Package ‘ChAMP’

December 21, 2016

Type Package

Title Chip Analysis Methylation Pipeline for Illumina HumanMethylation450 and EPIC

Version 2.4.1

Date 2016-12-09

Description The package includes quality control metrics, a selection of normalization methods and novel methods to identify differentially methylated regions and to highlight copy number alterations.

License GPL-3

VignetteBuilder knitr

Depends R (>= 3.3), minfi, ChAMPdata (>= 2.0), FEM (>= 3.1), DMRcate, Illumina450ProbeVariants.db, IlluminaHumanMethylationEPICmanifest

Imports sva, IlluminaHumanMethylation450kmanifest, limma, RPMM, DNAcopy, preprocessCore, impute, marray, wateRmelon, plyr, GenomicRanges, RefFreeEWAS, qvalue, isva, doParallel, bumphunter, quadprog, shiny, shinythemes, plotly, RColorBrewer, matrixStats

biocViews Microarray, MethylationArray, Normalization, TwoChannel, CopyNumber, DNAMethylation

Suggests knitr, rmarkdown

Author Yuan Tian [cre,aut], Tiffany Morris [ctb], Lee Stirling [ctb], Andrew Feber [ctb], Andrew Teschendorff [ctb], Ankur Chakravarthy [ctb]

Maintainer Yuan Tian <champ450k@gmail.com>

NeedsCompilation no

LazyData true

R topics documented:

ChAMP-package .................................................. 2
Block.GUI ..................................................... 3
champ.Block ................................................... 5
champ.CNA ..................................................... 6
A pipeline that enables pre-processing of 450K or EPIC data, a selection of normalization methods and a bundle of analysis method including SVD checking, Batch effect correction, DMP, DMR, Block detection, Cell proportion detection, GSEA pathway detection, EpiMod module detection, and copy number variance detection. ChAMP provided a very comprehensive analysis pipeline for EPIC or 450K data set.

The full analysis pipeline can be run with all defaults using champ.process() Alternatively, it can be run in steps using all functions separately.

Yuan Tian, Tiffany Morris, Lee Stirling, Andy Feber, Andrew Teschendorff, Ankur Chakravarthy, Stephen Beck

Maintainer: Yuan Tian <champ450k@gmail.com>
 directory = system.file('extdata', package = 'ChAMPdata')
champ.process(directory = directory)
  ### run champ functions separately.
myLoad <- champ.load(directory)
myImpute <- champ.impute()
champ.QC()
myNorm <- champ.norm()
champ.SVD()
myCombat <- champ.runCombat()
myDMP <- champ.DMP()
myDMR <- champ.DMR()
myBlock <- champ.Block()
myGSEA <- champ.GSEA()
myEpiMod <- champ.EpiMod()
myCNA <- champ.CNA()
myRefFree <- champ.reffree()
myRefbase <- champ.refbase() ### for blood sample only

CpG.GUI()
QC.GUI()
DMP.GUI()
DMR.GUI()
Block.GUI()

---

**Block.GUI**

*Generate interactive plot for the result of champ.Block() function,*

**Description**

A Shiny, Plotly and Web Browser based analysis interface. *Block.GUI()* is aimed to provide a comprehensive interactive analysis platform for the result of *champ.Block()*. The left panel indicate parameters user may be used to select significant Block, here I only provided minium number of clusters and p value as two threshold cutoff. After opening this web page, user may select their cutoff, then press submit, the webpage would calculate the result automatically. User could check the Blocktable in first tab easily, users can rank and select certain genes in the table, the content of the table might be changed based on the cutoff you selected in left panel. The second tab provide the mapping information from CpGs to Blocks, which will makes your easier to find connection between CpGs to clusters then Blocks. The third tab is the plot of Block and the clusters’ differential methylation information, you may search the Block you want to check by left panel, note that if there is only one significant cluster in the Block you selected, the plot might not be show properly.

**Usage**

```r
Block.GUI(Block = myBlock,
          beta = myNorm,
          pheno = myLoad$pd$Sample_Group,
          runDMP = TRUE,
          compare.group = NULL,
          arraytype = "450K")
```
Arguments

Block  The result from champ.Block(). (default = myBlock)

beta  A matrix of values representing the methylation scores for each sample (M or B). Better to be imputed and normalized data. (default = myNorm)

pheno  This is a categorical vector representing phenotype of factor wish to be analysed, for example "Cancer", "Normal"... Tow or even more phenotypes are allowed. (default = myLoad$pd$Sample_Group)

runDMP  If DMP result sould be calculated and combined into the result of CpGs annotation.

compare.group  compare.group is a parameter to assign which two phenotypes you wish to analysis, if it’s missed(NULL) or can not fulfill the condition of the dataset, the first two phenotypes in your pheno would be selected as compare.group automatically. (default = NULL)

arraytype  Choose microarray type is 450K or EPIC. (default = "450K")

Value

Totally three tabs would be generated on opened webpage.

Blocktable  The Block list of all significant Blocks selected by cutoff in left panel.

CpGtable  Information of all significant CpGs selected by cutoff in left panel. More importantly, it also contains mapping information each between CpG ID, Cluster ID and Block ID.

BlockPlot  Dots and lines of all clusters involved in one Block, the xaix is based on real Map information of clusters. Above the plot, is the differential methylation information of clusters contained in this Block.

Note

Please make sure you are running R locally or connected with local graph software(X11) remotely.

Author(s)

Yuan Tian

Examples

```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
myBlock <- champ.Block()
Block.GUI()

## End(Not run)
```
champ.Block

Identify Differential Methylation Blocks in Illumina HumanMethylation450 or HumanMethylationEPIC data.

Description

This function would detect all methylation Blocks exist in your dataset, methylation Block should be calculated based on the average value of clusters across whole genome. Firstly champ.Block would calculate all clusters in the dataset with clustermaker() function provided by Bumphunter package. Then, only OpenSea Clusters would be picked out to calculate Block. Eventually, Bumphunter Algorithms would be applied on averaged clusters to get final Blocks.

Usage

champ.Block(beta=myNorm, pheno=myLoad$pd$Sample_Group, arraytype="450K", maxClusterGap=250000, B=500, bpSpan=250000, minNum=10, cores=3)

Arguments

beta A matrix of values representing the methylation scores for each sample (M or B). Better to be imputed and normalized data. (default = myNorm)

pheno This is a categorical vector representing phenotype of factor wish to be analysed, for example "Cancer", "Normal"... Tow or even more phenotypes are allowed. (default = myLoad$pd$Sample_Group)

arraytype Choose microarray type is 450K or EPIC. (default = "450K")

maxClusterGap Max gap between clusters when calculating region at first step. (default = 250000)

B An integer denoting the number of resamples to use when computing null distributions. If permutations is supplied that defines the number of permutations/bootstrap and B is ignored. (default = 250)

bpSpan The maximum length for a Block should be detected, regions longer then this would be discarded. (default = 250000)

minNum Threshold to filtering Blocks with too few probes in it. After region detection, champ.Block will only select Blocks contain more than minNum clusters(OpenSea Regions) to continue the program. (default = 10)

cores The embeded DMR detection function, bumphunter, could automatically use more parallel to accelerate the program. User may assign number of cores could be used on users’s computer. User may use detectCore() function to detect number of cores in total. (default = 3)
champ.CNA

Value

Block
A data.frame contains all detected Blocks, with colnames as chr, start, end, value, area, cluster, indexStart, indexEnd, L, clusterL, p.value, fwer, p.valueArea, fwerArea. The result format is actually the same as Bumphunter, you may refer to Bumphunter packages to get more explanation about the result.

clusterInfo
When champ.Block() detection significant Blocks, a group of candidate Blocks would be detected out at first, this is the data frame of all candidate Blocks. The “TRUE” Blocks in above value are located in these candidate Blocks.

allCLID.v
The first step of detecting methylation Blocks is to get each probes into a cluster(region). This value is the clustering result of each probes.

avbetaCL.m
The beta matrix for each cluster. The value is calculated by taking mean value of all probes located in each cluster.

posCL.m
Position of each cluster, which is calculated by average all probes’ position in each cluster.

