The SDAMS package

Yuntong Li[1], Chi Wang[2,3*], Li Chen[2,3†]

[1]Department of Statistics , University of Kentucky,Lexington, KY;

[2]Markey Cancer Center, University of Kentucky, Lexington, KY ;

[3]Department of Biostatistics, University of Kentucky, Lexington, KY;

yuntong.li@uky.edu

chi.wang@uky.edu

lichenuky@uky.edu

April 30, 2018

Abstract

This vignette introduces the use of the Bioconductor package SDAMS, which is designed for differential abundance analysis for metabolomics and proteomics data from mass spectrometry. These data may contain a large fraction of zero values and non-zero part may not be normally distributed. SDAMS considers a two-part semi-parametric model, a logistic regression for the zero proportion and a semi-parametric log-linear model for the non-zero values. A kernel-smoothed likelihood method is proposed to estimate regression coefficients in the two-part model and a likelihood ratio test is constructed for differential abundant analysis.

*to whom correspondence should be addressed

†to whom correspondence should be addressed

Contents

1 Citation **3**

2 Quick Start **3**

3 Data Input **3**

3.1 Create SummarizedExperiment object from csv files 4

3.2 Create SummarizedExperiment object from seperate matrix 6

4 Data Analysis **7**

5 Theory for SDAMS **9**

5.1 A two-part semi-parametric model 9

5.2 Identification of differentially abundant features 10

6 Session Info **10**

1 Citation

The package SDAMS implements statistical methods from the following publication. If you use SDAMS in the published research, please cite:
Yuntong Li, Teresa W.M. Fan, Andrew N. Lane, Woo-Young Kang, Susanne M. Arnold, Arnold J. Stromberg, Chi Wang and Li Chen: A Two-Part Semi-Parametric Model for Metabolomics and Proteomics Data. (Manuscript)

2 Quick Start

This section show the most basic SDAMS work flow for a differential abundance analysis for metabolomics and proteomics data from mass spectrometry:

1. Create a SummarizedExperiment object using function createSEFromMatrix or createSEFromCSV. In this section we use an example SummarizedExperiment object directly, which is an object of SummarizedExperiment class named exampleSumExp contained in this package.
2. Perform a differential abundance analysis using SDA.

```
> library("SDAMS")
> data("exampleSumExp")
> results <- SDA(exampleSumExp)
```

Here, the SummarizedExperiment class object exampleSumExp contained in the package is the proteomics dataset, which a matrix-like container for proteomic features with experimental subject grouping information. There are 560 features for 202 experimental subjects with 49 prostate cancer subjects and 153 healthy subjects (0 for healthy control and 1 for patient in this case). This is a 10% subsample of the original dataset. The features are stored as a matrix in the assay slot. Each row in this matrix represents a proteomic feature and each column represents a subject. See Reference [1] for detailed information regarding this dataset.

3 Data Input

3.1 Create SummarizedExperiment object from csv files

The proteomics or metabolomics data is stored as a matrix with each row being a feature and each column corresponding to a subject. All data in this matrix are non-negative. Another information required is the phenotype covariates. Here we focus on the binary grouping information, such as numeric 1 for control group and 0 for case group. But it can also be characters, such as "healthy" and "disease". To utilize `SDAMS` package, we should have two separate csv files (for example 'feature.csv' and 'group.csv') as inputs for `createSEfromCSV` to creat a `SummarizedExperiment` object.

Note:

1. The 1^{st} column in 'feature.csv' represents feature names and the 1^{st} row represents subject codes.
2. The 1^{st} column in 'group.csv' represents subject codes, for example, Subject1, Subject2....

The format for "csv files" should look like as Figure 1 and Figure 2:

	9308	9324	9447	9449	9457	9459	9464	9466	9467	9471
54	0	0	116.9700012	0	0	0	0	19.01000023	0	0
94	52.40000153	0	0	139.1199951	82.37000275	0	58.22999954	0	22.32999992	25.09000015
97	0	0	53.45000076	62.72999954	47.15999985	0	0	24.44000053	0	22.77000046
191	0	0	22.18000031	170.6499939	0	0	0	0	0	26.52000046
219	58.47000122	0	0	64.5	28.11000061	0	39.77999878	7.429999828	0	5.230000019
286	0	0	0	19.26000023	89.55999756	0	162.8200073	109.8600006	75.48999786	115.6999969
366	0	0	0	0	28.97999954	0	0	0	0	0
388	29.43000031	0	0	0	0	0	0	32.20000076	0	59.97000122
420	65.79000092	0	87.44000244	314.0100098	160.4400024	0	0	0	0	26.85000038
463	0	0	0	12.15999985	38.52999878	0	0	0	0	0
499	0	0	0	26.12999916	14.22000027	0	0	0	0	1.919999957
583	43.74000168	0	0	59.29000092	7.099999905	0	0	0	0	3.809999943
605	0	0	0	657.8400269	661.9500122	0	0	32.22999954	0	0
624	0	96.94999695	0	97.40000153	167.0599976	232.9499969	89.54000092	55.5	35.15999985	0
652	0	0	9.68999958	0	0	0	0	0	0	3.599999905
709	0	0	38.45000076	0	0	0	0	8.229999542	0	6.380000114
889	0	0	0	0	0	0	0	0	0	0
912	569.3200073	0	0	0	0	0	23.57999992	120.6800003	73.69000244	100.4100037

Figure 1: Example of 'feature.csv' pattern

After creating the two csv files, we need the paths for the two csv files:

```
> path1 <- "/path/to/your/feature.csv/"
> path2 <- "/path/to/your/group.csv/"
```

Here for demonstration purpose, we use the data stored in `inst/extdata` directory. This is the csv format of the data in exampleSumExp which is a `SummarizedExperiment` object we described before.

	grouping
9308	0
9324	0
9447	0
9449	0
9457	0
9459	0
9464	0
9466	0
9467	0
9471	0
9495	0
9497	0
9507	0
9512	0
9959	1
9960	1

Figure 2: Example of 'group.csv' pattern

```
> directory1 <- system.file("extdata", package = "SDAMS", mustWork = TRUE)
> path1 <- file.path(directory1, "ProstateFeature.csv")
> directory2 <- system.file("extdata", package = "SDAMS", mustWork = TRUE)
> path2 <- file.path(directory2, "ProstateGroup.csv")
```

then use the function `createSEFromCSV` after loading the `SDAMS` package.

```
> library("SDAMS")
> exampleSE1 <- createSEFromCSV(path1, path2)
> exampleSE1
```

```
class: SummarizedExperiment
dim: 560 202
metadata(0):
assays(1): counts
rownames(560): 93922 87209 ... 180624 130855
rowData names(0):
colnames(202): 9512 9963 ... 49341 49586
colData names(1): grouping
```

The feature data and grouping information can be accessed using `SummarizedExperiment` commands:

```
> head(assay(exampleSE1)[,1:10])
```

```
        9512   9963  9965  9975   9979  9997 10015  10034  10044 10047
93922   0.00   0.00  0.00  0.00   0.00  0.00 68.97   0.00   0.00  0.00
87209   0.00   0.00  0.00  0.00   0.00  0.00  0.00   0.00   0.00 43.87
29633   0.00   0.00  0.00  0.00   0.00  0.00  0.00  57.21   0.00  0.00
40225   0.00 226.40  0.00 19.65   0.00  0.00  0.00   0.00   0.00  0.00
126342  0.00   0.00  0.00  0.00   0.00 20.43  0.00   0.00 109.93  0.00
42832  52.32 137.76 70.25  0.00 453.23 92.20  0.00 352.55 496.71  0.00
```

```
> head(colData(exampleSE1)$grouping)
```

```
[1] 0 1 1 1 1 1
```

