true relationships yield no missclassified sample relationships.

# 3 An example using real world methylation data from a SummarizedExperiment

Here we will show how you could varify sample relationships on publicly available DNA methylation data. The dataset used here contains pairs of monozygotic twins. We will extract the beta-value matrix from GEO [GSE100940](https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE100940), [paper in press](http://www.sciencedirect.com/science/article/pii/S1875176817300872).

First we extract the data from GEO using the [GEOquery](http://bioconductor.org/packages/GEOquery/)-package and convert the returned object into a [SummarizedExperiment](http://bioconductor.org/packages/SummarizedExperiment/):

library(GEOquery)

library(SummarizedExperiment)

gset <- getGEO("GSE100940", GSEMatrix=TRUE)

## https://ftp.ncbi.nlm.nih.gov/geo/series/GSE100nnn/GSE100940/matrix/

## OK

## Found 1 file(s)

## GSE100940\_series\_matrix.txt.gz

## File stored at:

## /tmp/RtmpvP2MPj/GPL13534.soft

## Warning in read.table(file = file, header = header, sep = sep, quote =

## quote, : not all columns named in 'colClasses' exist

gset[[1]]

## ExpressionSet (storageMode: lockedEnvironment)

## assayData: 485572 features, 24 samples

## element names: exprs

## protocolData: none

## phenoData

## sampleNames: X0.78198 X0.7367728 ... X0.8701804 (24 total)

## varLabels: title geo\_accession ... data\_row\_count (32 total)

## varMetadata: labelDescription

## featureData

## featureNames: cg00000289 cg00000292 ... rs9839873 (485572 total)

## fvarLabels: ID Name ... SPOT\_ID (37 total)

## fvarMetadata: Column Description labelDescription

## experimentData: use 'experimentData(object)'

## Annotation: GPL13534

se <- makeSummarizedExperimentFromExpressionSet(gset[[1]])

se

## class: RangedSummarizedExperiment

## dim: 485572 24

## metadata(3): experimentData annotation protocolData

## assays(1): exprs

## rownames(485572): cg00000289 cg00000292 ... rs966367 rs9839873

## rowData names(37): ID Name ... RANGE\_GB SPOT\_ID

## colnames(24): X0.78198 X0.7367728 ... X0.8016921 X0.8701804

## colData names(32): title geo\_accession ... supplementary\_file.1

## data\_row\_count

Sample data can be extracted from the SummarizedExperiment-object using the colData-function and we can see which pair of twins each sample belongs to through the source\_name\_ch1 field. Using this knowledge we can construct a table of expected relationships:

r <- expand.grid(idx=colnames(se), idy=colnames(se))

r$Xpair <- sapply(strsplit(as.character(colData(se)[r$idx, "source\_name\_ch1"]),

split = "\_"), head, 1)

r$Ypair <- sapply(strsplit(as.character(colData(se)[r$idy, "source\_name\_ch1"]),

split = "\_"), head, 1)

r$relation\_type <- "unrelated"

r$relation\_type[r$Xpair == r$Ypair] <- "twin"

r$relation\_type[r$idx == r$idy] <- "identical"

head(r)

## idx idy Xpair Ypair relation\_type

## 1 X0.78198 X0.78198 M01 M01 identical

## 2 X0.7367728 X0.78198 M01 M01 twin

## 3 X0.7796773 X0.78198 M02 M01 unrelated

## 4 X0.746217 X0.78198 M02 M01 unrelated

## 5 X0.7572916 X0.78198 M03 M01 unrelated

## 6 X0.791267 X0.78198 M03 M01 unrelated

Several probes on the array contain SNPs occurring frequently in different populations(Chen et al. 2013; Zhou, Laird, and Shen 2016). We can use these to verify the expected relationships. We have made these data available from inside of this package.

Now we make a selection of CpGs probably affected by polymorphic SNPS in populations from East Asian, as these samples are from South Korea:

data(hm450.manifest.pop.GoNL)

cpgs <- names(hm450.manifest.pop.GoNL[

mcols(hm450.manifest.pop.GoNL)$MASK.snp5.EAS])

se <- se[cpgs,]

Next the beta-values are converted to genotypes using our enhanced K-means algorithm:

dnamCalls <- beta2genotype(se, assayName = "exprs")

dim(dnamCalls)

## [1] 821 24

dnamCalls[1:5, 1:5]

## X0.78198 X0.7367728 X0.7796773 X0.746217 X0.7572916

## cg09762182 3 3 1 1 3

## cg25282454 3 3 2 2 2

## cg24345856 3 3 2 2 2

## cg13167158 3 3 3 3 1

## cg12213037 2 2 3 3 3

The DNA methylation based genotype calls can be directly supplied to the allelesharing algorithm to perform the intra-omic sample matching:

data <- alleleSharing(dnamCalls, relations = r, verbose = FALSE)

mismatches <- inferRelations(data)

