

# Package ‘omicsPrint’

February 1, 2025

**Title** Cross omic genetic fingerprinting

**Version** 1.27.0

**Description** omicsPrint provides functionality for cross omic genetic fingerprinting, for example, to verify sample relationships between multiple omics data types, i.e. genomic, transcriptomic and epigenetic (DNA methylation).

**Depends** R ( $\geq 3.5$ ), MASS

**Imports** methods, matrixStats, graphics, stats, SummarizedExperiment, MultiAssayExperiment, RaggedExperiment

**Suggests** BiocStyle, knitr, rmarkdown, testthat, GEOquery, VariantAnnotation, Rsamtools, BiocParallel, GenomicRanges, FDb.InfiniumMethylation.hg19, snpStats

**License** GPL ( $\geq 2$ )

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**biocViews** QualityControl, Genetics, Epigenetics, Transcriptomics, DNAMethylation, Transcription, GeneticVariability, ImmunoOncology

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/omicsPrint>

**git\_branch** devel

**git\_last\_commit** 8c45530

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2025-01-31

**Author** Maarten van Iterson [aut],  
Davy Cats [cre]

**Maintainer** Davy Cats <davycats.dc@gmail.com>

## Contents

alleleSharing . . . . .	2
beta2genotype . . . . .	3
hg19.GoNLsnps . . . . .	4
hm450.manifest.pop.GoNL . . . . .	6
inferRelations . . . . .	8

<b>Index</b>	<b>9</b>
--------------	----------

---

alleleSharing	<i>allele sharing based on ibs</i>
---------------	------------------------------------

---

## Description

Run the allele sharing algorithm based on ibs

## Usage

```
alleleSharing(x, y = NULL, relations = NULL, idx.col = "idx",
  idy.col = "idy", rel.col = "relation_type", callRate = 0.95,
  coverageRate = 2/3, alpha = 0, maf = 0, alignment = FALSE,
  assayNameX = NULL, assayNameY = NULL, verbose = TRUE)
```

## Arguments

x, y	genotype matrix with row and column names; if this is a SummarizedExperiment or a MultiAssayExperiment assayName must also be specified
relations	'data.frame' with relations and their mapping identifiers, provide columns if different from the default and beware identifiers should match with colnames of x and y
idx.col	column name containing mapping identifiers
idy.col	column name containing mapping identifiers
rel.col	column name containing the relations, i.e. identical, parentoffspring, etc.
callRate	default 0.95 SNPs that are called in less then the threshold are dropped
coverageRate	default 2/3 samples with less then threshold SNPs called are set to NA
alpha	significance level for Hardy-Weinberg test default alpha = 0, no filtering, internally Bonferonni multiple testing will be applied
maf	minor allele frequency threshold, variants with lower frequency (default 0 no filtering) will be dropped
alignment	default FALSE
assayNameX	the name of the assay to be used for x (see x, y)
assayNameY	same as assayNameX, but for y; if y is not specified, the assay will be retrieved from x
verbose	show progress default TRUE

**Details**

calculate mean and variance of identity by state between genotypes, coded as 1,2,3, of all sample pairs either give one omic inferred set of SNPs or two from different omic types.

'Strand Alignment' is required if methylation data inferred genotypes are compared with DNA based genotypes, i.e., DNA based genotype is 3 whereas methylation is 1. The strand alignment step will fix this.

Notice that there are two algorithms for calculating allele sharing. One for the case one matrix is provided and one for the case two, x and y, are provided. If one is provided the algorithm takes in account the symmetric relations between pairs i.e.  $x_{12} = x_{21}$  etc.

To improve the lookup of relations, which can be millions of say 1000 samples are provided, a hash-table is created from the provided data.frame with relations.

**Value**

data.frame with mean and variance ibs between all pairs

**Author(s)**

mvaniterson

**Examples**

```
set.seed(12345)
beta <- matrix(runif(100*10, 0,1), nrow=100)
beta[1:5, 1:5]
colnames(beta) <- paste0("sample", 1:10)
genotype <- beta2genotype(beta)
genotype[1:5, 1:5]
data <- alleleSharing(genotype)
head(data)
```

---

beta2genotype	<i>converts beta-values to genotypes (1, 2 and 3)</i>
---------------	---

---

**Description**

convert DNA methylation beta-value to inferred genotypes

**Usage**

```
beta2genotype(betas, na.rm = TRUE, minSep = 0.25, minSize = 5,
  centers = c(0.2, 0.5, 0.8), assayName = NULL)
```

**Arguments**

betas	beta matrix of probes possibly affected SNPs; if this is a SummarizedExperiment or a MultiAssayExperiment assayName must also be specified
na.rm	TRUE drop cpg for which no clustering was observed
minSep	minimal separation between clusters
minSize	size of smallest cluster (in percentage)
centers	center of clusters, defaults to 0.2, 0.5, 0.8.
assayName	the name of the assay to be used (see betas)

**Details**

Using kmeans unsupervised clustering to infer genotypes based on idea's from Leonard Schalkwyk; watermelon packages.

