An Introduction to BSDE

Mengqi Zhang

2021-10-24

This is a short introduction to the package BSDE. It consists of two parts: the first part illustrates the usage of the core function; the second part includes an example of using BSDE on real data.

Installation

BSDE can be installed from GitHub.

```
# install.packages("devtools")
devtools::install_github("mqzhanglab/BSDE")
```

Part I: Computing BSDE p-value

The core function from BSDE is `cal_w2_pval`. It calculates the p-values gene by gene. In most situation, the log-transformed expressions are recommended.

`cal_w2_pval` has three required inputs: `count_per_gene`, `meta_individual` and `meta_phenotype`.

This function return a list, which contains:

- `pval`: p-value
- `case_bc_ob`: density vector of the Barycenter distribution from cases
- `ctrl_bc_ob`: density vector of the Barycenter distribution from controls

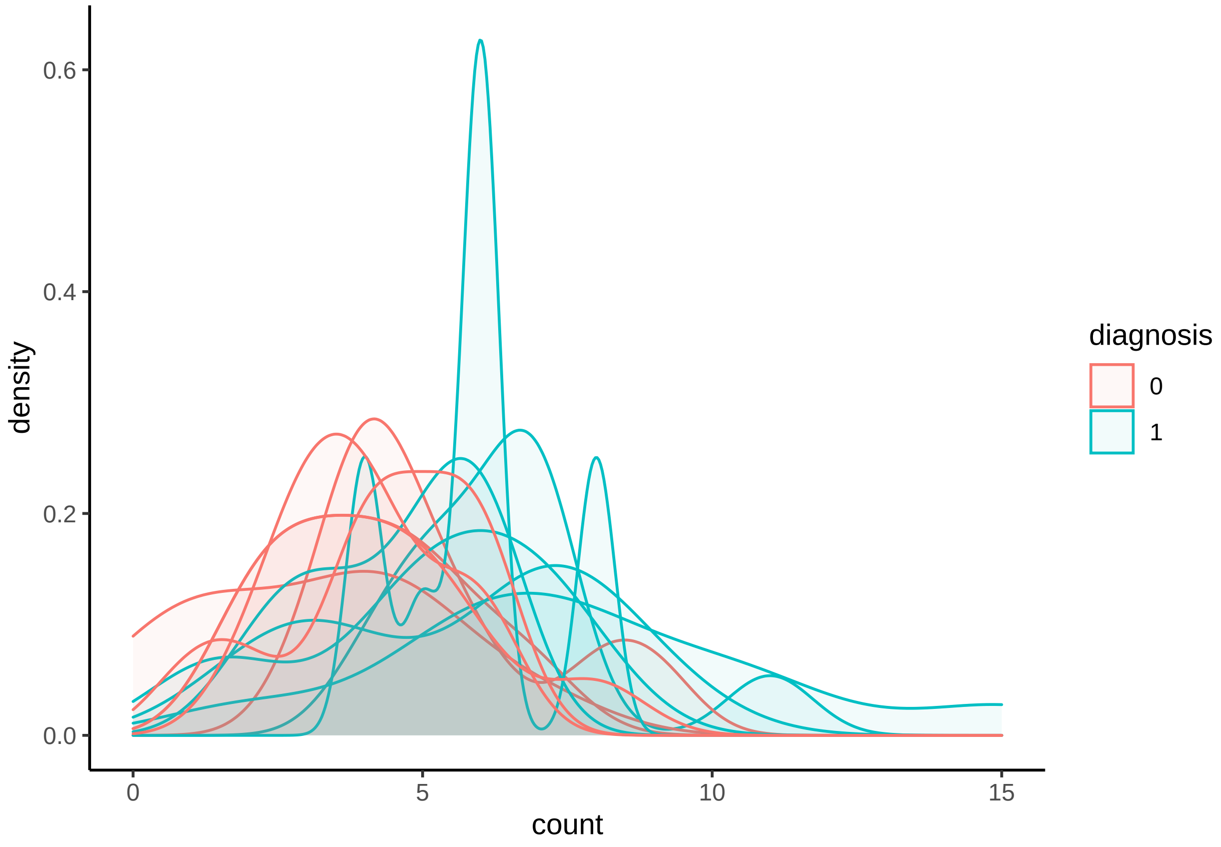
Example

```
library("BSDE")
library("ggplot2")
```

Suppose we have 12 subjects (6 cases and 6 controls), where each subject has 10 cells.

