DASE User's Guide

Xiang Liu xiang.liu@moffitt.org
Mingxiang Teng mingxiang.teng@moffitt.org
Department of Biostatistics and Bioinformatics
Moffit Cancer Center, Tampa, FL, USA

2023-01-05

Contents

Introduction

Super enhancers (SEs) are broad enhancer domains usually containing multiple constituent enhancers that hold elevated activities in gene regulation. Disruption in one or more constituent enhancers causes aberrant SE activities that lead to gene dysregulation in diseases. To quantify SE aberrations, differential analysis is performed to compare SE activities between cell conditions. The state-of-art strategy in estimating differential SEs relies on overall activities and neglect the changes in length and structure of SEs. DASE uses a weighted spline model to identify differential SEs between two conditions by accounting for the combinatorial effects of constituent enhancers weighted with their activities and locations (internal dynamics). In addition to overall changes, our method finds four novel types (shortened, shifted, hollowed and other complex scenarios) of differential SEs pointing to the structural differences within SEs.

Getting Started

Load the package in R

```
library(DASE)
```

Preparing Inputs

DASE requires the following input files:

1. Enhancer BED file: a merged enhancer BED file including the enhancers (e.g. H3K27Ac sequencing peaks) from all compared samples.
2. SE BED file: a merged SE BED file including the SE regions from all compared samples.
3. Coverage files: either the path of BAM/BigWig files for each condition and replicate, or a sequencing count table for all enhancers across conditions and replicates.

Below are the examples of the enhancer and SE BED files, following the UCSC definition (https://genome.ucsc.edu/FAQ/FAQformat.html).

Enhancer BED file

The enhancer BED file can be of any table formats as long as the first 6 columns contain the information of "chromosome","start","end","name","score" and "strand". Here is an example of BED files with only 6 columns.

```
# load enhancer BED file
enhancer_path <- system.file("extdata","enhancer.bed",package="DASE")
enhancer_region <- read.table(enhancer_path,sep="\t",header=F)
head(enhancer_region)
#>      V1       V2       V3         V4   V5 V6
#> 1 chr21 10119622 10119934 Peak_59320   58  .
#> 2 chr21 10413373 10414538  Peak_2651 1000  .
#> 3 chr21 13973708 13974647 Peak_51112   67  .
#> 4 chr21 14027434 14027662 Peak_40070   88  .
#> 5 chr21 14381282 14381485 Peak_44344   77  .
#> 6 chr21 14382640 14384785  Peak_2461 1000  .
```

SE BED file

The SE BED file can be of any formats as long as the first 6 columns contain the information of "chromosome","start","end","name","score",and "strand". Here is an example of BED files with only 6 columns.

```
# load SE BED file
se_path <- system.file("extdata","SE.bed",package="DASE")
se_region <- read.table(se_path,sep="\t",header=F)
head(se_region)
#>      V1       V2       V3                         V4   V5 V6
#> 1 chr21 21145883 21202220 17_Peak_26941_lociStitched  664  .
#> 2 chr21 43767083 43811262 10_Peak_18475_lociStitched 1080  .
#> 3 chr21 39347640 39389180 11_Peak_22013_lociStitched  887  .
#> 4 chr21 14737140 14764803  12_Peak_1337_lociStitched  847  .
#> 5 chr21 29634547 29660543 14_Peak_44019_lociStitched   81  .
#> 6 chr21 43880647 43903183  5_Peak_41657_lociStitched  949  .
```

Basic Usage of *DASE*

In this section, we use DASE to find differential SEs by comparing SE profiles on **chromosome 21** between two cancer cell lines (K562 and MCF7). Here, we focus on the basic usage of running DASE with different coverage input files (BAM, BigWig, and enhancer raw count table). By default, DASE will run permutations to determine a significant threshold to aid the defining of different SE categories. Additional options, such as blacklist region removal and running without permutation can be found later in the "Additional Options" section.

