Batchevaluation wrapper: Short introduction

Madhulika Mishra, Guillaume HEGER, Pablo Moreno, Irene Papatheodorou, Janet M Thronton

01/06/2020

Contents

1 Introduction

Over time, a vast amount of genomic information has been accumulated in the public repository mainly in GEO and ArrayExpress for the given phenotype. However, it is still challenging to integrate different high throughput experiments reflecting similar phenotype because of various other non-biological confounding factors such as type of array, date of experiment, a laboratory where data was generated, etc and, can be summarized as `batch effects`. In order to solve this issue, various batch-effect correction algorithms(BECAs) such as ComBat, SVA, and RUV have been developed to remove such batch effects from the integrated data and have shown promising results in mining biological signals. Evaluation of batch correction protocols involves mainly looking at the Principle component plots or RLE plot or sometimes by measuring batch entropy.

This project is to increase the reusability and reproducibility of these workflows in order- to facilitate batch correction, comparison, and benchmarking for ExpressionAtlas. Our strategy is to provide command-line access to individual library functions through simple wrapper scripts packaged in R wrapper. Batchevaluation R Wrapper contain implements of a variety of methods for batch correction of microarray as well as for bulk RNA-seq data.

2 Project overview

Batchevaluation analyses include step such as:

- Fetching experiments from ExpressionAtlas based on a biological factor of interest, such as disease, species and tissue
- Removing isolated experiments
- Merging experiments in a single dataset
- Evaluation of batch effect in the merged dataset
- Correcting batch effects
- Evaluation of batch correction methods

These steps may be implemented in a variety of ways including stand-alone tools, scripts or R package functions. To compare equivalent logical steps between workflows, and to 'mix and match' those components for optimal workflows is, therefore a challenging exercise without having additional infrastructure. The current R package provides a flexible to perform batch correction and `evaluate effect` of the batch correction on the given dataset.

3 Installation

Installation should take less than 5 min.

3.1 Via Github and devtools

If you want to install the package directly from Github, I recommend using the `devtools` package.

```
library(devtools)
install_github('XXXXXXXX') link is not up yet
```

3.2 Manually

Please download the package as `zip` archive and install it via

```
install.packages('Batchevaluation.zip', repos = NULL, type = 'source')
```

3.3 Installing the dependencies (first use)

To run the following functions, you will need some packages that can be installed using these commands in R :

```
install.packages(c('magrittr','stringr','purrr','dplyr','tibble','ggplot2','igraph','gtools','lme4','rea
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install('preprocessCore')
BiocManager::install('limma')
BiocManager::install('sva')
BiocManager::install('RUVSeq')
BiocManager::install('RUVnormalize')
BiocManager::install('SummarizedExperiment')
BiocManager::install('batchelor')
devtools::install_github('tengfei-emory/scBatch')
```

4 Overview of steps available in Batchevaluation

Batchevaltaion package has several steps ranging from meta-experiment creation to batcheffect evaluation step (`Figure1`). In current wrapper, scripts are written in user-friendly way. Short description of each step and example is given below -

