The crmn Package

Henning Redestig
henning@psc.riken.jp
RIKEN Plant Science Center
Yokohama, Japan
http://www.metabolome.jp/

February 10, 2020

Overview

CRMN (Cross-contribution Robust Multiple standard Normalization) is a normalization method that can be used to normalize data that has been generated using *internal standards* (ISs). It is mainly intended (but not restricted) to normalize GC-MS metabolomics data. An IS is a chemical compound that is added to the sample in a known concentration and can be used to remove bias that arise from technical variation. Unfortunately, sometimes analytes may directly cause variation in these standards thereby rendering them difficult to use for normalization purposes.

CRMN attempts to solve this issue by correcting the ISs for covariance with the experimental design before using them for normalization. In short, this achieved by the following algorithm:

1. Pre-process data by log-transforming and z-transformation.
2. Divide data into analytes, Y_A, and standards, Y_S.
3. Remove the correlation between experimental design matrix, G and the Y_S using linear regression.
4. Perform PCA on the residual of the regression model between Y_S and G to extract the systematic error, T_Z.
5. Remove correlation between Y_A and T_Z using linear regression.
6. Undo pre-processing steps.

Please see Redestig et al. [1] further description.

1 Normalization methods

The package comes with access to several different normalization methods to use as reference or alternative to the CRMN method. These are:

NOMIS The method proposed by Sysi-Aho et al. [2]. Note that this implementation was not done by the authors of the NOMIS method and any errors should be blamed on author of this package.

One Divide each analyte by the abundance estimate of a single user-defined IS. Also known as *Single*.

RI Divide each analyte by the abundance estimate of the IS that is closest in terms of its retention index.

totL2 Does not use internal standards, normalization is done by ensuring that the square sum of each sample is the same.

Median/Avg Does not use internal standards, normalization is done by ensuring that the median/average of each sample equals one.

2 Getting started

Installing the package: For Windows, start `R` and select the `Packages` menu, then `Install package from local zip file`. Find and highlight the location of the zip file and click on `open`.

For Linux/Unix, use the usual command `R CMD INSTALL` or use the command `install.packages` from within an `R` session.

Loading the package: To load the `crmn` package in your `R` session, type `library(crmn)`:

```
> library(crmn)
> help(package="crmn")
```

Help files: Detailed information on `crmn` package functions can be obtained from the help files. For example, to get a description of the normalization function `normalize` type `help("normalize")`.

Sample data: A sample data set is included under the name `mix`. This data has 46 analytes of which 11 are internal standards. There are 42 samples containing known compositions of the measured analytes (see Redestig et al. (Unpublished)). The samples were measured in three different batches as indicated in the phenotypical data.

3 Examples

3.1 Input data

There are two slightly different flavors for how the data can be provided to the normalization functions. One way is to use the `ExpressionSet` object type as defined by the Biobase package. This is convenient because object type holds both information about the samples (columns) and the analytes (rows) of the data matrix and gives programmatically useful ways to access the different types of data. To use this way you must first format your data to such an object, please read the documentation from Biobase on how to do this. The `ExpressionSet`-object must contain information about which features are the internal standards coded by a `tag` (or another column name but then you have to specify the "where" argument to `analytes` and `standards`) component which should be equal `"IS"` (or something else which you have to specify via the `what` argument) for the standards. To use the `RI` method the feature data must also contain the retention index of each analyte. See the dataset `mix` for an example.

Alternatively you can provide the data as a simple matrix (as for example read by the `read.table` function). In that case you must make sure to always pass the extra argument `standards` which is a logical vector indicating which rows are the internal standards. You must also specify the design matrix to the experiment yourself. This is of course an option when using an `ExpressionSet` as well.