Run DASE with BAM or BigWig coverage files

This section demonstrates *DASE* with BAM or BigWig input files. The BAM or BigWig files are used to estimate the reads counts of enhancers in different samples. *DASE* uses *featureCounts* to count the enhancer abundance from BAM files or uses *rtracklayer* for BigWig files. The first step is to concatenated the paths of all BAM or BigWig files for each sample/condition into one string separated by ",". Then, with the specified enhancer and SE regions, we can run *DASE* as follow. (Here we use BigWig files as an example).

```
# path of BigWig file for each condition
s1_r1_bw <- system.file("extdata","K562_1_chr21.bw",package="DASE")
s1_r2_bw <- system.file("extdata","K562_2_chr21.bw",package="DASE")
s1_r3_bw <- system.file("extdata","K562_3_chr21.bw",package="DASE")
s2_r1_bw <- system.file("extdata","MCF7_1_chr21.bw",package="DASE")
s2_r2_bw <- system.file("extdata","MCF7_2_chr21.bw",package="DASE")
s2_r3_bw <- system.file("extdata","MCF7_3_chr21.bw",package="DASE")
s2_r4_bw <- system.file("extdata","MCF7_4_chr21.bw",package="DASE")

# concatenated path into one string for each sample/condition
c1_path <- paste(s1_r1_bw,s1_r2_bw,s1_r3_bw,sep=",")
c2_path <- paste(s2_r1_bw,s2_r2_bw,s2_r3_bw,s2_r4_bw,sep=",")

# running DASE with BigWig files
DASE_out <- DASE(se_in=se_region,e_in=enhancer_region,
                 data_type = "bw",
                 condition_1=c1_path,
                 condition_2=c2_path)
#> [1] "Step 1: merge and filter SE"
#> [1] "Step 2: calculate log2FC of constituent enhancer using Deseq2 with bw file"
#> [1] "Step 3: b-spline fit log2FC"
#> [1] "Processing total of 34 SEs"
#> [1] "Step 4: permutation with bs-spline to get log2FC cutoff"
#> [1] "Permutation: 1"
#> [1] "Permutation: 2"
#> [1] "Permutation: 3"
#> [1] "Permutation: 4"
#> [1] "Permutation: 5"
#> [1] "Permutation: 6"
#> [1] "Permutation: 7"
#> [1] "Permutation: 8"
#> [1] "Permutation: 9"
#> [1] "Permutation: 10"
#> [1] "Step 5: pattern segments process"
#> [1] "Step 6: final category estimate"
```

In order to run *DASE* with BAM file, two more parameters (c1_pair and c2_pair) need to be specified indicating whether the BAM files are paired-reads or not. An example command is shown as following. Please refer to ?DASE for more information.

```
# running DASE with BigWig files
DASE_out <- DASE(se_in=se_region,e_in=enhancer_region,
                 data_type = "bw",
                 condition_1="path_to_bam/c1_r1.bam,path_to_bam/c1_r2.bam,...",
                 condition_2="path_to_bam/c2_r1.bam,path_to_bam/c2_r2.bam,
                              path_to_bam/c2_r3.bam,...",
                 c1_pair = "F,T,...",
                 c2_pair = "T,F,F,...")
```