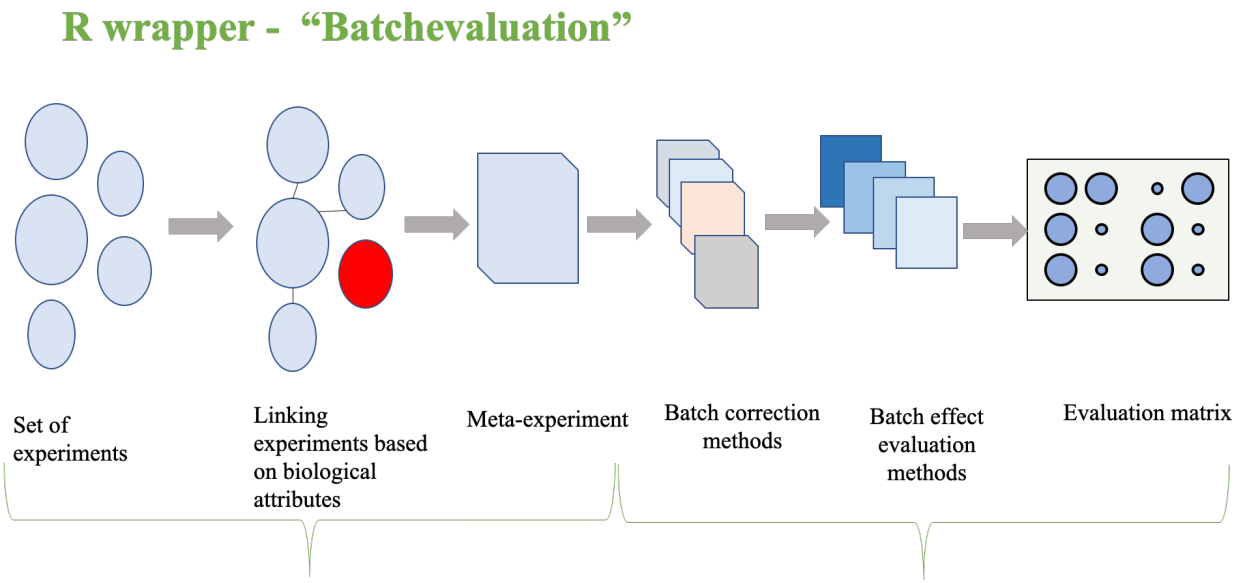


Figure 1: Workflow of the Batchevaluation package.

4.1 Loading data in R

If the data files are already on your computer, you can use this step. If you want to download data from Expression Atlas, skip this part and go directly to "Downloading data from Expression Atlas".

Recommendation: Microarray input data should be preprocessed- appropriately back-ground corrected without any normalization step, probe-to-gene level mapped and log-transformed. It is also recommended to remove low-expressed genes from data if possible. It should be count matrix for bulk-RNaseq.

Steps:

- Put the data files in `.Rdata` format in a directory containing only them. (The R objects contained in those files must be either `SummarizedExperiment` or `ExpressionSet` experiment.)
- Load all the experiments in a list using the function `load_experiments`

```
experiments <- load_experiments('directory_path')
experiments<-load_experiments('./')
```

Example Microarray data is provided in **example1** folder of the package.

4.2 Downloading data from Expression Atlas

If you want to use data from Expression Atlas that can be downloaded in `.Rdata` format, you can use the function `download_experiments_from_ExpressionAtlas` in this way :

```
experiments <- download_experiments_from_ExpressionAtlas('E-MTAB-3718','E-MTAB-3725','E-GEOD-44366','E-
```

This downloads the experiments in a new directory called **"experiments"** in your working directory and loads all the experiments in R within a list, using `load_experiments` function. After having loaded the experiments, you get a list of either `SummarizedExperiment` or list of `ExpressionSet` object.

Caution: Avoid mixing experiments of `SummarizedExperiment` with `ExpressionSet`. Experiment can only belong from any one of the class only.

Example of loaded data:

```
> experiments
$GSE1297_rawdata.Rdata
ExpressionSet (storageMode: lockedEnvironment)
assayData: 8872 features, 29 samples
  element names: exprs
protocolData: none
phenoData
  rowNames: 1 2 ... 3 (29 total)
  varLabels: ID Assay ... Batch (9 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:

$GSE1711_rawdata.Rdata
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12386 features, 24 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: SAMPLE212704SUB4576 SAMPLE212705SUB4576 ... SAMPLE212727SUB4576 (24 total)
  varLabels: AtlasAssayGroup age ... biosource_type (9 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

4.3 Removing the isolated experiments

To correct the `batch effect`, one needs to take the biological characteristics of the samples into account (`Tissue` in our example). If no sample of an experiment shares biological characteristics with samples from other batches, it is not possible to correct batch effect with these batches because one cannot distinguish the biological difference from the artifact. The function `remove_isolated_experiments` removes the isolated experiments and plots graphs of intersections between the experiments before and after removal.

```
experiments %<>% remove_isolated_experiments('Tissue')
```

WARNING: this function only removes the isolated experiment. Although it is still possible that two or more unconnected groups of experiments remain, within which the experiments are connected. In this case, batch effect correction is not possible neither and one has to choose a group of experiments manually.

The two following plots are displayed by the function. The first one shows the graph of intersections of all the experiments before the removal of isolated ones. The second shows the same graph after their removal.

4.4 Merging experiments in a single dataset

The function `merge_experiments` merges all the experiments in the list in a single `SummarizedExperiment` or `ExpressionSet` object and doesn't perform any correction. This function has two additional arguments `log` and `filter` (respectively set to `TRUE` and `FALSE` by default).

- The `log` argument determines whether to perform log transformation on the data (recommended for bulk RNAseq).

- The `filter` argument determines whether to filter genes for which all the samples of a batch have zero-counts. Set it to `TRUE` if you have issues in running ComBat at the next step.

