simpleVIPER

Denis Torre

July 14, 2016

simpleVIPER - Overview

The following code is a simplified, commented implementation of the VIPER algorithm, written by Mariano J. Alvarez. The algorithm produces the same results of a default call of the viper() function, when additional parameters are not specified.

The original code can be found on the 'viper' package, available on Bioconductor: https://www.bioconductor.org/packages/release/bioc/html/viper.html.

Step-by-step algorithm analysis

1. Load sample data

```
# 1.1 Load VIPER library
library(viper)

# 1.2 Load data
data(bcellViper, package="bcellViper")

# 1.3 Get expression matrix
eset <- exprs(dset)
```

Expression data

The 'eset' object is an expression matrix. The matrix has 6249 genes on rows, 211 samples on columns, and normalized expression levels as values.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
ADA	9.57	10.60	8.74	10.10	8.92
CDH2	6.18	5.31	5.65	4.46	4.81
MED6	9.67	8.77	9.19	9.53	8.80
NR2E3	9.85	9.58	9.63	8.82	9.27
ACOT8	9.08	9.04	8.01	8.68	8.74

Regulon object

The ARACNe network data is contained in the 'regulon' object. It contains information about 621 regulators, with a median number of targets of 259.

2. Data preprocessing

Step 1 - Filter the expression data.

Purpose: Remove expression data of genes which aren't targets of any regulator in the network.

```
# 1.1 Get a list of all targets of every regulator in the network.
tmp <- c(names(regulon), unlist(lapply(regulon, function(x) names(x$tfmode)), use.names = FALSE))

# 1.2 Remove expression data of all the genes which aren't in the list identified above.
eset <- eset[rownames(eset) %in% unique(tmp),]

# 1.3 Scale the expression matrix on rows [i.e. row = (row - mean(row)) / sd(row) ].
tt <- t(scale(t(eset)))
```

Scaled expression matrix

The 'tt' object contains the scaled expression matrix. It has 6249 genes on rows, 211 samples on columns, and scaled normalized expression levels as values.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
ADA	0.10	0.72	-0.41	0.42	-0.30
CDH2	0.43	-0.38	-0.06	-1.17	-0.85
MED6	2.10	0.41	1.19	1.83	0.45
NR2E3	0.76	0.42	0.48	-0.59	0.01
ACOT8	1.24	1.15	-0.95	0.42	0.55

Step 2 - Filter the regulon object (i.e. the regulatory network).

Purpose: Filter the network by removing regulators with small regulons and targets of which there is no expression data. Step 2.1 is especially required for microarray-based expression data, where genes might be missing in the network due to lack of probes in the platform used.

```
# 2.1 Remove targets in the regulon which are not in the rownames of the expression matrix.
regulon <- lapply(regulon, function(x, genes) {
                          filtro <- names(x$tfmode) %in% genes
                          x$tfmode <- x$tfmode[filtro]
                          if (length(x$likelihood) == length(filtro))
                              x$likelihood <- x$likelihood[filtro]
                          return(x)
                          }, genes = rownames(eset))

# 2.2 Define minimum regulon size for filtering (default is 20).
minsize <- 20

# 2.3 Remove regulators with a regulon size below the 'minsize' parameter.
regulon <- regulon[sapply(regulon, function(x) length(x$tfmode)) >= minsize]
```

The filtered 'regulon' object contains information about 621 regulators, with a median number of targets of 259.

3. aREA

aREA (analytic Rank-based Enrichment Analysis) is the core of the VIPER algorithm, as it allows to calculate the Normalized Enrichment Scores (i.e. the VIPER scores) for the regulators in the network.

These scores represent an inferred estimate of the activity of each regulator. If the positive targets of the regulator are highly ranked in expression in a sample - while the negative targets are lowly ranked - the regulator will have a positive estimated VIPER activity for that sample. In the opposite case, the regulator will have a negative estimated VIPER score.

The following implementation uses the 'matrix' method of the aREA algorithm, available in the official viper package on Bioconductor.

