Package: COCONUT (via r-universe)

October 31, 2024

Type Package

Title COmbat CO-Normalization Using conTrols (COCONUT)

Version 1.0.2

Date 2017-09-18

Author Timothy E Sweeney, MD, PhD [aut,cre]

Maintainer Timothy E Sweeney <tes17@alumni.stanford.edu>

Depends stats

Suggests limma, parallel

Description Allows for pooled analysis of microarray data by batch-correcting control samples, and then applying the derived correction parameters to non-control samples to obtain bias-free, inter-dataset corrected data.

License GPL-3

LazyData TRUE

NeedsCompilation no

RoxygenNote 6.0.1

Repository CRAN

Date/Publication 2017-09-19 07:55:13 UTC

Contents

COCONUT-package *COmbat CO-Normalization Using conTrols: COmbat CO-Normalization Using conTrols (COCONUT)*

Description

Allows for pooled analysis of microarray data by batch-correcting control samples, and then applying the derived correction parameters to non-control samples to obtain bias-free, inter-dataset corrected data.

Details

The DESCRIPTION file:

Package:	COCONUT
Type:	Package
Title:	COmbat CO-Normalization Using conTrols (COCONUT)
Version:	1.0.2
Date:	2017-09-18
Author:	Timothy E Sweeney, MD, PhD [aut,cre]
Maintainer:	Timothy E Sweeney <tes17@alumni.stanford.edu>
Depends:	stats
Suggests:	limma, parallel
Enhances:	
Description:	Allows for pooled analysis of microarray data by batch-correcting control samples, and then applying th
License:	GPL-3
LazyData:	TRUE
NeedsCompilation:	no
RoxygenNote:	6.0.1
Packaged:	2017-09-18 23:03:57 UTC; timsweeney

Index of help topics:

COCONUT	COmbat CO-Normalization Using conTrols: COCONUT
COCONUT-package	COmbat CO-Normalization Using conTrols: COmbat CO-Normalization Using conTrols (COCONUT)
GSEs.test	COCONUT test data
combineCOCOoutput	Combine COCONUT output from multiple objects into a single object

Direct comparison of different microarray cohorts is impossible due both to inherent differences in underlying microarray platform and processing (technical) batch effects. In order to make use of these data, we need to co-normalize cohorts in such a way that (1) no bias is introduced (i.e., the normalization protocol should be blind to disease state); (2) there should be no change to the distribution of a gene within a study, and (3) a gene should show the same distributions between studies after normalization.

We thus developed a modified version of the ComBat empiric Bayes normalization method (Johnson et al., Biostatistics 2007) to co-normalize control samples from different cohorts to allow for direct comparison of diseased samples from those same cohorts. We call this method COmbat CO-Normalization Using conTrols, or 'COCONUT' . COCONUT makes one strong assumption, which is that it forces controls/healthy patients from different cohorts to represent the same distribution.

Briefly, all cohorts are split into the control and diseased components. The control components undergo ComBat co-normalization without covariates. The ComBat estimated parameters are obtained for each dataset for the control component, and then applied onto the diseased component. This forces the diseased components of all cohorts to be from the same background distribution, but retain their relative distance from the control component . Importantly, it also does not require any a priori knowledge of what type of disease is present in the diseased portion of the data. This method does have the notable requirement that controls/healthy patients are required to be present in a dataset in order for it to be pooled with other available data. Also, since control/healthy patients are set to be in the same distribution, it should only be used where such an assumption is reasonable (i.e., within the same tissue type, among the same species, etc.).

COCONUT requires a list of objects with two components, $gene and $pheno. It is assumed that each item in the list represents a different study, and that these have already been internally batch-corrected and normalized as appropriate. It is assumed that data object structure $gene is a matrix (genes in rows, samples in columns) and that $pheno is a data.frame (samples in rows, variables in columns). Note that COCONUT (like ComBat) requires identical rownames (genes) across all batches; so probes data will not work unless all matrices are from the same manufacturer (common probe names).

Also note that, unlike sva::ComBat, no co-variates are allowed.