```
experiments %<>% merge_experiments
#OR
experiments %<>% merge_experiments(log=TRUE,filter=FALSE)
```

`experiments` is now an only `ExpressionSet` object containing the information about batches both in its `@metadata` and `@colData` slots :

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 8872 features, 56 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: GSM21215.cel GSM21218.cel ... GSM697308.CEL (56 total)
  varLabels: ID Assay ... batch (10 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

Caution: Sometimes during merging experiment, phenodata (SDRF) file gets corrupted, hence, it is advised to always check meta-data before proceeding further.

4.5 Correcting batch effect

The function `batch_correction` perform various type of batch-correction on given merged dataset and output batch-corrected data as a list.

Short detail of methods implemented in `batch_correction` function are given below-

- For Microarray, `batch_correction` function has following batch correction methods-
  - **limma**- default limma setting, more detail about method can be learnt from Limma documentation.
  - **GFS**- Gene Fuzzy Score, more detail about method can be learnt from publication.
  - **Robust quantile normalization**- quantile normalisation method from PreprocessCore package. More information can be obtained from their documentation.
  - **ComBat**- implemented from SVA package. More information can be obtained from their documentation. In current `batch_correction` method, there are two versions of ComBat- 1) **ComBat1**- for parametric adjustment and, 2) **ComBat2** - for non-parametric adjustment, mean-only version.
  - **Q_ComBat** - is Quantile+ parametric adjustment of ComBat.
  - **MNN** - default mnnCorrect function from batchelor package. More information can be obtained from their documentation.
  - **naiveRandRUV_HK** - default naiveRandRUV using Human HK genes from RUVnormalize package. More information can be obtained from their documentation. Human Housekeeping gene list is from this publication. No. of confounders were estimated using both "leek" and "be" method and then least no. of estimated confounders were used as input for all variants of **naiveRandRUV** method.
  - **Q_naiveRandRUV_HK** - is Qunatile+ naiveRandRUV_HK
  - **naiveRandRUV_empi.controls** - default naiveRandRUV using data-empirically derived HK genes from RUVnormalize package.
  - **Q_naiveRandRUV_empi.controls** -is Qunatile+ naiveRandRUV_empi.controls.
- For bulk-RNAseq, `batch_correction` function has the following batch correction methods-
  - **limma**- default limma setting, more detail about method can be learnt from Limma documentation.

- **ComBat**- implemented from SVA package. More information can be obtained from their documentation.
- **ComBat_seq** - RNAseq version of ComBat, now implemented in `SVA` package. More information can be obtained from this publication.
- **MNN** - default mnnCorrect function from batchelor package. More information can be obtained from their documentation.
- **RUVs** - default RUVs function from RUVSeq package.Here, k=1 is used. More information can be obtained from their documentation.
- **scBatch** - batch correction method implemented in scBatch package. More information can be obtained from their publication.

This function has following arguments-

- The `experiment` argument is the input merged dataset.
- The `model` argument is a R formula mentioning the biological factor to take into account during correction.
- The `batch` argument is a meta-data column which has information about batch-label.
- The `k` argument is a no. of confounders for RUV methods and denotes the "number of factors of unwanted variation to remove". If k is not provided it will then estimated from input data.
- The `filter` argument specify gene label for the given dataset. It Should be one of following string- 'symbol', 'ensembl_gene_id' or 'entrezgene_id' depending on gene label for the given dataset.

Since, this merged dataset had some issue while merging, phenodata was added separately. Phenodata is also available with the package in `data` folder and can be accessed like this:

```
load("~/Batchevaluation/data/pheno.example1.Rdata")
experiments <- ExpressionSet(exprs(experiments), phenoData=pheno.experiment1)
```

After assigning phenodata, batch correction can be performed like this:

```
result <- batch_correction(experiment= experiments,model=~Disease, batch = "batch",'symbol')
```

Result is the list of batch-corrected data:

```
> summary(result)
                                 Length Class         Mode
data.limma                       1      ExpressionSet S4
data.GFS                         1      ExpressionSet S4
data.quantile                    1      ExpressionSet S4
data.ComBat1                     1      ExpressionSet S4
data.ComBat2                     1      ExpressionSet S4
data.Q_ComBat                    1      ExpressionSet S4
data.mnncorrect                  1      ExpressionSet S4
data.Q_naiveRandRUV_HK           1      ExpressionSet S4
data.naiveRandRUV_HK             1      ExpressionSet S4
data.naiveRandRUV_empi.controls  1      ExpressionSet S4
data.Q_naiveRandRUV_empi.controls 1     ExpressionSet S4
```

4.6 Assessment of batch-correction methods

The function `evalution_matrix` performs various evaluation on batch-corrected data and output performance list of each individual method. Since, there are no. of ways batch-correction can be evaluated and each method has its own limitation, we have used cocktail of methods to perform analysis. This function has both PCA-inspired as well as biology inspired qualitative assessment protocol for batch-correction.

Short detail of methods implemented in `evalution_matrix` function is given below-

- **pvca**- A function for principal variance component analysis. The function is written based on the 'pvcaBatchAssess' function of the PVCA R package and slightly changed to make it more efficient and

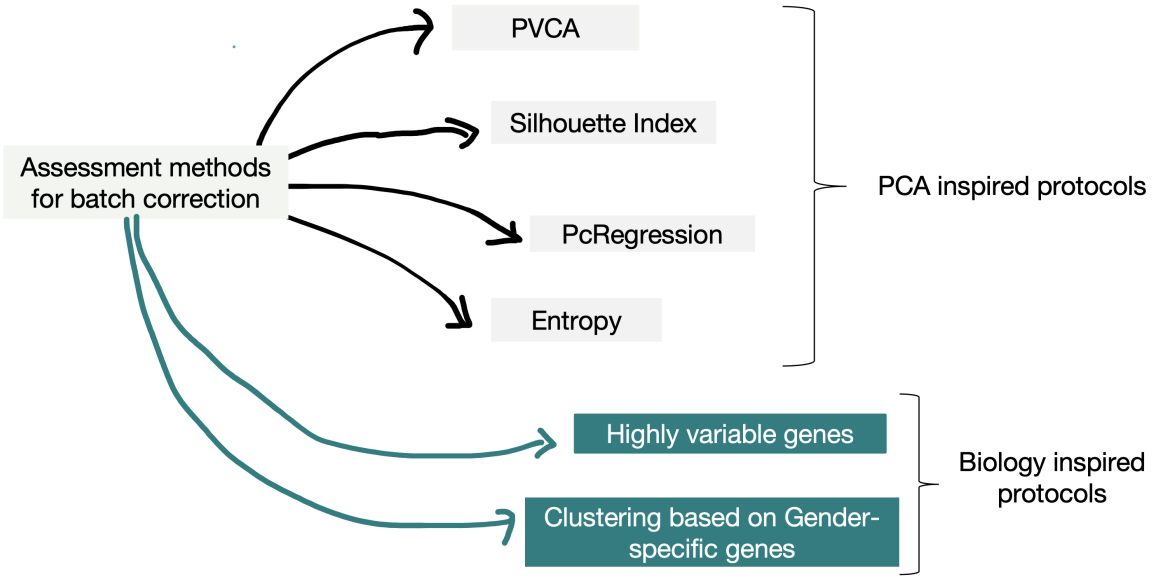


Figure 2: Overview of implemented assessment methods.

flexible for both microarray and bulk RNAseq gene-expression data. From https://github.com/dleelab/pvca.

- **silhouette**- Determine `batch effect` using the silhouette coefficient (adopted from scone) with default setting (nPcs=3). Taken from `kBET` package.
- **pcRegression**- Determine `batch effect` by a linear model fit of principal components and a batch (categorical) variable with a default setting (n_top=20). Taken from `kBET` package.
- **entropy**- Determine `batch effect` by computed the entropy of mixing to quantify the extent of intermingling of cells from different batches. Taken from `MNN` package.For calculation, first two Pcs are used as input. Since,depending on no. of samples in merged dataset, it is important to choose N1 and N2 for Batchentropy calculation, we have put both argument as variable here.
- **gender**- Determine `overfitting issue` using the silhouette coefficient with default setting (nPcs=3). Method computes silhouette coefficient using gender-specific genes and gender/sex meta-data column. Higher the silhouette coefficient, lesser overfitting will be expected because of `batch-correction`.**Warning**- If phenodata(SDRF) file doesn't contain sex/Gender/Sex/gender column, than this analysis will be skipped.
- **HVG.intersection**- This analysis calculate fraction of conserved highly variable genes (HVG) according to Brennecke et al., 2013 publication. This indirectly reflects whether biological heterogeneity is preserved or not during `batch-correction`. For this calculation, we consider ratio of HVG genes after correction/conserved HVG genes among different batches.
- **HVG.union**- This calculation also accounts for perseverance of biological heterogeneity by measuring ratio of conservation of HVG genes after batch correction. This function calculate ratio of no.of conserved HVG after batch correction/ Union of HVG genes among all the batches. Therefore, this function is less stringent compared to `HVG.intersection` function.

This function has following arguments-

- The `result` argument is a list of wrapped batch-corrected experiments obtained from last step ('batch_correction').
- The `batch.factors` is a list of factors to perform `PVCA` analysis. Along with `batch` as factor, one `biological factor` which can be used to assess over-fitting should be provided. Providing more than one `biological factor` here, will create issue while plotting the results.

- The `experiment` argument is the input merged dataset.
- The `N1` is the number of randomly picked cells for BatchEntropy function.
- The `N2` is the number of nearest neighbours of cell (from all batches) to check (for BatchEntropy function).
- The `filter` argument specify gene label for the given dataset. It Should be one of following string- 'symbol', 'ensembl_gene_id' or 'entrezgene_id' depending on gene label for the given dataset.

Assessment of batch-corrected data can be performed like this:

```
assessment <- evalution_matrix(result, batch.factors=c("batch","sex"), experiments,10,10,'symbol')
```

Result is the nested list of evalation scores for each of the evaluation protocol.

```
> assessment

$pvca
# A tibble: 2 x 11
  data.limma data.GFS data.quantile data.ComBat1 data.ComBat2 data.Q_ComBat data.mnncorrect data.Q_naive
*      <dbl>    <dbl>         <dbl>        <dbl>        <dbl>         <dbl>           <dbl>           <
1   8.64e-10   0.0326      9.99e- 1 0.0000000153 0.0000000783      3.60e-12          0.0122   0.0000000
2   7.14e- 2   0.0261      4.75e-12 0.0708       0.0718            7.55e- 2          0.0700   0.0760
# ... with 3 more variables: data.naiveRandRUV_HK <dbl>, data.naiveRandRUV_empi.controls <dbl>,
#   data.Q_naiveRandRUV_empi.controls <dbl>

$silhouette
# A tibble: 1 x 11
  data.limma data.GFS data.quantile data.ComBat1 data.ComBat2 data.Q_ComBat data.mnncorrect data.Q_naive
*      <dbl>    <dbl>         <dbl>        <dbl>        <dbl>         <dbl>           <dbl>           <
1   -0.00793   0.0853         0.744     -0.00849     -0.00743      -0.00498          0.0726         -0.0
# ... with 3 more variables: data.naiveRandRUV_HK <dbl>, data.naiveRandRUV_empi.controls <dbl>,
#   data.Q_naiveRandRUV_empi.controls <dbl>

$pcRegression
# A tibble: 1 x 11
  data.limma data.GFS data.quantile data.ComBat1 data.ComBat2 data.Q_ComBat data.mnncorrect data.Q_naive
*      <dbl>    <dbl>         <dbl>        <dbl>        <dbl>         <dbl>           <dbl>           <
1          0        0         0.872            0            0             0               0
# ... with 3 more variables: data.naiveRandRUV_HK <dbl>, data.naiveRandRUV_empi.controls <dbl>,
#   data.Q_naiveRandRUV_empi.controls <dbl>

$entropy
# A tibble: 1 x 11
  data.limma data.GFS data.quantile data.ComBat1 data.ComBat2 data.Q_ComBat data.mnncorrect data.Q_naive
*      <dbl>    <dbl>         <dbl>        <dbl>        <dbl>         <dbl>           <dbl>           <
1          0        0             0            0            0             0               0
# ... with 3 more variables: data.naiveRandRUV_HK <dbl>, data.naiveRandRUV_empi.controls <dbl>,
#   data.Q_naiveRandRUV_empi.controls <dbl>