3.2 Create SummarizedExperiment object from seperate matrix

If the two datasets have been already claeaned and loaded into R as matrices, then we can use `createSEFromMatrix` to create a `SummarizedExperiment` object.

```
> set.seed(100)
> featureInfo <- matrix(runif(800, -2, 5), ncol = 40)
> featureInfo[featureInfo<0] <- 0
> rownames(featureInfo) <- paste("feature", 1:20, sep = '')
> colnames(featureInfo) <- paste('subject', 1:40, sep = '')
> groupInfo <- data.frame(grouping=matrix(sample(0:1, 40, replace = TRUE),
+                         ncol = 1))
> rownames(groupInfo) <- colnames(featureInfo)
> exampleSE2 <- createSEFromMatrix(feature = featureInfo, group = groupInfo)
> exampleSE2
```

```
class: SummarizedExperiment
dim: 20 40
metadata(0):
assays(1): counts
rownames(20): feature1 feature2 ... feature19 feature20
rowData names(0):
colnames(40): subject1 subject2 ... subject39 subject40
colData names(1): grouping
```

```
> head(assay(exampleSE2)[,1:10])
```

```
          subject1  subject2  subject3  subject4 subject5  subject6  subject7
feature1 0.1543628 1.7506781 0.3146237 1.2459082 1.216678 0.2919056 0.7766364
feature2 0.0000000 2.9756269 4.0558438 2.5297084 2.195787 0.7263509 0.7489158
feature3 1.8662570 1.7684409 3.4430911 4.7240116 4.438053 0.0000000 1.3078982
feature4 0.0000000 3.2428056 3.7911241 2.7347872 4.879769 0.5297764 2.0855984
feature5 1.2798450 0.9407102 2.2232705 1.1160362 0.000000 1.9968466 0.4667102
feature6 1.3863951 0.0000000 1.4386228 0.5044165 2.045562 2.7941617 0.0000000
         subject8 subject9  subject10
feature1 2.712745 3.705652 0.02105356
feature2 0.000000 1.978800 3.02481591
feature3 0.000000 1.360440 4.46378210
feature4 4.359349 0.000000 2.72457641
feature5 0.000000 0.000000 0.00000000
feature6 3.102040 0.000000 0.43647014
```

```
> head(colData(exampleSE2)$grouping)
```

```
[1] 0 0 0 1 1 0
```

```
>
```

4 Data Analysis

Finally, we perform differential abundance analyais using `SummarizedExperiment` object created in the last section. This can be done by using function `SDA`. The theory behind `SDA` can be reached at section 5. A list with point estimates, p-values, q-values and corresponding feature names is returned. Below is the results generated by using the `SummarizedExperiment` object exampleSE1.

```
> results <- SDA(exampleSE1)
> head(results$gamma)
```

```
[1]  0.1100009  0.8629447 -0.7151261  0.2876821 -0.1251631  0.6292037
```

```
> head(results$pv_gamma)
```

```
[1] 0.8290701 0.1703578 0.2356975 0.5571851 0.7227301 0.1248023
```

```
> head(results$qv_gamma)

[1] 0.4092097 0.1914385 0.2191259 0.3505466 0.3887992 0.1651784
```

In this example , there is only one group covariate applied to each subject. Here $\mathbf{X}_i$ is one dimension. The covariate effect on the fraction of zero values for feature g is γ_g, which is estimated to be 0.11 for the first feature, and 0.86 for the second feature, etc. The corresponding hypothesis is H_0: $\gamma_g = 0$ vs. H_1: $\gamma_g \neq 0$. The p-values calculated from likelihood ratio test are returned in `pv_gamma`. Users can determine their own significance level to make inference, such as 0.05 nominal level. We also provide a FDR adjustment method [2] used in `SDA` for multiple comparison issues. Those results for γ_g are stored in `qv_gamma`.

```
> head(results$beta)

[1] -0.04912170 -1.11354659 -1.30566809  0.02484749  0.53967121 -0.22075205

> head(results$pv_beta)

[1] 0.88988924 0.11334266 0.06081223 0.95500620 0.06770261 0.13832035

> head(results$qv_beta)

[1] 0.7466379 0.2979437 0.2525717 0.7599717 0.2673415 0.3235222
```

The model parameter β_g is the log fold change in the non-zero abundance comparing different values of the single group covariate for feature g. The corresponding two-sided hypothesis is H_0: $\beta_g = 0$ vs. H_1: $\beta_g \neq 0$. Again, `SDA` will return p-values and adjusted p-values(q-values) for parameter β_g, and they are stored in `pv_beta` and `qv_beta` respectively.

```
> head(results$pv_2part)

           [,1]
[1,] 0.96764635
[2,] 0.11153821
[3,] 0.08538208
[4,] 0.84038731
[5,] 0.17695215
[6,] 0.10266642

> head(results$qv_2part)

          [,1]
```

```
[1,] 0.4622205
[2,] 0.1380966
[3,] 0.1192246
[4,] 0.4323376
[5,] 0.1806273
[6,] 0.1345612
```

Hypothesis testing on overall effect of group covariate on the gth feature is performed by assessing γ_g and β_g. The null hypothesis $H_0 : \gamma_g = 0$ and $\beta_g = 0$ against alternative hypothesis H_1 : at least one of the two parameters is non-zero. The p-values are calculated based on chi-square distribution with 2 degrees of freedom. And the corresponding q-values are calculated using the same procedure as in one-part model.

```
> head(results$feat.names)
```

```
[1] "93922"  "29633"  "40225"  "126342" "42832"  "127351"
```

A vector of feature names is returned for convenience which corresponds to the results in the other components.

