The NanoStringDiff package

Hong Wang[1], Chi Wang[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;

hong.wang@uky.edu

November 24, 2026

Abstract

This vignette introduces the use of the Bioconductor package NanoStringDiff, which is designed for differential analysis based on NanoString nCouner Data. NanoStringDiff considers a generalized linear model of the negative binomial family to characterize count data and allows for multi-factor design. Data normalization is incorporated in the model framework by including data normalization parameters estimated from positive controls, negative controls and housekeeping genes embedded in the nCounter system. Present method propose an empirical Bayes shrinkage approach to estimate the dispersion parameter and a likelihood ratio test to identify differential expression genes.

*to whom correspondence should be addressed

Contents

1 Citation 3

2 Quick Start 3

3 Data Input 4

3.1 Create NanoStringSet from csv.file 4

3.2 Create NanoStringSet from matrix 6

4 Differential Expression Analysis for Single Factor Experiment 8

4.1 Two Group Comparisons 8

4.2 Pairwise Comparisons 10

4.3 Multigroup Comparisons 14

5 Differential Expression Analysis for Multifactor Experiment 14

5.1 Nested Interactions 14

5.2 All Interactions 15

6 Session Info 16

1 Citation

The package `NanoStringDiff` implements statistical methods from the following publication. If you use `NanoStringDiff` in the published research, please cite:
Hong Wang, Craig Horbinski, Hao Wu, Yinxing Liu,Arnold J. Stromberg and Chi Wang: A Negative Binomial Model-Based Method for Differential Expression Analysis Based on NanoString nCounter Data.(Manuscript)

2 Quick Start

This section show the most basic steps for a differential expression analysis for NanoString nCounter Data:

1. Create a `NanoStringSet` object using `createNanoStingSet` or `createNanoStringSetFromCsv` (see examples in the Data Input section). In this section we use `NanoStringData` directly, which is an object of `NanoStringSet` contained in the package.
2. Make a design matrix to describe treatment conditions.
3. Estimate norlamization factors including positive size factors, housekeeping size factors and background noise using `estNormalizationFactors`
4. Perform a likelihood ratio test using `glm.LRT`.

```
> library("Biobase")
> library("NanoStringDiff")
> data(NanoStringData)
> pheno=pData(NanoStringData)
> design.full=model.matrix(~0+pheno$group)
> NanoStringData=estNormalizationFactors(NanoStringData)
> result=glm.LRT(NanoStringData,design.full,contrast=c(1,-1))
```

Here, the data NanoStringData contained in the package is an animal data, we called `MoriData`[1]. Mori et al tried to study the possible reasons responsible for the widespread miRNA repression observed in cancer, global microRNA expression in mouse liver normal tissues and liver tumors induced by deletion of Nf2 (merlin) profiled by nCounter Mouse miRNA Expression Assays. Expressions of 599 miRNAs were profiled for two replicates in each group.

3 Data Input

NanoStringDiff works on matrix of integer read counts from NanoString nCounter analyzer with endogenes, positive control genes, negative control genes and housekeeping control genes. For the matrix, rows corresponding to genes and columns to independent samples or replicates. The counts represent the total number of reads aligning to each gene (or other genomic locus).

The count values must be raw counts of NanoString nCounter data, since data normalization is incorporated in the model framework by including data normalization parameters estimated from positive controls, negative controls and housekeeping genes using function `estNormalizationFactors`. Hence, please do not supply normalized counts.

There must be have six positive control genes order by different concentrations from high to low, since NanoString nCounter analyzer provide six different samll positive controls with six different concentrations in the 30 uL hybridization: 128 fM, 32 fM, 8 fM, 2 fM, 0.5 fM, and 0.125 fM. No such restriction for negative control genes and housekeeping control genes.Nanostring recommends at least three housekeeping genes, but the more that are included, the more accurate the normalization will be.

3.1 Create NanoStringSet from csv.file

The data produced by the nCounter Digital Analyzer are exported as a Reporter Code Count (RCC) file. RCC files are comma-separated text(.csv) files that contain the counts for each gene in a sample and the data for each sample hybridization are contained in a separate RCC file. So before you call `createNanoStringSetFromCsv` to creat a `NanoStringSet` object, you should create a csv.file to combine all interesting samples together, and the first three columns shoud be "CodeClass", "Name" ane "Accession", the counts data contained from the 4th column to the last column.