Author(s)

Timothy E Sweeney, MD, PhD [aut,cre]

Maintainer: Timothy E Sweeney <tes17@alumni.stanford.edu>

References

Sweeney TE et al., "Robust classification of bacterial and viral infections via integrated host gene expression diagnostics", Science Translational Medicine, 2016

`COCONUT` *COmbat CO-Normalization Using conTrols: COCONUT*

Description

COCONUT is a modified version of the ComBat empiric Bayes batch correction method (Johnson et al., Biostatistics 2007). It allows for batch correction of microarray datasets using control samples, which allows for diseased samples to be compared in pooled analysis. It makes a strong assumption that all controls come from the same distribution.

Usage

```
COCONUT(GSEs, control.0.col, disease.col=NULL, byPlatform = FALSE, platformCol,
        par.prior=TRUE, itConv=1e-04, parallel=FALSE, mc.cores=1)
```

Arguments

GSEs	A list of data objects. See details below.
control.0.col	The column name in the $pheno data.frames (in GSEs) that notes which samples are controls. These samples MUST be marked with a 0 (zero).
disease.col	Optional; if passed, refers to a column name in the $pheno data.frames (in GSEs) from which disease samples are returned. Only checks to remove missing (NA) samples from disease.col. Useful if there is a class of samples that need to be removed from the analysis (i.e., samples that are not controls but also not the disease of interest). If NOT supplied, COCONUT assumes all non-0 rows in control.0.col are diseased.
byPlatform	Natively, byPlatform=F. If T, will group datasets by the batches found in platformCol.
platformCol	If byPlatform=T, platformCol is the name of a column in $pheno data.frames (in GSEs) that indicates platform type. For instance, in the data example, each $pheno has a $platform_id which contains that dataset's GPL ID. Note: the microarray ID type supplied should be constant within a column (but of course can vary between datasets).
par.prior	Whether to use parametric or non-parametric priors in empiric Bayes updates. Defaults to parametric. Non-parametric can be quite time-consuming.
itConv	Allows user to change threshold for iterative solver. For advanced users only.
parallel	Parallel derivation of priors. Uses parallel:mclapply, and so will not work on Windows machines (sorry).
mc.cores	If parallel=T, mc.cores should be set to the desired number of cores. Defaults to 1, so unless this is changed, functionality will be serial.

Details

GSEs: A list of (named) data objects. Each data object must have two components, $pheno (a data.frame of phenotype information with samples in rows and phenotype variables in columns), and $genes (a matrix of genes in rows and samples in columns). Further, the rownames of $pheno should match the colnames of $genes within each dataset. See the example data object for details.

byPlatform: Natively, COCONUT will assume each dataset in GSEs is a batch. However, there is enough similarity between microarrays (if the same normalization protocols are used) that each TYPE of microarray can be considered a batch. The advantage to this process is that datasets that share platforms can pool control samples, meaning datasets without controls can be potentially brought into the pool. The drawback is that there is still a substantial batch effect among datasets that used the same microarray type but were processed separately. Quantile normalization is used to overcome this to some degree, but it cannot be fixed altogether.

Value

COCONUT returns a list of lists. In the main list:

`COCONUTList` COCONUTList is itself a list, with the same names as the datasets in the input objects. Only diseased (or non-control) samples are passed back here (controls are dealt with separately, as below). The post-COCONUT-conormalized values are found in $COCONUTList[GSEname]$genes. See example below for single-line code to collapse these into a single matrix for pooled analysis.

`rawDiseaseList` rawDiseaseList is returned so that the user can make easy comparisons between pre- and post-COCONUT-co-normalized disease data. This contains the same data as the input object, except that all control samples have been removed.

`controlList` controlList returns the ComBat-normalized controls in $GSEs, and the derived empiric Bayes parameters in $bayesParams. It is generally assumed that these will be useful mainly for proving what COCONUT has done, etc., and not for downstream analyses. Note that this does NOT contain the non-co-normalized control data. To compare distributions, for example, you will need the original data object. See example below.

Warning : COCONUT makes the strong assumption that the control data are from the same distribution. This may not always be an appropriate assumption. Users are advised to think carefully about how to apply COCONUT locally.

