Two-Tier Mapper: a user-independent clustering method for global gene expression analysis based on topology

Rachel Jeitziner
UPBRI& ISREC
EPFL Lausanne

Contents

1 Introduction **1**

2 Prepare the data **2**

3 TTMap_part1 **3**

4 TTMap_part2 **5**

1 Introduction

We developed a new user-independent analytical framework, called *Two-Tier Mapper (TTMap)*. This tool is separated into two parts.

TTMap consists of two separated and independent parts : 1. Hyperrectangle Deviation assessment (HDA) and 2. Global-to-Local Mapper (GtLMap), where the first step establishes properties of the control group and removes outliers in order to calculate the deviation of each vector in the test group from the corrected control group. The second step uses the traditional Mapper algorithm **?** with a two-tier cover and a special distance.

This topological tool detects both global and local differences in the patterns of deviations and thereby captures the structure of the test group. The samples are clustered according to the shape of their deviation (do they both deviate positively, negatively or are they as the control). To still keep on

the information about the amount of deviation, one separates the data into 4 clusters according to a function measuring the amount of deviation. These represent then the second tier. Each cluster is colored by the extent of the deviation. A list of the differentially expressed genes is also provided

The functions and methods presented on this *vignette* provide explanation on how to use TTMap, by default and what can be changed by the user.

2 Prepare the data

Upload the file(s) to compare in R. Use *make_matrices* to create the needed files for the first function of TTMap since it generates the control and the test matrice in the right format. As an example, we generate two random files. For that we generate control samples $C_1, \ldots C_6$ and test samples a composed of two subgroups TA and TB, given by $TA_1, TA_2, TA_3, TB_1, TB_2, TB_3$, each with 10,000 features. The subgroups TA and TB have the same mean per gene as the mean of the control group, except for $C0$ genes for which the mean is Δ times higher for TA, respectively lower for TB.

```
> Aa = 6 # number of control samples
> B1 = 3 # number of samples in TA
> B2=3 # number of samples in TB
> C0=100 # number of differentially expressed genes
> D0 = 10000 # number of total genes
> a0 = 4 # average control
> b0=0.1 #variance control
> a1=6 # average TA for the C0 genes
> b1= 0.1 # variance TB for the C0 genes
> a2=2 # average TB for the C0 genes
> b2=0.5 # variance TB for the C0 genes
> ### Create the matrices
> set.seed(12)
> RA<- lapply(1:(D0-C0),function(i) rnorm(Aa, mean = a0, sd = sqrt(b0)))
> RA<-do.call(rbind,RA)
> RB1<- lapply(1:(D0-C0),function(i) rnorm(B1, mean = a0, sd = sqrt(b0)))
> RB1<-do.call(rbind,RB1)
> RB2<- lapply(1:(D0-C0),function(i) rnorm(B2, mean = a0, sd = sqrt(b0)))
> RB2<-do.call(rbind,RB2)
> RA_c<- lapply(1:C0,function(i) rnorm(Aa, mean = a0, sd = sqrt(b0)))
> RA_c<-do.call(rbind,RA_c)
> RB1_c<- lapply(1:C0,function(i) rnorm(B1, mean = a1, sd = sqrt(b1)))
> RB1_c<-do.call(rbind,RB1_c)
> RB2_c<- lapply(1:C0,function(i) rnorm(B2, mean = a2, sd = sqrt(b2)))
> RB2_c<-do.call(rbind,RB2_c)
> norm1 <- rbind(RA,RA_c)
> dis <- cbind(rbind(RB1,RB1_c),rbind(RB2,RB2_c))
> colnames(norm1)<- paste("N",c(1:Aa),sep="")
> rownames(norm1)<-c(paste("norm",c(1:(D0-C0)),sep=""),paste("diff",c(1:C0),sep=""))
> colnames(dis) <- c(paste("B1",c(1:B1),sep=""),paste("B2",c(1:B2),sep=""))
> rownames(dis)<-c(paste("norm",c(1:(D0-C0)),sep=""),paste("diff",c(1:C0),sep=""))
> junk <- TTMap::make_matrices(cbind(norm1,dis),col_ctrl = colnames(norm1),
+ col_test = colnames(dis),NAME=rownames(norm1),CLID=rownames(norm1))
```

This function can directly be used on a normalised count table from RNA-seq precising what are the columns of the control group (in col_ctrl) and what

are the columns in the test group (in col_test) .