Note
The internal structure of the result of champ.Block() function should not be modified if it’s not necessary caused it would be assigned as inpute for some other functions like Block.GUI(). You can try to use Block.GUI() to do interactively analysis on the result of champ.Block().

Author(s)
Yuan Tian

References

Examples
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
myBlock <- champ.Block()
Block.GUI()
## End(Not run)
champ.CNA

Description

This function enables CNA profiles to be built using methylation data from Illumina HumanMethylation450K and HumanMethylationEPIC BeadChips. This function provide options to find Copy Number Abberrations between two phenotype (e.g. Cancer & Normal), or the function would take the average value of your dataset as control and detect if some value are out of average status. For user want to detect abberrations between phenotypes, they can specify controlGroup in parameter, or they can simply used packaged dataset as control. Two kinds of plot would be returned, the abberrations of each sample, and the abberrations of each phenotype. The older version of ChAMP provide batchcorrect for intensity dataset, but it’s no longer provided here, user may use champ.runCombat() function to correct batch effect just like they correct beta matrix.

Usage

champ.CNA(intensity=myLoad$intensity,
          pheno=myLoad$pd$Sample_Group,
          control=TRUE,
          controlGroup="champCtls",
          sampleCNA=TRUE,
          groupFreqPlots=TRUE,
          Rplot=FALSE,
          PDFplot=TRUE,
          freqThreshold=0.3,
          resultsDir="./CHAMP_CNA",
          arraytype="450K")

Arguments

intensity A matrix of intensity values for each sample. (default = myLoad$intensity)
pheno This is a categorical vector representing phenotype of factor wish to be analysed, for example "Cancer", "Normal"...Tow or even more phenotypes are allowed. (default = myLoad$pd$Sample_Group)
control If champ.CNA() should calculate the difference between groups(controls and case) of not(with average). (default = TRUE)
controlGroup which phenotype in your pheno parameter shall be treated as control type is you want to comparision between two groups. If this value was missing or invalid, the function would automatically use packaged Blood sample(champCtls) as control. (default = "champCtls")
sampleCNA If sampleCNA=TRUE, then each sample’s Copy Number Abberrations would be calculated and plotted. (default = TRUE)
groupFreqPlots If groupFreqPlots=TRUE, then each group’s Copy Number Abberrations Frequency would be calculated and plotted. (default = TRUE)
freqThreshold If groupFreqPlots=T, then freqThreshold will be used as the cutoff for calling a gain or loss. (default = 0.3)
PDFplot If PDFplot would be generated and save in resultsDir. (default = TRUE)
Rplot If Rplot would be generated and save in resultsDir. Note if you are doing analysis on a server remotely, please make sure the server could connect your local graph applications. (For example X11 for linux.) (default = TRUE)
arraytype Choose microarray type is 450K or EPIC.
resultsDir The directory where PDF files would be saved. (default = "/CHAMP_CNA")
Value

sampleResult  The Copy Number Aberrations result calculated and ploted for each Sample.
groupResult   The Copy Number Aberrations result calculated and ploted for each Group.

Author(s)

Feber, A
adapted by Yuan Tian

References


Examples

## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myCNA <- champ.CNA()
## End(Not run)

champ.DMP

Identify Differential Methylation Positions (DMP) in Illumina Human-Methylation450 or HumanMethylationEPIC data.

Description

This function would use limma package to calculate differential methylation probes between two phenotypes. It’s easy to use but do remember to specify which two phenotypes you want to calculate in compare.group parameter. Note that, if the compare.group parameter is NULL, or the factor in it are not find in pheno, the first two phenotypes would be analysed automatically. Note that the result of champ.DMR() would be used as inpute of champ.GSEA() function, thus we suggest user not change the internal structure of the result of champ.DMR() function.

Usage

champ.DMP(beta = myNorm,
pheno = myLoad$pd$Sample_Group,
adjPVal = 0.05,
adjust.method = "BH",
compare.group = NULL,
arraytype = "450K")

Arguments

beta        A matrix of values representing the methylation scores for each sample (M or B). Better to be imputed and normalized data. (default = myNorm)
pheno       This is a categorical vector representing phenotype of factor wish to be analysed. for example "Cancer", "Normal"... Tow or even more phenotypes are allowed. (default = myLoad$pd$Sample_Group)
champ.DMR

adjPVal The minimum threshold of significance for probes to be considered an DMP. (default = 0.05)
adjust.method The p-value adjustment method to be used for the limma analysis, (default= BH (Benjamini-Hochberg))
compare.group compare.group is a parameter to assign which two phenotypes you wish to analysis, if it’s missed(NULL) or can not fulfill the condition of the dataset, the first two phenotypes in your pheno would be selected as compare.group automatically. (default = NULL)
arraytype Choose microarray type is 450K or EPIC. (default = "450K")

Value

DMP A data frame of all probes with an adjusted p-value for significance of differential methylation containing columns for logFC, AveExpr, t.P.Value, adj.P.Val, B, C_AVG, T_AVG, deltaBeta, CHR, MAPINFO, Strand, Type, gene, feature, cgi, feat.cgi, UCSC_CpG_Islands_Name, DHS, Enhancer, Phantom, Probe_SNPs, Probe_SNPs_10

Note

The internal structure of the result of champ.DMP() function should not be modified if it’s not necessary caused it would be assigned as inpute for some other functions like DMP.GUI(), champ.DMR() or champ.GSEA(). You can try to use DMP.GUI() to do interactively analysis on the result of champ.DMP().

Author(s)

Yuan Tian

Examples

## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
myDMP <- champ.DMP()
DMP.GUI()
## End(Not run)

champ.DMR Applying Bumphunter, DMRcate or ProbeLasso Algorithms to detect Different Methylation Regions in a beta valued Methylation Dataset.

Description

Applying Bumphunter, DMRcate or ProbeLasso Algorithms to Estimate regions for which a genomic profile deviates from its baseline value. Originally implemented to detect differentially methylated genomic regions between two populations. By default, we recommend user do champ.DMR on normalized beta value on two populations, like case to control. The function will return detected DMR and estimated p value. The three algorithms specified in this function is different, while Bumphunter and DMRcate calculated averaged candidate bumps methylation value between case
and control, ProbeLasso need Different Methylated Probes (DMP) from champ.DMP as input parameter and find DMRs around those DMPs. Thus parameters is different for three algorithms. Note that the result of champ.DMR() would be used as inpute of champ.GSEA() function, thus we suggest user not change the internal structure of the result of champ.DMR() function.