3.2 Normalization of a `ExpressionSet`

```
> data(mix)
> head(fData(mix))[,1:4]
```

```
     mark tag               synonym     RI
15 Marked   I  Glycolic acid (2TMS) 1066.5
18 Marked   I   Alanine, DL- (2TMS) 1097.3
47 Marked   I Nicotinic acid (1TMS) 1298.5
51 Marked   I   Fumaric acid (2TMS) 1340.6
52 Marked   I    Serine, DL- (3TMS) 1348.3
56 Marked   I Threonine, DL- (3TMS) 1373.1
```

```
> head(pData(mix))
```

```
              type experiment runorder
STDs_1_2_1 STDs_1         uv        3
STDs_1_2_2 STDs_1         uv        3
STDs_1_2_3 STDs_1         uv        3
STDs_1_3_1 STDs_1         uv        3
STDs_1_3_2 STDs_1         uv        3
STDs_1_3_3 STDs_1         uv        3
```

Division of the dataset should now be possible as following.

```
> Ys <- standards(mix)
> Ya <- analytes(mix)
> dim(Ys)
```

```
Features  Samples
      11       42
```

```
> dim(Ya)
```

```
Features  Samples
      35       42
```

These two functions must work as they should for your data too (if you want to the the ExpressionSet interface, otherwise see Section ??) so make sure that they do.

To proceed with normalization we first fit a normalization model. The complexity, number of principal components, is decided by the cross-validation functionality of the *pcaMethods* package. To use CRMN we also need access to the experimental design. This can be done automatically by specifying the relevant factors in the `pData` object of the data or by directly providing a design matrix. I.e:

```
> nfit <- normFit(mix, "crmn", factor="type", ncomp=2)
```

Is the same as doing:

```
> G <- model.matrix(~-1+mix$type)
> nfit <- normFit(mix, "crmn", factor=G, ncomp=2)
```

We proceed by not specifying the complexity but letting the cross-validation take care of this step.

```
> nfit <- normFit(mix, "crmn", factor="type")
> #complexty (number of PC's):
> sFit(nfit)$ncomp
```

```
PC 1
   1
```

The variance that CRMN identified as systematic error can be visualized using `slplot`, see Figure 1.

```
> slplot(sFit(nfit)$fit$pc, scol=as.integer(mix$runorder))
```

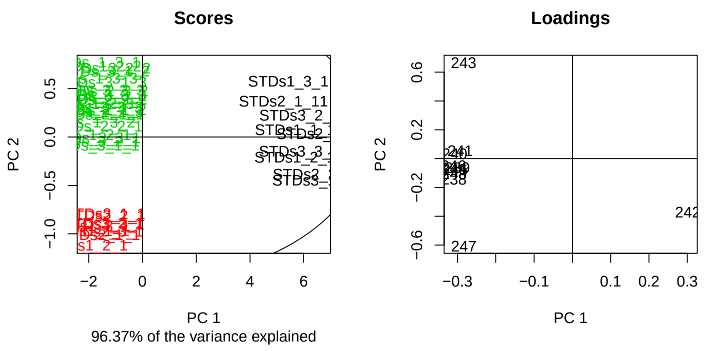

Figure 1: PCA of the systematic error T_Z. Colors correspond to the known batches.

The output from `normFit` is an object of class `nFit` and has a simple plot and print/show function which can give basic statistics about the normalization model, see Figure 2

To normalize the data we predict the training data. Note that we could also have held some samples out from the training to obtain sample-independent normalization (potentially useful for quality control purposes).

```
> nfit

crmn normalization model
========================
Effect of experiment design on standards:
-----------------------------------------
Analysis of Variance Table

          Df Pillai approx F num Df den Df  Pr(>F)
I(X)       3 1.1339   1.7123     33     93 0.02344 *
Residuals 39
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Captured Tz:
--------------
svd calculated PCA
Importance of component(s):
                PC1     PC2     PC3     PC4     PC5     PC6     PC7    PC8
R2            0.934 0.02973 0.01409 0.00784 0.00732 0.00344 0.00171 0.0010
Cumulative R2 0.934 0.96369 0.97778 0.98562 0.99294 0.99638 0.99809 0.9991
                PC9   PC10
R2            0.00053 0.0003
Cumulative R2 0.99962 0.9999
                  PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8
R2            0.93396 0.02973 0.01409 0.00784 0.00732 0.00344 0.00171 0.00100
Cumulative R2 0.93396 0.96369 0.97778 0.98562 0.99294 0.99638 0.99809 0.99909
                  PC9    PC10
R2            0.00053 0.00030
Cumulative R2 0.99962 0.99992

R2 from Tz to analytes:
-----------------------
[1] 0.6472936

> plot(nfit)
```