Note:

1. The 1st column "CodeClass" should specify the function of genes as "Positive", "Negative","Housekeeping" or "Endogenous".
2. Some data set have "Spikein" genes, you need delete these "spikein" genes or you could just leave there if you use `createNanoStringSetFromCsv` to creat `NanoStringSet` object(See example in the Data Input section). But NanoStringDiff only works with "positive", "negative", housekeeping" and "endogenous".

The "csv.file" should looks like as following Figure:

When you created a csv.file, you will specify a variable which points on the directory in which your

Code Class	Name	Accession	Normal 1	Normal 2	Tumor 1	Tumor 2
Positive	POS_A(128)	nmiR00813.1	4869	4416	10069	8018
Positive	POS_B(32)	nmiR00809.1	1427	1171	2809	2289
Positive	POS_C(8)	nmiR00811.1	277	213	400	318
Positive	POS_D(2)	nmiR00822.1	111	91	262	191
Positive	POS_E(0.5)	nmiR00801.1	33	25	56	33
Positive	POS_F(0.125)	nmiR00825.1	9	8	24	10
Negative	NEG_B(0)	nmiR00828.1	3	8	6	9
Negative	NEG_C(0)	nmiR00803.1	11	5	3	12
Negative	NEG_A(0)	nmiR00810.1	7	10	5	8
Negative	NEG_F(0)	nmiR00805.1	5	7	9	2
Negative	NEG_E(0)	nmiR00827.1	16	7	5	5
Negative	NEG_D(0)	nmiR00823.1	11	9	17	10
Housekeeping	B2m|0	NM_009735.3	874	698	3591	2377
Housekeeping	Rpl19|0	NM_009078.1	175	180	1099	1103
Housekeeping	Actb|0	NM_007393.1	356	332	1608	1651
Housekeeping	Gapdh|0	NM_008084.1	147	126	348	212
Endogenous1	mmu-miR-18a|0	MIMAT0000528	2	4	5	7
Endogenous1	mmu-miR-880|0	MIMAT0004844	3	6	15	10
Endogenous1	mmu-miR-686|0	MIMAT0003464	3	7	8	4
Endogenous1	mmu-miR-466f-5p|0	MIMAT0004881	8	5	4	5
Endogenous1	mmu-miR-106b|0	MIMAT0000386	30	26	310	106
Endogenous1	mmu-miR-1969|0	MIMAT0009442	3	4	5	1
Endogenous1	mmu-miR-193|0	MIMAT0000223	39	33	81	58

Figure 1: Example of csv.file pattern

csv.file is located

```
> directory <- "/path/to/your/files/"
```

However, for demonstration purposes only, the following line of code points to the directory for the "Mori.csv" in the NanoStringDiff package.

```
> directory <- system.file("extdata", package="NanoStringDiff", mustWork=TRUE)
> path<-paste(directory,"Mori.csv",sep="/")
```

The phenotype informations of the data should be stored as data frame.

```
> designs=data.frame(group=c("Normal","Normal","Tumor","Tumor"))
> designs
```

```
   group
1 Normal
2 Normal
3  Tumor
```

```
4  Tumor
```

```
> library("NanoStringDiff")
> NanoStringData=createNanoStringSetFromCsv(path,header=TRUE,designs)
```

```
 There are 4 samples imported;
 There are  618 genes imported with:code.class
 endogenous1  endogenous2 housekeeping     negative     positive      spikein
         566           33            4            6            6            3
```

```
> NanoStringData
```

```
NanoStringSet (storageMode: lockedEnvironment)
assayData: 599 features, 4 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: Normal.1 Normal.2 Tumor.1 Tumor.2
  varLabels: group
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

3.2 Create NanoStringSet from matrix

If you already read your positive control genes, negative control genes, housekeeping control genes and endogous into R session separately and stored as matrix, then you can use `createNanoStringSet` to create a `NanoStringSet` object.