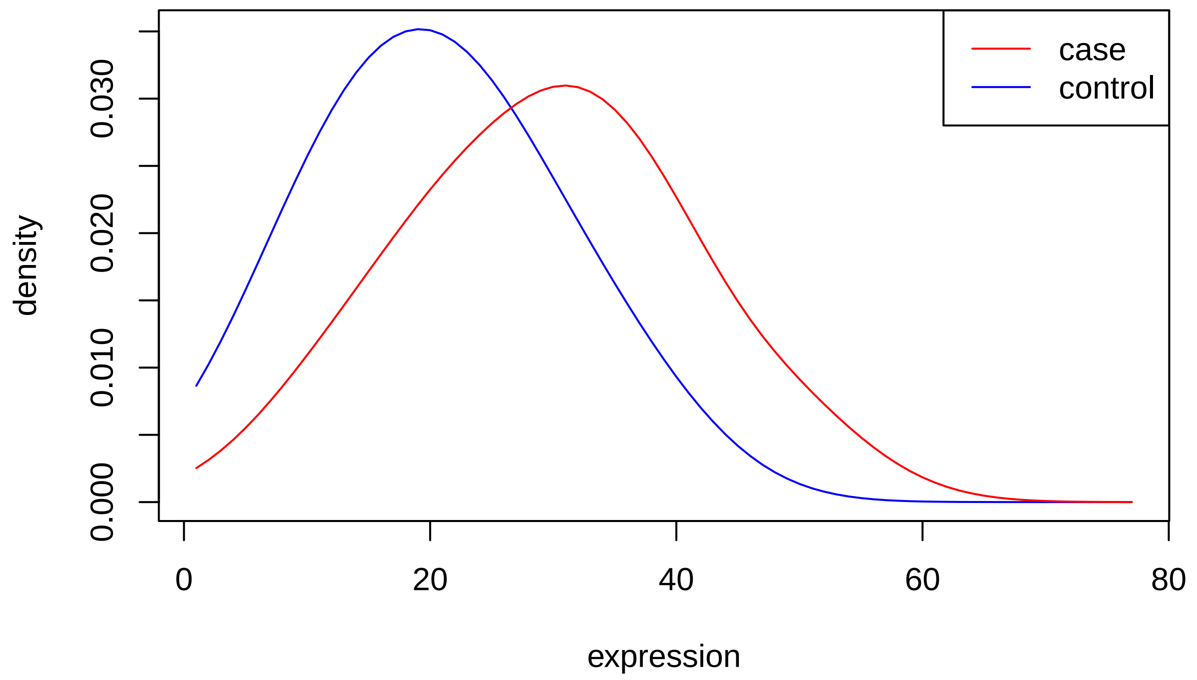
```
count_per_gene=c(rpois(60,6),c(rpois(60,4)))
meta_individual=paste0("ind",rep(1:12,each=10))
meta_phenotype=factor(c(rep(1,60),rep(0,60)))

# show dataset
table(meta_individual)
#> meta_individual
#>  ind1 ind10 ind11 ind12  ind2  ind3  ind4  ind5  ind6  ind7  ind8  ind9
#>    10    10    10    10    10    10    10    10    10    10    10    10
table(meta_phenotype)
#> meta_phenotype
#>  0  1
#> 60 60
```

```
# expressions
df=data.frame(count=count_per_gene,ind=meta_individual, diagnosis=meta_phenotype)
ggplot(df,aes(x=count,color=diagnosis,fill=diagnosis,group=ind)) +
  geom_density(alpha=0.05)+ theme_classic()
```



Let us calculate the p-value.

```
results <- cal_w2_pval(count_per_gene,meta_individual,meta_phenotype)
print(results[[1]])
#> [1] 0.15
plot(results[[3]], type="l", col="blue", xlab="expression", ylab="density")
lines(results[[2]], col="red")
legend("topright", c("case", "control"), lwd=c(1,1), col=c("red", "blue"))
```



Part II: Data analysis example

Here we present a demo simulation here. The simulation is based on a real autism dataset from this *paper*.

We generate 30 genes (from a particular cell type) for 20 case subjects and 20 control subjects. Each subject comes with 20 cells.

We simulate 4 types of differential expressions (DEs). The size of the differential expression is specified by a size factor.

1. **mean DE**: 3 genes (size factor 1.2).
2. **variance DE**: 3 genes (size factor 1.2).
3. **multimodality DE**: 3 genes(size factor 0.6).
4. **dispersion DE**: 3 genes (size factor 0.2).