Step 1 - Create the *Mode of Regulation* and *Scaled weights* matrices.

Purpose: Create the matrices to use to calculate the Normalized Enrichment Scores using data contained in the network inferred by ARACNe.

```
# 1.1 Get a list of all targets of every regulator in the network.
targets <- unique(unlist(lapply(regulon, function(x) names(x$tfmode)), use.names = FALSE))

# 1.2 Create the Mode of Regulation matrix from the regulon object.
mor <- sapply(regulon, function(x, genes) {
                        return(x$tfmode[match(genes, names(x$tfmode))])
                    }, genes = targets)

# 1.2 Create the Weights matrix from the regulon object.
wts <- sapply(regulon, function(x, genes) {
                        tmp <- x$likelihood[match(genes, names(x$tfmode))]
                        tmp[is.na(match(genes, names(x$tfmode)))] <- NA
                        return(tmp/max(tmp, na.rm = T))
                    }, genes = targets)

# 1.3 For each regulator, assign values of 0 to genes which are not listed as its targets.
mor[is.na(mor)] <- 0
wts[is.na(wts)] <- 0

# 1.4 Scale the columns of the 'Weights' matrix to the sum of the weights.
wtss <- scale(wts, center = FALSE, scale = colSums(wts))
```

Mode of Regulation matrix

The 'mor' object contains the *Mode of Regulation* matrix. It has 6249 genes on rows, 621 regulators on columns, and Mode of Regulation (MoR) estimates as values. The MoR values, calculated based on Spearmans's correlation between the regulator's and the target's expression, range between -1 and 1 and specify the strength and sign of the regulatory interaction (i.e. positive or negative regulation).

	AATF	ADNP	ADNP2	AEBP1	AFF3
SAMM50	1.00	0.00	0	0	0
DRG1	1.00	0.00	0	0	0
ATIC	0.99	0.42	0	0	0
SMARCC1	0.98	0.00	0	0	0
AHCY	0.97	0.00	0	0	0

Scaled weights matrix

The 'wts' object contains the *Scaled weights* matrix. It has 6249 genes on rows, 621 regulators on columns, and scaled likelihood estimates as values. The likelihood values are an estimate of the probabilistic significance of the regulatory relationship inferred by ARACNe. The scaling ensures that the sum of each column = 1.

	AATF	ADNP	ADNP2	AEBP1	AFF3
SAMM50	1	0.00	0	0	0
DRG1	1	0.00	0	0	0
ATIC	1	0.93	0	0	0
SMARCC1	1	0.00	0	0	0
AHCY	1	0.00	0	0	0

Step 2 - Calculate the two-tail enrichment scores.

Purpose: Calculate the two-tail enrichment scores for every regulator. These scores are non-normalized, signed estimates of the activity of the regulator. They are calculated using the relative ranking of its target genes in each sample, as well as the MoR and likelihood of the regulatory interactions.

A high two-tail enrichment score indicates that the positive targets of the regulator tend to lie in the upper portion of the distribution of gene expression ranks, whereas the negative targets tend to lie in the lower portion of the distribution. The opposite is true for low two-tail enrichment scores.

These scores will be integrated with the one-tail enrichment score estimate (see Step 3) to obtain the final VIPER score.

```
# 2.1 Calculate the 'T2 rank' matrix from the expression dataset.
t2 <- apply(tt, 2, rank)/(nrow(tt) + 1)

# This line of code is necessary to match the order of genes
# for the subsequent matrix multiplication steps.
pos <- match(targets, rownames(tt))

# 2.2 Transform T2 ranks to Gaussian values.
t2q <- qnorm(filterRowMatrix(t2, pos))

# 2.3 Matrix multiplication.
sum1 <- t(mor * wtss) %*% t2q
```

