Package 'blima'

November 16, 2026

Encoding UTF-8

Type Package

Title Tools for the preprocessing and analysis of the Illumina microarrays on the detector (bead) level

Version 1.47.0

Date 2017-09-23

Author Vojtěch Kulvait

Maintainer Vojtěch Kulvait <kulvait@gmail.com>

Description Package blima includes several algorithms for the preprocessing of Illumina microarray data. It focuses to the bead level analysis and provides novel approach to the quantile normalization of the vectors of unequal lengths. It provides variety of the methods for background correction including background subtraction, RMA like convolution and background outlier removal. It also implements variance stabilizing transformation on the bead level. There are also implemented methods for data summarization. It also provides the methods for performing T-tests on the detector (bead) level and on the probe level for differential expression testing.

License GPL-3

LazyLoad yes

Depends R(>= 3.3)

Imports beadarray(>= 2.0.0), Biobase(>= 2.0.0), Rcpp (>= 0.12.8), BiocGenerics, grDevices, stats, graphics

LinkingTo Rcpp

Suggests xtable, blimaTestingData, BiocStyle, illuminaHumanv4.db, lumi, knitr

URL https://bitbucket.org/kulvait/blima

biocViews Microarray, Preprocessing, Normalization, DifferentialExpression, GeneRegulation, GeneExpression

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/blima

git_branch devel

git_last_commit 55531c0

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-11-16

Contents

blima-package . . . 3
aggregateAndPreprocess . . . 3
bacgroundCorrect . . . 4
bacgroundCorrectSingleArray . . . 5
backgroundChannelSubtract . . . 6
backgroundChannelSubtractSingleArray . . . 7
channelExistsIntegrityWithLogicalVectorList . . . 7
checkIntegrity . . . 8
checkIntegrityLogical . . . 9
checkIntegrityOfListOfBeadLevelDataObjects . . . 9
checkIntegrityOfSingleBeadLevelDataObject . . . 10
chipArrayStatistics . . . 10
createSummarizedMatrix . . . 11
doAction . . . 12
doProbeTTests . . . 13
doTTests . . . 14
filterBg . . . 16
getNextVector . . . 17
initMeanDistribution . . . 17
insertColumn . . . 18
interpolateSortedVector . . . 18
interpolateSortedVectorRcpp_ . . . 19
log2TransformPositive . . . 19
meanDistribution . . . 20
nonParametricEstimator . . . 21
nonPositiveCorrect . . . 22
nonPositiveCorrectSingleArray . . . 23
numberOfDistributionElements . . . 23
performXieCorrection . . . 24
plotBackgroundImageAfterCorrection . . . 24
plotBackgroundImageBeforeCorrection . . . 25
quantileNormalize . . . 26
readToVector . . . 27
selectedChannelTransform . . . 28
selectedChannelTransformSingleArray . . . 29
singleArrayNormalize . . . 29
singleChannelExistsIntegrityWithLogicalVector . . . 30
singleCheckIntegrityLogicalVector . . . 31
singleNumberOfDistributionElements . . . 31
updateMeanDistribution . . . 32
varianceBeadStabilise . . . 32
varianceBeadStabiliseSingleArray . . . 33
vstFromLumi . . . 34
writeBackgroundImages . . . 35
xieBacgroundCorrect . . . 36
xieBacgroundCorrectSingleArray . . . 37

Index **38**

`blima-package` *Package for the preprocessing and analysis of the Illumina microarrays on the detector (bead) level.*

Description

Package blima includes several algorithms for the preprocessing of Illumina microarray data. It focuses to the bead level analysis and provides novel approach to the quantile normalization of the vectors of unequal lengths. It provides variety of the methods for background correction including background subtraction, RMA like convolution and background outlier removal. It also implements variance stabilizing transformation on the bead level. There are also implemented methods for data summarization. It provides the methods for performing T-tests on the detector (bead) level and on the probe level for differential expression testing.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Author(s)

Vojtěch Kulvait Maintainer: Vojtěch Kulvait <kulvait@gmail.com>

`aggregateAndPreprocess` *Aggregate data*

Description

This function is not intended to direct use. It helps perform work of doProbeTTests function. For each probe it prints mean and sd of an quality.

Usage

```
aggregateAndPreprocess(x, quality = "qua", transformation = NULL)
```

Arguments

`x`	Two column matrix to agregate with columns "ProbeID" and quality.
`quality`	Quality to analyze, default is "qua".
`transformation`	Function of input data trasformation, default is NULL. Any function which for input value returns transformed value may be supplied. T-test then will be evaluated on transformed data, consider use log2TranformPositive.

Value

Some return value

Author(s)

Vojtěch Kulvait

`bacgroundCorrect` *Data background correction.*

Description

Background correction procedure selecting beads with background Intensity I_b |mean - I_b | > k*SD(I_bs) for exclusion.

Usage

```
bacgroundCorrect(b, normalizationMod = NULL, channelBackground = "GrnB",
    k = 3, channelBackgroundFilter = "bgf", channelAndVector = NULL)
```

Arguments

`b`	List of beadLevelData objects (or single object).
`normalizationMod`	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
`channelBackground`	Name of channel to normalize.
`k`	Parameter of method stringency (default is 3).
`channelBackgroundFilter`	Filtered beads will have weight 0 and non filtered weight 1.
`channelAndVector`	Represents vector to bitvise multiple to the channelBackgroundFilter vector.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && interactive())
{
   #To perform background correction on blimatesting object for two groups. Background correction is followed by
    data(blimatesting)
    #Prepare logical vectors corresponding to conditions A and E.
    groups1 = "A";
    groups2 = "E";
    sampleNames = list()
    c = list()
    for(i in 1:length(blimatesting))
    {
        p = pData(blimatesting[[i]]@experimentData$phenoData)
        c[[i]] = p$Group %in% c(groups1, groups2);
        sampleNames[[i]] = p$Name
```

```
    }
  #Background correction and quantile normalization followed by testing including log2TransformPositive trans
  blimatesting = bacgroundCorrect(blimatesting, normalizationMod=c, channelBackgroundFilter="bgf")
  blimatesting = nonPositiveCorrect(blimatesting, normalizationMod=c, channelCorrect="GrnF", channelBackgro
}else
{
  print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

---

bacgroundCorrectSingleArray *Data background correction.*

Description

Background correction procedure selecting beads with background Intensity I_b |mean - I_b | > k*SD(I_bs) for exclusion, internal.