```
> endogenous=matrix(rpois(100,50),25,4)
> colnames(endogenous)=paste("Sample",1:4)
> positive=matrix(rpois(24,c(128,32,8,2,0.5,0.125)*80),6,4)
> colnames(positive)=paste("Sample",1:4)
> negative=matrix(rpois(32,10),8,4)
> colnames(negative)=paste("Sample",1:4)
> housekeeping=matrix(rpois(12,100),3,4)
> colnames(housekeeping)=paste("Sample",1:4)
```

```
> designs=data.frame(group=c("Control","Control","Treatment","Treatment"),
+                    gender=c("Male","Female","Female","Male"),
+                    age=c(20,40,39,37))
> NanoStringData1=createNanoStringSet(endogenous,positive,
+                                     negative,housekeeping,designs)
> NanoStringData1

NanoStringSet (storageMode: lockedEnvironment)
assayData: 25 features, 4 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: Sample 1 Sample 2 Sample 3 Sample 4
  varLabels: group gender age
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:

> pData(NanoStringData1)

             group gender age
Sample 1   Control   Male  20
Sample 2   Control Female  40
Sample 3 Treatment Female  39
Sample 4 Treatment   Male  37

> head(exprs(NanoStringData1))

  Sample 1 Sample 2 Sample 3 Sample 4
1       56       50       43       68
2       50       54       44       56
3       48       51       70       69
4       33       46       52       45
5       52       45       55       45
6       41       48       45       60
```

4 Differential Expression Analysis for Single Factor Experiment

For general experiments, once normalization factors obtained using `estNormalizationFactors`, given a design matrix and contrast, we can proceed with testing procedures for determing differential expression using the generalized linear model (GLM) likelihood ratio test. The testing can be done by using the function `glm.LRT` and return a list with a component is table including: logFC ,lr, pvalue and qvalue(adjust p value using the procedure of Benjamini and Hochberg).

4.1 Two Group Comparisons

The simplest and most common type of experimental design is two group comparison,like treatment group vs control group. As a brief example, consider a simple situation with control group and treatment group, each with two replicates, and the researcher wants to make comparisons between them. Here, we still use `NanoSreingData1` we created above to demonstrate this example.

Make design matrix using `model.matrix`:

```
> pheno=pData(NanoStringData1)
> group=pheno$group
> design.full=model.matrix(~0+group)
> design.full
  groupControl groupTreatment
1            1              0
2            1              0
3            0              1
4            0              1
attr(,"assign")
[1] 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

Note that there is no intercept column in the dasign matrix, each column is for each group, since we include `0+` in the model formula.

If the researcher want compare Treatment to Control, that is Treatment- Control, the contrast vector

is try to the comparison -1*Control+1*Treatment, so the contrast is :

```
> contrast=c(-1,1)
```

Estimate normalization factors and return the same object with positiveFactor, negativeFactor and housekeepingFactor slots filled or replaced:

```
> NanoStringData1=estNormalizationFactors(NanoStringData1)
> positiveFactor(NanoStringData1)
```

```
 Sample 1  Sample 2  Sample 3  Sample 4
1.0094161 0.9983484 0.9950470 0.9971885
```

```
> negativeFactor(NanoStringData1)
```

```
Sample 1 Sample 2 Sample 3 Sample 4
      10        9        8       11
```

```
> housekeepingFactor(NanoStringData1)
```

```
 Sample 1  Sample 2  Sample 3  Sample 4
1.0018800 1.0117776 1.0106659 0.9865717
```

Generalize linear model likelihood ratio test:

```
> result=glm.LRT(NanoStringData1,design.full,contrast=contrast)
> head(result$table)
```

```
        logFC         lr     pvalue    qvalue
1  0.09890867 0.14015920 0.70812280 0.8046850
2 -0.05122735 0.03508642 0.85141469 0.8548503
3  0.60810570 5.77889349 0.01621975 0.2449138
4  0.40083683 1.78259975 0.18183042 0.7392735
5  0.08442086 0.09179463 0.76190793 0.8281608
6  0.31406951 1.23257384 0.26690707 0.7392735
```

```
> str(result)
```

```
List of 11
 $ table         :'data.frame':        25 obs. of  4 variables:
  ..$ logFC : num [1:25] 0.0989 -0.0512 0.6081 0.4008 0.0844 ...
  ..$ lr    : num [1:25] 0.1402 0.0351 5.7789 1.7826 0.0918 ...
```

```
  ..$ pvalue: num [1:25] 0.7081 0.8514 0.0162 0.1818 0.7619 ...
  ..$ qvalue: num [1:25] 0.805 0.855 0.245 0.739 0.828 ...
 $ dispersion    : num [1:25] 0.00623 0.00623 0.00632 0.00623 0.00623 ...
 $ log.dispersion: num [1:25] -5.08 -5.08 -5.06 -5.08 -5.08 ...
 $ design.full   : num [1:4, 1:2] 1 1 0 0 0 0 1 1
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:4] "1" "2" "3" "4"
  .. ..$ : chr [1:2] "groupControl" "groupTreatment"
  ..- attr(*, "assign")= int [1:2] 1 1
  ..- attr(*, "contrasts")=List of 1
  .. ..$ group: chr "contr.treatment"
 $ design.reduce : num [1:4, 1] 0.707 0.707 0.707 0.707
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:4] "1" "2" "3" "4"
  .. ..$ : NULL
 $ Beta.full     : num [1:25, 1:2] 3.76 3.74 3.68 3.39 3.65 ...
 $ mean.full     : num [1:25, 1:4] 43 42 39.6 29.8 38.5 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:4] "1" "2" "3" "4"
 $ Beta.reduce   : num [1:25] 3.8 3.72 3.91 3.54 3.68 ...
 $ mean.reduce   : num [1:25, 1:4] 44.5 41.3 49.9 34.5 39.7 ...
 $ m0            : num -9.49
 $ sigma         : num 1.75
```