'minSep' and 'minSize' ensure good clusters are found. This function is similar to the gaphunter approach implemented in minfi.

**Value**

matrix with genotypes

**Author(s)**

mvaniterson

**Examples**

```
set.seed(12345)
beta <- matrix(runif(100*10, 0,1), nrow=100)
beta[1:5, 1:5]
genotype <- beta2genotype(beta)
genotype[1:5, 1:5]
```

---

hg19.GoNLsnps

*Dataframe with overlaps GoNL variants and 450K probes*


---

**Description**

Dataframe containing all SNPs and short INDELS from GoNLv5 that overlap with 450K probes. This release does not include X and Y chromosomes, so only information for autosomal probes is available. For each overlap there is an unique row. Consequently, some probes are duplicated (probes that overlap with multiple variants) and some variants are duplicated (some variants overlap with more than one probe).

**Usage**

```
data(hg19.GoNLsnps)
```

**Format**

A data frame with 207866 rows and 19 variables:

**CHROM** chromosome, X and Y chromosomes are not available, since they are not included in this GoNL release

**probe** probe ID

**type** Infinium probedesign

**strand** orientation of the probe

**probeType** whether the probe measures a CpG-site (cg) or a non-CpG site (ch)

**location\_c** Location of the queried 'C' of the CpG dinucleotide. Note that this is the location of the C that is actually measured. For probes that interrogate the reverse strand (plus-strand probes) this is one base downstream of the C nucleotide on the forward strand

**location\_g** Location of the G nucleotide of the CpG dinucleotide. Note that this is the location of the queried G. For probes that interrogate the reverse strand (plus-strand probes) this is one base upstream of the G nucleotide on the forward strand

**ID** SNP ID

**snpBeg** Start coordinate of the variant. Identical to snpEnd for SNPs.

**snpEnd** End coordinate of the variant. Identical to snpBeg for SNPs

**AF** Allele frequency of alternative allele

**REF** Reference allele

**ALT** Alternative allele

**FILTER** Filter information from GoNL.

**MAF** Minor allele frequency

**variantType** SNP or INDEL

**distance\_3end** Distance between SNP and 3'end of the probe. For type I probes the 3'end of the probe coincides with the queried C nucleotide. For type II probes the 3'end of the probe coincides with the G nucleotide directly after the C nucleotide.

**distance\_c** Distance from queried C nucleotide. A distance of -1 indicates that the SNPs overlaps the SBE-position for type I probes.

**channel\_switch** Indicates whether a variant in the SBE-location of type I probes causes a color-channel-switch or overlap with an INDEL. For plus-strand probes C/T, C/A and C/G SNPs are expected to cause a color-channel switch. For min-strand probes A/G, G/T and C/G SNPs are expected to cause a color-channel switch.

**Source**

<http://zwdzwd.github.io/InfiniumAnnotation>

[https://molgenis26.target.rug.nl/downloads/gonl\\_public/variants/release5/](https://molgenis26.target.rug.nl/downloads/gonl_public/variants/release5/)

## Examples

```
data(hg19.GoNLsnps)

# Select variants that overlap with queried C nucleotide
snps_c <- hg19.GoNLsnps[hg19.GoNLsnps$distance_c == 0, ]

# Select all INDELS
indels <- hg19.GoNLsnps[hg19.GoNLsnps$variantType == "INDEL",]

# Select SNPs that cause a channel-switch
channel_switch <- hg19.GoNLsnps[!is.na(hg19.GoNLsnps$channel_switch)
& hg19.GoNLsnps$channel_switch == "Yes",]
```

---

hm450.manifest.pop.GoNL

*HM450 population-specific probe-masking recommendations*

---

## Description

Adapted version of the annotation file provided by Zhou et al. (see source, Mar-13-2017 release). This annotation file contains population-specific probe-masking recommendations based on SNPs within 5 bases from the 3' end of the probe, mapping issues, non-unique 3' 30bp subsequence and channel-switching SNPs in the single-base-extension for type I probes. We added population-specific masking recommendations for the Dutch population using GoNL release 5. This release does not include X and Y chromosomes, so for the Dutch population, only masking information for the autosomal probes is available.