Usage

```
bacgroundCorrectSingleArray(b, normalizationMod = NULL, channelBackground = "GrnB",
    k = 3, channelBackgroundFilter = "bgf", channelAndVector = NULL)
```

Arguments

b	List of beadLevelData objects (or single object).
normalizationMod	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
channelBackground	Name of channel to normalize.
k	Parameter of method stringency (default is 3).
channelBackgroundFilter	Filtered beads will have weight 0 and non filtered weight 1.
channelAndVector	Represents vector to bitvise multiple to the channelBackgroundFilter vector.

Author(s)

Vojtěch Kulvait

backgroundChannelSubtract *Background channel subtraction*

Description

Function to subtract one channel from another producing new channel. Standard graphic subtraction.

Usage

```
backgroundChannelSubtract(b, normalizationMod = NULL, channelSubtractFrom = "GrnF",
    channelSubtractWhat = "GrnB", channelResult = "Grn")
```

Arguments

b	List of beadLevelData objects (or single object).
normalizationMod	NULL for performing on all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
channelSubtractFrom	Name of channel to subtract from.
channelSubtractWhat	Name of channel to subtract.
channelResult	Result channel, if this channel exists it will be overwritten.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && interactive())
{
   #To perform background correction on blimatesting object for two groups. Background correction is followed by
    data(blimatesting)
    #Prepare logical vectors corresponding to conditions A and E.
    groups1 = "A";
    groups2 = "E";
    sampleNames = list()
    c = list()
    for(i in 1:length(blimatesting))
    {
        p = pData(blimatesting[[i]]@experimentData$phenoData)
        c[[i]] = p$Group %in% c(groups1, groups2);
        sampleNames[[i]] = p$Name
    }
   #Background correction and quantile normalization followed by testing including log2TransformPositive trans
   blimatesting = bacgroundCorrect(blimatesting, normalizationMod=c, channelBackgroundFilter="bgf")
   blimatesting = nonPositiveCorrect(blimatesting, normalizationMod=c, channelCorrect="GrnF", channelBackgro
}else
```

```
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

`backgroundChannelSubtractSingleArray` *Background channel subtraction*

Description

INTERNAL FUNCTION Correction for positive values only

Usage

```
backgroundChannelSubtractSingleArray(b, normalizationMod = NULL,
    channelSubtractFrom = "GrnF", channelSubtractWhat = "GrnB",
    channelResult = "Grn")
```

Arguments

`b`	List of beadLevelData objects (or single object).
`normalizationMod`	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
`channelSubtractFrom`	Name of channel to subtract from.
`channelSubtractWhat`	Name of channel to subtract.
`channelResult`	Result channel, if this channel exists it will be overwritten.

Author(s)

Vojtěch Kulvait

`channelExistsIntegrityWithLogicalVectorList` *Internal function*

Description

Test existence of channel slot based on vector list

Usage

```
channelExistsIntegrityWithLogicalVectorList(b, spotsToCheck = NULL,
    slotToCheck, action = c("returnText", "warn", "error"))
```

Arguments

b	List of beadLevelData objects.
spotsToCheck	NULL for check all spots from b. Otherwise specifies logical vector of the length equals to the number of arrays in b with TRUE for checking.
slotToCheck	Slot name to check
action	What type of action is required in case of invalid object structure. Either return text different from TRUE, warn or error.

Author(s)

Vojtěch Kulvait

---

checkIntegrity *Internal function*

Description

Check integrity of the list of beadLevelData objects or single beadLevelData object returns waslist.

Usage

```
checkIntegrity(b, action = c("warn", "error"))
```

Arguments

b	List of beadLevelData objects or single.
action	What type of action is required in case of invalid object structure. Either return text different from TRUE, warn or error.

Value

Returns value if the object was list or not before calling this function.

Author(s)

Vojtěch Kulvait

checkIntegrityLogical *Internal function*

Description

Check integrity of the list of logical objects, internal.

Usage

```
checkIntegrityLogical(xx, b, action = c("returnText", "warn",
    "error"))
```

Arguments

xx	List of logical objects compatible with a list b.
b	List of beadLevelData objects.
action	What type of action is required in case of invalid object structure. Either return text different from TRUE, warn or error.

Author(s)

Vojtěch Kulvait

checkIntegrityOfListOfBeadLevelDataObjects *Internal function*

Description

Check integrity of the list of beadLevelData objects, internal.

Usage

```
checkIntegrityOfListOfBeadLevelDataObjects(listb, action = c("returnText",
    "warn", "error"))
```

Arguments

listb	List of beadLevelData objects.
action	What type of action is required in case of invalid object structure. Either return text different from TRUE, warn or error.

Author(s)

Vojtěch Kulvait

checkIntegrityOfSingleBeadLevelDataObject *Internal function*

Description

Check integrity of single beadLevelData object, internal.

Usage

```
checkIntegrityOfSingleBeadLevelDataObject(b, action = c("returnText",
    "warn", "error"))
```

Arguments

b	beadLevelData object.
action	What type of action is required in case of invalid object structure. Either return text different from TRUE, warn or error.

Author(s)

Vojtěch Kulvait

chipArrayStatistics *Statistics of beadLevelData*

Description

This function returns table with statistics of single beadLevelData object indexed by order of spots. It prints number of beads on each array spot mean foreground intensity and optionally mean background intensity, mean number of beads in probe set and unbiased estimate of standard deviations of these parameters. Optionaly you can also obtain percentage of removed beads within excludedOnSDMultiple multiple of standard deviations from the background value.

Usage

```
chipArrayStatistics(b, includeBeadStatistic = TRUE, channelForeground = "GrnF",
   channelBackground = "GrnB", includeBackground = TRUE, excludedOnSDMultiple = NA)
```

Arguments

b	Single beadLevelData object.
includeBeadStatistic	Include number of beads per probe in output.
channelForeground	Name of channel of foreground.
channelBackground	Name of channel of background.

includeBackground
: Whether to output background data.

excludedOnSDMultiple
: If positive number, print how much percents of the background lies more than excludedOnSDMultiple multipliers of standard deviation estimate away from background mean.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && interactive())
{
    #To print basic statistic data about blimatesting[[1]] object.
    data(blimatesting)
    array1stats = chipArrayStatistics(blimatesting[[1]], includeBeadStatistic=TRUE,
            excludedOnSDMultiple=3)
    array1pheno = pData(blimatesting[[1]]@experimentData$phenoData)
    array1stats = data.frame(array1pheno$Name, array1stats)
    colnames(array1stats)[1] <- "Array";
    print(array1stats);
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

---

createSummarizedMatrix

Summarized value matrix.

Description

This function creates summarized matrix of values of certain type.