Author(s)

Timothy E Sweeney, MD, PhD (tes17 [at] stanford [dot] edu)

References

Sweeney TE et al., "Robust classification of bacterial and viral infections via integrated host gene expression diagnostics", Science Translational Medicine, 2016

See Also

COCONUT-package

Examples

```
data(GSEs.test)

## apply COCONUT to a very small test case
## (3 datasets with 10 patients and 2000 genes)
GSEs.COCONUT <- COCONUT(GSEs=GSEs.test,
                        control.0.col="Healthy0.Sepsis1",
                        byPlatform=FALSE)

## make gene matrices
COCONUTgenes <- Reduce(cbind, lapply(GSEs.COCONUT$COCONUTList, function(x) x$genes))
rawgenes <- Reduce(cbind, lapply(GSEs.COCONUT$rawDiseaseList, function(x) x$genes))

### plot not run; (uncomment for plot)
```

```
### plot pre- and post-normalized data
# plot(x=1:ncol(COCONUTgenes), y=COCONUTgenes["ATP6V1B1", ], ylim=c(0,6), pch=20, col=1)
# points(x=1:ncol(rawgenes), y=rawgenes["ATP6V1B1", ], ylim=c(0,6), pch=20, col=2)


## compare distributions before and after COCONUT
classvec <- GSEs.test$GSE28750$pheno$Healthy0.Sepsis1
prior <- GSEs.test$GSE28750$genes
post <- cbind(GSEs.COCONUT$controlList$GSEs$GSE28750$genes,
              GSEs.COCONUT$COCONUTList$GSE28750$genes)

prior.t.stats <- apply(prior, 1, function(geneRow){
    geneByClass <- split(geneRow, classvec)
    gene.test <- t.test(geneByClass[[1]], geneByClass[[2]])
    gene.test$statistic
})

post.t.stats <- apply(post, 1, function(geneRow){
    geneByClass <- split(geneRow, classvec)
    gene.test <- t.test(geneByClass[[1]], geneByClass[[2]])
    gene.test$statistic
})

summary(prior.t.stats-post.t.stats)

## thus gene distributions are preserved within datasets, but normalized
## between datasets
```

`combineCOCOoutput` *Combine COCONUT output from multiple objects into a single object*

Description

Combine COCONUT output from multiple objects into a single object. Makes pooled analysis of COCONUT-co-normalized data easier.

Usage

```
combineCOCOoutput(COCONUT.out)
```

Arguments

`COCONUT.out`	Output from a call to COCONUT().

Details

The output from COCONUT() can be a bit daunting, and the separate dataobjects remain separated by input cohort, plus are separated into control and diseased components.

Value

This function will knit all data together into a list with three parts:

gene	contains a single matrix with all COCONUT-conormalized data (both control and disease)
pheno	contains a single data.frame with all phenotype info from the input samples, but ONLY from those columns whose colnames are same across all cohorts
class.cntl0.dis1	a binary vector that contains control/disease assignment for all columns in $genes.

Author(s)

Timothy E Sweeney, MD, PhD (tes17 [at] stanford [dot] edu)

References

Sweeney TE et al., "Robust classification of bacterial and viral infections via integrated host gene expression diagnostics", 2016

Examples

```
data(GSEs.test)

## apply COCONUT to a very small test case
## (3 datasets with 10 patients and 2000 genes)
GSEs.COCONUT <- COCONUT(GSEs=GSEs.test,
                        control.0.col="Healthy0.Sepsis1",
                        byPlatform=FALSE)

## combine output
GSEs.COCO.combined <- combineCOCOoutput(GSEs.COCONUT)
str(GSEs.COCO.combined)
```

GSEs.test *COCONUT test data*

Description

A list of lists, specifically, a list of three data objects (GSEs) from the NIH GEO repository. Each has been converted from a probe matrix to a gene matrix, and subsetted to have only 10 samples (5 healthy and 5 diseased) with only 2000 genes.

Usage

```
data(GSEs.test)
```

Format

A list of lists. Within the list, each named object consists of:

`genes` a numeric matrix, gene names in rows and sample IDs in columns.

`pheno` a data.frame, with sample IDs in rows and phenotype variables in columns.

Details

The data all come from the NIH GEO repository, and are subsets of their respective GSE IDs.

Source

http://www.ncbi.nlm.nih.gov/geo/

Examples

```
## see help(COCONUT) for further example
data(GSEs.test)
str(GSEs.test)
```

Index