Usage

champ.DMR(beta=myNorm, 
    pheno=myLoad$pd$Sample_Group, 
    arraytype="450K", 
    method = "Bumphunter", 
    minProbes=7, 
    adjPvalDmr=0.05, 
    cores=3, 
    ## following parameters are specifically for Bumphunter method. 
    maxGap=300, 
    cutoff=0.5, 
    smooth=TRUE, 
    smoothFunction=loessByCluster, 
    useWeights=FALSE, 
    permutations=NULL, 
    B=250, 
    nullMethod="bootstrap", 
    ## following parameters are specifically for probe ProbeLasso method. 
    DMP=myDMP, 
    meanLassoRadius=375, 
    minDmrSep=1000, 
    minDmrSize=50, 
    adjPvalProbe=0.05, 
    Rplot=T, 
    PDFplot=T, 
    resultsDir="./CHAMP_ProbeLasso/", 
    ## following parameters are specifically for DMRcate method. 
    rmSNPCH=T, 
    dist=2, 
    mafcut=0.05, 
    lambda=1000, 
    C=2)

Arguments

Since there are three methods incorporated to detect DMRs, user may specify which function to do DMR detection, Bumphunter DMRcate or ProbeLasso. All three methods are available for both 450K and EPIC beadarray. But they are controlled by different parameters, thus users shall be careful when they specify parameters for corresponding algorithm. Parameters shared by three algorithms:

Methylation beta valued dataset user want to detect DMR. We recommend to use normalized beta value. In Bumphunter method, beta value will be transformed to M value. NA value is NOT allowed into this function, thus user may need to do some imputation work beforehand. This parameter is essential for both two algorithms. (default = myNorm)
champ.DMR

**phenom**
This is a categorical vector representing phenotype of factor wish to be analysed, for example "Cancer", "Normal"... Tow or even more phenotypes are allowed. (default = myLoad$pd$Sample_Group)

**arraytype**
Choose microarray type is 450K or EPIC. (default = "450K")

**method**
Specify the method users want to use to do DMR detection. There are three options: "Bumphunter", "DMRcate" or "ProbeLasso". (default = "Bumphunter").

**minProbes**
Threshold to filtering clusters with too few probes in it. After region detection, champ.DMR will only select DMRs contain more than minProbes to continue the program. (default = 7)

**adjPvalDmr**
This is the significance threshold for including DMRs in the final DMR list. (default = 0.05)

**cores**
The embedded DMR detection function, bumphunter and DMRcate, could automatically use parallel to accelerate the program. User may assign number of cores could be used on user's computer. User may use detectCore() function to detect number of cores in total. (default = 3)

Parameters specific for Bumphunter algorithm:

**maxGap**
The maximum length for a DMR should be detected, regions longer then this would be discarded. (default = 300)

**cutoff**
A numeric value. Values of the estimate of the genomic profile above the cutoff or below the negative of the cutoff will be used as candidate regions. It is possible to give two separate values (upper and lower bounds). If one value is given, the lower bound is minus the value. (default = 0.5)

**smooth**
A logical value. If TRUE the estimated profile will be smoothed with the smoother defined by smoothFunction. (default = TRUE)

**smoothFunction**
A function to be used for smoothing the estimate of the genomic profile. Two functions are provided by the package: loessByCluster and runmedByCluster. (default = loessByCluster)

**useWeights**
A logical value. If TRUE then the standard errors of the point-wise estimates of the profile function will be used as weights in the loess smoother loessByCluster. If the runmedByCluster smoother is used this argument is ignored. (default = FALSE)

**permutations**
is a matrix with columns providing indexes to be used to scramble the data and create a null distribution when nullMethod is set to permutations. If the bootstrap approach is used this argument is ignored. If this matrix is not supplied and B>0 then these indexes are created using the function sample. (default = NULL)

**B**
An integer denoting the number of resamples to use when computing null distributions. If permutations is supplied that defines the number of permutations/bootstrap and B is ignored. (default = 250)

**nullMethod**
Method used to generate null candidate regions, must be one of 'bootstrap' or 'permutation' (defaults to 'permutation'). However, if covariates in addition to the outcome of interest are included in the design matrix (ncol(design)>2), the 'permutation' approach is not recommended. See vignette and original paper for more information. (default = "bootstrap")

Parameters specific for ProbeLasso algorithm:

**DMP**
Different Methylated Probes (DMP) detected from champ.DMP() function, which used limma function to find all CpGs show significant different methylation value. It's a MUST provided parameter for ProbeLasso algorithm. (default = myDMP)
champ.DMR

### Parameters specific for DMRcate algorithm:

- **rmSNPCH**: Filters a matrix of M-values (or beta values) by distance to SNP. Also (optionally) removes crosshybridising probes and sex-chromosome probes. (default = TRUE)

- **dist**: Maximum distance (from CpG to SNP) of probes to be filtered out. See details for when Illumina occasionally lists a CpG-to-SNP distance as being < 0. (default = 2)

- **mafcut**: Minimum minor allele frequency of probes to be filtered out. (default = 0.05)

- **lambda**: Gaussian kernel bandwidth for smoothed-function estimation. Also informs DMR bookend definition; gaps >= lambda between significant CpG sites will be in separate DMRs. Support is truncated at 5*lambda. See DMRcate package for further info. (default = 1000)

- **C**: Scaling factor for bandwidth. Gaussian kernel is calculated where lambda/C = sigma. Empirical testing shows that when lambda=1000, near-optimal prediction of sequencing-derived DMRs is obtained when C is approximately 2, i.e. 1 standard deviation of Gaussian kernel = 500 base pairs. Cannot be < 0.2. (default = 2)

### Value

- **myDmrs**: A data.frame in a list contains Different Methylation Regions detected by champ.DMR. For different algorithms, myDmrs would be in different structure and named as "BumphunterDMR", "DMRcateDMR" and "ProbeLassoDMR". They may contain some different informations, caused by their method. However all three kinds of result are already suitable for champ.GSEA() analysis, so please don’t modify the structure if it’s not necessary.

### Note

The internal structure of the result of champ.DMR() function should not be modified if it’s not necessary caused it would be assigned as inpute for some other functions like champ.GSEA(). You can try to use DMR.GUI() to do interactively analysis on the result of champ.DMR().

### Note

The internal structure of the result of champ.DMR() function should not be modified if it’s not necessary caused it would be assigned as inpute for some other functions like DMR.GUI() and champ.GSEA(). You can try to use DMR.GUI() to do interactively analysis on the result of champ.DMR().
**champ.EpiMod**

**Author(s)**

Butcher, L., Aryee MJ, Irizarry RA, Andrew Teschendorff, Yuan Tian

**References**


**Examples**

```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
myDMR <- champ.DMR()
DMR.GUI()
## End(Not run)
```

**champ.EpiMod()** infer differential methylation interactome hotspots.

**Description**

champ.EpiMod() used FEM package to identify interactome hotspots of differential promoter methylation. By "interactome hotspot" we mean a connected subnetwork of the protein interaction network (PIN) with an exceptionally large average edge-weight density in relation to the rest of the network. The weight edges are constructed from the statistics of association of DNA methylation with the phenotype of interest. Thus, the EpiMod algorithm can be viewed as a functional supervised algorithm, which uses a network of relations between genes (in our case a PPI network), to identify subnetworks where a significant number of genes are associated with a phenotype of interest (POI). We call these "hotspots" also Functional Epigenetic Modules (FEMs). You can get more detailed information in FEM package.