Run DASE with enhancer raw count table

This section demonstrates DASE with enhancer raw count table. Instead of count enhancer reads from BAM or BigWig files, the function adapts count table directly to save time and space. The format of count table is shown below. The first column must be the enhancer name with "chr_start_end" format. The order of following columns need to be condition 1 replicates and condition 2 replicates. The count table is adapted with parameter *enhancer_count_table*.

```
# read enhancer count table
enhancer_count_path <- system.file("extdata",
                                   "chr21_enhancer_count_mutiple_replicates.txt",
                                   package="DASE")
enhancer_count <- read.table(enhancer_count_path,sep="\t",header=T)
head(enhancer_count)
#>                  enhancer S1_r1 S1_r2 S1_r3 S2_r1 S2_r2 S2_r3 S2_r4
#> 1 chr21_5128185_5128529    12     2     4    21     5    42    10
#> 2 chr21_5240507_5241144    29    20    40    15     1    30     2
#> 3 chr21_5241953_5242568    33    29    58    11     2    22     4
#> 4 chr21_5242733_5243984   160    68   136    12     7    24    14
#> 5 chr21_5244027_5244554    52    29    58    13     4    26     8
#> 6 chr21_5244634_5245418    59    19    38    13     1    26     2

# run DASE, need to specify the number of replicates
# in each condition with parameter c1_n, c2_n
DASE_out_count <- DASE(se_in = se_region,
                       e_in = enhancer_region,
                       enhancer_count_table=enhancer_count,
                       c1_n=3,
                       c2_n=4)
#> [1] "Step 1: merge and filter SE"
#> Step 2: calculate log2FC of constituent enhancer using Deseq2
#>    with raw enhancer count table.
#> [1] "Step 3: b-spline fit log2FC"
#> [1] "Processing total of 34 SEs"
#> [1] "Step 4: permutation with bs-spline to get log2FC cutoff"
#> [1] "Permutation: 1"
#> [1] "Permutation: 2"
#> [1] "Permutation: 3"
#> [1] "Permutation: 4"
#> [1] "Permutation: 5"
#> [1] "Permutation: 6"
#> [1] "Permutation: 7"
#> [1] "Permutation: 8"
#> [1] "Permutation: 9"
#> [1] "Permutation: 10"
#> [1] "Step 5: pattern segments process"
#> [1] "Step 6: final category estimate"
```

Run DASE with different spline functions

DASE uses spline functions to fit log2 fold change values of constituent enhancers between conditions. We chose to implement 3 widely-adapted spline functions (b-spline, natural spline, and smooth.spline) in *DASE* to provide flexibility, although we only reported results from b-splines in our manuscript due to its superiority in practice. *DASE* uses b-spline as default. The three functions provide similar results overall. However, they could have very different performance in some cases. The examples below show a brief comparison between spline functions.

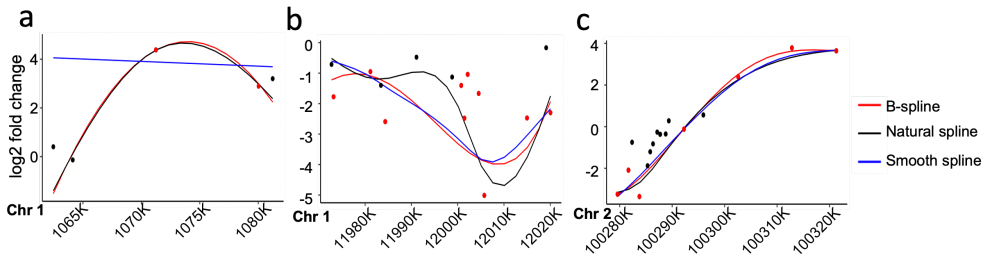


Figure 1: Spline fitting on different SEs.

Smooth spline performs badly on SEs with a small number of constituent enhancers (fitting points) (Figure 1a). When there are more number of data points (usually larger than 6), all spline functions show similar performance (Figure 1c). However, in practice, natural splines might overfit when regression weights are considered based on our algorithms (Figure 1b). Note that we implement a strategy to control overfitting with b-spline (see our manuscript).

Users can chose different spline functions with *spline_fun* parameter in "DASE". Following are some examples of run *DASE* with different spline functions.

