Introduction to Clipper

Xinzhou Ge

2020-09-14

The most common goal of analyzing high-throughput data is to contrast two conditions to reliably screen "interesting features", where "interesting" means "differential" or "enriched". Differential features are defined as those that have different expected measurements (without measurement errors) between two conditions, and the detection of such differential features is called differential analysis. For example, popular differential analyses include the identification of differentially expressed genes (DEGs) from genome-wide gene expression data (e.g., microarray and RNA sequencing (RNA-seq) data) and differentially chromosomal interaction regions (DIRs) from Hi-C data. In contrast, enriched features are defined as those that have higher expected measurements under the experimental/treatment condition than the background condition, i.e., the negative control. The detection of such enriched features is called enrichment analysis. For example, common enrichment analyses include calling protein-binding sites in a genome from chromatin immunoprecipitation sequencing (ChIP-seq) data and identifying peptides from mass spectrometry (MS) data.

The package `Clipper` provides methods to identify interesting features without using p-values in differential or enrichment analysis. This vignette explains the general use of and typical workflows in DEG identification using RNA-seq data, peak calling using ChIP-Seq data, peptide identification from mass spectrometry data, and DIR analysis from Hi-C data. The details and examples of all these analyses will be further introduced below.

Standard workflow

Note: if you use `Clipper` in published research, please cite:

Input data

`Clipper` requires at least four inputs:

- `score.exp` the measurements from the background condition
- `score.back` the measurements from the experimental condition
- `analysis` the type of analysis, "differential" or "enrichment"
- `FDR` the target FDR threshold(s), set to 0.05 by default

In a differential analysis, the two conditions can be treated equally thus you can assign either condition as background/experimental.

```
library(Clipper)
```

```
exp_d[c(1:3, 1001:1003), ]
#>         replicates1 replicates2 replicates3
#> gene1            40          43          29
#> gene2            27          46          39
#> gene3            33          27          35
#> gene1001           6           9           4
#> gene1002           2           1           2
#> gene1003           1           3           5
back_d[c(1:3, 1001:1003), ]
#>          replicates1 replicates2 replicates3
#> gene1             19          28          27
#> gene2             29          18          29
#> gene3             17          17          13
#> gene1001          24          28          23
#> gene1002          15          16          18
#> gene1003           8          12           6
```

General pipeline

Before introducing the application of `Clipper` in specific biological analysis, we first introduce a general case where we use `Clipper` to find interesting features by contrasting two measurement matrices, one from the experimental condition and one from the background condition. The inputs of `score.exp` and `score.back` should be numeric matrices. The rows of `score.back` and `score.exp` should match and represent the same feature, and their columns represent replicates. For differential and enrichment analysis, set `analysis` to be `"differential"("d")` or `"enrichmentl"("e")` respectively.

For example, we can use the simulated dataset `exp_d` and `back_d` as inputs of `score.back` and `score.exp`. In these two dataset, there are 10,000 features and the first 2000 features are interesting. For the target FDR threshold(s), we can then use `Clipper` to perform differential analysis:

```
#use three FDR thresholds: 0.01, 0.05, 0.1
re1 <- Clipper(score.exp = exp_d, score.back = back_d, analysis = "differential", FDR = c(0.01, 0.05, 1
names(re1)
#> [1] "contrast.score"       "contrast.score.value" "FDR"
#> [4] "contrast.score.thre"  "discoveries"
```

`Clipper` returns a list and its component `discovery` is a list of indices of identified interesting discoveries corresponding to the FDR threshold(s):

```
#indices of identified interesting genes with the second input FDR threshold (0.05)
re1$discoveries[[2]][1:5]
#> [1] 1 3 5 6 7
```

We can then calculate the resulting false discovery proportion (FDP) and power:

```
#FDP (the first 2000 genes are true positives)
sum(re1$discoveries[[1]]>2000)/length(re1$discoveries[[1]])
#> [1] 0.003556188
#power
sum(re1$discoveries[[1]]<=2000)/2000
#> [1] 0.7005
```

DEG identification

To use `Clipper` for DEG analysis, set `analysis` to be `"differential"`. The input of `score.exp` and `score.back` should be read count matrices of transcriptomic data, such as RNA sequencing or microarray analysis. The rows of `score.back` and `score.exp` should match and represent the same set of genes, and their columns represent replicates. `Clipper` requires either `score.exp` or `score.back` to have at least two replicates. Users are recommended to normalize and log-tranform read count matrices before inputing them to `Clipper` (see examples below). They can use their preferred preprocessing steps.