3 TTMap_part1

The first part of the method checks if the control and the test matrices have the same row-names, and if not the method subselects the common rows. It outputs the files with the common rows subselected (with the extension mesh). It then calculates the corrected control matrix, which removes outliers and replaces them by a chosen method (given by a function with input the matrix with NAs where there is an outlier and should return a matrix without NAs), or by the median of the other values by default. The inputs can even be given by the CTRL and TEST variables of the list given by the output of *make_matrices* or by imputed control and test matrices in pcl format (see **?**). The name of the control group and the project name need to be inputed as well as the working directory, in which the output files will be created. A value for what to consider as an outlier (called e) can be imputed or use the data-driven default value given by the method. If there are any batch effects to consider, they can be imputed using the variable B, which is a vector of numbers representing the batches. Last, the parameter P is a value which will remove the genes that have a higher percentage than P of outlier values.

```
> E=1
> Pw=1.1
> Bw=0
> TTMAP_part1prime <-TTMap::ttmap_part1(normal.pcl = junk$CTRL,tumor.pcl = junk$TEST,
+ normalname = "Hi", dataname = "Hello", org.directory = getwd(),
+ e=E,P=Pw,B=Bw);

[1] "el= 1"
[1] 3.315239
```

This outputs:

- A file with the number of outliers per sample (Dataname followed by the number of the batch followed by na_numbers_per_col.txt)
- A file with the number of outliers per row (Dataname followed by the number of the batch followed by na_numbers_per_col.txt)
- A picture of the distribution of the mean against variance for each gene, before (Dataname followed by _mean_vs_variance.pdf) and
- after correction of outliers (Dataname followed by _mean_vs_variance_after_correction.pdf).

The corrected control matrix is output in the next step. A possible output after this first step is shown in figure 1.

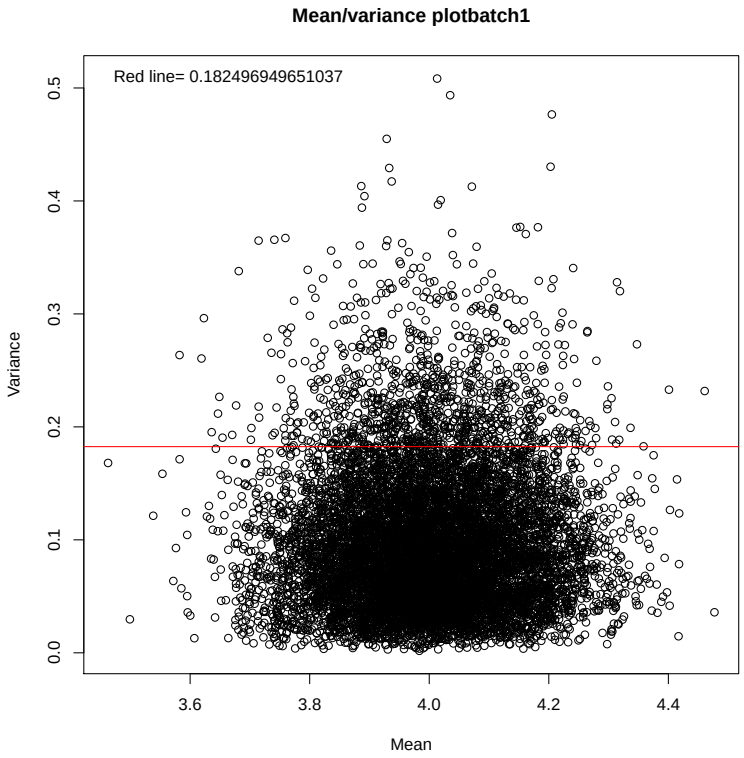

Figure 1: `barplotSignifSignatures`: Plot of mean against variance per gene.

4 TTMap_part2

The second part of the HDA step consists of calculating deviation components. This is enables the calculation in the third function of the shape of deviation. One parameter k is determining if all the vectors of the control group should be kept or if only the the top k-dimensional principal component approximation of the control matrix should be kept using the singular value decomposition (as in **?**. The default is to keep all the vectors.

```
> TTMAP_part2 <- TTMap::ttmap_part2(x = TTMAP_part1prime,k=dim(TTMAP_part1prime$Normal.mat)[2],
+ dataname = "Hello", normalname = "Hi");

           N1       N2       N3       N4       N5       N6
norm1 3.531803 4.498745 3.697451 3.709069 3.368290 3.913892
norm2 3.900278 3.801328 3.966333 4.135350 3.754063 3.590838
norm3 3.753479 4.003779 3.951802 3.777545 4.375957 4.107679
norm4 4.160317 3.907249 4.070722 4.634733 4.320016 3.904354
norm5 3.675789 3.915445 3.937037 4.041465 4.046106 4.114495
norm6 4.213132 4.655235 3.828912 3.661481 3.882219 3.846585

> head(TTMAP_part2$Dc.Dmat)

         B11.Dis   B12.Dis     B13.Dis   B21.Dis     B22.Dis B23.Dis
norm1  0.0000000 0.0000000  0.22416801  0.000000  0.00000000       0
norm2  0.1177527 0.0000000  0.06566366  0.000000 -0.03925713       0
norm3  0.0000000 0.3168190  0.00000000  0.000000 -0.33495136       0
norm4 -0.2970056 0.0000000 -0.17786351 -0.106407 -0.21233618       0
norm5  0.0000000 0.0911299  0.00000000  0.000000  0.00000000       0
norm6  0.0000000 0.0000000  0.00000000  0.000000  0.00000000       0
```

The outputs of this step are the following.

- The corrected control matrix, calculated at the first step is given in *Hi.NormalModel.pcl*, with a possible trimming of columns if k is different than the number of columns in the corrected matrix.