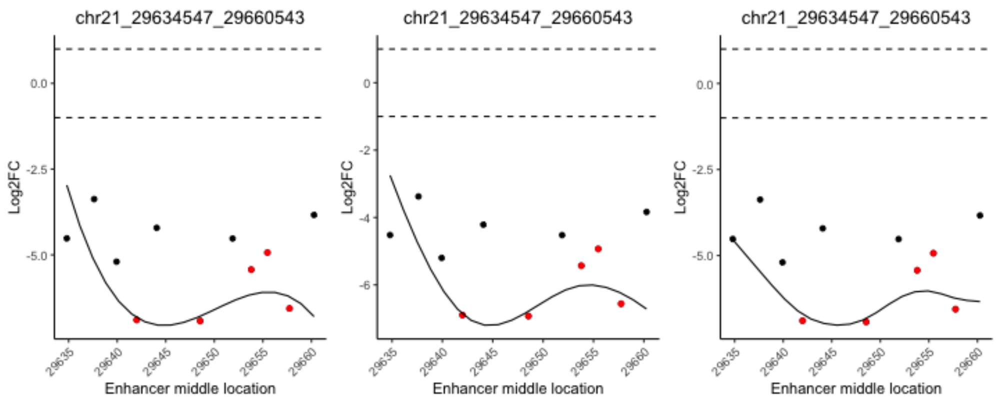
```
# rund DASE with b-spline function (default)
DASE_out_bs <- DASE(se_in=se_region,e_in=enhancer_region,
                    enhancer_count_table=enhancer_count,
                    c1_n=3,c2_n=4,
                    permut=F)
#> [1] "Step 1: merge and filter SE"
#> Step 2: calculate log2FC of constituent enhancer using Deseq2
#>    with raw enhancer count table.
#> [1] "Step 3: b-spline fit log2FC"
#> [1] "Processing total of 34 SEs"
#> [1] "Step 4: no permutation, skipping"
#> [1] "Step 5: pattern segments process"
#> [1] "Step 6: final category estimate"

# run DASE with natural spline function
```

```
DASE_out_ns <- DASE(se_in=se_region,e_in=enhancer_region,
                    enhancer_count_table=enhancer_count,
                    c1_n=3,c2_n=4,
                    permut=F,
                    spline_fun = "ns",)
#> [1] "Step 1: merge and filter SE"
#> Step 2: calculate log2FC of constituent enhancer using Deseq2
#>    with raw enhancer count table.
#> [1] "Step 3: natural spline fit log2FC"
#> [1] "Processing total of 34 SEs"
#> [1] "Step 4: no permutation, skipping"
#> [1] "Step 5: pattern segments process"
#> [1] "Step 6: final category estimate"

# run DASE with smooth spline function
DASE_out_smooth <- DASE(se_in=se_region,e_in=enhancer_region,
                        enhancer_count_table=enhancer_count,
                        c1_n=3,c2_n=4,
                        permut=F,
                        spline_fun = "smooth")
#> [1] "Step 1: merge and filter SE"
#> Step 2: calculate log2FC of constituent enhancer using Deseq2
#>    with raw enhancer count table.
#> [1] "Step 3: smooth spline fit log2FC"
#> [1] "Processing total of 34 SEs"
#> [1] "Step 4: no permutation, skipping"
#> [1] "Step 5: pattern segments process"
#> [1] "Step 6: final category estimate"
```

```
# spline fitted plots of different spline functions
library(gridExtra)
grid.arrange(DASE_out_bs$pattern_list[[7]],
             DASE_out_ns$pattern_list[[7]],
             DASE_out_smooth$pattern_list[[7]],nrow=1)
```



Left is b-spline; middle is natural spline; right is smooth spline.

Interpretation of *DASE* Outputs

The output of *DASE* is a list with multiple data types including:

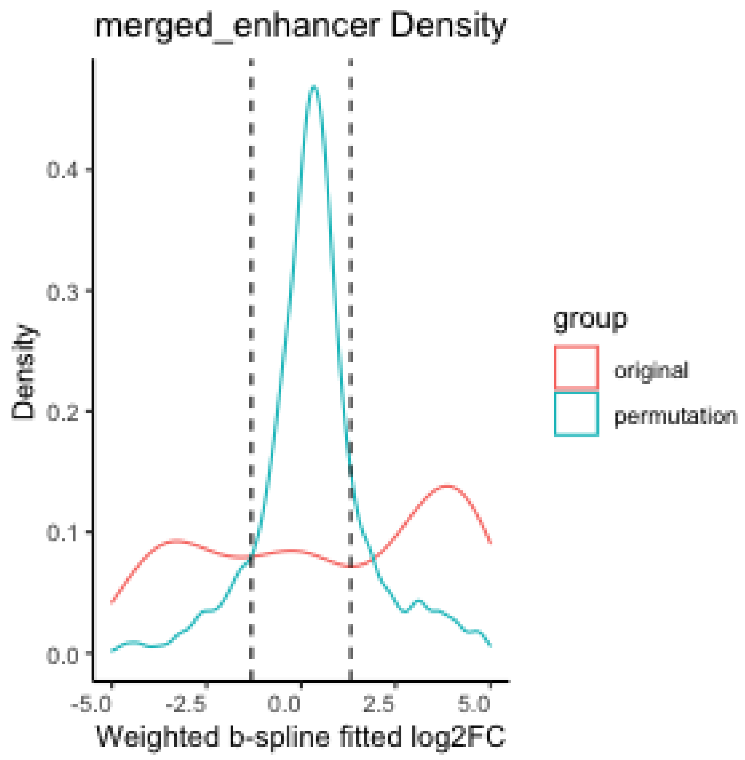
1. lfc_shrink: a shrinking lfc object from *DESeq2* for all enhancers. It can be used to creat MA plot.
2. cutoff: significant threshold for fitted log2 fold changes.
3. density_plot: a density plot of permutation and original fitted log2 fold changes, if *permut=T*.
4. boxplot: a boxplot of final SE categories.
5. se_category: a data frame containing final SE categories.
6. pattern_list: a list containing figures for each SE pattern.
7. ce_fit: a data frame containing DESeq2 output and spline-fitted log2 fold change of all constitute enhancers.

Significant threshold

We use permutation of spline fitted log2 fold changes to decide the significant threshold. Under default settings, *DASE* will run permutation 10 times with *SEpermut* function. If permutation is disabled, the default significant threshold is -1 and 1. Users can also choose their own thresholds with *cutoff_v* parameter. Please refer to the *Additional options* section for this.

```
# Significant threshold
DASE_out_count$cutoff
#> [1] -1.313807  1.313807
```

```
# Permutation density plot
DASE_out_count$density_plot
```



Black dash lines indicate the thresholds which are obtained based on the inflection points of the permutation distribution.

Super-enhancer internal dynamic categories

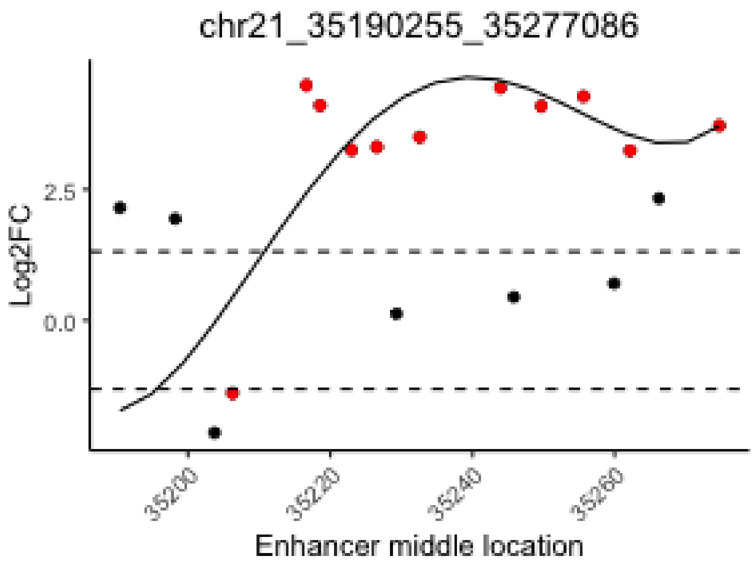
SE_category, pattern_list, ce_fit are the outputs related to SE internal dynamics.

```
# se_categories
head(DASE_out_count$se_category)
#>              se_merge_name total_width number_enhancer category direction
#> 36 chr21_40366721_40390218    9296.007               5  Overall         +
#> 9  chr21_29634547_29660543    6045.861              11  Overall         -
#> 35 chr21_40301259_40351803   17076.367              11  Overall         +
#> 37 chr21_41677251_41688506    2818.363               2  Overall         +
#> 38 chr21_41750932_41787919    1506.602               7  Overall         +
#> 30 chr21_38903295_38921596    1709.170               8  Overall         +
#>    non_mid_percent   mean_FC rank
#> 36               1  5.794619    1
#> 9                1 -5.128361    2
#> 35               1  4.763041    3
#> 37               1  4.698742    4
#> 38               1  4.472774    5
#> 30               1  4.116843    6
```

