A fast negative binomial mixed model for analyzing multi-subject single-cell data

Liang He

2020-09-23

Contents

Overview

The R package, *nebula*, provides fast algorithms for fitting negative binomial and Poisson mixed models for analyzing large-scale multi-subject single-cell data. The package *nebula* accounts for the hierarchical structure of the data by decomposing the total overdispersion into between-subject and within-subject components using a negative binomial mixed model (NBMM). The package nebula can be used for e.g., identifying marker genes, testing treatment effects, detecting genes with differentail expression, and performing cell-level co-expression analysis.

Installation

Most recent version

To install the lastest version from github:

```
install.packages("devtools")
library(devtools)
install_github("lhe17/nebula")
```

Because the package *nebula* uses the R package *Rfast*, the installation process may first install *Rfast*, which requires that GSL is installed or available in the environment.

Please contact liang.he@duke.edu for more information.

Functions

The current version provides the following functions.

- `nebula`: performs association analysis given a count matrix and subject IDs.
- `group_cell`: reorders cells to group them by the subject IDs.

Basic usage

We use an example data set to illustrate how to use nebula to perform an association analysis of multi-subject single-cell data. The example data set attached to the R package can be loaded as follows.

```
library(nebula)
data(sample_data)
```

The example data set includes a count matrix of 6030 cells from 30 subjects for 10 genes.

```
dim(sample_data$count)
#> [1]   10 6176
```

The count matrix can be a matrix object or a sparse dgCMatrix object. The elements should be integers.

```
sample_data$count[1:5,1:5]
#> 5 x 5 sparse Matrix of class "dgCMatrix"
#>
#> A . . . . .
#> B . . . . .
#> C . 1 2 . .
#> D . . . . .
#> E . . . . .
```

The subject IDs of each cell are stored in `sample_data$sid`. The subject IDs can be a character or numeric vector, the length of which should equal the number of cells.

```
head(sample_data$sid)
#> [1] "1" "1" "1" "1" "1" "1"
table(sample_data$sid)
#>
#>   1  10  11  12  13  14  15  16  17  18  19   2  20  21  22  23  24  25  26  27
#> 187 230 185 197 163 216 211 195 200 239 196 223 198 202 213 210 199 214 237 200
#>  28  29   3  30   4   5   6   7   8   9
#> 205 183 222 191 205 225 211 197 215 207
```

The next step is to build a design matrix for the predictors. The example data set includes a data frame consisting of three predictors stored in `sample_data$pred`. To build the design matrix, we can use the function `model.matrix`.

```
head(sample_data$pred)
#>            X1        X2      cc
#> 1  0.6155094 0.9759191 control
#> 2  1.4608092 0.9759191    case
#> 3  1.6675054 0.9759191 control
#> 4 -0.1717715 0.9759191    case
#> 5  0.2277492 0.9759191 control
#> 6 -0.2635516 0.9759191 control
df = model.matrix(~X1+X2+cc, data=sample_data$pred)
head(df)
#>   (Intercept)         X1        X2 cccontrol
#> 1           1  0.6155094 0.9759191         1
#> 2           1  1.4608092 0.9759191         0
#> 3           1  1.6675054 0.9759191         1
#> 4           1 -0.1717715 0.9759191         0
#> 5           1  0.2277492 0.9759191         1
#> 6           1 -0.2635516 0.9759191         1
```