Usage

```
createSummarizedMatrix(b, spotsToProcess = NULL, quality = "qua",
    channelInclude = "bgf", annotationTag = NULL)
```

Arguments

b	List of beadLevelData objects (or single object).
spotsToProcess	NULL for processing all spots in b. Otherwise specifies logical vector of the length equals to the number of arrays in b.
quality	Quality to matrize.
channelInclude	This field allows user to set channel with weights which have to be from 0,1. All zero weighted items are excluded from summarization. You can turn this off by setting this NULL. This option may be used together with bacgroundCorrect method or/and with beadarray QC (defaults to "bgf").
annotationTag	Tag from annotation file which to use in resulting matrix as colname.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && require("illuminaHumanv4.db") && interactive())
{
    #Create summarization of nonnormalized data from GrnF column.
    data(blimatesting)
    blimatesting = bacgroundCorrect(blimatesting, channelBackgroundFilter="bgf")
  blimatesting = nonPositiveCorrect(blimatesting, channelCorrect="GrnF", channelBackgroundFilter="bgf", cha
  #Prepare logical vectors corresponding to conditions A(groups1Mod), E(groups2Mod) and both(processingMod).
  nonnormalized = createSummarizedMatrix(blimatesting, quality="GrnF", channelInclude="bgf",
            annotationTag="Name")
    head(nonnormalized)
}else
{
  print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

`doAction` *Internal function*

Description

Performs action of certain type

Usage

```
doAction(message, action = c("returnText", "warn", "error"))
```

Arguments

`message`	Text message.
`action`	What type of action is required in case of invalid object structure. Either return text different from TRUE, warn or error.

Author(s)

Vojtěch Kulvait

doProbeTTests *T-test for probe level data.*

Description

This function does aggregated probe level t-tests on the data provided by the object beadLevelData from package beadarray.

Usage

```
doProbeTTests(b, c1, c2, quality = "qua", channelInclude = "bgf",
    correction = "BY", transformation = NULL)
```

Arguments

b	List of beadLevelData objects (or single object).
c1	List of logical vectors of data to assign to the first group (or single vector).
c2	List of logical vectors of data to assign to the second group (or single vector).
quality	Quality to analyze, default is "qua".
channelInclude	This field allows user to set channel with weights which have to be 0,1. All zero weighted items are excluded from t-test. You can turn this off by setting this NULL. This option may be used together with bacgroundCorrect method or/and with beadarray QC (defaults to "bgf").
correction	Multiple testing adjustment method as defined by p.adjust function, default is "BY".
transformation	Function of input data trasformation, default is NULL. Any function which for input value returns transformed value may be supplied. T-test then will be evaluated on transformed data, consider use log2TranformPositive.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && require("illuminaHumanv4.db") && interactive())
{
   #To perform background correction, variance stabilization and  quantile normalization then test on probe leve
    data(blimatesting)
   #Prepare logical vectors corresponding to conditions A(groups1Mod), E(groups2Mod) and both(processingMod).
    groups1 = "A";
    groups2 = "E";
    sampleNames = list()
    groups1Mod = list()
    groups2Mod = list()
    processingMod = list()
    for(i in 1:length(blimatesting))
    {
        p = pData(blimatesting[[i]]@experimentData$phenoData)
        groups1Mod[[i]] = p$Group %in% groups1;
        groups2Mod[[i]] = p$Group %in% groups2;
```

```
        processingMod[[i]] = p$Group %in% c(groups1, groups2);
        sampleNames[[i]] = p$Name
    }
   #Background correction and quantile normalization followed by testing including log2TransformPositive trans
   blimatesting = bacgroundCorrect(blimatesting, normalizationMod =processingMod, channelBackgroundFilter="bg
   blimatesting = nonPositiveCorrect(blimatesting, normalizationMod=processingMod, channelCorrect="GrnF", ch
    blimatesting = varianceBeadStabilise(blimatesting, normalizationMod = processingMod,
            quality="GrnF", channelInclude="bgf", channelOutput="vst")
    blimatesting = quantileNormalize(blimatesting, normalizationMod = processingMod,
            channelNormalize="vst", channelOutput="qua", channelInclude="bgf")
    beadTest = doTTests(blimatesting, groups1Mod, groups2Mod, "qua", "bgf")
    probeTest = doProbeTTests(blimatesting, groups1Mod, groups2Mod, "qua", "bgf")
   adrToSymbol <- merge(toTable(illuminaHumanv4ARRAYADDRESS), toTable(illuminaHumanv4SYMBOLREANNOTATED))
    adrToSymbol <- adrToSymbol[,c("ArrayAddress", "SymbolReannotated") ]
    colnames(adrToSymbol) <- c("Array_Address_Id", "Symbol")
    probeTestID = probeTest[,"ProbeID"]
    beadTestID = beadTest[,"ProbeID"]
    probeTestFC = abs(probeTest[,"mean1"]-probeTest[,"mean2"])
    beadTestFC = abs(beadTest[,"mean1"]-beadTest[,"mean2"])
    probeTestP = probeTest[,"adjustedp"]
    beadTestP = beadTest[,"adjustedp"]
    probeTestMeasure = (1-probeTestP)*probeTestFC
    beadTestMeasure = (1-beadTestP)*beadTestFC
    probeTest = cbind(probeTestID, probeTestMeasure)
    beadTest = cbind(beadTestID, beadTestMeasure)
    colnames(probeTest) <- c("ArrayAddressID", "difexPL")
    colnames(beadTest) <- c("ArrayAddressID", "difexBL")
    tocmp <- merge(probeTest, beadTest)
    tocmp = merge(tocmp, adrToSymbol, by.x="ArrayAddressID", by.y="Array_Address_Id")
    tocmp = tocmp[, c("ArrayAddressID", "Symbol", "difexPL", "difexBL")]
    sortPL = sort(-tocmp[,"difexPL"], index.return=TRUE)$ix
    sortBL = sort(-tocmp[,"difexBL"], index.return=TRUE)$ix
    beadTop10 = tocmp[sortBL[1:10],]
    probeTop10 = tocmp[sortPL[1:10],]
    print(beadTop10)
    print(probeTop10)
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

---

doTTests *T-test for bead (detector) level data.*

Description

This function does t-tests on the data provided by the object beadLevelData from package beadarray.