- The deviation component of each test sample is written in *Hello.Tdis.pcl*. An example of the deviation component is found in the previous script by writing $head(TTMAP_part2\$Dc.Dmat)The normal component of each test sample is written in Hello.Tnorm.$ The two values of this function are the deviation component matrix and the overall deviation (calculated by summing in absolute values the deviation components).

5 TTMap_part3

The third part corresponds to the Global-to-local Mapper part. One starts with an annotation file of our samples, in order to annotate the obtained clusters. In this example here we just copied several times the column

names. This annotation file needs to have as rownames the columns of the test samples followed by ".Dis". We then calculate the distance matrix between the samples using the *generate_mismatch_distance* function, which uses a cutoff parameter α in order to decide what is a considered as noise. Any other distance matrix can be computed here and used for the next step. Then, we calculate and output the clusters using *ttmap_part3*, which needs as inputs the values of *ttmap_part1, ttmap_part2.* The default parameter uses all the genes to calculate the overall deviation, but if a subset should be selected (only one pathway for example), it can be imputed here. TTMap$_part3then calculatesusing calcul_ea parameter of closeness using the data, in order to know wh$

```
> library(rgl)
> ALPHA <- 1
> annot <- c(paste(colnames(junk$TEST[,-c(1:3)]),"Dis",sep="."),paste(colnames(junk$CTRL[,-c(1:3)]),"Dis",sep="."))
>  annot <- cbind(annot,annot)
>  rownames(annot)<-annot[,1]
> dd5_sgn_only <-TTMap:::generate_mismatch_distance(TTMAP_part2,select=rownames(TTMAP_part2$Dc.Dmat),alpha = ALPHA)
>  de1 <- TTMap::ttmap_part3(TTMAP_part2,TTMAP_part2$m,select=rownames(TTMAP_part2$Dc.Dmat),annot,e= TTMap::calcul_e(d

[1] 10000
       norm1        norm2        norm3        norm4        norm5        norm6
7.034105e-06 4.836165e-12 1.024042e-09 6.749577e-08 4.622052e-14 2.897064e-06
[1] 4.834517
[1] "coucou"
[1] "0.0391050519118232:1e-04"
[1] "0.0964797118783403:2e-04"
[1] "0.0964797118783403:2e-04"
[1] "1:0.0101"
[1] "1:0.01"
[1] "1:0.01"
[1] "1:0.0104"
[1] "1:0.0101"
[1] "1:0.0098"
[1] "0.350753159900109:4e-04"
[1] "1:0.0099"
[1] "1:0.0098"
[1] "1:0.0101"
[1] "0.0391050519118232:1e-04"
[1] "0.19973141077333:3e-04"
[1] "e_map= 4e-04"
[1] "e_map= 4e-04"
[1] "e_map= 4e-04"
[1] "e_map= 4e-04"
[1] "e_map= 4e-04"
[1] "B11.Dis" "B12.Dis" "B13.Dis"
[1] "B23.Dis"
[1] "B12.Dis"
[1] "B13.Dis"
[1] "B21.Dis" "B22.Dis"

> rgl.postscript("test.pdf","pdf")
```

6 TTMap_part4

This last function analyses the different clusters for significant features. It outputs a file per level (one for overall, called all, one for the lower quartile, called low, one for the second quartile, called mid1, the third, mid2, and the higher quartile, called high). In each of them one file per cluster is given, with the list of significant genes linked to the cluster.

```
> TTMap:::ttmap_part4(de1, TTMAP_part2, TTMAP_part1prime, annot, n = 2, a = ALPHA, filename = "TEST2", annot = TTMAP_p

           [,1]
diff3  -1.716508
diff5  -1.423398
diff8  -1.798135
diff10 -1.220030
diff11 -1.263687
diff12 -1.663409
       [,1]     [,2]
diff3  "diff3"  "-1.71650836662187"
diff5  "diff5"  "-1.42339772701599"
diff8  "diff8"  "-1.79813490249846"
diff10 "diff10" "-1.22002983533383"
diff11 "diff11" "-1.26368744716498"
diff12 "diff12" "-1.66340872770607"
            [,1]
norm3302 1.031810
norm4659 1.027210
diff1    1.772229
diff2    1.237586
diff3    1.638131
diff4    1.344975
         [,1]       [,2]
norm3302 "norm3302" "1.0318104363959"
norm4659 "norm4659" "1.02721041305251"
diff1    "diff1"    "1.77222885905873"
diff2    "diff2"    "1.23758588950026"
diff3    "diff3"    "1.63813149827498"
diff4    "diff4"    "1.34497531755836"
                V1
norm3566 -1.000295
norm6592 -1.014647
diff1     1.361391
diff2     1.727683
diff3     1.872485
diff4     1.765962
[1] norm3566 norm6592 diff1    diff2    diff3    diff4
10000 Levels: diff1 diff10 diff100 diff11 diff12 diff13 diff14 diff15 diff16 diff17 diff18 diff19 diff2 diff20 diff21
```