T2 rank matrix

The 't2' object contains the *T2 rank* matrix. It has 6249 genes on rows, 211 samples on columns, and its values represent the scaled expression rank of every gene within the sample.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
ADA	0.57	0.79	0.39	0.67	0.39
CDH2	0.69	0.38	0.51	0.17	0.21
MED6	0.99	0.68	0.88	0.97	0.69
NR2E3	0.79	0.69	0.70	0.35	0.51
ACOT8	0.91	0.89	0.22	0.67	0.73

***T2 q-value* matrix**

The 't2q' object contains the *T2 q-value* matrix. It has 6249 genes on rows, 211 samples on columns, and its values represent the q values which satisfy $P[X \leq q] = p$ on a standard Gaussian curve, where p is represented by the values in each cell of the 'T2 rank' matrix. The transformation is performed to ensure a uniform distribution of the targets, so that the ES will be normally distributed with mean 0 and variance 1.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
ADA	0.16	0.81	-0.29	0.44	-0.28
CDH2	0.48	-0.32	0.03	-0.94	-0.81
MED6	2.36	0.48	1.16	1.86	0.51
NR2E3	0.81	0.49	0.51	-0.39	0.04
ACOT8	1.32	1.24	-0.78	0.44	0.63

***Two-tail enrichment score* matrix**

The 'sum1' object contains the *Two-tail enrichment score* matrix. It has 621 regulators on rows, 211 samples on columns, and its values are the two-tailed enrichment score estimates. The matrix is calculated by matrix multiplication (see step 2.3) between two matrices with dimensions $621x6249$ and $6249x211$ respectively.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
AATF	0.70	0.33	0.75	0.65	0.71
ADNP	0.13	-0.09	0.17	0.44	0.30
ADNP2	0.12	-0.04	0.14	0.20	0.20
AEBP1	-0.40	-0.14	-0.44	-0.39	-0.51
AFF3	-0.13	-0.01	-0.32	-0.29	-0.12

Step 3 - Calculate the one-tail scores.

Purpose: Calculate the one-tail enrichment scores for every regulator. These scores are non-normalized, unsigned estimates of the activity of the regulator. They are calculated using the relative ranking of its target genes in each sample, as well as the absolute MoR and likelihood of the regulatory interactions.

A high one-tail enrichment score indicates that the targets of the regulator tend to lie at the extremes of the distribution of gene expression ranks, independently of the sign of the regulatory interaction. A low one-tail enrichment score indicates that the targets tend to lie towards the center of the distribution. It is a measure of the absolute 'dysregulation' of the expression of the regulator's targets.

These scores will be integrated with the two-tail enrichment score estimate (see Step 2) to obtain the final VIPER score.

```
# 3.1 Calculate the 'T1 Rank' matrix from the 'T2 Rank' matrix.
t1 <- abs(t2 - 0.5) * 2
t1 <- t1 + (1 - max(t1))/2

# 3.2 Transform T1 ranks to Gaussian values.
t1q <- qnorm(filterRowMatrix(t1, pos))

# 3.3 Matrix multiplication.
sum2 <- t((1 - abs(mor)) * wtss) %*% t1q
```