Note that the text Treatment compare to Control tells you that the estimates of logFC is log2(Treatment/Control).

4.2 Pairwise Comparisons

Now consider a researcher has three treatment groups say A, B and C, and researcher wants to compare each groups like: B compare to A, C compare to A, and B compare to C.

First create an object of `NanoStringSet` with pseudo data:

```
> endogenous=matrix(rpois(300,50),25,12)
> colnames(endogenous)=paste("Sample", 1:12)
> colnames(endogenous)=paste("Sample",1:12)
```

```
> positive=matrix(rpois(72,c(128,32,8,2,0.5,0.125)*80),6,12)
> negative=matrix(rpois(96,10),8,12)
> housekeeping=matrix(rpois(36,100),3,12)
> designs=data.frame(group=c(rep("A",4),rep("B",4),rep("C",4)),
+                    gender=rep(c("Male","Male","Female","Female"),3),
+                    age=c(20,40,39,37,29,47,23,45,34,65,35,64))
> NanoStringData2=createNanoStringSet(endogenous,positive,
+                                     negative,housekeeping,designs)
> NanoStringData2

NanoStringSet (storageMode: lockedEnvironment)
assayData: 25 features, 12 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: Sample 1 Sample 2 ... Sample 12 (12 total)
  varLabels: group gender age
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:

> pData(NanoStringData2)

          group gender age
Sample 1      A   Male  20
Sample 2      A   Male  40
Sample 3      A Female  39
Sample 4      A Female  37
Sample 5      B   Male  29
Sample 6      B   Male  47
Sample 7      B Female  23
Sample 8      B Female  45
Sample 9      C   Male  34
Sample 10     C   Male  65
Sample 11     C Female  35
```

```
Sample 12     C Female  64
```

Make design matrix only consider one experimential factor group:

```
> pheno=pData(NanoStringData2)
> group=pheno$group
> design.full=model.matrix(~0+group)
> design.full
```

```
   groupA groupB groupC
1       1      0      0
2       1      0      0
3       1      0      0
4       1      0      0
5       0      1      0
6       0      1      0
7       0      1      0
8       0      1      0
9       0      0      1
10      0      0      1
11      0      0      1
12      0      0      1
attr(,"assign")
[1] 1 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

Estimate normalization factors:

```
> NanoStringData2=estNormalizationFactors(NanoStringData2)
```

B compare to A, the contrast should be (-1,1,0)

```
> result=glm.LRT(NanoStringData2,design.full,contrast=c(-1,1,0))
```

C compare to A, the contrast should be (-1,0,1)

```
> result=glm.LRT(NanoStringData2,design.full,contrast=c(-1,0,1))
```

B compare to c, the contrast should be (0,1,-1)

```
> result=glm.LRT(NanoStringData2,design.full,contrast=c(0,1,-1))
```

The other way to creat a design matrix in R is to include an intercept term to represents the base level of the factor. We just omitte the 0+ in the model formula when we create design matrix using model.matrix function:

```
> design.full=model.matrix(~group)
> design.full
```

```
   (Intercept) groupB groupC
1            1      0      0
2            1      0      0
3            1      0      0
4            1      0      0
5            1      1      0
6            1      1      0
7            1      1      0
8            1      1      0
9            1      0      1
10           1      0      1
11           1      0      1
12           1      0      1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

In this sitution, the first coefficient measure the log scaler of baseline mean expression level in group A, and the second and third column are now relative to the baseline, not represent the mean expression level in group B and C. So, if we want to compare B to A, now that is equivalent to test the 2nd coefficient equal to 0.

```
> result=glm.LRT(NanoStringData2,design.full,Beta=2)
```

