User Manual for rSeqNP

Yang Shi (email: shyboy@umich.edu)
and
Hui Jiang (email: jianghui@umich.edu)

02/01/2015

1. Introduction

rSeqNP is an R package using a non-parametric approach to test for differential expression (DE) and differential splicing (DS) of genes from RNA-seq data. It performs two types of tests. The first test is to identify DE genes and isoforms using standard non-parametric tests based on ranks of expression values of genes and isoforms. Table 1 summarizes experimental designs that can be handled by rSeqNP and the non-parametric statistics used in each scenario.

Table 1. Non-parametric statistics used by rSeqNP

Study design	Test statistic
Two condition comparison	Wilcoxon rank-sum statistic
Multiple condition comparison	Kruskal-Wallis statistic
Paired two condition comparison	Wilcoxon singed-rank statistic
Quantitative outcomes	Spearman's rank correlation coefficient
Survival outcomes	Score statistic of the Cox proportional hazard model

The second test is, as a complement to the first test, to identify differential genes (either DE or DS) based on permutation tests of the gene differential score (*GDS*). Suppose a gene has *J* distinct isoforms, and T_j is the statistic for testing DE of the *j*th isoform as described in Table 1. The *GDS* is computed as:

$$GDS = \sum_{j=1}^{J} T_j^2 \quad (1)$$

See [1] for details of how the permutation test is performed. The package is currently in its beta version. The authors appreciate any feedbacks and suggestions. The package is submitted to CRAN and the relevant website and documents will be updated later.

2. Usage

2.1 Download and installation: The package installation files ("`rSeqNP_1.0.tar.gz`" and "`rSeqNP_1.0.zip`") and testing dataset ("`test_data_real.zip`" file) are available at the authors' webpage: http://www-personal.umich.edu/~jianghui/rseqnp/. The statistical software R (version≥3.0.0) should be installed and the following R packages are also required:

combinat, *plyr*, *exactRankTests*.

For UNIX/Linux machines, download the package installation source file ("`rSeqNP_1.0.tar.gz`") and use the following R command to install the package:

```
>install.packages(pkgs='Directory_where_you_download_the_
package/rSeqNP_1.0.tar.gz',          lib='Directory_where
you_want_to_install_the_package', repos=NULL)
```

For Windows machines, download the package installation binary file ("`rSeqNP_1.0.zip`") and use the following R command to install the package:

```
>install.packages(pkgs='Directory_where_you_download_the_
package/rSeqNP_1.0.zip',             lib='Directory_where
you_want_to_install_the_package', repos=NULL)
```

Upon completion of installation, you can use the following R command to load the package and view the package documentation (package description, user manual and help pages for the functions in the package)

```
> library(rSeqNP)
> help(package="rSeqNP")
```

Below we demonstrate the usage of the package in details.

2.2 Data preprocessing: Before applying rSeqNP, the raw sequence reads (usually as FASTA or FASTQ files) need to be processed to obtain the expression estimates (e.g., in RPKM, FPKM or other normalized unit) of all the genes and their isoforms for each sample in the RNA-Seq study. The authors recommend rSeq [2], which is available at http://www-personal.umich.edu/~jianghui/rseq/, or RSEM [3], which is available at http://deweylab.biostat.wisc.edu/rsem/. See the websites for the detailed documents for the usage of rSeq and RSEM. The output files of rSeq and RSEM can be directly read by rSeqNP. Other tools that can be used is Cuffdiff 2 [4], but the users may need to re-format the output files of that package to make them consistent with the input format of rSeqNP (see the *Usage* section for details).

2.3 Usage: We demonstrate the usage of rSeqNP using two examples. The first example is from simulated single-end reads of two groups with 20 samples (10 samples in each group) using the R package *polyester* [5], which is distributed with the package. The second example (in the "`test_data_real.zip`" file) is from a real RNA-Seq experiment, where paired-end reads were generated from prostate cancer samples and matched benign samples of 14 Chinese prostate cancer patients [6]. In both examples, the raw sequence reads were pre-processed by rSeq.