**Usage**

```r
champ.EpiMod(beta=myNorm,
pheno=myLoad$pd$Sample_Group,
nseeds=100,
gamma=0.5,
nMC=1000,
sizeR.v=c(1,100),
minsizeOUT=10,
resultsDir="/CHAMP_EpiMod/",
PDFplot=TRUE,
arraytype="450K")
```
champ.EpiMod

Arguments

beta A matrix of values representing the methylation scores for each sample (M or B). Better to be imputed and normalized data. (default = myNorm)

pheno This is a categorical vector representing phenotype of factor wish to be analysed, for example "Cancer", "Normal"... Tow or even more phenotypes are allowed. (default = myLoad$pd$Sample_Group)

nseeds An integer specifying the number of seeds and therefore modules to search for. (default = 100)

gamma A parameter of the spin-glass algorithm, which determines the average module size. Default value generally leads to modules in the desired size range (10-100 genes). (default = 0.5)

nMC Number of Monte Carlo runs for establishing statistical significance of modularity values under randomisation of the molecular profiles on the network. (default = 1000)

sizeR.v Desired size range for modules. (default = c(1,100))

minsizeOUT Minimum size of modules to report as interesting. (default = 10)

resultsDir The directory where PDF files would be saved. (default = "./CHAMP_QCimages")

PDFplot If PDFplot would be generated and save in resultsDir. (default = TRUE)

arraytype Choose microarray type is 450K or EPIC. (default = "450K")

Value

EpiMod.o A data frame of all probes with an adjusted p-value for significance of differential methylation containing columns for logFC, AveExpr, t, P.Value, adj.P.Val, B, C_AVG, T_AVG, deltaBeta, CHR, MAPINFO, Strand, Type, gene, feature, cgi, feat.cgi, UCSC_CpG_Islands_Name, DHS, Enhancer, Phantom, Probe_SNPs, Probe_SNPs_10, you can turn to FEM package for more informations.

Author(s)

Teschendorff, A
adapted by Yuan Tian

References


Examples

```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
myEpiMod <- champ.EpiMod()

## End(Not run)
```
Do GSEA for DMP, DMR and other methylation data related results.

Description

This function would do GSEA on the results of champ functions like DMP and DMR. However users may also add individual CpGs and genes in it.

Usage

champ.GSEA(beta=myNorm, 
DMP=myDMP, 
DMR=myDMR, 
CpGlist=NULL, 
Genelist=NULL, 
arraytype="450K", 
adjPval=0.05)

Arguments

beta A matrix of values representing the methylation scores for each sample (M or B). Better to be imputed and normalized data. (default = myNorm)

DMP Results from champ.DMP() function. (default = myDMP)

DMR Results from champ.DMR() function. (default = myDMR)

CpGlist Apart from previous parameters, if you have any other CpGs list want to do GSEA, you can input them here as a list. (default = NULL)

Genelist Apart from previous parameters, if you have any other Gene list want to do GSEA. you can input them here as a list. (default = NULL)

arraytype Which kind of array your data set is? (default = "450K")

adjPval Adjusted p value cutoff for all calculated GSEA result. (default = 0.05)

Value

Block A data.frame contains all detected Blocks, with colnames as chr, start, end, value, area, cluster, indexStart, indexEnd, L, clusterL, p.value, fwer, p.valueArea, fwerArea. The result format is actually the same as Bumphunter, you may refer to Bumphunter packages to get more explanation about the result.

clusterInfo When champ.Block() detection significant Blocks, a group of candidate Blocks would be detected out at first, this is the data frame of all candidate Blocks. The "TRUE" Blocks in above value are located in these candidate Blocks.

allCLID.v The first step of detecting methylation Blocks is to get each probes into a cluster(region). This value is the clustering result of each probes.

avbetaCL.m The beta matrix for each cluster. The value is calculated by taking mean value of all probes located in each cluster.

posCL.m Position of each cluster, which is calculated by average all probes’ position in each cluster.

adjPval P value cutoff for calculated GSEA results. (default = 0.05)
champ.impute

Author(s)

Yuan Tian

Examples

```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
myDMP <- champ.DMP()
myDMR <- champ.DMR()
myGSEA <- champ.GSEA()

## End(Not run)
```

champ.impute  Conduct imputation for NA value on beta matrix and corresponding pd(Sample_sheet.csv) file.

Description

champ.impute will conduct imputation on beta matrix contains missing value. This function can be used for any beta dataset, along with their corresponding pd files. If you loaded this file with champ.load(), champ.impute() function will automatically loaded myLoad$beta as inputted beta matrix, while take myLoad$pd as pd input. There are totally three method provided in champ.impute() function. "Delete" is simply remove all NA related CpGs and Samples contain certian proportion of missing value, which is suitable for Small DataSets. "KNN" method use impute.knn() function from "impute" to do imputation on all missing value, which is rather popular but would cause trouble if DataSets contains few samples, no CpGs or samples woule be deleted. "Combine" method would remove all Samples and CpGs with certian proportions of missing value, then do KNN imputation for the rest (Default).

Usage

```r
champ.impute(beta=myLoad$beta, 
    pd=myLoad$pd, 
    method="Combine", 
    k=5, 
    ProbeCutoff=0.2, 
    SampleCutoff=0.1)
```

Arguments

- **beta**: Data matrix want to be imputed, user can input M matrix or intensity matrix even. (default = myLoad$beta)
- **pd**: Phenotype file for your data set. It’s optional for this function, but if during imputation some samples contain too many NA values dicarded, your old pd file might not be able to work for imputed data properly any more. (default = myLoad$pd)
- **method**: Imputation method optional, only "Combine","KNN","Delete" are feasible. (default = "Combine").
- **k**: Number of neighbors to be used in the imputation (default = 5)
champ.load

ProbesCutoff  Proportion of for probes shall be removed. Any probes with NA value proportion above this parameter will be removed. (default = 0.2)

SampleCutoff  Proportion of for Sample shall be removed. Any Sample with NA value proportion above this parameter will be removed. (default = 0.1)

Value

beta  The matrix get imputed

pd  The pd file corresponding to imputed matrix, if provided.

Author(s)

Yuan Tian

Examples

## Not run:
myLoad <- champ.load(directory=getwd(),
methValue="B",
filterDetP=TRUE,
detPcut=0.01,
removeDetP = 0,
filterBeads=TRUE,
beadCutoff=0.05,
filterNoCG=TRUE,
filterSNPs=TRUE,
filterMultiHit=TRUE,
filterXY=TRUE,
arraytype="450K")

## End(Not run)

champ.load  Upload of raw HumanMethylation450K or HumanMethylationEPIC data from IDAT files.

Description

Function that loads data from IDAT files to calculate intensity. Some kinds of filtering will be conducted as well such as unqualified CpGs, SNP, multihit sites, and XY chromosomes related CpGs.