Usage

```
doTTests(b, c1, c2, quality = "qua", channelInclude = "bgf",
    correction = "BY", transformation = NULL)
```

Arguments

b	List of beadLevelData objects (or single object).
c1	List of logical vectors of data to assign to the first group (or single vector).
c2	List of logical vectors of data to assign to the second group (or single vector).
quality	Quality to analyze, default is "qua".
channelInclude	This field allows user to set channel with weights which have to be 0,1. All zero weighted items are excluded from t-test. You can turn this off by setting this NULL. This option may be used together with bacgroundCorrect method or/and with beadarray QC (defaults to "bgf").
correction	Multiple testing adjustment method as defined by p.adjust function, default is "BY".
transformation	Function of input data trasformation, default is NULL. Any function which for input value returns transformed value may be supplied. T-test then will be evaluated on transformed data, consider use log2TransformPositive.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && require("illuminaHumanv4.db") && interactive())
{
   #To perform background correction, variance stabilization and  quantile normalization then test on probe leve
    data(blimatesting)
   #Prepare logical vectors corresponding to conditions A(groups1Mod), E(groups2Mod) and both(processingMod).
    groups1 = "A";
    groups2 = "E";
    sampleNames = list()
    groups1Mod = list()
    groups2Mod = list()
    processingMod = list()
    for(i in 1:length(blimatesting))
    {
        p = pData(blimatesting[[i]]@experimentData$phenoData)
        groups1Mod[[i]] = p$Group %in% groups1;
        groups2Mod[[i]] = p$Group %in% groups2;
        processingMod[[i]] = p$Group %in% c(groups1, groups2);
        sampleNames[[i]] = p$Name
    }
   #Background correction and quantile normalization followed by testing including log2TransformPositive trans
   blimatesting = bacgroundCorrect(blimatesting, normalizationMod =processingMod, channelBackgroundFilter="bg
   blimatesting = nonPositiveCorrect(blimatesting, normalizationMod=processingMod, channelCorrect="GrnF", ch
    blimatesting = varianceBeadStabilise(blimatesting, normalizationMod = processingMod,
            quality="GrnF", channelInclude="bgf", channelOutput="vst")
    blimatesting = quantileNormalize(blimatesting, normalizationMod = processingMod,
            channelNormalize="vst", channelOutput="qua", channelInclude="bgf")
    beadTest = doTTests(blimatesting, groups1Mod, groups2Mod, "qua", "bgf")
    probeTest = doProbeTTests(blimatesting, groups1Mod, groups2Mod, "qua", "bgf")
   adrToSymbol <- merge(toTable(illuminaHumanv4ARRAYADDRESS), toTable(illuminaHumanv4SYMBOLREANNOTATED))
    adrToSymbol <- adrToSymbol[,c("ArrayAddress", "SymbolReannotated") ]
    colnames(adrToSymbol) <- c("Array_Address_Id", "Symbol")
    probeTestID = probeTest[,"ProbeID"]
```

```
    beadTestID = beadTest[,"ProbeID"]
    probeTestFC = abs(probeTest[,"mean1"]-probeTest[,"mean2"])
    beadTestFC = abs(beadTest[,"mean1"]-beadTest[,"mean2"])
    probeTestP = probeTest[,"adjustedp"]
    beadTestP = beadTest[,"adjustedp"]
    probeTestMeasure = (1-probeTestP)*probeTestFC
    beadTestMeasure = (1-beadTestP)*beadTestFC
    probeTest = cbind(probeTestID, probeTestMeasure)
    beadTest = cbind(beadTestID, beadTestMeasure)
    colnames(probeTest) <- c("ArrayAddressID", "difexPL")
    colnames(beadTest) <- c("ArrayAddressID", "difexBL")
    tocmp <- merge(probeTest, beadTest)
    tocmp = merge(tocmp, adrToSymbol, by.x="ArrayAddressID", by.y="Array_Address_Id")
    tocmp = tocmp[, c("ArrayAddressID", "Symbol", "difexPL", "difexBL")]
    sortPL = sort(-tocmp[,"difexPL"], index.return=TRUE)$ix
    sortBL = sort(-tocmp[,"difexBL"], index.return=TRUE)$ix
    beadTop10 = tocmp[sortBL[1:10],]
    probeTop10 = tocmp[sortPL[1:10],]
    print(beadTop10)
    print(probeTop10)
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

`filterBg` *Bg correct vector*

Description

Background correction procedure selecting beads with background Intensity I_b |mean - I_b | > k*SD(I_bs) for exclusion, internal.

Usage

```
filterBg(x, k = 3)
```

Arguments

x	Vector to correct
k	Parameter of method stringency (default is 3).

Author(s)

Vojtěch Kulvait

getNextVector *Support probe and beadl level testing.*

Description

Internal function supporting probe and beadl level testing.

Usage

```
getNextVector(what, from, length)
```

Arguments

what	Two column sorted matrix with probe values.
from	Index to start on
length	nrow(what)

Author(s)

Vojtěch Kulvait

initMeanDistribution *initMeanDistribution*

Description

This is internal function not intended to direct use which initializes mean distribution.

Usage

```
initMeanDistribution(srt, prvku)
```

Arguments

srt	vector of sorted values
prvku	number of items in meanDistribution

Author(s)

Vojtěch Kulvait

`insertColumn` *Internal function to support chipArrayStatistics*

Description

Internal

Usage

```
insertColumn(matrix, column, name)
```

Arguments

`matrix`	Object to insert column to
`column`	Column to insert
`name`	Name of column to assign.

Author(s)

Vojtěch Kulvait

`interpolateSortedVector` *Interpolate sorted vector*

Description

Interpolates given sorted vector to the vector of different length. It does not sort input vector thus for unsorted vectors do not guarantee functionality. Internal function.

Usage

```
interpolateSortedVector(vector, newSize)
```

Arguments

`vector`	Sorted vector to interpolate.
`newSize`	Size of the vector to produce.

Author(s)

Vojtěch Kulvait

interpolateSortedVectorRcpp_ *interpolateSortedVectorRcpp*

Usage

```
interpolateSortedVectorRcpp_(vector, newSize)
```

Arguments

vector

newSize

Author(s)

Vojtěch Kulvait

log2TransformPositive *Log2 transform of numbers >1.*

Description

Transformation function are popular in beadarray package. Here this is similar concept. This function allow user to perform log transformation before doing t-tests.

Usage

```
log2TransformPositive(x)
```

Arguments

x	Number to transform.