2.3.1 Example 1: For demonstration purpose, this example takes only about 2 minutes to run.

Step 1 - load the files containing the estimated expression values of genes and isoforms: This dataset is incorporated in the package folder "`rSeqNP\inst\extdata\test_data_simulated`" (you can locate this folder at the directory where you installed the package), which contains the output files from rSeq (files with extension "`.xls`") that provide the estimated expression values of genes and isoforms in RPKM unit for each sample, and the "`samplelist_simulated.txt`" file in that folder contains the file names of those output files ("`.xls`" files) from rSeq, with one file per line (these lines can also be the absolute path of the "`.xls`" files). The R script for reading the expression data and its output are given below:

```
>simulated_data_dir = system.file('extdata', 'rseq_data_simulated',
                                  package='rSeqNP')
>setwd(simulated_data_dir)
>mydata = read.rseq('samplelist_rseq.txt')

Number of samples in total:20
Number of genes in total:2585
Number of transcripts(isoforms) in total:4937
Processing file: sample01.xls .
Processing file: sample02.xls .
……
Processing file: sample20.xls .
```

The function "`read.rseq`" will return a list that contains the following two data-frames: 1). The "`iso.exp`" data-frame contains the expression values of the isoforms, with the 1st column as the gene names, the 2nd column as the isoform names, the 3rd column as the number of isoforms corresponding to the genes and the rest of the columns as the expression values of the isoforms in each sample. 2). The "`gene.exp`" data-frame contains the expression values for the genes, with the 1st column as the gene names, the 2nd column as the number of isoforms of the genes and the rest columns as the expression values of the genes in each sample.

If the user would like to check the expression values of a particular gene (isoform) at this step, the following R script can be used:

```
>data$gene.exp[data$gene.exp$gene.name==' IRF6', ]
>data$iso.exp[data$iso.exp$gene_name==' IRF6',]
```

Upon typing the above two lines of codes, the expression values of the IRF6 gene and its isoforms in all the samples will be shown. Note that rSeq put a blank space before

the gene names, so that they can display correctly when writing the results to text files and opening them in Microsoft Excel.

The expression values in the output files of RSEM can be read and formatted by the function `read.rsem`. See the help page of that function for details. If other tools are used for obtaining the estimated expression values of genes and isoforms (for example, Cuffdiff), please format the expression data as the same as described above.

Step 2 - test for differential expression and splicing: Next we can use the following R script to invoke the main function of the package to perform the tests:

```
> result = rSeqNP.Main(exp_object=mydata, type='twoclass',
                       gamma=NULL, outcome=c(rep(1,10), rep(2,10)),
                       nperm=1e5, seed=100, exp.mean=0.5, low.exp=0.5)
```

Below is the explanation of the arguments of this function:

1. `exp_object`: the list returned by the function "`read.rseq`" or "`read.rsem`", as described above.
2. `type`: This indicates the type of study design. It should be one of the following: "`twoclass`" – unpaired two group comparison; "`twopaired`" – paired two group comparison; "`multiclass`" – multi-group comparison; "`quant`" – quantitative outcome data; "`survi`" – survival outcome data.
3. `outcome`: the outcome vector. For two group data, the first group must be denoted by 1, and the second group must be denoted by 2. For two paired group data, this vector should be like `c(1,2,1,2...,1,2)`, where the first two samples form a pair, and the third and fourth form a pair, and so on. For *K* class data, Class *k* must be denoted by *k*, *k* = 1,…,*K*. For quantitative data, this vector is the quantitative outcome data (real numbers) corresponding to each sample. For survival outcome data, this vector is the survival time of each subject or the time at which the observation for the subject is censored.
4. `gamma` optional, only required for survival outcome data, which is the censoring status vector for survival outcomes. 1 for observed (died) and 0 for censored. Default value is `NULL`.
5. `nperm`: number of permutations for the permutation test based on the *GDS*. Default value is `1e6`. The user can set it to larger values, as larger values give more stable results under different seed. But the computing time will also increase proportionally. In the above R script, we set it as `1e5` to shorten the running time.
6. `seed`: random seed to generate the permutation indexes. Different seed usually gives slightly different estimated *p*-values and FDR. Default value is 100.
7. `exp.mean`: the parameter to filter out the genes and isoforms with very low expressions. If the average expression levels of a gene or an isoform across all the samples is less than `exp.mean`, the gene or isoform will be filtered out in the analysis. Default value is 0.5.
8. `low.exp`: this argument is used for two group comparison (paired or unpaired).