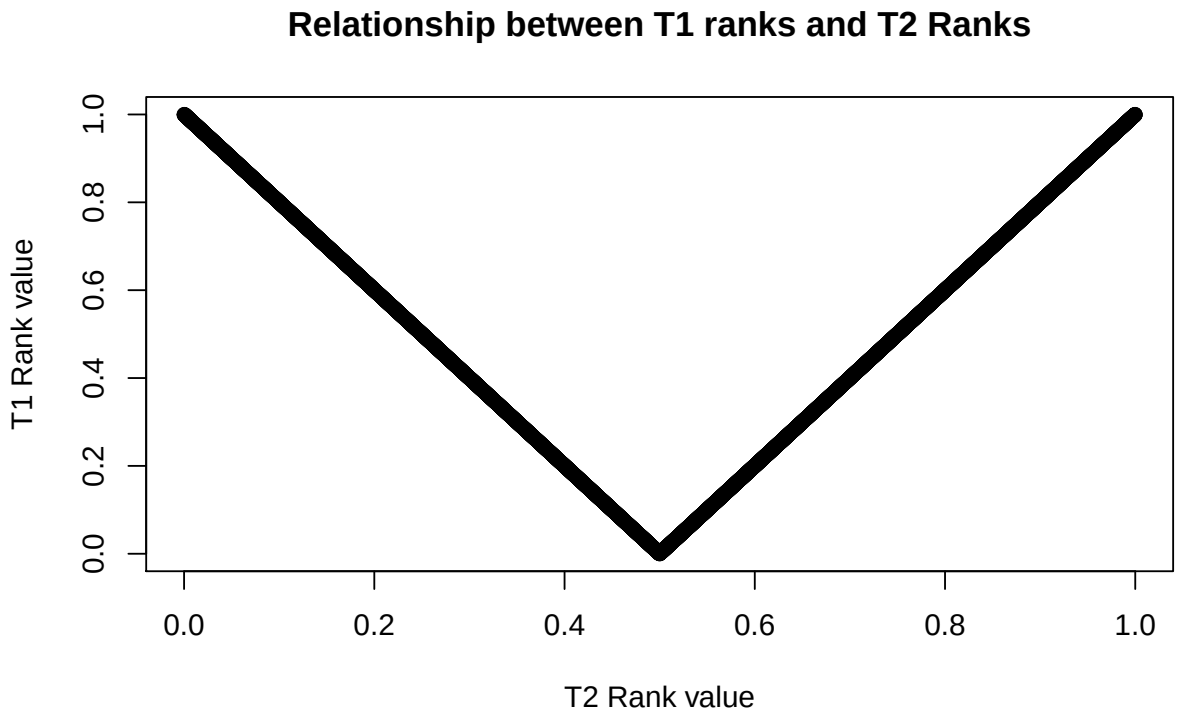
***T1 rank* matrix**

The 't1' object contains the *T1 rank* matrix. It has 6249 genes on rows, 211 samples on columns, and its values represent the scaled expression rank of every gene within the sample.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
ADA	0.13	0.58	0.23	0.34	0.22
CDH2	0.37	0.25	0.03	0.65	0.58
MED6	0.98	0.37	0.76	0.94	0.39
NR2E3	0.58	0.38	0.39	0.30	0.03
ACOT8	0.81	0.78	0.57	0.34	0.47

Comparison between T1 ranks and T2 ranks

The following plot displays the relationship between T1 ranks and T2 ranks. Both ranks are limited between 0 and 1; however, T1 Ranks have high values when the T2 Ranks approach the limits.



***T1 q-value* matrix**

The 't1q' object contains the *T1 q-value* matrix. It has 6249 genes on rows, 211 samples on columns, and its values represent the q values which satisfy $P[X \leq q] = p$ on a standard Gaussian curve, where p is represented by the values in each cell of the 'T1 rank' matrix. The transformation is performed to ensure a uniform distribution of the targets, so that the ES will be normally distributed with mean 0 and variance 1.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
ADA	-1.12	0.21	-0.76	-0.41	-0.77
CDH2	-0.33	-0.68	-1.92	0.39	0.20
MED6	2.10	-0.34	0.69	1.53	-0.28
NR2E3	0.21	-0.31	-0.28	-0.52	-1.91
ACOT8	0.90	0.78	0.17	-0.41	-0.08

***One-tail enrichment score* matrix**

The 'sum2' object contains the *One-tail enrichment score* matrix. It has 621 regulators on rows, 211 samples on columns, and its values are the two-tailed enrichment score estimates. The matrix is calculated by matrix multiplication (see step 3.3) between two matrices with dimensions $621x6249$ and $6249x211$ respectively.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
AATF	0.02	0.01	-0.01	-0.01	0.03
ADNP	0.02	0.03	0.03	-0.01	0.00
ADNP2	-0.03	0.09	0.01	0.03	-0.02
AEBP1	0.04	-0.02	0.02	0.00	0.03
AFF3	-0.05	0.04	-0.05	-0.01	-0.03

Step 4 - Calculate the Three-tail enrichment score.