Each column represents the following:

1. se_merge_name: name of merged SE,"chr_start_end".
2. total_width: width of merged SE (unit=k).
3. number_enhancer: number of CEs in each SE.
4. category: SE category identified by *DASE*.
5. direction: enrichment direction of SEs (none: Other or non-differential category; +: enriched in sample 2; -: enriched in sample 1; l: sample 1 shifted into 5' direction; r: sample 2 shifted into 5' direction).
6. non_mid_percent: total activity occupancy of the segments that go beyond the threshold cutoffs.
7. mean_FC: mean of the log2 fold change of SE coverage.
8. rank: SE category ranking based on *non_mid_percent* and *mean_FC* in each SE category. (rank=1 means the most changed in the corresponding SE category.)

```
# an example of one sample
DASE_out_count$pattern_list[[11]]
```



This figure shows the constituent enhancer patterns within a SE example which is identified as *Shortened*. Black line is the fitted log2 fold change curve; dots indicate constituent enhancers. Red dots indicate the constituent enhancers with heavy weights.

```
# example of ce_fit
head(DASE_out_count$ce_fit)
#>                e_merge_name   chr     start       end width C1_r1 C1_r2 C1_r3
#> 1: chr21_14751738_14756723 chr21 14751738 14756723  4986   463   471   942
#> 2: chr21_14760734_14764325 chr21 14760734 14764325  3592   254   303   606
#> 3: chr21_14750604_14751557 chr21 14750604 14751557   954    61    49    98
#> 4: chr21_14749603_14750540 chr21 14749603 14750540   938    52    46    92
#> 5: chr21_14757516_14758773 chr21 14757516 14758773  1258    63    33    66
#> 6: chr21_14737131_14737833 chr21 14737131 14737833   703    35    31    62
#>    C2_r1 C2_r2 C2_r3 C2_r4 C1_r1_norm C1_r2_norm C1_r3_norm C2_r1_norm
#> 1:    48    13    96    26  838.97559  1990.2043  1990.2043  16.043634
#> 2:    91    30   182    60  460.25875  1280.3225  1280.3225  30.416055
#> 3:    14     9    28    18  110.53458   207.0489   207.0489   4.679393
#> 4:    14     8    28    16   94.22620   194.3724   194.3724   4.679393
#> 5:    11     7    22    14  114.15867   139.4411   139.4411   3.676666
#> 6:    13     6    26    12   63.42148   130.9901   130.9901   4.345151
#>    C2_r2_norm C2_r3_norm C2_r4_norm baseMean_shrink log2FoldChange_shrink
#> 1:  19.862262  16.043634  19.862262       698.74227             -6.508346
#> 2:  45.835989  30.416055  45.835989       453.34397             -4.743660
#> 3:  13.750797   4.679393  13.750797        80.21324             -4.387851
#> 4:  12.222930   4.679393  12.222930        73.82509             -4.390356
#> 5:  10.695064   3.676666  10.695064        60.25489             -4.383653
#> 6:   9.167198   4.345151   9.167198        50.34662             -4.136973
#>    lfcSE_shrink pvalue_shrink  padj_shrink         padj   C2_mean  max_mean
#> 1:    0.3770104    -17.263041 8.928198e-67 9.292194e-65 17.952948 1606.4614
#> 2:    0.4090548    -11.596638 4.285867e-31 8.283968e-30 38.126022 1006.9679
#> 3:    0.5087667     -8.624486 6.438137e-18 5.184999e-17  9.215095  174.8774
#> 4:    0.5006936     -8.768549 1.809836e-18 1.530443e-17  8.451162  160.9903
#> 5:    0.4868648     -9.003841 2.179562e-19 1.979159e-18  7.185865  131.0136
#> 6:    0.4864625     -8.504197 1.828566e-17 1.397768e-16  6.756174  108.4672
#>    width_mid   percent   cumsum spline_bs
#> 1:  14754.23 42.378660 42.37866 -6.112893
#> 2:  14762.53 26.563945 68.94260 -4.469933
#> 3:  14751.08  4.613289 73.55589 -5.208772
#> 4:  14750.07  4.246946 77.80284 -4.772417
#> 5:  14758.15  3.456156 81.25900 -6.032781
#> 6:  14737.48  2.861379 84.12037 -2.690306
```