Let us load the pre-simulated data from the package.

```
data("test_data")
```

Differential expression analysis

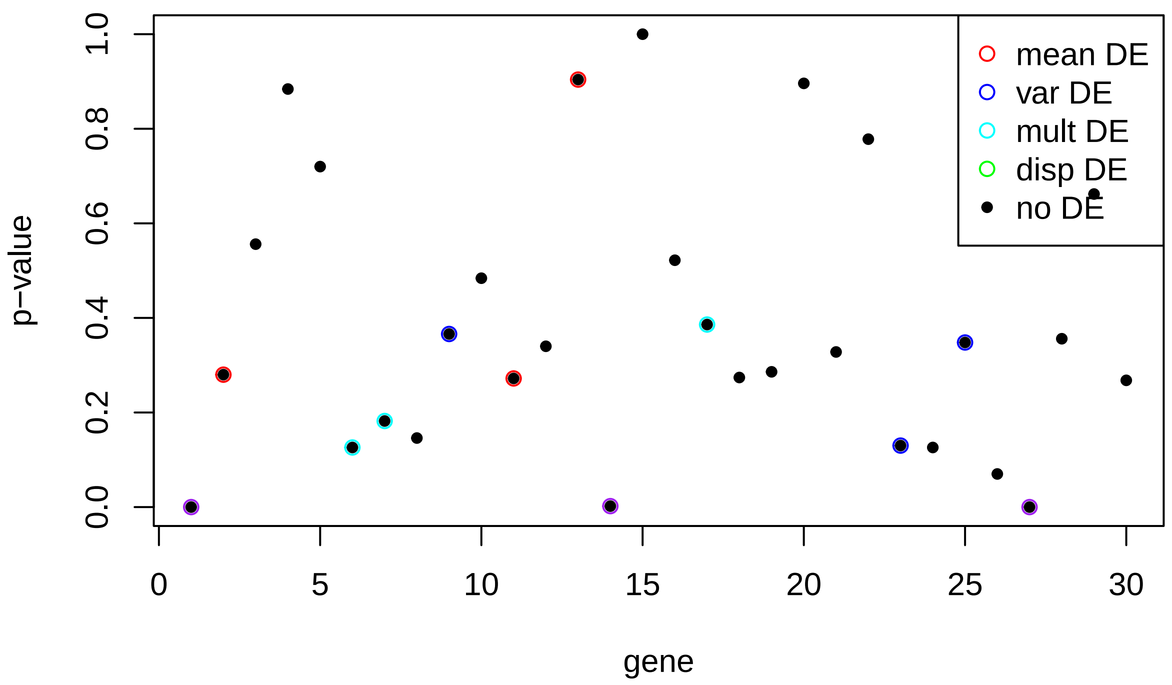
To calculate the p-values, we can either call `cal_w2_pval` for each gene or call `BSDE` on an assembled `SingleCellExperiment` object.

```
# log transform
sim_matrix_log = log2(1 + sim_matrix) #log transformed data

dim(sim_matrix_log)
#> [1]  30 800
sim_matrix_log[1:2, 1:5]
#>          cell1    cell2    cell3    cell4    cell5
#> gene1 3.906891 2.807355 6.392317 6.807355 4.584963
#> gene2 4.000000 8.317413 7.787903 7.312883 8.066089
```

```r
pvals = rep(0, nrow(sim_matrix_log))

print(date())
#> [1] "Sun Oct 24 16:19:20 2021"
print(gc())
#>           used  (Mb) gc trigger  (Mb) limit (Mb) max used  (Mb)
#> Ncells 1968267 105.2    3772024 201.5         NA  2773018 148.1
#> Vcells 3509066  26.8    8388608  64.0      16384  6158720  47.0

for (i_g in 1:nrow(sim_matrix_log)) {
  cur_sim = sim_matrix_log[i_g, ]
  cur_ind = meta$individual
  cur_pheno = meta$phenotype

  pvals[i_g] = tryCatch({
    cal_w2_pval(
      count_per_gene = cur_sim,
      meta_individual = cur_ind,
      meta_phenotype = cur_pheno,
      perm_num = 500,
      unif_round_unit = 0.2
    )[[1]]
  }, error = function(e) {
    NA
  })
}
print(date())
#> [1] "Sun Oct 24 16:27:54 2021"
print(gc())
#>           used  (Mb) gc trigger  (Mb) limit (Mb) max used  (Mb)
#> Ncells 1969348 105.2    3772024 201.5         NA  3772024 201.5
#> Vcells 3516424  26.9    8388608  64.0      16384  6454912  49.3

names(pvals) <- row.names(sim_matrix)
print(pvals)
#>  gene1  gene2  gene3  gene4  gene5  gene6  gene7  gene8  gene9 gene10 gene11
#>  0.000  0.280  0.556  0.884  0.720  0.126  0.182  0.146  0.366  0.484  0.272
#> gene12 gene13 gene14 gene15 gene16 gene17 gene18 gene19 gene20 gene21 gene22
#>  0.340  0.904  0.002  1.000  0.522  0.386  0.274  0.286  0.896  0.328  0.778
#> gene23 gene24 gene25 gene26 gene27 gene28 gene29 gene30
#>  0.130  0.126  0.348  0.070  0.000  0.356  0.662  0.268
```