## Assumed relation

## Predicted relation identical twin unrelated

## identical 24 12 .

## twin . . .

## unrelated . . 264

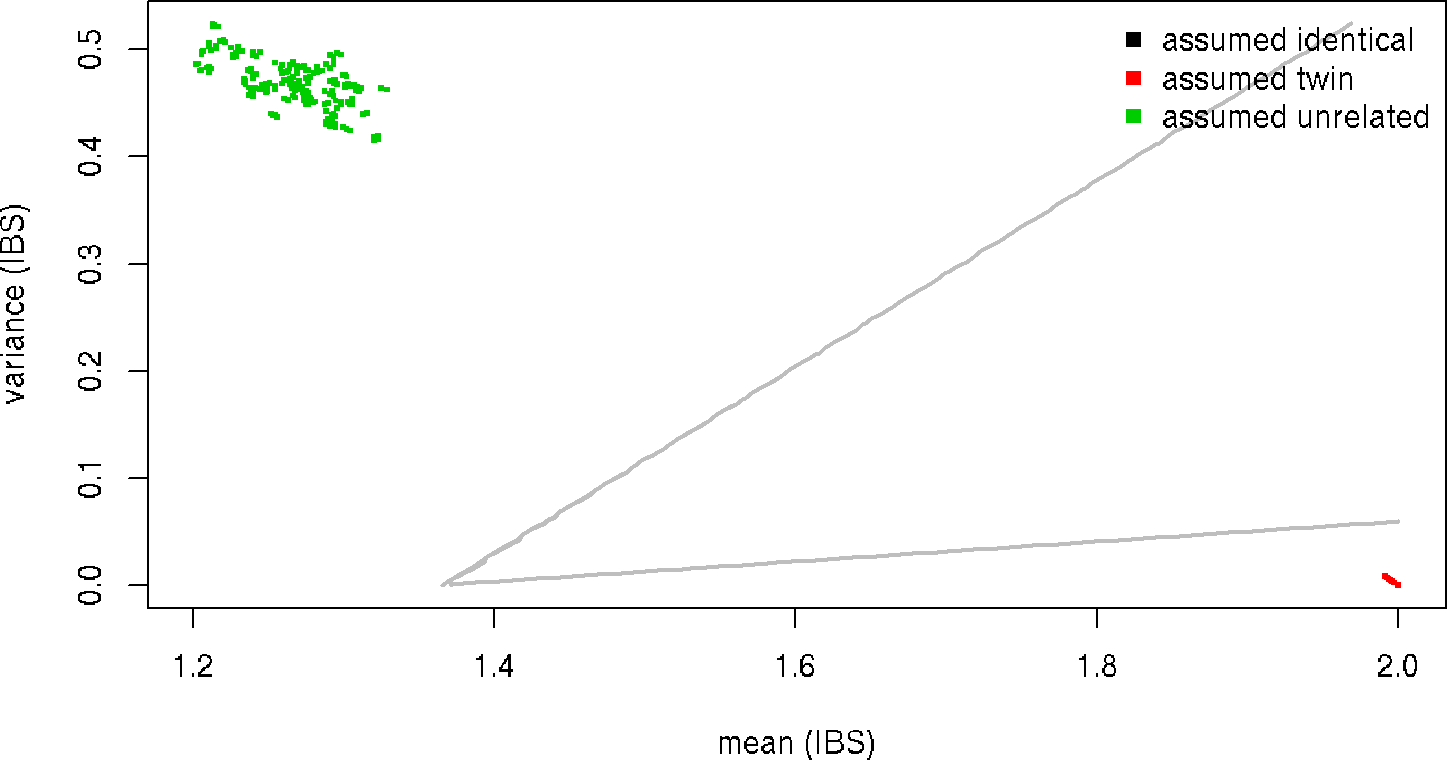


Figure 5: Scatter-plot of IBS mean and variance with classification boundaries  
For pairwise comparison between samples consiting of pairs of monozygotic twins.

mismatches

## mean var colnames.x colnames.y relation predicted

## 2 1.995128 0.004854282 X0.7367728 X0.78198 twin identical

## 49 1.997564 0.002433083 X0.746217 X0.7796773 twin identical

## 92 1.991474 0.008463801 X0.791267 X0.7572916 twin identical

## 131 1.995128 0.004854282 X0.7827371 X0.7656295 twin identical

## 166 1.995128 0.004854282 X0.7877404 X0.766961 twin identical

## 197 1.995128 0.004854282 X0.8374685 X0.8321307 twin identical

## 224 1.996346 0.003645168 X0.7442845 X0.7399069 twin identical

## 247 2.000000 0.000000000 X0.7982959 X0.8037069 twin identical

## 266 1.995128 0.004854282 X0.833827 X0.775218 twin identical

## 281 1.992692 0.007263599 X0.7498735 X0.793403 twin identical

## 292 1.993910 0.006060426 X0.7657012 X0.7426056 twin identical

## 299 1.993910 0.006060426 X0.8701804 X0.8016921 twin identical

The twins are predicted as being identical to each other. This is not unexpected as they are monozygotic.