If the average expression levels of a gene or an isoform across all the replicates in one group is less than `low.exp`, the gene or isoform will be given a lowly expressed label in the returned value (a list) of this function (the returned list contain the results of the tests for differential expression or splicing, see below).

Below is the screen output after running the above codes:

```
> result = rSeqNP.Main(exp_object=mydata, type='twoclass',
                gamma=NULL, outcome=c(rep(1,10), rep(2,10)),
                nperm=1e5, seed=100, exp.mean=0.5, low.exp=0.5)
246 isoforms has been filtered because their expression levels
are too low.
203 genes has been filtered because their expression levels are
too low.
This is unpaired two group comparison.
Isoform differential expression test done!
Gene differential expression test done!
Begin the permutation test based on GDS score.
Genes that contain greater or equal to 5 isoforms will be pooled
together.
Testing genes with 1 isoforms.
Testing genes with 2 isoforms.
Testing genes with 3 isoforms.
Testing genes with 4 isoforms.
Testing genes with more than 5 isoforms.
Permutation test based on GDS done!
Running time is:
   user  system elapsed
 81.518   0.000  81.373
```

The returned value of this function (`result` in this example) is a list that contains the following three data-frames: 1). The "`isoform.DE.result`" is a data-frame that contains the results from the two-sample Wilcoxon rank-sum test (for other study designs, the test is the corresponding one described in Table 1) for differential expression of isoforms; 2). The "`gene.DE.result`" is the data-frame that contains the results from the two-sample Wilcoxon rank-sum test (for other study designs, the test is the corresponding one described in Table 1) for differential expression of genes. 3). The "`GDS.result`" is the data-frame that contains the results from the permutation test based on the *GDS*. We explain the meanings of the column names of each data-frame below:

For the "`isoform.DE.result`" data-frame:

```
>result$isoform.DE.result[1:3,]
```

```
gene_name isoform_name isoform_number WRS.stat    p.WRStest       fdr.BH    log.FC if.low
     UTS2    NM_021995              2      -50 1.082509e-05 3.103942e-05 -4.862848  FALSE
      AGL    NM_000644              6       50 1.082509e-05 3.103942e-05  4.471919  FALSE
   ZRANB2    NM_005455              2      -50 1.082509e-05 3.103942e-05 -4.037039  FALSE
```

1. `gene_name`: this is the name of the gene that contains the isoform.
2. `isoform_name`: this is the name of the isoform tested.
3. `isoform_number`: this is the number of isoforms that corresponds to the gene in the analysis. Note it may be smaller than the number in the annotation database if you filter out those isoforms with low expressions.
4. `WRS.stat`: this is the centered Wilcoxon rank-sum statistic. For paired two-group data, this column will be "`WSR.stat`" (centered Wilcoxon signed-rank statistic); for multi-group data, this column will be "`KW.stat`" (Kruskal-Wallis statistic); for quantitative outcome data, this column will be "`SCC`" (Spearman's correlation coefficient); for survival outcome data, this column will be "`Score.stat`" (Score statistic of the Cox proportional hazard model).
5. `p.WRStest`: this is the *p*-values from the Wilcoxon rank-sum test for isoform DE. For other types of data, this column will be the *p*-values from the corresponding tests that are described in Table 1.
6. `fdr.BH`: this is the FDR estimated by the Benjamini–Hochberg (BH) procedure.
7. `Log.FC`: This column only exists for two group comparisons (paired or unpaired), which is the log2 of the average fold change between the two groups (Group 2 v.s. Group 1).
8. `if.low`: This column only exists for two group comparisons (paired or unpaired), which is a Boolean data type. It is `TRUE` if the average isoform expression levels of all the replicates in one group are less than the value of `low.exp` specified in the `rSeqNP.Main` function (in this situation, the fold change can be either very large or very small, which is not a good summary statistic of the data). It is `FALSE` otherwise.