$gender
# A tibble: 1 x 11
  data.limma data.GFS data.quantile data.ComBat1 data.ComBat2 data.Q_ComBat data.mnncorrect data.Q_naive
*      <dbl>    <dbl>         <dbl>        <dbl>        <dbl>         <dbl>           <dbl>           <
1     0.0549 -0.00168        0.0198       0.0502       0.0550         0.133           0.107           0
# ... with 3 more variables: data.naiveRandRUV_HK <dbl>, data.naiveRandRUV_empi.controls <dbl>,
#   data.Q_naiveRandRUV_empi.controls <dbl>
```

```
$HVG.intersection
# A tibble: 1 x 11
  data.limma data.GFS data.quantile data.ComBat1 data.ComBat2 data.Q_ComBat data.mnncorrect data.Q_naive
*      <dbl>    <dbl>         <dbl>        <dbl>        <dbl>         <dbl>           <dbl>           <dbl>
1          1   0.0395         0.355            1            1             1               1
# ... with 3 more variables: data.naiveRandRUV_HK <dbl>, data.naiveRandRUV_empi.controls <dbl>,
#   data.Q_naiveRandRUV_empi.controls <dbl>

$HVG.union
# A tibble: 1 x 11
  data.limma data.GFS data.quantile data.ComBat1 data.ComBat2 data.Q_ComBat data.mnncorrect data.Q_naive
*      <dbl>    <dbl>         <dbl>        <dbl>        <dbl>         <dbl>           <dbl>           <dbl>
1      0.716   0.0401         0.254        0.666        0.719         0.735           0.791          0.0
# ... with 3 more variables: data.naiveRandRUV_HK <dbl>, data.naiveRandRUV_empi.controls <dbl>,
#   data.Q_naiveRandRUV_empi.controls <dbl>
```

4.6.1 Interpretaion of result of evaluation protocols

Below is the table explaining how results from each evalaution protocol should be interpreted. PVCA, silhouette index and PcRegression measures residual `batch-effect` in corrected data, therefore for these measurements lower the score, better the performance will be. HVG (HVG.union & HVG.inersection) measures inherent biological heterogeneity, therefore higher the score, better will be method. Entropy measures entropy of batch-mixing, therefore, higher the score better the method is. Lastly, we also compute silhouette index using only gender-specific genes and gender meta-data. This gives us measurement of impact of batch-correction on gender-difference which is well-estabilshed biological phenotype. Ideally, any good batch-correction method should not decrease silhouette index of gender-based clustering after batch-correction.

Protocol	Factor	Score range	Interpretation	sign
PVCA	Batch	0-1	Higher the value, higher the batch effect	-
PcRegression	Batch	0-1	Higher the value, higher the batch effect	-
Silhouette Index	Batch	0-1	Higher the value, higher the batch effect	-
PVCA	Biological factor	0-1	Higher the value better is method	+
Entropy	Batch	-inf to + inf	Higher the value better batch mixing	+
HVG (highly variable genes)	Biological heterogeneity	0-1	Higher the value better is method	+
Gender specific silhouette index	Biological factor	0-1	Higher the value better sub-population preservation after batch correction	+

Figure 3: Result interpretation.

4.7 Diagnostic plot

Once, assessment is done, in next step, results obtained from `evaluation_matrix` step can be further analyszed using `Rank.plot` function which performs ranking and plotting of evaluation matrix obtained at previous step. Here, methods are ranked on their individual performance and finally `sumRank` is final Rank of each method for the given input dataset. Rank 1 will be the best performer method.

This function has only one argument-

- `evaluation` is a evaluation list obtained from previous step `evaluation_matrix`.