Peak calling (as an add-on to MACS)

Although `Clipper` could be used as a standalone peak calling method, we recommend applying `Clipper` as an add-on to existing peak calling methods such as MACS or HOMER. Below we show how to use `Clipper` to call peaks with FDR control based on outputs from MACS.

Input data

To implement `Clipper` for peak calling, users need to extract two outputs from MACS: a "∗_peaks.narrowPeak" file and two "∗_pileup.dbg" files, one for the experimental track and the other for the control track. The "∗_peaks.narrowPeak" file contains candidate peaks identified by MACS with its second and third rows indicating the start and end points of the peaks(see below). The "∗_pileup.dbg" files contain the base-pair read coverages for genomic regions of interest (see below). See below section for details about generation of the "∗_peaks.narrowPeak" file and the two "∗_pileup.dbg" files.

```
experimental <-  read.table(paste0("syn/rep14/rep1/experimental_treat_pileup.bdg"))
head(experimental)
#      V1    V2    V3 V4
# 1 chr1     0  9852  0
# 2 chr1  9852  9913  1
# 3 chr1  9913 10150  2
# 4 chr1 10150 10175  1
# 5 chr1 10175 10211  2
# 6 chr1 10211 10256  1
control <- read.table(paste0("syn/rep14/rep1/control_treat_pileup.bdg"))
head(control)
#      V1   V2    V3 V4
# 1 chr1    0  9811  0
# 2 chr1 9811  9819  1
# 3 chr1 9819  9896  2
# 4 chr1 9896  9947  3
# 5 chr1 9947  9997  4
# 6 chr1 9997 10106  5
macs2.peak <- read.table("syn/rep14/rep1/twosample_peaks.narrowPeak")
head(macs2.peak[,1:3])
#      V1      V2      V3
# 1 chr1  799321  799619
# 2 chr1  904144  905827
# 3 chr1  940097  943865
# 4 chr1 1058976 1060651
# 5 chr1 1247846 1249093
# 6 chr1 1344798 1345405
```

Use Clipper to screen candidate peaks

We use the following codes to generate the input of `score.exp` and `score.back` from the two "∗_pileup.dbg" files. The resulting vectors `s1` and `s2` contains the read coverages for each base pair. Then we supply `s1` to score.exp, s2 to score.back, and use Clipper to perform enrichment analysis to obtain a threshold on per-base-pair contrast scores.

```
# create two vectors for the two conditions, which contains the read coverages for each base pair
s1 <- rep(experimental$V4, experimental$V3- experimental$V2 )
s2 <- rep(control$V4, control$V3- control$V2)
s1[(length(s1)+1):length(s2)] <- 0
# use Clipper
re <- Clipper(score.exp = s1, score.back = s2, analysis = "enrichment")
# the threshold on contrast scores
re$thre
# 6
```

We then use this threshold to screen the candidate peaks identified by MACS. We compare the median of per-base-pair contrast scores within each candidate peak with the threshold output by Clipper. Only peaks whose median per-base-pair contrast score is greater or equal to the threshold are identified as peaks.

```
median.peak <- sapply(1:nrow(macs2.peak), function(i){
  # start point of the candidate peak
  start <- macs2.peak$V2[i]+1
  # end point of the candiate peak
  end <- macs2.peak$V3[i]
  # the different contrast score is the difference between the two condiitons
  return(median(s1[start:end] -(s2[start:end])))
})
# peaks identified by Clipper
clipper.peak <- macs2.peak[median.peak >= re$thre,]
```

We don't directly use the discoveries output by Clipper becasuse they are base-pair features and do not form biologically meaningful peaks. Instead, we use better-defined peak regions by MACS.

Generation of MACS output files

We start from two files applicable for MACS peak calling, such as two bam files: experimental.bam for experimental condition and control.bam for background/negative control condition. We use the following shell scripts to obtain the "∗_peaks.narrowPeak" file and two "∗_pileup.dbg" files. See links for tutorials of MACS.

```
macs2 callpeak -t experimental.bam -c control.bam -f BAM -n twosample -B -q 1 --outdir
results
```

```
macs2 callpeak -t experimental.bam -f BAM -n exp -B -q 1 --outdir results
```

```
macs2 callpeak -t control.bam -f BAM -n back -B -q 1 --outdir results
```