For the "`gene.DE.result`" data-frame:

```
>result$gene.DE.result[1:3,]
    gene.name WRS.stat    p.WRStest       fdr.BH    log.FC if.low
    LINC01525       50 1.082509e-05 1.329828e-05  3.273101  FALSE
       PIN1P1      -50 1.082509e-05 1.329828e-05 -2.631645  FALSE
 LOC101928043       50 1.082509e-05 1.329828e-05  2.621820  FALSE
```

1. `gene.name`: this is the name of the gene tested.
2. `WRS.stat`: this is the centered Wilcoxon rank-sum statistic. For paired two-group data, this column will be "`WSR.stat`" (centered Wilcoxon signed-rank statistic); for multi-group data, this column will be "`KW.stat`" (Kruskal-Wallis statistic); for quantitative outcome data, this column will be "`SCC`" (Spearman's correlation coefficient); for survival outcome data, this column will be "`Score.stat`" (Score statistic of the Cox proportional hazard model).

3. `p.WRStest`: this is the *p*-values from the Wilcoxon rank-sum test for gene DE. For other types of data, this column will be the *p*-values from the corresponding tests that are described in Table 1.
4. `fdr.BH`: this is the FDR estimated by the Benjamini–Hochberg (BH) procedure.
5. `Log.FC`: This column only exists for two group comparisons (paired or unpaired), which is the log 2 of the average fold change between the two groups (Group 2 v.s. Group 1).
6. `if.low`: This column only exists for two group comparisons (paired or unpaired), which is a Boolean data type. It is `TRUE` if the average gene expression levels of all the replicates in one group are less than the value of `low.exp` specified in the `rSeqNP.Main` function (in shis situation, the fold change can be either very large or very small, which is not a good summary statistic of the data). It is `FALSE` otherwise.

For the "`GDS.result`" data-frame:

```
>result$GDS.result[1:3,]
```

gene.name	GDS	p.plugin	fdr.plugin	iso.num	T.value
AADACL4	2500	0	0	1	0
ABCA4	2500	0	0	1	0
ACAP3	2500	0	0	1	0

1. `gene.name`: this is the name of the gene tested.
2. `GDS`: this is the gene differential score (*GDS*) as computed in Equation (1).
3. `p.plugin`: this is the *p*-values from the permutation test based on the *GDS*, which is estimated by the permutation plug-in method (see [1] and [7] for details).
4. `fdr.plugin`: this is the FDR from the permutation test based on the *GDS*, which is estimated by the permutation plug-in method (see [1] and [7] for details).
5. `iso.num`: this is the number of isoforms that the gene have in the analysis. Note it may be smaller than that in the annotation database if you filter out those isoforms with low expression values.
6. `T.value`: This is a statistic that measures the level of differential splicing. For a gene with J ($J \geq 2$) isoform, let θ_{1j} be the average expression level of isoform j in condition 1 ($j = 1, 2, \ldots, J$) and θ_{2j} be the same quantity in condition 2; $\theta_1 = [\theta_{11}, \theta_{12}, \ldots, \theta_{1J}]^T$ and $\theta_2 = [\theta_{21}, \theta_{22}, \ldots, \theta_{2J}]^T$ be the vector of isoform expressions for each condition. The T value is computed as:

$$T = \frac{1}{2} \left\| \frac{\theta_1}{\| \theta_1 \|_1} - \frac{\theta_2}{\| \theta_2 \|_1} \right\|_1$$

where $\| \cdot \|_1$ denotes the vector L_1 norm. Large T values indicate high level of differential splicing. For genes with only one isoform, T value is 0 [8].

2.3.2 Example 2: This example is a real RNA-Seq experiment performed on prostate cancer samples and matched benign samples from 14 Chinese prostate cancer patients [6] (the raw sequence files in FASTQ format are available at: http://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-567/samples/). Upon