Purpose: Combine the Two-tail and One-tail enrichment scores in a 'Three-tail' (3T) enrichment score matrix.

```
# 4.1 Extract the signs of the Two-tail enrichment scores
ss <- sign(sum1)
ss[ss == 0] <- 1

# 4.2 Combine the Two-tail and One-tail enrichment score matrices.
sum3 <- (abs(sum1) + sum2 * (sum2 > 0)) * ss
```

***Sign* matrix.**

The 'ss' object contains the *Sign* matrix. It has 621 regulators on rows, 211 samples on columns, and its values represent the signs of the Two-tail enrichment scores calculate in *2.3*. These signs will become the signs of the final VIPER scores.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
AATF	1	1	1	1	1
ADNP	1	-1	1	1	1
ADNP2	1	-1	1	1	1

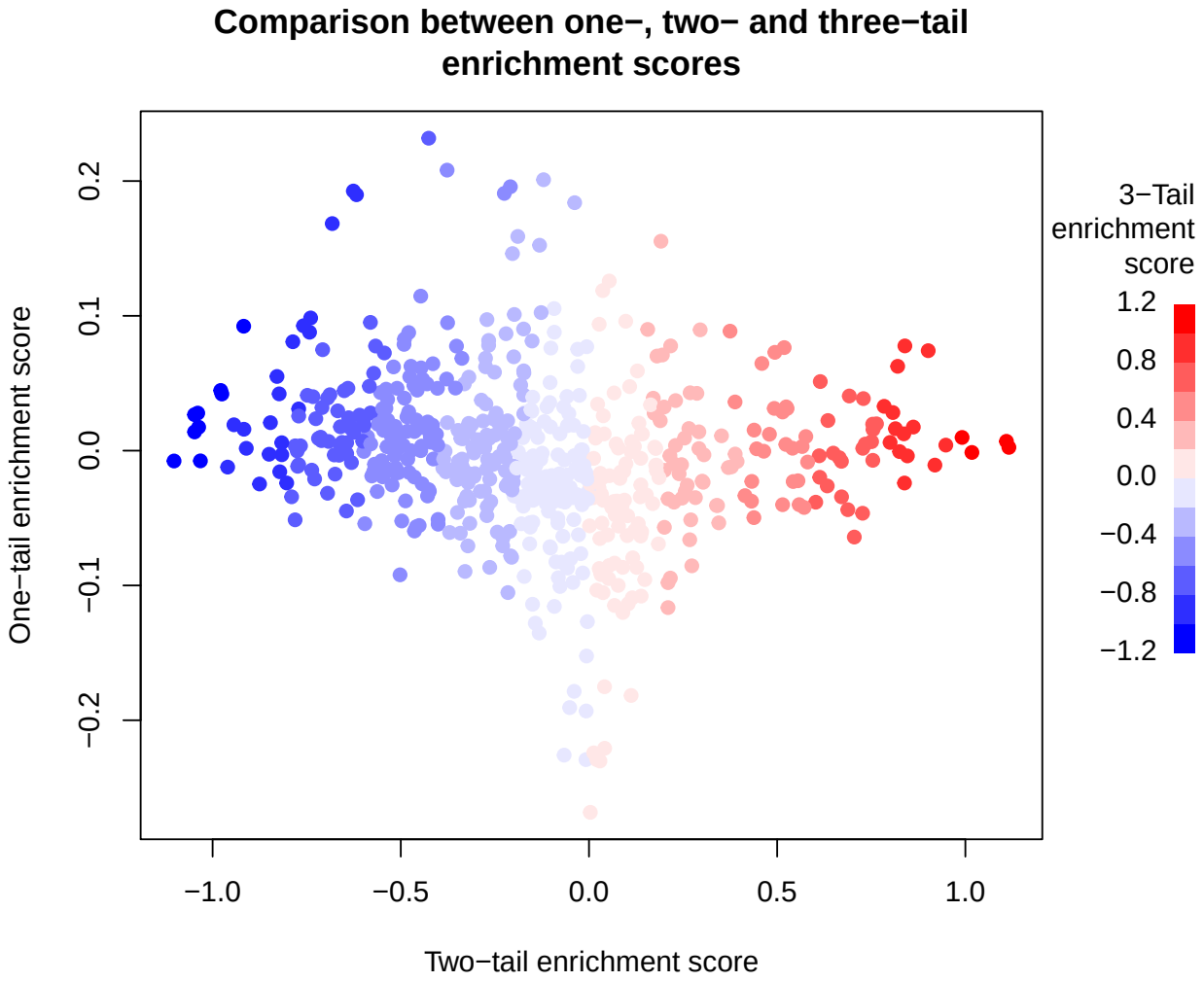
***Three-tail enrichment score* matrix.**