Usage

champ.load(directory = getwd(),
methValue="B",
filterDetP=TRUE,
detPcut=0.01,
removeDetP = 0,
filterBeads=TRUE,
beadCutoff=0.05,
filterNoCG=TRUE,
filterSNPs=TRUE,
filterMultiHit=TRUE,
filterXY=TRUE,
arraytype="450K")
Arguments

directory Location of IDAT files, default is current working directory.(default = getwd())
methValue Indicates whether you prefer m-values M or beta-values B. (default = "B")
filterDetP If filter = TRUE, then probes above the detPcut will be filtered out.(default = TRUE)
detPcut The detection p-value threshold. Probes about this cutoff will be filtered out. (default = 0.01)
removeDetP The removeDetP parameter represents the fraction of samples that can contain a detection p-value above the detPcut.(default = 0)
filterBeads If filterBeads=TRUE, probes with a beadcount less than 3 will be removed depending on the beadCutoff value.(default = TRUE)
beadCutoff The beadCutoff represents the fraction of samples that must have a beadcount less than 3 before the probe is removed.(default = 0.05)
filterNoCG If filterNoCG=TRUE, non-cg probes are removed.(default = TRUE)
filterSNPs If filterSNPs=TRUE, probes in which the probed CpG falls near a SNP as defined in Nordlund et al are removed.(default = TRUE)
filterMultiHit If filterMultiHit=TRUE, probes in which the probe aligns to multiple locations with bwa as defined in Nordlund et al are removed.(default = TRUE)
filterXY If filterXY=TRUE, probes from X and Y chromosomes are removed.(default = TRUE)
arraytype Choose microarray type is "450K" or "EPIC".(default = "450K")

Value

mset mset object from minfi package, with filtering CpGs discarded.
rgSet rgset object from minfi package function read.metharray.exp(), contains all information of a .idat methylation dataset. If you want to do more analysis than functions provided by ChAMP, you can take this as a start point.
pd pd file of all sample information from Sample Sheet, which would be very frequently by following functions as DEFAULT input, thus it’s not very necessarily, please don’t modify it.
intensity A matrix of intensity values for all probes and all samples, the information would be used in champ.CNA() function. CpGs has been filtered as well.
beta A matrix of methylation scores (M or beta values) for all probes and all samples.
detP A matrix of detection p-values for all probes and all samples.

Author(s)

Yuan Tian

Examples

## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))

## End(Not run)
**champ.norm**

Normalization for HumanMethylation450 or HumanMethylationEPIC data

**Description**

Option to normalize data with a selection of normalization methods. There are four functions could be selected: "PBC","BMIQ","SWAN" and "FunctionalNormalize". SWAN method call for BOTH rgSet and mset input, FunctionNormalization call for rgset only, while PBC and BMIQ only needs beta value. Please set parameter correctly. BMIQ method is the default function, which would also return normalised density plots in PDF format in results Dir. FunctionalNormalize is provided in minfi package, which ONLY support 450K data yet. Not that BMIQ function might fail if you sample’s beta value distribution is not beta distribution, which occasionally happen when too many CpGs are deleted while loading .idat files with champ.load() function. Also multi-cores parallel is conductable for BMIQ function, if your server or computer is good enough with more than one cores, you may assign more cores like 10 to accelerate the process. No matter what method you selected, they all will return the same result: Normalize beta matrix with effect of Type-I and Type-II probes corrected.

**Usage**

champ.norm(beta=myLoad$beta, 
            rgSet=myLoad$rgSet, 
            mset=myLoad$mset, 
            resultsDir="./CHAMP_Normalization/", 
            method="BMIQ", 
            plotBMIQ=FALSE, 
            arraytype="450K", 
            cores=3)

**Arguments**

- **beta**: Original beta matrix waiting to be normalized. NA value are not recommended, thus you may want to use champ.impute to impute data first. colname of each sample MUST be marked. (default = myLoad$beta)
- **rgSet**: Original full information matrix from champ.load(), which is required by "SWAN" and "FunctionNormalization" method. (default = myLoad$rgSet)
- **mset**: mset object from minfi package, with filtering CpGs discarded, which is required by "SWAN" method. (default = myLoad$mset)
- **resultsDir**: The folder where champ.norm()'s PDF file should be saved. (default = "./CHAMP_Normalization/"
- **method**: Method to do normalization: "PBC","BMIQ","SWAN" and "FunctionalNormalize". (default = "BMIQ")
- **plotBMIQ**: If "BMIQ" method is chosen, should champ.norm() plot normalized plot in PDF and save it in resultsDir. (default = FALSE)
- **arraytype**: Choose microarray type is "450K" or "EPIC". (default = "450K")
- **cores**: If "BMIQ" method is chosen, how many cores shall be used to run parallel. (default = 3)
champ.process

Value

beta.p
A matrix of normalised methylation scores (M or beta values) for all probes and all samples.

Author(s)

Yuan Tian wrote the wrappers

References


Examples

## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()

## End(Not run)

champ.process

Process function to run all methods in ChAMP pipeline.

Description

This function allows the user to run the entire pipeline in one function. Arguments allow user to select functions if desired. Note that it maybe run during champ.process() if there is any problem during the process, thus run ChAMP functions one by one is actually recommended.

Usage

champ.process(runload=TRUE, directory = getwd(), filters=c("XY","DetP","Beads","NoCG","SNP","MultiHit"), runimpute=TRUE, imputemethod="Combine", runQC=TRUE, QCplots=c("mdsPlot","densityPlot","dendrogram"))

#---champ.impute parameters below---#
#---champ.QC parameters below---#
#---champ.norm parameters below---#
runnorm=TRUE,
normalizationmethod="BMIQ",
#---champ.SVD parameters below---#
runSVD=TRUE,
RGEEffect=FALSE,
#---champ.runCombat parameters below---#
runCombat=TRUE,
batchname=c("Slide"),
#---champ.DMP parameters below---#
runDMP=TRUE,
#---champ.DMR parameters below---#
runDMR=TRUE,
DMRmethod="Bumphunter",
#---champ.Block parameters below---#
runBlock=TRUE,
#---champ.GSEA parameters below---#
runGSEA=TRUE,
#---champ.EpiMod parameters below---#
runEpiMod=TRUE,
#---champ.CNA parameters below---#
runCNA=TRUE,
control=TRUE,
controlGroup="champCtls",
#---champ.reffree parameters below---#
runRefFree=TRUE,
#---champ.refbase parameters below---#
runRefBase=FALSE,
#---universal settings---#
compare.group=NULL,
adjPVal=0.05,
resultsDir="./CHAMP_RESULT/",
arraytype="450K",
PDFplot=TRUE,
Rplot=TRUE,
cores=3,
saveStepResults=TRUE

Arguments

runload	If champ.load() should be run? (default = TRUE)
directory	The folder directory of .idat files. (default = getwd())
filters	A character vector indicates filters should be done if load data from .idat files.
You can remove some of the filters in it if you don’t need that much. (default = c("XY","DetP","Beads","NoCG","SNP","MultiHit"))
runimpute	If champ.impute() should be run? Note that if your data contains too many NA,
champ.impute() may remove not only CpGs, but also samples. (default = TRUE)
imputemethod	Which imputation method should be applied into champ.impute().
runQC	If champ.QC() should be run? (default = TRUE)
QCplots	A character vector indicates plots should be drawn by champ.QC(). You can re-
move some plots in it if you don’t need them. (default = c("mdsPlot","densityPlot","dendrogram"))
runnorm  If champ.norm() should be run? (default = TRUE)

normalizationmethod
  Which normalization method should be selected by champ.norm().

rnSVD  If champ.SVD() should be run? (default = TRUE)