Value

This function returns logarithm of base 2 for numbers >=1 and zero for numbers <1.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && require("illuminaHumanv4.db") && interactive())
{
   #To perform background correction, quantile normalization and then bead level t-test on log data run. Vst is r
    data(blimatesting)
   #Prepare logical vectors corresponding to conditions A(groups1Mod), E(groups2Mod) and both(c).
    groups1 = "A";
    groups2 = "E";
    sampleNames = list()
    groups1Mod = list()
    groups2Mod = list()
    c = list()
    for(i in 1:length(blimatesting))
    {
        p = pData(blimatesting[[i]]@experimentData$phenoData)
        groups1Mod[[i]] = p$Group %in% groups1;
        groups2Mod[[i]] = p$Group %in% groups2;
        c[[i]] = p$Group %in% c(groups1, groups2);
        sampleNames[[i]] = p$Name
    }
   #Background correction and quantile normalization followed by testing including log2TransformPositive trans
   blimatesting = bacgroundCorrect(blimatesting, normalizationMod =c, channelBackgroundFilter="bgf")
   blimatesting = nonPositiveCorrect(blimatesting, normalizationMod=c, channelCorrect="GrnF", channelBackgro
   blimatesting = quantileNormalize(blimatesting, normalizationMod=c, channelNormalize="GrnF", channelOutput=
    beadTest <- doTTests(blimatesting, groups1Mod, groups2Mod,
            transformation=log2TransformPositive, quality="qua", channelInclude="bgf")
   symbol2address <- merge(toTable(illuminaHumanv4ARRAYADDRESS), toTable(illuminaHumanv4SYMBOLREANNOTATED))
    symbol2address <- symbol2address[,c("SymbolReannotated", "ArrayAddress") ]
    colnames(symbol2address) <- c("Symbol", "ArrayAddressID")
    beadTest = merge(beadTest, symbol2address, by.x="ProbeID", by.y="ArrayAddressID")
    beadTestID = beadTest[,c("ProbeID", "Symbol")]
    beadTestFC = abs(beadTest[,"mean1"]-beadTest[,"mean2"])
    beadTestP = beadTest[,"adjustedp"]
    beadTestMeasure = (1-beadTestP)*beadTestFC
    beadTest = cbind(beadTestID, beadTestMeasure)
    colnames(beadTest) <- c("ArrayAddressID", "Symbol", "difexBL")
    sortBL = sort(-beadTest[,"difexBL"], index.return=TRUE)$ix
    beadTop10 = beadTest[sortBL[1:10],]
    print(beadTop10)
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

---

meanDistribution *Produce sorted double vector with mean distribution.*

Description

This function processes arrays in the object beadLevelData from package beadarray and returns sorted double vector. The vector has length prvku. And the distribution of this vector is a "mean" of all distributions of distributionChannel quantity in arrays. In case that probe numbers are different from prvku it does some averaging.

Usage

```
meanDistribution(b, normalizationMod = NULL, distributionChannel = "Grn",
    channelInclude = NULL, prvku)
```

Arguments

b	Object beadLevelData from package beadarray or list of these objects
normalizationMod	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes (defaults to NULL).
distributionChannel	Channel to do mean distribution from (defaults to "Grn").
channelInclude	This field allows user to set channel with weights which have to be in 0,1. All zero weighted items are excluded from quantile normalization and the value asigned to such probes is a close to value which would be assigned to them if not being excluded. You can turn this off by setting this NULL. This option may be used together with bacgroundCorrect method or/and with beadarray QC (defaults to NULL).
prvku	Number of items in a resulting double vector. Prvku must not be more than minimal number of indluded items in any distributionChannel.

Author(s)

Vojtěch Kulvait

---

nonParametricEstimator

INTERNAL FUNCTION Xie background correct.

Description

INTERNAL This function is not intended for direct use. Background correction according to non parametric estimator in Xie, Yang, Xinlei Wang, and Michael Story. "Statistical Methods of Background Correction for Illumina BeadArray Data." Bioinformatics 25, no. 6 (March 15, 2009): 751-57. doi:10.1093/bioinformatics/btp040. The method is applied on the bead level.

Usage

```
nonParametricEstimator(toCorrectAll, toCorrectNeg)
```

Arguments

toCorrectAll	
toCorrectNeg	

Author(s)

Vojtěch Kulvait

nonPositiveCorrect *Correct non positive*

Description

Correction for positive values only

Usage

```
nonPositiveCorrect(b, normalizationMod = NULL, channelCorrect = "GrnF",
    channelBackgroundFilter = "bgf", channelAndVector = NULL)
```

Arguments

b	List of beadLevelData objects (or single object).
normalizationMod	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
channelCorrect	Name of channel to correct.
channelBackgroundFilter	Filtered beads will have weight 0 and non filtered weight 1.
channelAndVector	Represents vector to bitvise multiple to the channelBackgroundFilter vector.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && interactive())
{
   #To perform background correction on blimatesting object for two groups. Background correction is followed by
    data(blimatesting)
    #Prepare logical vectors corresponding to conditions A and E.
    groups1 = "A";
    groups2 = "E";
    sampleNames = list()
    c = list()
    for(i in 1:length(blimatesting))
    {
        p = pData(blimatesting[[i]]@experimentData$phenoData)
        c[[i]] = p$Group %in% c(groups1, groups2);
        sampleNames[[i]] = p$Name
    }
   #Background correction and quantile normalization followed by testing including log2TransformPositive trans
   blimatesting = bacgroundCorrect(blimatesting, normalizationMod=c, channelBackgroundFilter="bgf")
   blimatesting = nonPositiveCorrect(blimatesting, normalizationMod=c, channelCorrect="GrnF", channelBackgro
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

nonPositiveCorrectSingleArray *Correct non positive*

Description

INTERNAL FUNCTION Correction for positive values only

Usage

```
nonPositiveCorrectSingleArray(b, normalizationMod = NULL, channelCorrect = "GrnF",
    channelBackgroundFilter = "bgf", channelAndVector = NULL)
```

Arguments

b	List of beadLevelData objects (or single object).
normalizationMod	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
channelCorrect	Name of channel to correct.
channelBackgroundFilter	Filtered beads will have weight 0 and non filtered weight 1.
channelAndVector	Represents vector to bitvise multiple to the channelBackgroundFilter vector.

Author(s)

Vojtěch Kulvait

numberOfDistributionElements *Internal*

Description

Internal function

Usage

```
numberOfDistributionElements(b, normalizationMod = NULL, channelInclude = NULL)
```

Arguments

b	Object beadLevelData from package beadarray or list of these objects
normalizationMod	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
channelInclude	

Author(s)

Vojtěch Kulvait

`performXieCorrection` *INTERNAL FUNCTION Xie background correct.*

Description

INTERNAL This function is not intended for direct use. Background correction according to non parametric estimator in Xie, Yang, Xinlei Wang, and Michael Story. "Statistical Methods of Background Correction for Illumina BeadArray Data." Bioinformatics 25, no. 6 (March 15, 2009): 751-57. doi:10.1093/bioinformatics/btp040. ###The method is applied on the bead level.