Let us plot the p-values.

```r
idx <- 1:length(pvals)
plot(idx, pvals, pch=20, xlab="gene", ylab="p-value")
points(idx[de.mean==1], pvals[de.mean==1], col="red", pch=1)
points(idx[de.var==1], pvals[de.var==1], col="blue", pch=1)
points(idx[de.mult==1], pvals[de.mult==1], col="cyan", pch=1)
points(idx[de.dp==1], pvals[de.dp==1], col="purple", pch=1)
legend("topright", c("mean DE", "var DE", "mult DE", "disp DE", "no DE"), pch=c(1,1,1,1,20),
       col=c("red", "blue", "cyan", "green", "black"))
```



Alternatively, the data can be assembled into a `SingleCellExperiment` object with the "counts" in assays, a rowData named "gene_id" as gene_names and two columns named as "individual" and "condition" respectively. Column `individual` is a factor that represents individual labels. Column `condition` is an indicator vector (1: case, 0: control).

```
gene_id <- rownames(sim_matrix)   #get the gene id from the data
meta$condition <- meta$phenotype
sce <- SingleCellExperiment::SingleCellExperiment(list(counts=sim_matrix),
                                                  colData=meta,rowData=gene_id)
```

Now we call the function `BSDE` to compute the p-values.

```
pvals.2 <- BSDE(sce, perm_num = 500)
#>   |                                                                      |
head(pvals.2)
#> [1] 0.002 0.456 0.956 0.842 0.658 0.374
```

Note: the p-values might be different from the previous result. This is due to the way log-normalized counts are computed. See also `scuttle::logNormCounts`.

Session Information

```
sessionInfo()
#> R version 4.1.1 (2021-08-10)
#> Platform: x86_64-apple-darwin17.0 (64-bit)
#> Running under: macOS Big Sur 10.16
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#> [1] doRNG_1.8.2    rngtools_1.5.2 foreach_1.5.1  ggplot2_3.3.5  BSDE_0.1.0
#>
#> loaded via a namespace (and not attached):
#>  [1] viridis_0.6.1               MatrixGenerics_1.4.3
#>  [3] Biobase_2.52.0              BiocSingular_1.8.1
#>  [5] viridisLite_0.4.0           jsonlite_1.7.2
#>  [7] DelayedMatrixStats_1.14.3   scuttle_1.2.1
#>  [9] highr_0.9                   stats4_4.1.1
#> [11] vipor_0.4.5                 GenomeInfoDbData_1.2.6
#> [13] yaml_2.2.1                  pillar_1.6.3
#> [15] lattice_0.20-45             glue_1.4.2
#> [17] beachmat_2.8.1              reticulate_1.22
#> [19] digest_0.6.28               GenomicRanges_1.44.0
#> [21] XVector_0.32.0              colorspace_2.0-2
#> [23] plyr_1.8.6                  htmltools_0.5.2
#> [25] Matrix_1.3-4                pkgconfig_2.0.3
#> [27] zlibbioc_1.38.0             purrr_0.3.4
#> [29] scales_1.1.1                ScaledMatrix_1.0.0
#> [31] BiocParallel_1.26.2         tibble_3.1.5
#> [33] generics_0.1.0              farver_2.1.0
#> [35] IRanges_2.26.0              ellipsis_0.3.2
#> [37] withr_2.4.2                 SummarizedExperiment_1.22.0
#> [39] BiocGenerics_0.38.0         magrittr_2.0.1
#> [41] crayon_1.4.1                evaluate_0.14
#> [43] fansi_0.5.0                 doParallel_1.0.16
#> [45] beeswarm_0.4.0              tools_4.1.1
#> [47] scater_1.20.1               lifecycle_1.0.1
#> [49] matrixStats_0.61.0          stringr_1.4.0
#> [51] S4Vectors_0.30.2            munsell_0.5.0
#> [53] DelayedArray_0.18.0         irlba_2.3.3
#> [55] compiler_4.1.1              GenomeInfoDb_1.28.4
#> [57] rsvd_1.0.5                  rlang_0.4.11
#> [59] grid_4.1.1                  RCurl_1.98-1.5
#> [61] iterators_1.0.13            BiocNeighbors_1.10.0
#> [63] SingleCellExperiment_1.14.1 bitops_1.0-7
#> [65] labeling_0.4.2              rmarkdown_2.11
#> [67] gtable_0.3.0                codetools_0.2-18
#> [69] R6_2.5.1                    gridExtra_2.3
#> [71] knitr_1.36                  dplyr_1.0.7
#> [73] fastmap_1.1.0               utf8_1.2.2
#> [75] ggbeeswarm_0.6.0            stringi_1.7.5
#> [77] parallel_4.1.1              Rcpp_1.0.7
#> [79] vctrs_0.3.8                 png_0.1-7
#> [81] tidyselect_1.1.1            xfun_0.26
#> [83] sparseMatrixStats_1.4.2
```