Columns from “e_merge_name” to “width” indicate the characteristics of each CE. Columns from “C1_r1” to “C2_r4_norm” indicate the coverage and normalized coverage of each CE in each condition. Columns from “baseMean” to “pvalue” indicate the differential testing results from *DESeq2*. Columns from “baseMean_shrink” to “padj_shrink” indicate shrinkage estimation of differential anlaysis from *DESeq2*. Please refer to help functions in *DASE* for more information.

Additional Options

In addition of default parameters, *DASE* can exclude the ENCODE blacklist regions from consideration. *DASE* can also adapt an user-define blacklist region with *custom_range* parameter. We have included the human blacklist region file from ENCODE (accession ID: ENCFF356LFX) in our package.

```
# blacklist file
blacklist_path <- system.file("extdata","region_blacklist.bed",package="DASE")
blacklist_region <- read.table(blacklist_path,sep="\t",header=F)
head(blacklist_region)
#>     V1       V2       V3
#> 1 chr1   628903   635104
#> 2 chr1  5850087  5850571
#> 3 chr1  8909610  8910014
#> 4 chr1  9574580  9574997
#> 5 chr1 32043823 32044203
#> 6 chr1 33818964 33819344
```

There is also an option to opt-out permutation calculation. The default setting of DASE will run permutation 10 times. Users can turn it off with *permut=F*. When there is no permutation, the default thresholds are -1 and 1, or users can choose the thresholds they like with *cutoff_v* parameter. However, defining customized thresholds are only available under *permut=F*.

DASE with enhancer blacklist region

Example of using blacklist option.

```
# run DASE with blacklist file and customized region
DASE_out_bl <- DASE(se_in=se_region,e_in=enhancer_region,bl_file = blacklist_region,
                  custom_range = c("chr21:14735763-29634779","chr21:33539902-43710703"),
                  enhancer_count_table=enhancer_count,
                  c1_n=3,c2_n=4)
#> [1] "Step 1: merge and filter SE"
#> Step 2: calculate log2FC of constituent enhancer using Deseq2
#>    with raw enhancer count table.
#> [1] "Step 3: b-spline fit log2FC"
#> [1] "Processing total of 8 SEs"
#> [1] "Step 4: permutation with bs-spline to get log2FC cutoff"
#> [1] "Permutation: 1"
#> [1] "Permutation: 2"
#> [1] "Permutation: 3"
#> [1] "Permutation: 4"
#> [1] "Permutation: 5"
#> [1] "Permutation: 6"
#> [1] "Permutation: 7"
#> [1] "Permutation: 8"
#> [1] "Permutation: 9"
#> [1] "Permutation: 10"
#> [1] "Step 5: pattern segments process"
#> [1] "Step 6: final category estimate"

# number of SEs with blacklist range
nrow(DASE_out_bl$se_category)
#> [1] 8

# number of SEs without blacklist range
nrow(DASE_out_count$se_category)
#> [1] 34
```

Users can find that the number of SEs (i.e. 8) in "DASE_out_bl" is less than that in "DASE_out_count" which is calculated without blacklist region (i.e. 34).