Usage

```
performXieCorrection(value, alpha, mu, sigma)
```

Arguments

`value`

`alpha`

`mu`

`sigma`

Author(s)

Vojtěch Kulvait

`plotBackgroundImageAfterCorrection`
Plot background image after correction

Description

This function plots image of background distribution versus to foreground after background subtraction.

Usage

```
plotBackgroundImageAfterCorrection(b, index, channelForeground = "GrnF",
    channelBackground = "GrnB", SDMultiple = 3, includePearson = FALSE)
```

Arguments

b	Single beadLevelData object.
index	Index of spot to generate.
channelForeground	Name of channel of foreground.
channelBackground	Name of channel of background.
SDMultiple	Correct on this level.
includePearson	Include Pearson corelation.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && interactive())
{
   #Write background images after correction. This function prints graph for condition D4. Call dev.off() to clo
    data(blimatesting)
    p = pData(blimatesting[[2]]@experimentData$phenoData)
    index = base::match("D4", p$Name)
    plotBackgroundImageAfterCorrection(blimatesting[[2]], index)
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

---

plotBackgroundImageBeforeCorrection

Plot background image before correction

Description

This function plots image of background distribution versus to foreground before background subtraction.

Usage

```
plotBackgroundImageBeforeCorrection(b, index, channelForeground = "GrnF",
    channelBackground = "GrnB", includePearson = FALSE)
```

Arguments

b	Single beadLevelData object.
index	Index of spot to generate.
channelForeground	Name of channel of foreground.
channelBackground	Name of channel of background.
includePearson	Include Pearson corelation.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && interactive())
{
   #Write background images before correction. This function prints graph for condition D4. Call dev.off() to cl
    data(blimatesting)
    p = pData(blimatesting[[2]]@experimentData$phenoData)
    index = base::match("D4", p$Name)
    plotBackgroundImageBeforeCorrection(blimatesting[[2]], index)
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

---

quantileNormalize *Bead level quantile normalization.*

Description

This function does quantile normalization of object beadLevelData from package beadarray.

Usage

```
quantileNormalize(b, normalizationMod = NULL, channelNormalize = "Grn",
    channelOutput = "qua", channelInclude = NULL, dst)
```

Arguments

b	Object beadLevelData from package beadarray or list of these objects
normalizationMod	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
channelNormalize	Name of channel to normalize.
channelOutput	Name of output normalized channel.
channelInclude	This field allows user to set channel with weights which have to be in 0,1. All zero weighted items are excluded from quantile normalization and the value asigned to such probes is a close to value which would be assigned to them if not being excluded. You can turn this off by setting this NULL. This option may be used together with bacgroundCorrect method or/and with beadarray QC (defaults to NULL).
dst	User can specify sorted vector which represents distribution that should be assigned to items.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && interactive())
{
    #To perform background correction, variance stabilization and quantile normalization.
    data(blimatesting)
   #Prepare logical vectors corresponding to conditions A(groups1Mod), E(groups2Mod) and both(c).
    groups1 = "A";
    groups2 = "E";
    sampleNames = list()
    processingMod = list()
    for(i in 1:length(blimatesting))
    {
        p = pData(blimatesting[[i]]@experimentData$phenoData)
        processingMod[[i]] = p$Group %in% c(groups1, groups2);
        sampleNames[[i]] = p$Name
    }
   #Background correction and quantile normalization followed by testing including log2TransformPositive trans
   blimatesting = bacgroundCorrect(blimatesting, normalizationMod = processingMod, channelBackgroundFilter="b
   blimatesting = nonPositiveCorrect(blimatesting, normalizationMod = processingMod, channelCorrect="GrnF",
    blimatesting = varianceBeadStabilise(blimatesting, normalizationMod = processingMod,
            quality="GrnF", channelInclude="bgf", channelOutput="vst")
    blimatesting = quantileNormalize(blimatesting, normalizationMod = processingMod,
            channelNormalize="vst", channelOutput="qua", channelInclude="bgf")
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

---

readToVector *Support doTTests function.*

Description

Internal function supporting doTTests function.

Usage

```
readToVector(what, from, length, quality)
```

Arguments

what	Item to read.
from	From index.
length	Length of vector.
quality	Column.

Author(s)

Vojtěch Kulvait

selectedChannelTransform *Channel transformation*

Description

Function to transform channel data.

Usage

```
selectedChannelTransform(b, normalizationMod = NULL, channelTransformFrom,
    channelResult, transformation = NULL)
```

Arguments

b	List of beadLevelData objects (or single object).
normalizationMod	NULL for performing on all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
channelTransformFrom	Name of channel to transform.
channelResult	Result channel, if this channel exists it will be overwritten.
transformation	Function of input data trasformation, default is NULL. Any function which for input value returns transformed value may be supplied. T-test then will be evaluated on transformed data, consider use log2TranformPositive.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && interactive())
{
   #To perform background correction on blimatesting object for two groups. Background correction is followed by
    data(blimatesting)
    #Prepare logical vectors corresponding to conditions A and E.
    groups1 = "A";
    groups2 = "E";
    sampleNames = list()
    c = list()
    for(i in 1:length(blimatesting))
    {
        p = pData(blimatesting[[i]]@experimentData$phenoData)
        c[[i]] = p$Group %in% c(groups1, groups2);
        sampleNames[[i]] = p$Name
    }
   #Background correction and quantile normalization followed by testing including log2TransformPositive trans
   blimatesting = bacgroundCorrect(blimatesting, normalizationMod=c, channelBackgroundFilter="bgf")
   blimatesting = nonPositiveCorrect(blimatesting, normalizationMod=c, channelCorrect="GrnF", channelBackgro
}else
```

```
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

`selectedChannelTransformSingleArray` *Channel transformation*

Description

Function to transform channel data.

Usage

```
selectedChannelTransformSingleArray(b, normalizationMod = NULL,
    channelTransformFrom, channelResult, transformation)
```

Arguments

`b`	List of beadLevelData objects (or single object).
`normalizationMod`	NULL for performing on all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
`channelTransformFrom`	Name of channel to transform.
`channelResult`	Result channel, if this channel exists it will be overwritten.
`transformation`	Function of input data trasformation, default is NULL. Any function which for input value returns transformed value may be supplied. T-test then will be evaluated on transformed data, consider use log2TranformPositive.