RGEffect  If Red Gree color Effect should be calculated in champ.SVD(). (default = FALSE)

runCombat  If champ.runCombat() should be run? (default = TRUE)

batchname  A character vector indicates what factors should be corrected by champ.runCombat().
  (default = c("Slide"))

runDMP  If champ.DMP() should be run? (default = TRUE)

runDMR  If champ.DMR() should be run? (default = TRUE)

DMRmethod  Which DMR method should be applied by champ.DMR()? (default = TRUE)

runBlock  If champ.Block() should be run? (default = TRUE)

runGSEA  If champ.GSEA() should be run? (default = TRUE)

runEpiMod  If champ.EpiMod() should be run? (default = TRUE)

runCNA  If champ.CNA() should be run? (default = TRUE)

ccontrol  If champ.CNA() should be calculate copy number variance between case and
c  control? (The other option for champ.CNA() is calculate copy number variance for
c  each sample to the averaged value). (default = TRUE)

ccontrolGroup  Which pheno should be treated as control group while running champ.CNA().
  (default = "champCtls")

runRefFree  If champ.reffree() should be run? (default = TRUE)

runRefBase  If champ.refbase() should be run? (default = TRUE)

compare.group  Which two phenos should be compared in champ.DMP()?

adjPVal  The adjusted p value for each function’s significant cutoff.

resultsDir  The directory where result should be stored. (default = "/CHAMP_RESULT/")

arraytype  If the data set under analysis is "450K" or "EPIC"? (default = "450K")

PDFplot  If PDF files should be plotted during running? (default = TRUE)

Rplot  If R plots should be plotted during running? (default = TRUE)

cores  How many cores should be used for parallel running during champ.process()? (default = 3)

saveStepresults  If result of each steps should be saved as .rd file into resultsDir folder? (default = TRUE)

Value

CHAMP_RESULT  A list contains all results from each champ.method.

Author(s)

Yuan Tian
Examples

## Not run:

directory=system.file("extdata",package="ChAMPdata")
champ.process(directory=directory)

## End(Not run)

champ.QC

Plot quality control plot, mdsplot, densityPlot, dendrogram for a data set.

Description

champ.QC() function would plot some summary plot for a dataset, including mdsplot, densityPlot, dendrogram. You may use QC.GUI() function to see even more plot interactively, like heatmap, Type-I and Type-II probes plot. Note that the dendrogram would do its best to modify plot size automatically, but if you have too many samples like 1000+, the speed would be slow and the plot might be hard to read.

Usage

champ.QC(beta = myLoad$beta,
pheno=myLoad$pd$Sample_Group,
mdsPlot=TRUE,
densityPlot=TRUE,
dendrogram=TRUE,
PDFplot=TRUE,
Rplot=TRUE,
Feature.sel="None",
resultsDir="./CHAMP_QCimages/")

Arguments

beta  beta matrix want to be analysed. NA value are not recommended, thus you may want to use champ.impute to impute data first. colname of each sample MUST be marked. (default = myLoad$beta)

pheno one Phenotype categorical vector for your dataset. NO list or dataframe or numeric. (default = myLoad$pd$Sample_Group)

mdsPlot If mdsPlot would be plotted. (default = TRUE)

densityPlot If densityPlot would be plotted. (default = TRUE)

dendrogram If dendrogram would be plotted. (default = TRUE)

PDFplot If PDFplot would be generated and save in resultsDir. (default = TRUE)

Rplot If Rplot would be generated and save in resultsDir. Note if you are doing analysis on a server remotely, please make sure the server could connect your local graph applications. (For example X11 for linux.) (default = TRUE)

Feature.sel Feature Selection method when champ.QC() calculate dendrogram. Two options are provided, "None" means no feature selection would be done, all probes would be used to calculate distance between each sample. "SVD" method means
champ.refbase

Applying References-Base Method to beta valued methylation data.

Description

Applying References-Based Method to correct cell-proportion in a methylation dataset. Reference-based method use purified whole blood cell-type specific methylation value to correct beta value dataset. Cell Proportions for each cell-type will be detected, and lm function will be used to correct beta value for 5 largest cell types. Cell type with smallest cell proportion will not be corrected.

Usage

champ.refbase(beta=myNorm, arraytype="450K")

Arguments

beta whole blood beta methylation dataset user want to correct. (default = myNorm)
arraytype There are two types of purified cell-type specific references can be chosen, "450K" and "27K". By default, 450K value will be used, but user may choose 27K as well. (default = myNorm)

Value

CorrectedBea A beta valued matrix, with all value get corrected with RefBaseEWAS method. Be aware, champ.refbase will only correct top 5 cell types with largest mean cell proportions, and leave the cell with smallest mean cell proportion. User may check CellFraction result to find out which cell types are get corrected.
CellFraction Proportion for each cell type.
champ.reffree

Author(s)
Houseman EA, Yuan Tian, Andrew Teschendorff

References

Examples
```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
myRefbase <- champ.refbase()
## End(Not run)
```

champ.reffree

Applying RefFreeEWAS Method to beta valued methylation data.

Description
Applying RefFreeEWAS method to beta valued methylation data. This method does not rely on purified cell reference, thus can be easily used on tissue data set, while RefbaseEWAS can only be used to whole blood samples. Reference-free method for conducting EWAS while deconvoluting DNA methylation arising as mixtures of cell types. This method is similar to surrogate variable analysis (SVA and ISVA), except that it makes additional use of a biological mixture assumption. Returns mixture-adjusted Beta and unadjusted Bstar, as well as estimates of various latent quantities.

Usage
```
champ.reffree(beta=myNorm,
              pheno=myLoad$pd$Sample_Group,
              K=NULL,
              nboot=50)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta</td>
<td>Methylation beta valued dataset user want to do RefFreeEWAS. (default = myNorm)</td>
</tr>
<tr>
<td>pheno</td>
<td>Phenotype user want to find differential methylation result. This parameter MUST be a vector or a matrix. Though Characters are allowed because inside the function, character covariates will be transformed into numeric, we still recommend user input numeric designed covariates matrix or vector. (default = myLoad$pd$Sample_Group)</td>
</tr>
<tr>
<td>K</td>
<td>Number of latent variable. If this value was ignored, function will use Random Matrix Theory from isva package to estimate latent variables. (default = NULL)</td>
</tr>
<tr>
<td>nboot</td>
<td>Number for Bootstrap on result of RefFreeEWAS. (default = 50)</td>
</tr>
</tbody>
</table>
champ.runCombat

Value

RefFreeEWASModel

RefFreeEWASModel S4 Object from RefFreeEWAS pacakge, contains adjusted beta value and unadjusted beta value (Bstar).

pvBeta

p value of each covariates, calculated from cell type mixture corrected Beta value.

qvBeta

q value of each covariates, calculated from cell type mixture corrected Beta value.

Author(s)

Houseman EA, Yuan Tian, Andrew Teschendorff

References


Examples

## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
myReffree <- champ.reffree()
## End(Not run)

champ.runCombat

Function that uses ComBat to correct for batch effects. Multiple batch effect correction is allowed.

Description

This function formats data to run through ComBat batch correction. If beta values are used the data is first logit transformed. Them Combat function from "sva" package would be used to do batch correction. Note that multi-batch correction is supported, user just need to assign name of batch need to be corrected. Note Combat function is a little bit critical to dataset, thus you have futher question or higher lever of application of Combat, you may turn to "sva" pacakge for help. After inputing pd file, champ.runCombat() would automatically detect all correctable factors and list them below, if your assigned batchname is correct, champ.runCombat() would start to do batch correction.

Usage

champ.runCombat(beta=myNorm,
    pd=myLoad$pd,
    batchname=c("Slide"),
    logitTrans=TRUE)
**champ.SVD**  

### Arguments

- **beta**: A matrix of values representing the methylation scores for each sample (M or B). (default = myNorm).
- **pd**: This data.frame includes the information from the sample sheet. (default = myLoad$pd).
- **batchname**: A character vector of name indicates which batch factors shall be corrected. (default = c("Slide"))
- **logitTrans**: If logitTrans=T then your data will be logit transformed before the Combat correction and inverse logit transformed after correction. This is T by default for Beta values but if you have selected M values, it should be FALSE. It is also FALSE when used with CNA as those are intensity values that don’t need to be transformed.

### Value

- **beta**: The matrix of values representing the methylation scores for each sample after ComBat batch correction.

### Author(s)

Yuan Tian

### Examples