The 'sum3' object contains the *Three-tail enrichment score* matrix. It has 621 regulators on rows, 211 samples on columns. Its values represent the Three-tailed enrichment score estimates, as calculated in *4.2*.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
AATF	0.72	0.34	0.75	0.65	0.73
ADNP	0.15	-0.12	0.20	0.44	0.30
ADNP2	0.12	-0.13	0.15	0.23	0.20
AEBP1	-0.44	-0.14	-0.46	-0.40	-0.54
AFF3	-0.13	-0.05	-0.32	-0.29	-0.12

Comparison between 1T, 2T and 3T enrichment score values

Here is a comparison between the values of the three types of enrichment scores. The 3T enrichment score is largely dependent on the 2T score, as well as the 1T score, albeit on a smaller level.



Step 5 - Calculate the Normalized Enrichment Scores.

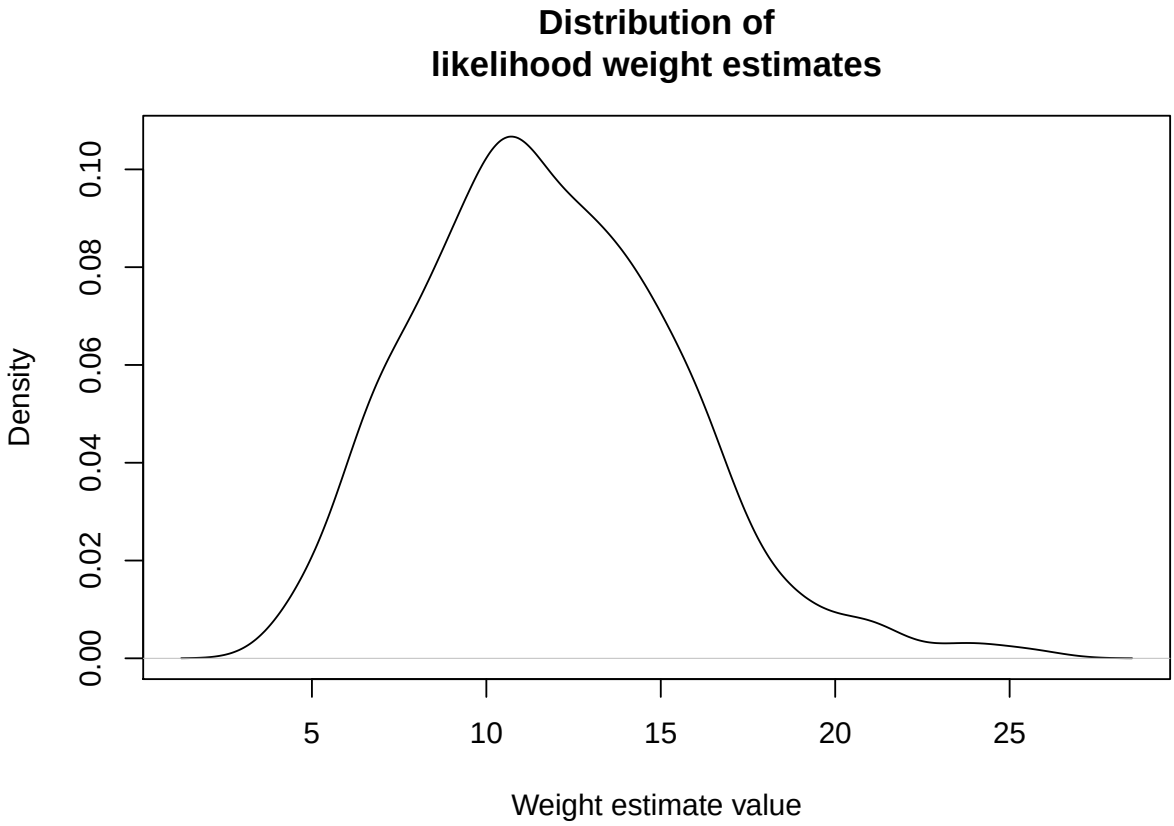
Purpose: Adjust the 3T enrichment scores by accounting for the overall likelihood estimates of the regulator's interactions. This step allows to increase the absolute activity scores of regulators which have high likelihood estimates for regulation of their targets, and decreases the absolute activity of regulators which have poor likelihood scores.

```
# 5.1 For each regulator, calculate an index proportional to the likelihood value
# of all its regulatory interactions.
lwt <- sqrt(colSums(wts^2))

# 5.2 Adjust 3T enrichment scores proportionally to the weights calculated above.
nes <- sum3 * lwt
```