Author(s)

Vojtěch Kulvait

`singleArrayNormalize` *Bead level quantile normalization.*

Description

This function does quantile normalization of object beadLevelData from package beadarray. Internal function not intended to direct use. Please use quantileNormalize.

Usage

```
singleArrayNormalize(b, normalizationMod = NULL, channelNormalize = "Grn",
    channelOutput = "qua", channelInclude = NULL, dst)
```

Arguments

b	Object beadLevelData from package beadarray
normalizationMod	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b.
channelNormalize	Name of channel to normalize.
channelOutput	Name of output normalized channel.
channelInclude	This field allows user to set channel with weights which have to be in 0,1. All zero weighted items are excluded from quantile normalization and the value asigned to such probes is a close to value which would be assigned to them if not being excluded. You can turn this off by setting this NULL. This option may be used together with bacgroundCorrect method or/and with beadarray QC (defaults to NULL).
dst	This field must be sorted. It is a distribution of values to assign to ports. By default this distribution is computed using meanDistribution function.

Author(s)

Vojtěch Kulvait

---

singleChannelExistsIntegrityWithLogicalVector

Internal function

Description

Test existence of channel slot based on logical list

Usage

```
singleChannelExistsIntegrityWithLogicalVector(b, spotsToCheck = NULL,
    slotToCheck, action = c("returnText", "warn", "error"))
```

Arguments

b	single beadLevelData object
spotsToCheck	NULL for check all spots from b. Otherwise specifies logical vector of the length equals to the number of arrays in b with TRUE for checking.
slotToCheck	Slot name to check
action	What type of action is required in case of invalid object structure. Either return text different from TRUE, warn or error.

Author(s)

Vojtěch Kulvait

singleCheckIntegrityLogicalVector

Internal function

Description

Check integrity of the logical object, internal.

Usage

```
singleCheckIntegrityLogicalVector(xx, b, action = c("returnText",
    "warn", "error"))
```

Arguments

xx	Logical object compatible with b.
b	Single beadLevelData object.
action	What type of action is required in case of invalid object structure. Either return text different from TRUE, warn or error.

Author(s)

Vojtěch Kulvait

singleNumberOfDistributionElements

Internal

Description

Internal function

Usage

```
singleNumberOfDistributionElements(b, normalizationMod = NULL,
    channelInclude = NULL)
```

Arguments

b	Object beadLevelData from package beadarray
normalizationMod	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
channelInclude	

Author(s)

Vojtěch Kulvait

updateMeanDistribution *updateMeanDistribution*

Description

This is internal function not intended to direct use. Updates mean distribution.

Usage

```
updateMeanDistribution(meanDistribution, srt, arraysUsed)
```

Arguments

`meanDistribution`

`srt`	vector of sorted values
`arraysUsed`	number of arrays allready used to create distribution

Author(s)

Vojtěch Kulvait

varianceBeadStabilise *Bead level VST.*

Description

This function does variance stabilising step on bead level.

Usage

```
varianceBeadStabilise(b, normalizationMod = NULL, quality = "qua",
    channelInclude = "bgf", channelOutput = "vst")
```

Arguments

`b`	List of beadLevelData objects (or single object).
`normalizationMod`	NULL for normalization of all input b. Otherwise specifies logical vector of the length equal to the number of arrays in b or list of such vectors if b is a list of beadLevelData classes.
`quality`	Quality to analyze, default is "qua".
`channelInclude`	This field allows user to set channel with weights which have to be in 0,1. All zero weighted items are excluded from t-test. You can turn this off by setting this NULL. This option may be used together with bacgroundCorrect method or/and with beadarray QC (defaults to "bgf").
`channelOutput`	Output from VST.

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && interactive())
{
    #To perform background correction, variance stabilization and quantile normalization.
    data(blimatesting)
   #Prepare logical vectors corresponding to conditions A(groups1Mod), E(groups2Mod) and both(c).
    groups1 = "A";
    groups2 = "E";
    sampleNames = list()
    processingMod = list()
    for(i in 1:length(blimatesting))
    {
        p = pData(blimatesting[[i]]@experimentData$phenoData)
        processingMod[[i]] = p$Group %in% c(groups1, groups2);
        sampleNames[[i]] = p$Name
    }
   #Background correction and quantile normalization followed by testing including log2TransformPositive trans
   blimatesting = bacgroundCorrect(blimatesting, normalizationMod = processingMod, channelBackgroundFilter="b
   blimatesting = nonPositiveCorrect(blimatesting, normalizationMod = processingMod, channelCorrect="GrnF",
    blimatesting = varianceBeadStabilise(blimatesting, normalizationMod = processingMod,
            quality="GrnF", channelInclude="bgf", channelOutput="vst")
    blimatesting = quantileNormalize(blimatesting, normalizationMod = processingMod,
            channelNormalize="vst", channelOutput="qua", channelInclude="bgf")
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

---

varianceBeadStabiliseSingleArray *Bead level VST.*

Description

This function is not intended to direct use it takes single beadLevelData object and do bead level variance stabilisation.

Usage

```
varianceBeadStabiliseSingleArray(b, normalizationMod = NULL,
    quality = "qua", channelInclude = "bgf", channelOutput = "vst")
```

Arguments

b	Object beadLevelData.
normalizationMod	NULL for normalization of all input b. Otherwise specifies logical vector of the length equals to the number of arrays in b.
quality	Quality to analyze, default is "qua".

channelInclude	This field allows user to set channel with weights which have to be in 0,1. All zero weighted items are excluded from t-test. You can turn this off by setting this NULL. This option may be used together with bacgroundCorrect method or/and with beadarray QC (defaults to "bgf").
channelOutput	Output from VST.

Author(s)

Vojtěch Kulvait

`vstFromLumi` *Function from LGPL lumi package 2.16.0*

Description

This function is derived from copy and paste of lumi::vst function. Since lumi package has extensive imports I decided to hardcode this function to the blima instead of importing lumi package.

Usage

```
vstFromLumi(u, std, nSupport = min(length(u), 500), backgroundStd = NULL,
    lowCutoff = 1/3)
```

Arguments

u	The mean of probe beads
std	The standard deviation of the probe beads
nSupport	Something for c3 guess.
backgroundStd	Estimate the background variance c3. Input should be variance according to article, not SD.
lowCutoff	Something for c3 guess.

Author(s)

authors are Pan Du, Simon Lin, the function was edited by Vojtěch Kulvait

References

http://www.bioconductor.org/packages/release/bioc/html/lumi.html

writeBackgroundImages *Write Background Images*

Description

This function writes images with background distribution according to foreground before and after background subtraction.