# 4 SessionInfo

sessionInfo()

## R Under development (unstable) (2017-09-30 r73418)

## Platform: x86\_64-pc-linux-gnu (64-bit)

## Running under: Ubuntu 16.04.3 LTS

##

## Matrix products: default

## BLAS: /usr/local/lib64/R/lib/libRblas.so

## LAPACK: /usr/local/lib64/R/lib/libRlapack.so

##

## locale:

## [1] LC\_CTYPE=en\_US.UTF-8 LC\_NUMERIC=C

## [3] LC\_TIME=nl\_NL.UTF-8 LC\_COLLATE=en\_US.UTF-8

## [5] LC\_MONETARY=nl\_NL.UTF-8 LC\_MESSAGES=en\_US.UTF-8

## [7] LC\_PAPER=nl\_NL.UTF-8 LC\_NAME=C

## [9] LC\_ADDRESS=C LC\_TELEPHONE=C

## [11] LC\_MEASUREMENT=nl\_NL.UTF-8 LC\_IDENTIFICATION=C

##

## attached base packages:

## [1] stats4 parallel stats graphics grDevices utils datasets

## [8] methods base

##

## other attached packages:

## [1] SummarizedExperiment\_1.6.5 DelayedArray\_0.2.7

## [3] matrixStats\_0.52.2 GenomicRanges\_1.28.6

## [5] GenomeInfoDb\_1.12.3 IRanges\_2.10.5

## [7] S4Vectors\_0.14.7 GEOquery\_2.42.0

## [9] Biobase\_2.36.2 BiocGenerics\_0.22.1

## [11] omicsPrint\_0.99.20 MASS\_7.3-47

## [13] BiocStyle\_2.4.1

##

## loaded via a namespace (and not attached):

## [1] reshape2\_1.4.2 purrr\_0.2.3

## [3] lattice\_0.20-35 colorspace\_1.3-2

## [5] htmltools\_0.3.6 yaml\_2.1.14

## [7] XML\_3.98-1.9 rlang\_0.1.2

## [9] glue\_1.1.1 GenomeInfoDbData\_0.99.0

## [11] plyr\_1.8.4 stringr\_1.2.0

## [13] zlibbioc\_1.22.0 munsell\_0.4.3

## [15] gtable\_0.2.0 evaluate\_0.10.1

## [17] knitr\_1.17 UpSetR\_1.3.3

## [19] httpuv\_1.3.5 MultiAssayExperiment\_1.2.1

## [21] highr\_0.6 Rcpp\_0.12.13

## [23] xtable\_1.8-2 backports\_1.1.1

## [25] scales\_0.5.0 RaggedExperiment\_1.0.0

## [27] XVector\_0.16.0 mime\_0.5

## [29] gridExtra\_2.3 ggplot2\_2.2.1

## [31] digest\_0.6.12 stringi\_1.1.5

## [33] bookdown\_0.5 shiny\_1.0.5

## [35] grid\_3.5.0 rprojroot\_1.2

## [37] tools\_3.5.0 bitops\_1.0-6

## [39] magrittr\_1.5 RCurl\_1.95-4.8

## [41] lazyeval\_0.2.0 tibble\_1.3.4

## [43] tidyr\_0.7.1 Matrix\_1.2-11

## [45] shinydashboard\_0.6.1 httr\_1.3.1

## [47] rmarkdown\_1.6 R6\_2.2.2

## [49] compiler\_3.5.0

# Reference

Abecasis, G. R., S. S. Cherny, W. O. Cookson, and L. R. Cardon. 2001. “GRR: graphical representation of relationship errors.” Bioinformatics 17 (8): 742–43.

Chen, Y. A., M. Lemire, S. Choufani, D. T. Butcher, D. Grafodatskaya, B. W. Zanke, S. Gallinger, T. J. Hudson, and R. Weksberg. 2013. “Discovery of cross-reactive probes and polymorphic CpGs in the Illumina Infinium HumanMethylation450 microarray.” Epigenetics 8 (2): 203–9.

Zhou, W., P. W. Laird, and H. Shen. 2016. “Comprehensive characterization, annotation and innovative use of Infinium DNA methylation BeadChip probes.” Nucleic Acids Res., Oct.

ScriptScriptScript