```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
champ.SVD()
myCombat <- champ.runCombat()
## End(Not run)
```

---

**champ.SVD**  

*Singular Value Decomposition analysis for batch effects prediction in HumanMethylation450 or HumanMethylationEPIC data*

### Description

Runs Singular Value Decomposition on a dataset to estimate the impact of batch effects. This function would run SVD deconvolution on beta matrix, get components explain most variance in original data set. Then use Random Matrix Theory to estimate numbers of latent variables. Then each significant components would be correlated with each phenotype, to see if this phenotype show significant correlation with this component. All suitable factors in your pd(Sample_Sheet.csv) file will be analysed. After champ.SVD(), used would get a heatmap indicating effect of factors on original data set. And decide if some batch effect shall be corrected before future analysis. Not all factors in your pd file would be analysis though, name information like Sample_Name, Pool_ID... would be discarded, covariates contain less then 2 variances shall be discarded as well. Note that numeric covariates like age would be calculated with linear regression, while factors and character covariates like Sample_Group would be calculated with Krustal Test. Thus please check your input pd file carefully as well.
champ.SVD

Usage

champ.SVD(beta = myNorm,
          rgSet=myLoad$rgSet,
          pd=myLoad$pd,
          RGEffect=FALSE,
          PDFplot=TRUE,
          Rplot=TRUE,
          resultsDir="./CHAMP_SVDimages/")

Arguments

beta  beta matrix waiting to be analysed, better to be one get Probe-Type normalized and imputed. (default = myNorm)
rgSet An rgSet object that was created when data was loaded the data from the .idat files, which contains green and red color information of original data set, might be used if RGEffect set TRUE. (default = myLoad$rgSet)
pd This data.frame includes the information from the sample sheet. (default = myLoad$pd)
RGEffect If Green and Red color control probes would be calculated. (default = FALSE)
PDFplot If PDFplot would be generated and save in resultsDir. (default = TRUE)
Rplot If Rplot would be generated and save in resultsDir. Note if you are doing analysis on a server remotely, please make sure the server could connect your local graph applications. (For example X11 for linux.) (default = TRUE)
resultsDir The directory where PDF files would be saved. (default = "./CHAMP_SVDimages/"")

Author(s)

Teschendorff, A
adapted by Yuan Tian

References


Examples

```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata", package="ChAMPdata"))
myNorm <- champ.norm()
champ.SVD()

## End(Not run)
```
Description

A Shiny, Plotly and Web Browser based analysis interface. CpG.GUI() is aimed to generate summary of a list of CpGs. Feature distribution, CpG island distribution, e.g. It’s call for X11 similar graph software locally if you are doing analysis on server. Also the RAM memory might be large if you have a very big dataset. This function can be used anytime you have a list of CpGs from any analysis, you simply need to input the CpGs and specify the array type, a web browser interactive interface would be generated automatically. The plots are interactive thus you can make easier and better analysis on your data, and also download them at any size (jpg only).

Usage

CpG.GUI(CpG=rownames(myLoad$beta),
arraytype="450K")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CpG</td>
<td>A list of CpG you want to do plot summary. MUST be a vector with CpG ID. (default = rownames(myLoad$beta))</td>
</tr>
<tr>
<td>arraytype</td>
<td>Choose microarray type is 450K or EPIC. (default = &quot;450K&quot;)</td>
</tr>
</tbody>
</table>

Value

Totally four plots would be generated on opened webpage.

- chromosome_barplot: A chromosome barplot for the CpG list
- feature_barplot: A feature barplot for the CpG list
- cgi_barplot: A cgi barplot for the CpG list
- type_barplot: A type-I and type-II barplot for the CpG list

Note

Please make sure you are running R locally or connected with local graph software(X11) remotely.

Author(s)

Yuan Tian

Examples

```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
CpG.GUI()
## End(Not run)
```
DMP.GUI

Generate interactive plot for the result of champ.DMP() function.

Description

A Shiny, Plotly and Web Browser based analysis interface. DMP.GUI() is aimed to provide a comprehensive interactive analysis platform for the result of champ.DMP(). The left panel indicate parameters user may be used to select significant CpGs, here I only provided abslogFC and p value as two threshold cutoff. After opening this web page, user may select their cutoff, then press submit, the webpage would calculate the result automatically. User could check the DMPtable in first tab easily, users can rank and select certain genes in the table, the content of the table might be changed based on the cutoff you selected in left panel. The second tab provide the heatmap of all significant CpGs you selected, be careful that if there are too many CpGs, the memory consumption might be large. The third tab provide barplots of proportions of feature and CpGs in for your selected CpGs. The fourth tab is the plot of gene and the wikigene information of certain gene, you may search the gene you want to check by left panel, note that if there is only one significant CpG in the gene you selected, the plot might not be show properly. The last panel provide a boxplot of CpGs and a gene enrichment plot, you may use this gene enrichment plot to find interesting genes.

Usage

```r
DMP.GUI(DMP=myDMP,
beta=myNorm,
pheno=myLoad$pd$Sample_Group)
```

Arguments

- **DMP**: The result from champ.DMP(). (default = myDMP)
- **beta**: A matrix of values representing the methylation scores for each sample (M or B). Better to be imputed and normalized data. (default = myNorm)
- **pheno**: This is a categorical vector representing phenotype of factor wish to be analysed, for example "Cancer", "Normal"... Tow or even more phenotypes are allowed. (default = myLoad$pd$Sample_Group)

Value

Totally five tabs would be generated on opened webpage.

- **DMPtable**: The DMP list of all significant CpGs selected by cutoff in left panel.
- **Heatmap**: Heatmap of all significant CpGs selected by cutoff in left panel.
- **Feature&CpG**: Barplot of feature and Cgi information for all significant CpGs selected by cutoff in left panel.
- **Gene**: Dots and lines of all significant CpGs involved in one gene, the distance between CpGs are equal, and the feature and Cgi information are marked down the plot. Below the plot, is the wikigene information extracted from website.
- **CpG**: Boxplot for CpGs you want to check, you can search CpGs based on the left panel. Below is the gene enrichment plot, hyper CpGs and hyper CpGs are separated.
Note

Please make sure you are running R locally or connected with local graph software (X11) remotely.

Author(s)

Yuan Tian

Examples