Usage

```
writeBackgroundImages(b, spotsToGenerate = NULL, imageType = c("jpg",
    "png", "eps"), channelForeground = "GrnF", channelBackground = "GrnB",
    SDMultiple = 3, includePearson = FALSE, outputDir = getwd(),
    width = 505, height = 505)
```

Arguments

b	Single beadLevelData object.
spotsToGenerate	NULL for generate images for all spots from b. Otherwise specifies logical vector of the length equals to the number of arrays in b with TRUE for images to generate.
imageType	Type of images produced, either jpg, png or eps
channelForeground	Name of channel of foreground.
channelBackground	Name of channel of background.
SDMultiple	Correct on this level.
includePearson	Include Pearson corelation.
outputDir	Directory where to output images.
width	Width of image (default 505 fits well for 86mm 150dpi illustration in Bioinformatics journal:)
height	Height of image

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && interactive())
{
   #Write background images before and after correction for background into /tmp directory. This function create
    data(blimatesting)
    p = pData(blimatesting[[2]]@experimentData$phenoData)
    spotsToGenerate = p$Group %in% "D";
   writeBackgroundImages(blimatesting[[2]], imageType="jpg", spotsToGenerate=spotsToGenerate, includePearson
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

xieBacgroundCorrect *Xie background correct.*

Description

Background correction according to non parametric estimator in Xie, Yang, Xinlei Wang, and Michael Story. "Statistical Methods of Background Correction for Illumina BeadArray Data." Bioinformatics 25, no. 6 (March 15, 2009): 751-57. doi:10.1093/bioinformatics/btp040.###The method is applied on the bead level.

Usage

```
xieBacgroundCorrect(b, normalizationMod = NULL, negativeArrayAddresses,
    channelCorrect, channelResult, channelInclude = NULL)
```

Arguments

b	List of beadLevelData objects (or single object).
normalizationMod	NULL for processing all spots in b. Otherwise specifies logical vector of the length equals to the number of arrays in b.
negativeArrayAddresses	Vector of addresses of negative control probes on array
channelCorrect	Slot to perform convolution correction.
channelResult	Result channel, if this channel exists it will be overwritten.
channelInclude	This field allows user to set channel with weights which have to be from 0,1. All zero weighted items are excluded from summarization. You can turn this off by setting this NULL. This option may be used together with bacgroundCorrect method or/and with beadarray QC (defaults to NULL).

Author(s)

Vojtěch Kulvait

Examples

```
if(require("blimaTestingData") && exists("annotationHumanHT12V4") && interactive())
{
    #Create vector of negative array addresses.
   negAdr = unique(annotationHumanHT12V4$Controls[annotationHumanHT12V4$Controls$Reporter_Group_Name=="negat
    #Create summarization of nonnormalized data from GrnF column.
    data(blimatesting)
    blimatesting = bacgroundCorrect(blimatesting, channelBackgroundFilter="bgf")
   blimatesting = nonPositiveCorrect(blimatesting, channelCorrect="GrnF", channelBackgroundFilter="bgf", cha
   blimatesting = xieBacgroundCorrect(blimatesting, negativeArrayAddresses=negAdr, channelCorrect="GrnF", cha
   #Prepare logical vectors corresponding to conditions A(groups1Mod), E(groups2Mod) and both(processingMod).
   xiecorrected = createSummarizedMatrix(blimatesting, quality="GrnFXIE", channelInclude="bgf",
             annotationTag="Name")
    head(xiecorrected)
}else
{
   print("To run this example, please install blimaTestingData package from bioconductor by running BiocManager
}
```

xieBacgroundCorrectSingleArray *INTERNAL FUNCTION Xie background correct.*

Description

INTERNAL This function is not intended for direct use. Background correction according to non parametric estimator in Xie, Yang, Xinlei Wang, and Michael Story. "Statistical Methods of Background Correction for Illumina BeadArray Data." Bioinformatics 25, no. 6 (March 15, 2009): 751-57. doi:10.1093/bioinformatics/btp040. The method is applied on the bead level.

Usage

```
xieBacgroundCorrectSingleArray(b, normalizationMod = NULL, negativeArrayAddresses,
    channelCorrect, channelResult, channelInclude = NULL)
```

Arguments

b	Single beadLevelData object.
normalizationMod	NULL for processing all spots in b. Otherwise specifies logical vector of the length equals to the number of arrays in b.
negativeArrayAddresses	Vector of addresses of negative control probes on array
channelCorrect	Slot to perform convolution correction.
channelResult	Result channel, if this channel exists it will be overwritten.
channelInclude	This field allows user to set channel with weights which have to be from 0,1. All zero weighted items are excluded from summarization. You can turn this off by setting this NULL. This option may be used together with bacgroundCorrect method or/and with beadarray QC (defaults to NULL).

Author(s)

Vojtěch Kulvait

Index

∗ **package**
blima-package, 3

aggregateAndPreprocess, 3

bacgroundCorrect, 4
bacgroundCorrectSingleArray, 5
backgroundChannelSubtract, 6
backgroundChannelSubtractSingleArray, 7
blima *(*blima-package*)*, 3
blima-package, 3

channelExistsIntegrityWithLogicalVectorList, 7
checkIntegrity, 8
checkIntegrityLogical, 9
checkIntegrityOfListOfBeadLevelDataObjects, 9
checkIntegrityOfSingleBeadLevelDataObject, 10
chipArrayStatistics, 10
createSummarizedMatrix, 11

doAction, 12
doProbeTTests, 13
doTTests, 14

filterBg, 16

getNextVector, 17

initMeanDistribution, 17
insertColumn, 18
interpolateSortedVector, 18
interpolateSortedVectorRcpp_, 19

log2TransformPositive, 19

meanDistribution, 20

nonParametricEstimator, 21
nonPositiveCorrect, 22
nonPositiveCorrectSingleArray, 23
numberOfDistributionElements, 23

performXieCorrection, 24
plotBackgroundImageAfterCorrection, 24
plotBackgroundImageBeforeCorrection, 25

quantileNormalize, 26

readToVector, 27

selectedChannelTransform, 28
selectedChannelTransformSingleArray, 29
singleArrayNormalize, 29
singleChannelExistsIntegrityWithLogicalVector, 30
singleCheckIntegrityLogicalVector, 31
singleNumberOfDistributionElements, 31

updateMeanDistribution, 32

varianceBeadStabilise, 32
varianceBeadStabiliseSingleArray, 33
vstFromLumi, 34

writeBackgroundImages, 35

xieBacgroundCorrect, 36
xieBacgroundCorrectSingleArray, 37