5 Theory for SDAMS

As mentioned in the abstract, MS data is a mixture of zero intensity values and possibly non-normally distributed non-zero intensity values. Therefore, the differential abundance analysis needs to be performed to compare both the zero proportion and the mean of non-zero values between groups. SDA is a two-part model which addresses these issues that uses a logistic regression model to characterize the zero proportion and a semiparametric model to characterize non-zero values.

5.1 A two-part semi-parametric model

The differential abundance analysis in SDAMS has the following forms. For binary part:

$$\log(\frac{\pi_{ig}}{1-\pi_{ig}}) = \gamma_{0g} + \boldsymbol{\gamma}_g \boldsymbol{X}_i,$$

For continuous non-zero part:

$$\log(Y_{ig}) = \boldsymbol{\beta}_g \boldsymbol{X}_i + \varepsilon_{ig},$$

where Y_{ig} is the random variable representing the abundance of feature g in subject i, $\pi_{ig} = Pr(Y_{ig} = 0)$ is the probability of point mass. $\boldsymbol{X}_i = (X_{i1}, X_{i2}, ..., X_{iQ})^T$ is a Q-vector covariates that specifies the treatment conditions applied to subject i. The corresponding Q-vector of model parameters $\boldsymbol{\gamma}_g = (\gamma_{1g}, \gamma_{2g}, ..., \gamma_{Qg})^T$ quantify the covariates effects on the fraction of zero values for feature g and γ_{0g} is the intercept. $\boldsymbol{\beta}_g = (\beta_{1g}, \beta_{2g}, ..., \beta_{Qg})$ is a Q-vector of model parameters quantifying the covariates effects on the non-zero values for the feature, and ε_{ig}'s $(i = 1, 2, ..n)$ are independent error terms with a common but completely unspecified density function f_g. Importantly, we do not impose any distributional assumption on f_g. Without assuming a specific parametric distribution for ε_{ig}, this model is much more flexible to characterize data with unknown and possibly non-normal distribution.

5.2 Identification of differentially abundant features

We replace f_g by its kernel density estimator in the likelihood function. The maximum likelihood estimator is obtained through a trust region maximization algorithm. The likelihood ratio test is performed on the null hypothesis $H_0 : \gamma_{qg} = 0$ and $\beta_{qg} = 0$ against alternative hypothesis H_1 : at least one of the two parameters is non-zero. We also consider the hypotheses for testing $\gamma_{qg} = 0$ and $\beta_{qg} = 0$ separately. To adjust for multiple comparisons across features, the false discovery discovery rate (FDR) q-value is calculated based on the `qvalue` function in `qvalue` package in R/Bioconductor.

6 Session Info

```
> toLatex(sessionInfo())
```

- R version 3.5.0 (2018-04-23), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 16.04.4 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.7-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.7-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils

- Other packages: Biobase 2.40.0, BiocGenerics 0.26.0, BiocParallel 1.14.0, DelayedArray 0.6.0, GenomeInfoDb 1.16.0, GenomicRanges 1.32.0, IRanges 2.14.0, S4Vectors 0.18.0, SDAMS 1.0.0, SummarizedExperiment 1.10.0, matrixStats 0.53.1
- Loaded via a namespace (and not attached): GenomeInfoDbData 1.1.0, Matrix 1.2-14, RCurl 1.95-4.10, Rcpp 0.12.16, XVector 0.20.0, bitops 1.0-6, colorspace 1.3-2, compiler 3.5.0, ggplot2 2.2.1, grid 3.5.0, gtable 0.2.0, lattice 0.20-35, lazyeval 0.2.1, magrittr 1.5, munsell 0.4.3, pillar 1.2.2, plyr 1.8.4, qvalue 2.12.0, reshape2 1.4.3, rlang 0.2.0, scales 0.5.0, splines 3.5.0, stringi 1.1.7, stringr 1.3.0, tibble 1.4.2, tools 3.5.0, trust 0.1-7, zlibbioc 1.26.0

References

[1] Justyna Siwy, William Mullen, Igor Golovko, Julia Franke, and Petra Zürbig. Human urinary peptide database for multiple disease biomarker discovery. *PROTEOMICS-Clinical Applications*, 5(5-6):367–374, 2011.

[2] John D Storey and Robert Tibshirani. Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences*, 100(16):9440–9445, 2003.