```
final <- Rank.plot(assessment)
```

final is a list of two data-frame- (a) raw - simple data-frame output of evaluation matrix and, (b) ranked- Ranked data-frame of evaluation matrix which has additional column `sumRank` containing final Rank of each method. Ranks are in descending performance order, i.e. method having score 1 will be the best method. This function also output `diagnostic plot`, where x-axis is the evaluation methods and y-axis is the Rank of each normalization method.

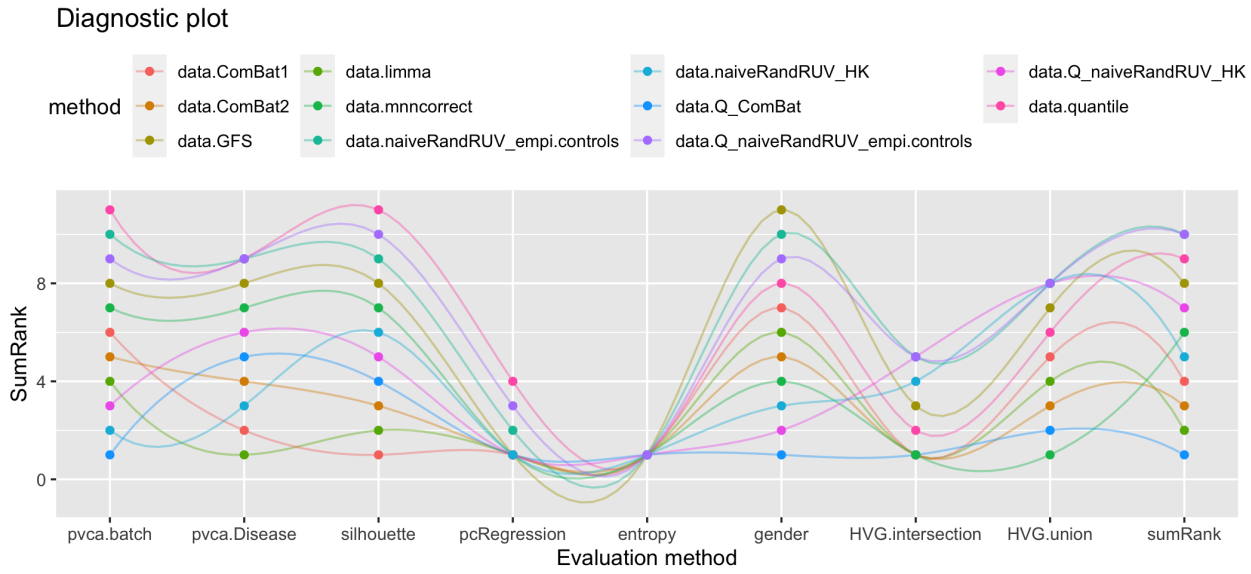


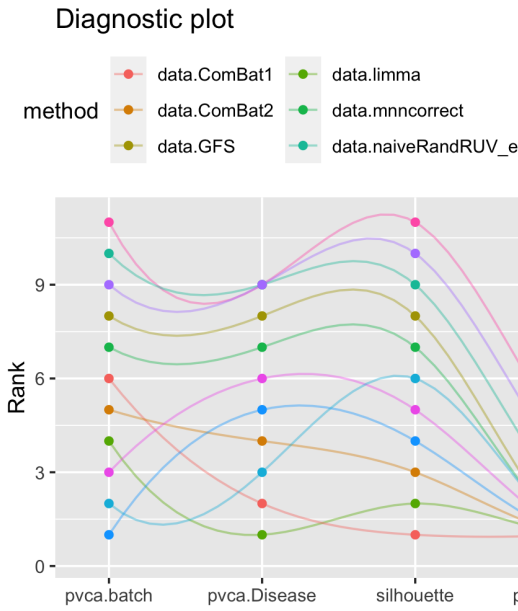
Figure 4: Diagnostic plot.

Result:

```
> final
$raw
                                  pvca.batch pvca.Disease   silhouette pcRegression entropy       genc
data.limma                     3.182752e-10   0.09378949 -0.007928513    0.0000000       0  0.0549238
data.GFS                       2.847282e-02   0.03073336  0.085294013    0.0000000       0 -0.0016819
data.quantile                  9.994937e-01   0.00000000  0.743891878    0.8721174       0  0.0197919
data.ComBat1                   2.605892e-09   0.09262298 -0.008489088    0.0000000       0  0.0502110
data.ComBat2                   5.605465e-10   0.09087029 -0.007427614    0.0000000       0  0.0549741
data.Q_ComBat                  4.700739e-12   0.09074923 -0.004983807    0.0000000       0  0.1326909
data.mnncorrect                1.490184e-02   0.08641440  0.072566372    0.0000000       0  0.1069008
data.Q_naiveRandRUV_HK         2.636718e-10   0.08810802 -0.001986466    0.0000000       0  0.1234779
data.naiveRandRUV_HK           5.259534e-11   0.09142561  0.003888988    0.0000000       0  0.1132437
data.naiveRandRUV_empi.controls 9.994633e-01   0.00000000  0.680552340    0.8204334       0  0.0185797
data.Q_naiveRandRUV_empi.controls 9.994511e-01   0.00000000  0.704169390    0.8687179       0  0.0193001
                                  HVG.union
data.limma                     0.715801887
data.GFS                       0.040094340
data.quantile                  0.253537736
data.ComBat1                   0.666273585
data.ComBat2                   0.719339623
data.Q_ComBat                  0.734669811
```

```
data.mnncorrect                    0.791273585
data.Q_naiveRandRUV_HK             0.001179245
data.naiveRandRUV_HK               0.001179245
data.naiveRandRUV_empi.controls    0.001179245
data.Q_naiveRandRUV_empi.controls 0.001179245

$ranked
                                  pvca.batch pvca.Disease silhouette pcRegression entropy gender HVG.in
data.limma                                 4            1          2            1       1      6
data.GFS                                   8            8          8            1       1     11
data.quantile                             11            9         11            4       1      8
data.ComBat1                               6            2          1            1       1      7
data.ComBat2                               5            4          3            1       1      5
data.Q_ComBat                              1            5          4            1       1      1
data.mnncorrect                            7            7          7            1       1      4
data.Q_naiveRandRUV_HK                     3            6          5            1       1      2
data.naiveRandRUV_HK                       2            3          6            1       1      3
data.naiveRandRUV_empi.controls           10            9          9            2       1     10
data.Q_naiveRandRUV_empi.controls          9            9         10            3       1      9
                                  sumRank                            method
data.limma                              2                        data.limma
data.GFS                                8                          data.GFS
data.quantile                           9                       data.quantile
data.ComBat1                            4                        data.ComBat1
data.ComBat2                            3                        data.ComBat2
data.Q_ComBat                           1                       data.Q_ComBat
data.mnncorrect                         6                     data.mnncorrect
data.Q_naiveRandRUV_HK                  7           data.Q_naiveRandRUV_HK
data.naiveRandRUV_HK                    5             data.naiveRandRUV_HK
data.naiveRandRUV_empi.controls        10  data.naiveRandRUV_empi.controls
data.Q_naiveRandRUV_empi.controls      10 data.Q_naiveRandRUV_empi.controls
```



Example of input dataset where Gender information is not present in meta-data.

5 Accessory functions

5.1 Detetction of batch-effects in raw merged dataset

This is a accessory function which perform subset of evaluation tests of `evaluation_matrix` function and provide estimates whether merged dataset obtained after 'merge_experiments' requires batch correction or not. Higher value of pvca.batch, silhouette, pcRegression and entropy is a direct indicative of batch-effects in raw merged dataset.

This function has following arguments-

- The `result` argument is a is the input merged dataset obtained from `merge_experiments` step.
- The `batch.factors` is a list of factors to perform `PVCA` analysis. Along with `batch` as factor, one `biological factor` which can be used to assess over-fitting should be provided. Providing more than one `biological factor` here, will create issue while plotting the results.
- The `experiment` argument is the again the same input merged dataset.
- The `N1` is the number of randomly picked cells for BatchEntropy function.
- The `N2` is the number of nearest neighbours of cell (from all batches) to check (for BatchEntropy function).
- The `filter` argument specify gene label for the given dataset. It Should be one of following string- 'symbol', 'ensembl_gene_id' or 'entrezgene_id' depending on gene label for the given dataset.

Assessment of batch-effect on raw merged data can be performed like this:

```
batch_effect.raw <- raw.evaluation(experiments,experiment = experiments, batch.factors=c("batch","Disea
Result <- do.call(rbind, lapply(batch_effect.raw, data.frame))
```

Result:

```
> Result
                   raw
pvca.batch   0.9994122
pvca.Disease 0.0000000
silhouette   0.7406456
pcRegression 0.9173469
entropy      0.0000000
```