```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
myDMP <- champ.DMP()
DMP.GUI()
## End(Not run)
```

**DMR.GUI**

*Generate interactive plot for the result of champ.DMR() function.*

**Description**

A Shiny, Plotly and Web Browser based analysis interface. DMR.GUI() is aimed to provide a comprehensive interactive analysis platform for the result of champ.DMR(). The last panel indicate parameters user may be used to select significant DMRs, here I only provided minprobes and p value as two threshold cutoff. After opening this web page, user may select their cutoff, then press submit, the webpage would calculate the result automatically. User could check the DMR table in first tab easily, users can rank and select certain genes in the table, the content of the table might be changed based on the cutoff you selected in left panel. The second tab is the CpG table, which extract all CpGs involved in selected CpGs. Note that maybe not all CpGs are DMPs. The third tab provide the plot of the DMR, just like gene plot in DMP.GUI(). Above the plot are CpGs information involved in this DMR. The fourth panel provide a heatmap of all CpGs involved in significant DMRs, and a gene enrichment plot. Both plot maybe not very clear to look, but user may zoom in for these two plots. Again be careful if you have a very big dataset. Note that the runDMP parameters will indicate if DMR.GUI() shall calculated DMP for all CpGs, which may cause slight different in the CpG table and the gene enrichment plot. And though there are three ways to calculate DMR, all three results from champ.DMR() are applicable for this function. The title would changed automatically for different result.

**Usage**

```r
DMR.GUI(DMR=myDMR,
        beta=myNorm,
        pheno=myLoad$pd$Sample_Group,
        runDMP=TRUE,
        compare.group=NULL,
        arraytype="450K")
```
Arguments

DMR  The result from champ.DMR(), all three DMR methods’ result are supported. (default = myDMR)
beta A matrix of values representing the methylation scores for each sample (M or B). Better to be imputed and normalized data. (default = myNorm)
pheno This is a categorical vector representing phenotype of factor wish to be analysed, for example "Cancer", "Normal"... Tow or even more phenotypes are allowed. (default = myLoad$pd$Sample_Group)
runDMP If DMP result should be calculated and combined into the result of CpGs annotation.
compare.group compare.group is a parameter to assign which two phenotypes you wish to analysis, if it’s missed(NULL) or can not fulfill the condition of the dataset, the first two phenotypes in your pheno would be selected as compare.group automatically. (default = NULL)
arraytype Choose microarray type is 450K or EPIC. (default = "450K")

Value

Totally four tabs would be generated on opened webpage.

DMRtable  The DMR list of all significant DMR you selected by cutoff in left panel.
CpGtable  A CpGs annotation (with p value and t value if runDMP=TRUE) of all CpGs related with selected DMRs in tab 1.
DMRPlot Dots and lines of all significant CpGs involved in one DMR, the distance between CpGs are equal, and the feature and Cgi information are marked down the plot. Above the plot, is the CpGs list involved in this DMR.
Summary CpG enrichment gene barplot, hyper CpGs and hyper CpGs may be marked if runDMP=TRUE. Below is the heatmap for all significant DMRs related CpGs. Both plots maybe not that clear but zoomable.

Note

Please make sure you are running R locally or connected with local graph software(X11) remotely.

Author(s)

Yuan Tian

Examples

```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
myNorm <- champ.norm()
myDMR <- champ.DMR() # All three methods supported.
DMR.GUI()

## End(Not run)
```
Description

A Shiny, Plotly and Web Browser based analysis interface. QC.GUI() is aimed to provide mdsplot, densityPlot, Type-I&Type-II densityplot, dendrogram(no interactable) and heatmap for top 1000 variable CpGs. In the first tab, mdsplot are plotted based on the distance calculated by top 1000 variable CpGs. For dendrogram, if there are only less than 10 samples, the distance between samples are calculated by all CpGs, if there are more than 10 samples, QC.GUI() would apply SVD deconvolution on the dataset first then extract top significant components as latent variables and calculate distance between samples. For the heatmap, if your dataset contains less than 1000 CpGs, all CpGs would be plotted, but if your dataset contains more than 1000 CpGs, the top 1000 variable CpGs would be selected and plot.

Usage

QC.GUI(beta=myLoad$beta,
        pheno=myLoad$pd$Sample_Group,
        arraytype="450K")

Arguments

beta A matrix of values representing the methylation scores for each sample (M or B). Better to be imputed and normalized data. (default = myNorm)

pheno This is a categorical vector representing phenotype of factor wish to be analysed, for example "Cancer", "Normal"... Tow or even more phenotypes are allowed. (default = myLoad$pd$Sample_Group)

arraytype Choose microarray type is 450K or EPIC. (default = "450K")

Value

Totally five tabs would be generated on opened webpage.

mdsplot A mdsplot used to see the clustering result and similarity between samples.

TypeDensity A two-line density Plot indicate Type-I CpGs and Type-II CpGs.

QCplot Beta distribution of each sample. You may use it to check samples with low qualities.

Dendrogram Dendrogram of all samples. If there are only less than 10 samples, the distance between samples are calculated by all CpGs, if there are more than 10 samples, QC.GUI() would apply SVD deconvolution on the dataset first then extract top significant components as latent variables and calculate distance between samples.

heatmap Heatmap for top 1000 variable CpGs.

Note

Please make sure you are running R locally or connected with local graph software(X11) remotely.
Author(s)

Yuan Tian

Examples

```r
## Not run:
myLoad <- champ.load(directory=system.file("extdata",package="ChAMPdata"))
QC.GUI()

## End(Not run)
```
Index

*Topic **450k**
  Champ-package, 2
*Topic **Beadchip**
  Champ-package, 2
*Topic **Block**
  champ.Block, 5
*Topic **ComBat**
  champ.runCombat, 26
*Topic **DMR**
  champ.DMR, 9
*Topic **DNAMethylation**
  Champ-package, 2
*Topic **EPIC**
  Champ-package, 2
*Topic **EpiMod**
  champ.EpiMod, 13
*Topic **GSEA**
  champ.GSEA, 15
*Topic **HumanMethylation450**
  Champ-package, 2
*Topic **RefFreeEWAS**
  champ.reffree, 25
*Topic **RefbaseEWAS**
  champ.refbase, 24
*Topic **array**
  Champ-package, 2
*Topic **batch effects**
  champ.SVD, 27
*Topic **copynumber**
  champ.CNA, 6
*Topic **limma**
  champ.DMP, 8
*Topic **methylation**
  Champ-package, 2
*Topic **normalization**
  champ.norm, 19
*Topic **package**
  Champ-package, 2
*Topic **plotly**
  Block.GUI, 3
  CpG.GUI, 29
  DMP.GUI, 30
  DMR.GUI, 31
  QC.GUI, 33
  Block.GUI, 3
  Champ (ChAMP-package), 2
  Champ-package, 2
  champ.Block, 5
  champ.CNA, 6
  champ.DMP, 8
  champ.DMR, 9
  champ.EpiMod, 13
  champ.GSEA, 15
  champ.impute, 16
  champ.load, 17
  champ.norm, 19
  champ.process, 20
  champ.QC, 23
  champ.refbase, 24
  champ.reffree, 25
  champ.runCombat, 26
  champ.SVD, 27
  CpG.GUI, 29
  DMP.GUI, 30
  DMR.GUI, 31
  QC.GUI, 33