The association analysis between the gene expression and the predictors can then be conducted using the function *nebula*. The count matrix is an M by N matrix, where M is the number of genes, and N is the number of cells.

```
re = nebula(sample_data$count,sample_data$sid,pred=df)
#> Remove  0  genes having low expression.
#> Analyzing  10  genes with  30  subjects and  6176  cells.
re
#> $summary
#>    logFC_(Intercept)     logFC_X1     logFC_X2 logFC_cccontrol se_(Intercept)
#> 1          -1.902455 -0.016755225 -0.097867225     0.047278197     0.06335820
#> 2          -2.046638 -0.002679074 -0.053812464    -0.022293899     0.06181112
#> 3          -2.033211  0.017954707  0.002398445    -0.048296661     0.08695028
#> 4          -2.008542 -0.005698984 -0.027780387     0.077357703     0.05509711
#> 5          -1.981546  0.011486801 -0.025051853     0.032882833     0.06280883
#> 6          -1.951561  0.013464590 -0.012557319    -0.031646274     0.07519685
#> 7          -1.969248 -0.003531361  0.075230699    -0.009075031     0.06185028
#> 8          -1.964371  0.013639930 -0.061302756    -0.059284665     0.07786371
#> 9          -2.072699 -0.017372176 -0.043828288     0.026624998     0.05737632
#> 10         -2.045646  0.030742876  0.022260806    -0.025516032     0.06842808
#>         se_X1      se_X2 se_cccontrol p_(Intercept)      p_X1      p_X2
#> 1  0.03494975 0.06413521   0.04864366 4.362881e-198 0.6354810 0.1291514
#> 2  0.03744907 0.06221023   0.05222322 2.052806e-240 0.9436079 0.3896818
#> 3  0.03654592 0.09186803   0.05132265 6.275233e-121 0.6271261 0.9792875
#> 4  0.03662965 0.05593511   0.05128078 5.822961e-291 0.8777381 0.6213846
#> 5  0.03709429 0.06189072   0.05184865 1.860076e-218 0.7594613 0.6872997
#> 6  0.03583152 0.07356394   0.05011851 1.694944e-148 0.7102234 0.8652074
#> 7  0.03590846 0.06034913   0.05044028 1.871848e-222 0.9225364 0.2151043
#> 8  0.03512025 0.07911592   0.04928082 1.959009e-140 0.7009654 0.4409832
#> 9  0.03773196 0.05735863   0.05269998 9.307378e-286 0.6489358 0.4473406
#> 10 0.03756047 0.06878990   0.05214337 2.296650e-196 0.4183421 0.7476010
#>    p_cccontrol gene_id gene
#> 1    0.4919443       1    A
#> 2    0.7627706       2    B
#> 3    0.5058082       3    C
#> 4    0.2861434       4    D
#> 5    0.6538444       5    E
#> 6    0.6552629       6    F
#> 7    0.8987718       7    G
#> 8    0.3949916       8    H
#> 9    0.7209245       9    I
#> 10   0.7293434      10    J
#>
#> $overdispersion
#>       Subject      Cell
#> 1  0.08125256 0.8840821
#> 2  0.07102681 0.9255032
#> 3  0.17159404 0.9266395
#> 4  0.05026165 0.8124118
#> 5  0.07507489 1.2674146
#> 6  0.12398378 1.1096065
#> 7  0.07360445 0.9112956
#> 8  0.13571262 0.7549629
#> 9  0.05541398 0.8139652
#> 10 0.09496649 0.9410035
#>
#> $convergence
#>  [1] 1 1 1 1 1 1 1 1 1 1
#>
#> $algorithm
#>  [1] "LN"    "LN"    "LN"    "LN"    "LN+HL" "LN+HL" "LN"    "LN"    "LN"
#> [10] "LN"
```

The function fitted the negative binomial gamma mixed model (NBGMM) for each of the genes, and return a list of summary statistics including the fold change, p-values, and both subject-level and cell-level overdispersions (σ^2 and ϕ^{-1}). The cells need to be grouped by the subjects before using as the input to the *nebula* function. If the cells are not grouped, the *group_cell* function can be used to first reorder the cells. If the cells are already grouped, the *group_cell* function will return NULL.

Example

```
data_g = group_cell(count=sample_data$count,id=sample_data$sid,pred=df)
re = nebula(data_g$count,data_g$id,pred=data_g$pred)
```

If `pred` is not specified, `nebula` will fit the model with an intecept term by default. This can be used when only the overdispersions are of interest.