unzipping the "test_data_real.zip" file, you will get the "test_data_real" folder that contains the output files from rSeq (files with extension ".xls") with the estimated expression values of genes and isoforms in RPKM unit for each sample, and the "Chinese_samplelist_twogroup.txt" file in that folder contains the file names of those output files. We can use the following R script to perform the tests for differential expression or splicing (It takes about 30 minutes of running):

```
>setwd('Directory_where_you_unzipped_the_data/test_data_real')
>data=read.rseq('Chinese_samplelist_twogroup.txt')
Number of samples in total: 28
Number of genes in total: 24900
Number of transcripts(isoforms) in total: 45714
Processing file: T1_ERR031028.xls .
Processing file: T2_ERR031030.xls .
……
Processing file: N14_ERR031025.xls .

> result = rSeqNP.Main (exp_object=data, type='twoclass',
                        outcome=c(rep(1,14), rep(2,14)))
22517 isoforms has been filtered because their expression
levels are too low.
9883 genes has been filtered because their expression levels
are too low.
This is unpaired two group comparison.
Isoform differential expression test done!
Gene differential expression test done!
Begin the permutation test based on GDS score.
Genes that contain greater or equal to 6 isoforms will be pooled
together.
Testing genes with 1 isoforms.
Testing genes with 2 isoforms.
Testing genes with 3 isoforms.
Testing genes with 4 isoforms.
Testing genes with 5 isoforms.
Testing genes with more than 6 isoforms.
Permutation test based on GDS done!
Running time is:
   user  system elapsed
 1463.95   0.069  1478.09
```

The result list returned by rSeqNP.Main contains the test results as described above in *Example 1*. We can use FDR≤0.05 as a cut-off to call a gene or isoform as differential expressed. Using the following R script, we can check the numbers of

differential events detected by rSeqNP, which are the numbers that we reported in Column 1 in Table 2 of Reference [1]:

```
> iso.DE.res=result$isoform.DE.result
> gene.DE.res=result$gene.DE.result
> GDS.res=result$GDS.result
> sum(iso.DE.res$fdr.BH<0.05)
[1] 2933
> sum(gene.DE.res$fdr.BH<0.05)
[1] 3346
> sum(GDS.res$fdr.plugin<0.05)
[1] 4122
```

We can also generate some graphical summaries, for example the "volcano plots" (plot of the fold changes of genes or isoforms v.s. the *p*-values of the tests), using the following R script:

```
>volcano.plot(iso.DE.res, filter=F, xlim=c(-10,10),
           pch=20, xlab='Log 2 fold change',
           ylab=expression(-log~10~italic(p)-value))

>volcano.plot(iso.DE.res, filter=T, xlim=c(-10,10),
           pch=20, xlab='Log 2 fold change',
           ylab=expression(-log~10~italic(p)-value))

>volcano.plot(gene.DE.res, filter=F, xlim=c(-10,10),
           pch=20, xlab='Log 2 fold change',
           ylab=expression(-log~10~italic(p)-value))

>volcano.plot(gene.DE.res, filter=T, xlim=c(-10,10),
           pch=20, xlab='Log 2 fold change',
           ylab=expression(-log~10~italic(p)-value))
```

In Figure 1, we create the volcano plots for genes and isoforms using the above R script, where we can see the *p*-values for the Wilcoxon rank-sum test show some discrete patterns (lines of dots). Also after removing those isoforms or genes that have low expression levels in one group (as specified by the argument `filter`, where `filter=FALSE` means do not remove those isoforms or genes when drawing the plot and `filter=TRUE` means remove those isoforms or genes), we can see the plots became "thinner" at the bottom part (Figure 1B and 1D).