Beta=3 means compare C to A:

```
> result=glm.LRT(NanoStringData2,design.full,Beta=3)
```

4.3 Multigroup Comparisons

NanoStringDiff can do multigroup comparisons, for example if we want compare group A to the average of group B and C,

```
> design.full=model.matrix(~0+group)
> result=glm.LRT(NanoStringData2,design.full,contrast=c(1,-1/2,-1/2))
```

5 Differential Expression Analysis for Multifactor Experiment

In this section, we still use `NanoStringData2` in examples, but the above examples all cotnsider single factor `treatment condition` , now we include the other experiment factor `gender` to consider multifactor problems.

5.1 Nested Interactions

First, we form the design matrix use factors `group` and interaction between `group` and `gender`:

```
> design=pData(NanoStringData2)[,c("group","gender")]
> group=design$group
> gender=design$gender
> design.full=model.matrix(~group+group:gender)
> design.full
```

```
   (Intercept) groupB groupC groupA:genderMale groupB:genderMale
1            1      0      0                 1                 0
2            1      0      0                 1                 0
3            1      0      0                 0                 0
4            1      0      0                 0                 0
5            1      1      0                 0                 1
6            1      1      0                 0                 1
7            1      1      0                 0                 0
8            1      1      0                 0                 0
9            1      0      1                 0                 0
10           1      0      1                 0                 0
11           1      0      1                 0                 0
```

```
12           1      0      1                   0                   0
   groupC:genderMale
1                  0
2                  0
3                  0
4                  0
5                  0
6                  0
7                  0
8                  0
9                  1
10                 1
11                 0
12                 0
attr(,"assign")
[1] 0 1 1 2 2 2
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"

attr(,"contrasts")$gender
[1] "contr.treatment"
```

Here, we consider the mean expression level of female in group A as the baseline expression level, if we want to test the effect of gender in group A, we can use Beta=4,

```
> result=glm.LRT(NanoStringData2,design.full,Beta=4)
```

Compare treatment group B to A ignore the effect the gender,

```
> result=glm.LRT(NanoStringData2,design.full,Beta=2)
```

Consider the interaction effect between gender and group B compare to A:

```
> result=glm.LRT(NanoStringData2,design.full,Beta=c(2,5))
```

5.2 All Interactions

We also can form a design matrix consider all interactions:

```
> design.full=model.matrix(~group+gender+group:gender)
```

Which is equivalent to:

```
> design.full=model.matrix(~group*gender)
> design.full
```

```
   (Intercept) groupB groupC genderMale groupB:genderMale groupC:genderMale
1            1      0      0          1                 0                 0
2            1      0      0          1                 0                 0
3            1      0      0          0                 0                 0
4            1      0      0          0                 0                 0
5            1      1      0          1                 1                 0
6            1      1      0          1                 1                 0
7            1      1      0          0                 0                 0
8            1      1      0          0                 0                 0
9            1      0      1          1                 0                 1
10           1      0      1          1                 0                 1
11           1      0      1          0                 0                 0
12           1      0      1          0                 0                 0
attr(,"assign")
[1] 0 1 1 2 3 3
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"


attr(,"contrasts")$gender
[1] "contr.treatment"
```

Test the gender effect in Treatment group B:

```
> result=glm.LRT(NanoStringData2,design.full,Beta=4:5)
```

6 Session Info

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

- R version 4.6.1 (2026-06-24), x86_64-pc-linux-gnu

- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_GB, 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
- Time zone: America/New_York
- TZcode source: system (glibc)
- Running under: Ubuntu 24.04.5 LTS
- Matrix products: default
- BLAS: /home/biocbuild/bbs-3.24-bioc/R/lib/libRblas.so
- LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Biobase 2.74.0, BiocGenerics 0.60.0, NanoStringDiff 1.44.0, generics 0.1.4
- Loaded via a namespace (and not attached): Rcpp 1.1.2, compiler 4.6.1, matrixStats 1.5.0, tools 4.6.1

References

[1] Masaki Mori, Robinson Triboulet, Morvarid Mohseni, Karin Schlegelmilch, Kriti Shrestha, Fernando D Camargo, and Richard I Gregory. Hippo signaling regulates microprocessor and links cell-density-dependent mirna biogenesis to cancer. *Cell*, 156(5):893–906, 2014.