Specifying scaling factors

The scaling factor for each cell is specified in `nebula` using the argument `offset`. The argument `offset` has to be a positive vector of length N. Note that log(offset) will be the offset in the NBMM. If not specified, `nebula` will set `offset` as 1 by default, which means that each cell is treated equally. Common scaling factors include the library size of a cell or a normalizing factor adjusted using e.g., TMM.

Example

```
re = nebula(sample_data$count,sample_data$sid,pred=df,offset=sample_data$offset)
```

Selection between NEBULA-LN and NEBULA-HL

In *nebula*, a user can choose one of the two algorithms to fit an NBGMM. NEBULA-LN uses an approximated likelihood based on the law of large numbers, and NEBULA-HL uses an h-likelihood. A user can select these methods through`method='LN'` or `method='HL'`. NEBULA-LN is faster and performs particularly well when the number of cells per subject is large. In the following analysis of the example data comprising ~200 cells per subject, the difference of the estimated cell-level overdispersions between NEBULA-LN and NEBULA-HL is ~5% for most genes.

```
re_ln = nebula(sample_data$count,sample_data$sid,pred=df,offset=sample_data$offset,method=’LN’)
#> Remove  0  genes having low expression.
#> Analyzing  10  genes with  30  subjects and  6176  cells.
re_hl = nebula(sample_data$count,sample_data$sid,pred=df,offset=sample_data$offset,method=’HL’)
#> Remove  0  genes having low expression.
#> Analyzing  10  genes with  30  subjects and  6176  cells.
## compare the estimated overdispersions
cbind(re_hl$overdispersion,re_ln$overdispersion)
#>       Subject      Cell    Subject      Cell
#> 1  0.08432304 0.9284704 0.08125256 0.8840821
#> 2  0.07455454 0.9726513 0.07102681 0.9255032
#> 3  0.17403265 0.9817571 0.17159404 0.9266395
#> 4  0.05352150 0.8516686 0.05026165 0.8124118
#> 5  0.07480036 1.3254378 0.07507489 1.2674146
#> 6  0.12372437 1.1653120 0.12398378 1.1096065
#> 7  0.07724763 0.9578175 0.07360445 0.9112956
#> 8  0.13797543 0.7991951 0.13571262 0.7549629
#> 9  0.05879494 0.8568852 0.05541398 0.8139652
#> 10 0.09782377 0.9940205 0.09496649 0.9410035
```

Such difference has little impact on testing fixed-effects predictors.

```
## compare the p-values for testing the predictors using NEBULA-LN and NEBULA-HL
cbind(re_hl$summary[,10:12],re_ln$summary[,10:12])
#>         p_X1      p_X2 p_cccontrol      p_X1      p_X2 p_cccontrol
#> 1  0.6373036 0.1346295   0.4950795 0.6354810 0.1291514   0.4919443
#> 2  0.9444825 0.3977107   0.7626827 0.9436079 0.3896818   0.7627706
#> 3  0.6282384 0.9787882   0.5087304 0.6271261 0.9792875   0.5058082
#> 4  0.8786074 0.6278827   0.2868256 0.8777381 0.6213846   0.2861434
#> 5  0.7596198 0.6872260   0.6544751 0.7594613 0.6872997   0.6538444
#> 6  0.7134192 0.8656693   0.6576835 0.7102234 0.8652074   0.6552629
#> 7  0.9216994 0.2230951   0.8977251 0.9225364 0.2151043   0.8987718
#> 8  0.7017085 0.4443591   0.3955344 0.7009654 0.4409832   0.3949916
#> 9  0.6505414 0.4561469   0.7238323 0.6489358 0.4473406   0.7209245
#> 10 0.4199830 0.7510846   0.7308111 0.4183421 0.7476010   0.7293434
```

The bias of NEBULA-LN in estimating the cell-level overdispersion gets larger when the cell count per subject becomes lower or the gene expression is more sparse. In contrast, NEBULA-HL is slower, but its accuracy of estimating the overdispersions depends less on these factors. NEBULA-HL will underestimate the subject-level overdispersion if the gene expression is very low. Filtering out low-expressed genes (e.g., counts per cell<0.5%) can be specified by `cpc=0.005`.

