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

March 3, 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 mdoel, 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 3

3.2 Create SummarizedExperiment object from R global environment 6

4 Data Analysis 7

5 Session Info 8

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:

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

```
> library("SDAMS")
> data("exampleSumExpset")
> results=SDA(exampleSumExpset)
```

Here, the SummarizedExperiment class exampleSumExpset contained in the package is the proteomics dataset, which a matrix-like container for proteomics features with experimental subject grouping information. There are 560 features for 202 experimental subjects (0 for control and 1 for patient). This is a 10% subsample of the original dataset. Each row in this matrix repressents a proteomics feature. 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 stroed 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 utlize `SDAMS` package, we should have two separate csv.files (for example 'feature.csv' and 'group.csv') as inputs for `createSEsetfromCSV` to creat a `SummarizedExperiment` object.

Note:

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

The format for "csv.files" should looks 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 purposes, we use the data in the `SDAMS` package

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

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

	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

```
> library("SDAMS")
> exampleSEset1 = createSEsetFromCSV(path1,path2)
> exampleSEset1
```

```
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` command:

```
> head(assay(exampleSEset1)[,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(exampleSEset1)$grouping)
```

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

3.2 Create SummarizedExperiment object from R global environment

If the two datasets have been already claeaned and loaded into the R global environment, we can use `createSEsetFromEnvir` 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)
> exampleSEset2 = createSEsetFromEnvir(feature = featureInfo,group = groupInfo)
> exampleSEset2
```

```
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(exampleSEset2)[,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(exampleSEset2)$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`. And a list with point estimates, p-values, q-values and corresponding feature names is returned. Below is results generated by using the `SummarizedExperiment` object exampleSEset1.

```
> results = SDA(exampleSEset1)
> head(results$gamma)
```

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

```
> head(results$beta)
```

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

```
> head(results$qv_gamma)
```

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

```
> head(results$qv_beta)
```

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

```
> head(results$qv_2part)
```

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

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

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

5 Session Info

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

- R version 3.4.3 (2017-11-30), `x86_64-apple-darwin15.6.0`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Running under: `macOS High Sierra 10.13.1`
- Matrix products: default
- BLAS: `/Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib`
- LAPACK: `/Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, stats4, utils
- Other packages: Biobase 2.36.2, BiocGenerics 0.22.1, DelayedArray 0.2.7, GenomeInfoDb 1.12.3, GenomicRanges 1.28.6, IRanges 2.10.5, matrixStats 0.53.1, S4Vectors 0.14.7, SDAMS 0.99.4, SummarizedExperiment 1.6.5
- Loaded via a namespace (and not attached): bitops 1.0-6, colorspace 1.3-2, compiler 3.4.3, GenomeInfoDbData 0.99.0, ggplot2 2.2.1, grid 3.4.3, gtable 0.2.0, lattice 0.20-35, lazyeval 0.2.0, magrittr 1.5, Matrix 1.2-12, munsell 0.4.3, plyr 1.8.4, qvalue 2.8.0, Rcpp 0.12.11, RCurl 1.95-4.10, reshape2 1.4.2, rlang 0.1.1, scales 0.4.1, splines 3.4.3, stringi 1.1.5, stringr 1.2.0, tibble 1.3.3, tools 3.4.3, trust 0.1-7, XVector 0.16.0, zlibbioc 1.22.0

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.