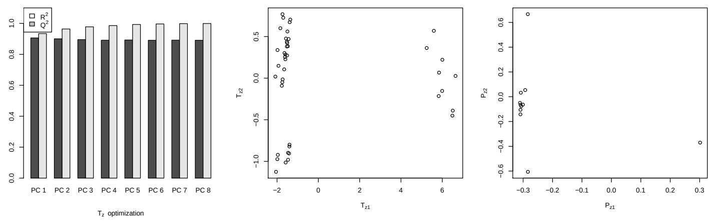

Figure 2: Basic plot function.

```
> normed.crmn <- normPred(nfit, mix, factor="type")
```

We can compare the result with other methods. Now we do this using the wrapper function `normalize` that combines `normFit` and `normPred`. See side-by-side PCA score plots of CRMN normalized data versus *One* and NOMIS normalized data in Figure 3.

```
> normed.one <- normalize(mix, "one", one="Hexadecanoate_13C4")
> normed.nomis <- normalize(mix, "nomis")
```

```
> pca.crmn <- pca(scale(log(t(exprs(normed.crmn)))))
> pca.one <- pca(scale(log(t(exprs(normed.one)))))
> pca.nomis <- pca(scale(log(t(exprs(normed.nomis)))))
> par(mfrow=c(1,3))
> plot(scores(pca.one), col=as.integer(mix$type),
+      pch=as.integer(mix$runorder),
+      main="Single IS")
> plot(scores(pca.nomis), col=as.integer(mix$type),
+      pch=as.integer(mix$runorder),
+      main="NOMIS")
> plot(scores(pca.crmn), col=as.integer(mix$type),
+      pch=as.integer(mix$runorder),
+      main="CRMN")
>
```

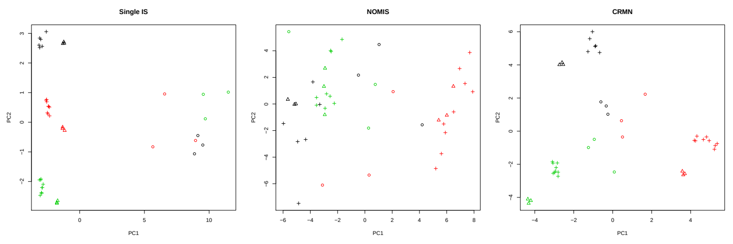

Figure 3: PCA of the `mix` using three different normalizations. Colors indicate the true concentration groups and plot character indicate the different batches (unwanted effect).

3.3 Normalization of a `matrix`

First we construct the required input parameters. This would of course normally be done by using `read.table` to read data as obtained by programs such as TargetSearch, HDA, metAlign etc.

```
> Y <- exprs(mix)
> replicates <- factor(mix$type)
> G <- model.matrix(~-1+replicates)
> isIS <- fData(mix)$tag == 'IS'
```

Division of the dataset should now be possible as following (results hidden).

```
> standards(Y, isIS)
> analytes(Y, isIS)
```

The main business is the same as when normalizing an `ExpressionSet` except that we now have to remember to pass the vector speciying the standards.

```
> nfit <- normFit(Y, "crmn", factors=G, ncomp=2, standards=isIS)
```

To normalize the data predict the training data.

```
> normed.crmn <- normPred(nfit, Y, factors=G, standards=isIS, ncomp=2)
```

and this could also have been done directly by:

```
> normed.crmn <- normalize(Y, "crmn", factors=G, standards=isIS, ncomp=2)
```

References

[1] Redestig, H., Fukushima, A., H., Stenlud, Moritz, T., Arita, M., Saito, K. and Kusano, M. *Compensation for systematic cross-contribution improves normalization of mass spectrometry based metabolomics data* Anal Chem, 2009, 81, 7974-7980

[2] Sysi-Aho, M., Katajamaa, M., Yetukuri, L. and Oresic, M *Normalization method for metabolomics data using optimal selection of multiple internal standards* BMC Bioinformatics, 2007, 8, 93