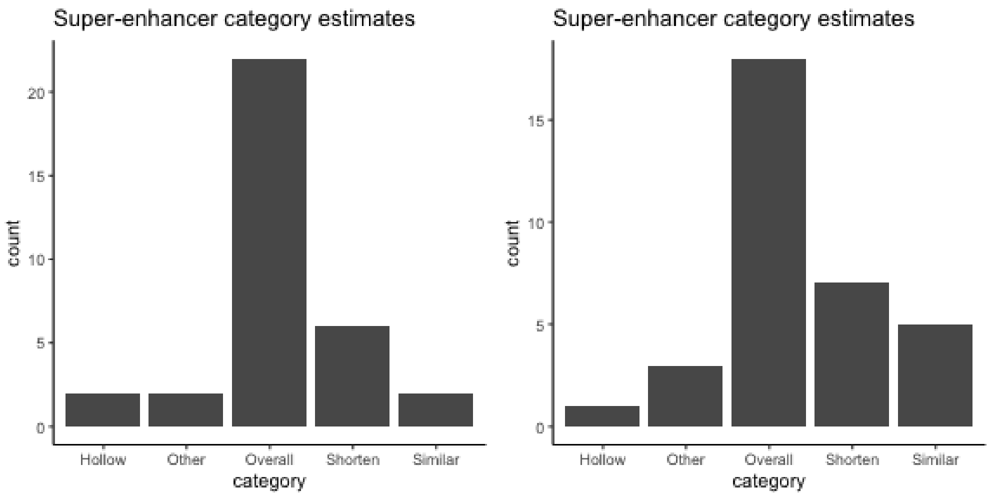
DASE with no permutation

Two examples of different permutation options.

```
# run DASE with permutation 3 times
DASE_out <- DASE(se_in=se_region,e_in=enhancer_region,times=3,
                 cutoff_v = c(-2,2),enhancer_count_table=enhancer_count,
                 c1_n=3,c2_n=4)
#> [1] "Step 1: merge and filter SE"
#> Step 2: calculate log2FC of constituent enhancer using Deseq2
#>    with raw enhancer count table.
#> [1] "Step 3: b-spline fit log2FC"
#> [1] "Processing total of 34 SEs"
#> [1] "Step 4: permutation with bs-spline to get log2FC cutoff"
#> [1] "Permutation: 1"
#> [1] "Permutation: 2"
#> [1] "Permutation: 3"
#> [1] "Step 5: pattern segments process"
#> [1] "Step 6: final category estimate"
```

```
# run DASE with customized threshold
DASE_out_p <- DASE(se_in=se_region,e_in=enhancer_region,permut = F,
                 cutoff_v = c(-3,3),enhancer_count_table=enhancer_count,
                 c1_n=3,c2_n=4)
#> [1] "Step 1: merge and filter SE"
#> Step 2: calculate log2FC of constituent enhancer using Deseq2
#>    with raw enhancer count table.
#> [1] "Step 3: b-spline fit log2FC"
#> [1] "Processing total of 34 SEs"
#> [1] "Step 4: no permutation, skipping"
#> [1] "Step 5: pattern segments process"
#> [1] "Step 6: final category estimate"
```

```
# boxplots of different threshold
library(gridExtra)
grid.arrange(DASE_out_count$boxplot,
             DASE_out_p$boxplot,nrow=1)
```



By comparing results based on different threshold cutoffs, the right plot (with larger threshold cutoff) identified more SEs in the *similar* category than the left plot (with smaller threshold cutoff).

Citation

Please cite our paper when using DASE: [https://doi.org/10.1093/nar/gkac141]