## Usage

```
data(hm450.manifest.pop.GoNL)
```

## Format

A GRanges object with 485577 ranges and 65 metadata columns:

**MASK.general.pop** Recommended general purpose masking merged from "MASK.sub30.copy", "MASK.mapping" (in either the hg38 or hg19 genome), "MASK.extBase", "MASK.typeINextBaseSwitch" and "MASK.snp5.pop" from the "hm450.manifest" file and the "hm450.manifest.pop" file (see source). For GoNL, "MASK.typeINextBaseSwitchandINDEL.GoNL" is used instead of "MASK.typeINextBaseSwitch"

**MASK.snp5.pop** Whether the 5bp 3'-subsequence (including extension for type II) overlap with a SNP with population-specific AF > 0.01

**MASK.typeINextBaseSwitchandINDEL.GoNL** SNPs (that cause a color-channel switch) and INDELS with AF > 0.01 in GoNL. In contrast, "MASK.typeINextBaseSwitch" column is based on all SNPs in 1000 genomes and dbSNP, regardless of population or allele frequency

## Details

Note: Zhou et al. identified several probes that match to a different location than annotated in the original Illumina manifest file. The authors have used the 'updated' location in their annotation file. Therefore, a handful of probes in this annotation file are annotated to a different location than the original Illumina manifest file. For the identification of the overlaps we used the locations as annotated in the original Illumina file. Therefore, a few of the identified overlaps do not match the locations as specified in this annotation file. This also explains why GoNL masking information is available for a couple of probes that are located in the X- and Y-chromosome in this annotation: these probes map to autosomal probes in the original Illumina file. All probes that map to a different location than originally annotated are recommended to be masked (in the MASK.mapping and MASK.general column), so generally they won't be included in further analyses.

## Source

<http://zwdzwd.github.io/InfiniumAnnotation>

## References

Zhou W, Laird PW and Shen H: Comprehensive characterization, annotation and innovative use of Infinium DNA Methylation BeadChip probes. Nucleic Acids Research 2016

## Examples

```
# Select probes that should be masked in Dutch population
# (note that X and Y chromosomes are not included)
hm450.manifest.pop.GoNL <- hm450.manifest.pop.GoNL[!is.na(
  hm450.manifest.pop.GoNL$MASK.general.GoNL) &
  hm450.manifest.pop.GoNL$MASK.general.GoNL == TRUE, ]

# Select probes that should be masked in Dutch population because there is
# a SNP within 5 bases of the 3'end of the probe
# (note that X and Y chromosomes are not included)
hm450.manifest.pop.GoNL <- hm450.manifest.pop.GoNL[!is.na(
  hm450.manifest.pop.GoNL$MASK.snp5.GoNL) &
  hm450.manifest.pop.GoNL$MASK.snp5.GoNL == TRUE, ]

# When studying a Dutch population and one wants to include X and Y
# chromosomal probes, the EUR or CEU population can be used.
# Select probes that should be masked in European population
# (these include X and Y chromosomes)
hm450.manifest.pop.GoNL <- hm450.manifest.pop.GoNL[
  hm450.manifest.pop.GoNL$MASK.general.EUR == TRUE, ]
```

---

inferRelations	<i>predict mismatches</i>
----------------	---------------------------

---

### Description

predict mismatches

### Usage

```
inferRelations(data, n = 100, plot.it = TRUE, verbose = FALSE, ...)
```

### Arguments

data	output from allelesharing
n	= 100 default interpolation for showing the classification boundaries
plot.it	= TRUE default plot classification graph and returning mismatches otherwise return all
verbose	default FALSE, if TRUE show confusion matrix
...	optional plotting argument passed to plot

### Details

based on all data a classifier is build using Linear Discriminant Analysis and on the same data a prediction is performed in order to detect wrong sample relationships. The assumption is that the majority of sample relations is correct otherwise we could not do this!

### Value

predicted mismatches

### Author(s)

mvaniterson

### Examples

```
set.seed(12345)
beta <- matrix(runif(100*10, 0,1), nrow=100)
beta[1:5, 1:5]
colnames(beta) <- paste0("sample", 1:10)
genotype <- beta2genotype(beta)
genotype[1:5, 1:5]
data <- alleleSharing(genotype)
head(data)
inferRelations(data)
```



# Index

## \* **datasets**

hg19.GoNLsnps, [4](#)

hm450.manifest.pop.GoNL, [6](#)

alleleSharing, [2](#)

beta2genotype, [3](#)

hg19.GoNLsnps, [4](#)

hm450.manifest.pop.GoNL, [6](#)

inferRelations, [8](#)