Likelihood weight estimates

The 'lwt' object contains indices which are proportional to the estimates of likelihood between the regulator and all of its targets.



***Normalized Enrichment Score* matrix**

The 'nes' object contains the *Normalized Enrichment Score* matrix, commonly known as the VIPER matrix. It has 621 regulators on rows, 211 samples on columns. Its values represent VIPER score estimates of the activity of the regulator.

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
AATF	12.62	5.95	13.30	11.54	12.93
ADNP	2.08	-1.63	2.81	6.22	4.19
ADNP2	0.89	-0.94	1.10	1.67	1.43
AEBP1	-5.58	-1.81	-5.85	-5.03	-6.86
AFF3	-1.37	-0.52	-3.44	-3.19	-1.36

Comparison to the original viper function

We can compare the result to the default result from the viper() function of the official viper package.

```
# Remove all variables except for the NES results of 5.2
rm(list=setdiff(ls(), 'nes'))

# Load VIPER library
library(viper)

# Load data
data(bcellViper, package="bcellViper")

# Get expression matrix
eset <- exprs(dset)

# Run VIPER
nes_original <- viper(eset, regulon, verbose=FALSE)
```

The following are the results from the original viper function:

	GSM44075	GSM44078	GSM44080	GSM44081	GSM44082
AATF	12.62	5.95	13.30	11.54	12.93
ADNP	2.08	-1.63	2.81	6.22	4.19
ADNP2	0.89	-0.94	1.10	1.67	1.43
AEBP1	-5.58	-1.81	-5.85	-5.03	-6.86
AFF3	-1.37	-0.52	-3.44	-3.19	-1.36

The results are consistent:

```
# Check whether the two results are identical
identical(round(nes, digits=5), round(nes_original, digits=5))
```

```
## [1] TRUE
```