When NEBULA-LN is used, the user can opt for better accuracy of estimating a smaller subject-level overdispersion through the argument κ. NEBULA first fits the data using NEBULA-LN. If the estimated κ for a gene is smaller than the user-defined value, NEBULA-HL will be used to estimate the subject-level overdispersion for the gene. The default value of κ is 200, which can provide a good estimate of the subject-level overdispersion as low as ~0.02. This value is sufficent for well controlled false positive rate of testing a cell-level predictor. We do not recommend using a smaller κ than the default value. Specifying a larger κ can obtain a more accurate estimate of a smaller subject-level overdispersion when the cell-level overdispersion is large, but will be computationally slower. On the other hand, testing a subject-level predictor (i.e., a variable whose values are shared across all cells from a subject, such as age, sex, treatment, genotype, etc) is more sensitive to the accuracy of the estimated subject-level overdispersion. So we recommend using $\kappa = 1000$ or even larger when testing a subject-level predictor. Another option to testing a subject-level predictor is to use a Poisson gamma mixed model, which is extremely fast (>50x faster than NEBULA-LN) and decribed below.

Using other mixed models

In addition to the NBGMM, the *nebula* package provides efficient estimation implementation for a Poisson gamma mixed model and a negative binomial lognormal mixed model (NGLMM). This can be specified through `model="PMM"` and `model="NBLMM"`, respectively. The NBLMM is the same model as that adopted in the `glmer.nb` function in the *lme4* R package. The only difference between NBGMM and NBLMM is that NBGMM uses a gamma distribution for the random effects while the NBLMM uses a lognormal distribution. When choosing the NBLMM, `method='HL'` is set automatically. The PMM is the fastest among these models. Note that the Poisson mixed model (PMM) should not be used to test a cell-level predictor because it only estimates the subject-level overdispersion. Here is an example of using the PMM to fit the example data set.

Example

```
re = nebula(sample_data$count,sample_data$sid,pred=df,offset=sample_data$offset,model='PMM')
#> Remove  0  genes having low expression.
#> Analyzing  10  genes with  30  subjects and  6176  cells.
```

logFC_(Intercept)	logFC_X1	logFC_X2	logFC_cccontrol	se_(Intercept)	se_X1	se_X2	se_cccontrol
-1.903571	-0.0155809	-0.0976660	0.0511060	0.0661297	0.0325420	0.0651904	0.0454149
-2.047864	-0.0032670	-0.0536887	-0.0189269	0.0644332	0.0351088	0.0631912	0.0491308
-2.032645	0.0179777	0.0009387	-0.0505390	0.0908196	0.0341618	0.0927258	0.0478477
-2.009746	-0.0054963	-0.0278602	0.0782074	0.0573209	0.0346807	0.0571261	0.0485713
-1.980528	0.0106338	-0.0248791	0.0312190	0.0644287	0.0339500	0.0618116	0.0474898
-1.950451	0.0160341	-0.0134775	-0.0345244	0.0778198	0.0330110	0.0734396	0.0459884
-1.970271	-0.0026753	0.0750060	-0.0063677	0.0645989	0.0338097	0.0611736	0.0472832
-1.964311	0.0141532	-0.0610984	-0.0578672	0.0809943	0.0332801	0.0796531	0.0464402
-2.074031	-0.0178190	-0.0436094	0.0259745	0.0597947	0.0358136	0.0584407	0.0500542
-2.046055	0.0307026	0.0227238	-0.0246112	0.0714158	0.0350860	0.0698359	0.0489158