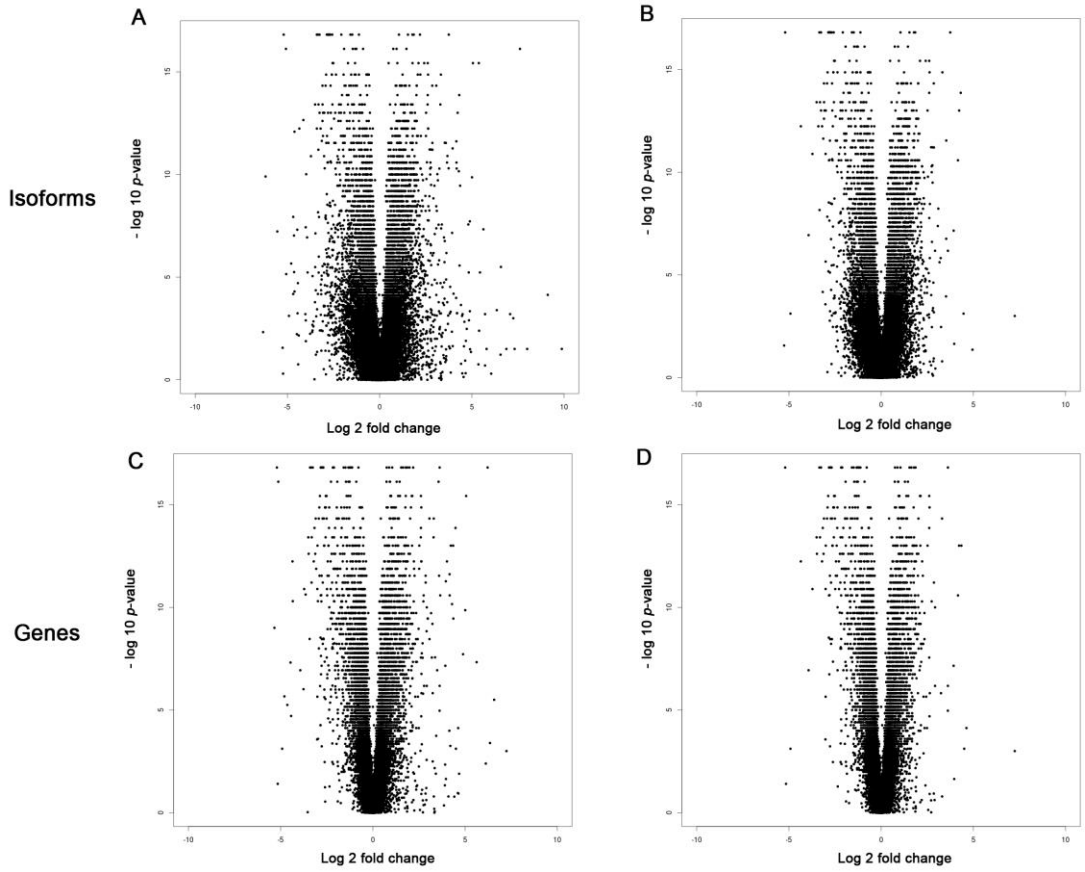

Figure 1. Volcano plots for Example 2. (A) Plot for all isoforms. (B) Plot for isoforms after removing those isoforms with average expression levels less than the value of `low.exp` (0.5) in one group. (C) Plot for all genes. (D) Plot for genes after removing those genes with average expression levels less than the value of `low.exp` (0.5) in one group.

Acknowledgements

We thank Jun Li for sharing the source R scripts for calculating the score statistic of the Cox proportional hazard model for survival outcome data and the permutation plug-in method for estimating the *p*-values and FDR of the permutation test based on the *GDS*. The R scripts for those two parts in rSeqNP was adapted from the R package *npSeq* (version 1.1) written by Jun Li [7].

References

1. Shi Y, Chinnaiyan AM, Jiang H (2015) rSeqNP: A non-parametric approach for detecting differential expression and splicing from RNA-seq data (in revision).

2. Jiang H, Wong WH (2009) Statistical inferences for isoform expression in RNA-Seq. Bioinformatics 25: 1026-1032.

3. Li B, Dewey CN (2011) RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics 12: 323.

4. Trapnell C, Hendrickson DG, Sauvageau M, Goff L, Rinn JL, et al. (2013) Differential analysis of gene regulation at transcript resolution with RNA-seq. Nat Biotechnol 31: 46-53.

5. Frazee AC, Jaffe AE, Langmead B, Leek J (2014) Polyester: simulating RNA-seq datasets with

differential transcript expression.
6. Ren S, Peng Z, Mao JH, Yu Y, Yin C, et al. (2012) RNA-seq analysis of prostate cancer in the Chinese population identifies recurrent gene fusions, cancer-associated long noncoding RNAs and aberrant alternative splicings. Cell Res 22: 806-821.
7. Li J, Tibshirani R (2013) Finding consistent patterns: a nonparametric approach for identifying differential expression in RNA-Seq data. Stat Methods Med Res 22: 519-536.
8. Shi Y, Jiang H (2013) rSeqDiff: detecting differential isoform expression from RNA-Seq data using hierarchical likelihood ratio test. PLoS One 8: e79448.