Raw code

simpleVIPER

```
simpleVIPER <- function (eset, regulon, minsize=20)
{
    # Data Preprocessing
    # Step 1 - Filter the expression data.
    # 1.1 Get a list of all targets of every regulator in the network.
    tmp <- c(names(regulon), unlist(lapply(regulon, function(x) names(x$tfmode)), use.names = FALSE))

    # 1.2 Remove expression data of all the genes which aren't in the list identified above.
    eset <- eset[rownames(eset) %in% unique(tmp),]

    # 1.3 Scale the expression matrix on rows [i.e. row = (row - mean(row)) / sd(row) ].
    tt <- t(scale(t(eset)))

    # Step 2 - Filter the regulon object (i.e. the regulatory network).
    # 2.1 Remove targets in the regulon which are not in the rownames of the expression matrix.
    regulon <- lapply(regulon, function(x, genes) {
                              filtro <- names(x$tfmode) %in% genes
                              x$tfmode <- x$tfmode[filtro]
                              if (length(x$likelihood) == length(filtro))
                                  x$likelihood <- x$likelihood[filtro]
                              return(x)
                              }, genes = rownames(eset))

    # 2.2 Define minimum regulon size for filtering (default is 20).
    # The 'minsize' parameter is specified in the function parameters.

    # 2.3 Remove regulators with a regulon size below the 'minsize' parameter.
    regulon <- regulon[sapply(regulon, function(x) length(x$tfmode)) >= minsize]

    # aREA
    nes <- simpleaREA(tt, regulon)

    # Return VIPER matrix
    return(nes)
}
```

simpleaREA

```
simpleaREA <- function (tt, regulon)
{
    # Step 1 - Create the 'Mode of Regulation' and 'Weights' matrices.
    # 1.1 Get a list of all targets of every regulator in the network.
    targets <- unique(unlist(lapply(regulon, function(x) names(x$tfmode)), use.names = FALSE))

    # 1.2 Create the Mode of Regulation matrix from the regulon object.
    mor <- sapply(regulon, function(x, genes) {
                            return(x$tfmode[match(genes, names(x$tfmode))])
                        }, genes = targets)

    # 1.2 Create the Weights matrix from the regulon object.
    wts <- sapply(regulon, function(x, genes) {
                            tmp <- x$likelihood[match(genes, names(x$tfmode))]
                            tmp[is.na(match(genes, names(x$tfmode)))] <- NA
                            return(tmp/max(tmp, na.rm = T))
                        }, genes = targets)

    # 1.3 For each regulator, assign values of 0 to genes which are not listed as its targets.
    mor[is.na(mor)] <- 0
    wts[is.na(wts)] <- 0

    # 1.4 Scale the columns of the 'Weights' matrix to the sum of the weights.
    wtss <- scale(wts, center = FALSE, scale = colSums(wts))

    # Step 2 - Calculate the two-tail enrichment scores.
    # 2.1 Calculate the 'T2 rank' matrix from the expression dataset.
    t2 <- apply(tt, 2, rank)/(nrow(tt) + 1)

    # This line of code is necessary to match the order of genes
    # for the subsequent matrix multiplication steps.
    pos <- match(targets, rownames(tt))

    # 2.2 Transform T2 ranks to Gaussian values.
    t2q <- qnorm(filterRowMatrix(t2, pos))

    # 2.3 Matrix multiplication.
    sum1 <- t(mor * wtss) %*% t2q

    # Step 3 - Calculate the one-tail score matrix.
    # 3.1 Calculate the 'T1 Rank' matrix from the 'T2 Rank' matrix.
    t1 <- abs(t2 - 0.5) * 2
    t1 <- t1 + (1 - max(t1))/2

    # 3.2 Get qnorm values
    t1q <- qnorm(filterRowMatrix(t1, pos))

    # 3.3 Matrix multiplication.
    sum2 <- t((1 - abs(mor)) * wtss) %*% t1q
```

```
    # Step 4 - Calculate the Three-tail enrichment score.
    # 4.1 Extract the signs of the Two-tail enrichment scores
    ss <- sign(sum1)
    ss[ss == 0] <- 1

    # 4.2 Combine the Two-tail and One-tail enrichment score matrices.
    sum3 <- (abs(sum1) + sum2 * (sum2 > 0)) * ss

    # Step 5 - Calculate the Normalized Enrichment Scores.
    # 5.1 For each regulator, calculate an index proportional to the likelihood value
    # of all its regulatory interactions.
    lwt <- sqrt(colSums(wts^2))

    # 5.2 Adjust 3T enrichment scores proportionally to the weights calculated above.
    nes <- sum3 * lwt
}
```