Using "npSeq" (Version 1.1) to discover differential expression based on sequencing data

Jun Li
Department of Statistics, Stanford University
junli07@stanford.edu

September 7, 2011

The main purpose of this package is for people to reproduce the results in Li et al. ([2]), and develop new methods based on it. We do not expect people to use it on new data. For people who want to use methods in Li et al. ([2]) on new data, we recommend them to use SAM (samr, SAMseq) instead of this package.

1 Introduction

npSeq is an R package that implements all methods described in Li *et al.* ([2]). Like edgeR ([4]), DESeq ([1]) and PoissonSeq ([3]), npSeq is used to discover differentially expressed elements (genes, tags, et al.) based on sequencing (RNA-Seq, Tag-Seq, ...) data. It gives a list of significant genes, as well as the estimated false discovery rate.

This package is NOT available on CRAN, to install it (under Linux), follow these two steps.

1. Install CRAN R package `combinat` by typing on the R command
   `install.packages("combinat")`

2. Download `npSeq_1.1.tar.gz` to your current R directory, and then type on the R command
   `install.packages("npSeq_1.1.tar.gz", type="source")`

Every time before you use this package, load it by typing on the R command:

`library("npSeq")`

The main/unique features of npSeq are:

1. It uses nonparametric statistic to judge the significance of differential expression, and thus it is very robust to violation of distributional assumptions of other methods, and very robust to outliers and other types of noise/errors.

2. It can be used not only for data with two-class outcome (eg. normal vs. diseased), but also for data with multiple-class outcome (eg. cancer type 1 vs. cancer type 2 vs. cancer type 3), data with quantitative outcome (eg. patients with different blood pressures), and data with *survival* outcome (eg. patients with different survival time).

3. It uses permutation to generate the null distribution of the test statistic, rather than full trusts the asymptotic distribution. Therefore, it may be more robust to gene-gene correlation.

Please refer to the paper ([2]) for more details. This instruction document can be downloaded from `http://www.stanford.edu/~junli07/research.html`.

2 Relationship to SAM (samr, SAMseq)

This package implements exactly the method described in Li *et al.* ([2]). The SAMseq function in the newest version of SAM (samr) is slightly different. npSeq and SAMseq use the same nonparametric statistic, and both uses permutation to generate the null distribution. However, npSeq uses symmetric cutoffs for the nonparametric statistic ([2]), while SAMseq uses asymmetric cutoffs ([6]). Therefore, for some datasets, SAMseq will only give significant genes that are up-regulated or only give significant genes that are down-regulated, but npSeq will almost always give both genes that are up-regulated and genes that are down-regulated.

3 Usage

npSeq is very easy to use. Only two functions in the package are available to the users. Note: npSeq uses the same method as PoissonSeq to estimate the sequencing depth. npSeq does not output the estimated sequencing depth. For users who want to get the sequencing depth, please use the `PS.Est.Depth` function in PoissonSeq.

3.1 `npSeq.Main`

This is the main function of this package. Given a sequencing dataset and optionally the running parameters, this function will do all the following things for the user:

1. Filter out genes with too small number of reads across experiments. The default is to filter out genes with no more than 5 reads totally across all experiments, AND to filter out genes with no more than 0.5 reads averagely across all experiments. For example, if you have 6 samples totally, then all genes with 0, 1, 2, 3, 4, or 5 reads totally across the 6 samples will be discarded. As another example, if you have 20 samples totally, then all genes with 0$\sim$10 reads totally across the 20 samples will be discarded.

2. Estimate the sequencing depth.

3. Calculate the statistic, permute and estimate the FDR for all possible cutoffs.

The input of `PS.Main` are two lists: `dat` and `para`. `dat` contains all information about the sequencing dataset, and `para` contains all parameters that control the running of the program. `dat` must be given by the users, while `para` can be left as blank, in which case its default values will be used. In most cases, the default values should be good enough.

3.1.1 `dat`

`dat` contains the following elements:

1. `n` (required): the data matrix. Rows for genes, columns for experiments (samples). Note that *these are the original counts without any normalization.* The normalization will be done by npSeq. RPKM cannot be used as the input data matrix. However, non-integer counts, which can be generated by some mapping algorithms for reads mapped with uncertainty, can be used as the input.

2. `y` (required): the outcome vector. For two class data, the first class must be denoted by `1`, and the second class must be denoted by `2`. For K class data, Class k must be denoted by `k`, $k = 1, \ldots, K$. For quantitative data, `y` are real numbers.

3. `type` (required): `twoclass`, `multiclass`, `quant`, or `survi`. No other values are accepted.

4. `gname` (optional): gene names. Default value: `1 : nrow(n)`. That is, the i'th gene is named `i`.

5. `gamma` (optional): censoring statuses. `1` for observed (died), `0` for censored.

6. `delta` (optional): true significance of the genes. `TRUE` for significance. `FALSE` for insignificance. This can only be known in simulated data. When `delta` is not null, true false discovery rates will be calculated and returned.

Note: now npSeq cannot be used for paired twoclass data, but extending to paired data is straightforward and has been implemented in SAM (samr, SAMseq). We'll add it to npSeq in the next version.

For example, if you have a sequencing dataset from 3 normal samples and 5 cancer samples. Each experiment is mapped to 10,000 genes. Then your data `n` should be a 10000×10 matrix. `y=c(1,1,1,2,2,2,2,2)` if you put the normal samples in the first three columns of `n`. `type` should be `twoclass`, and `gname` is set to be the names of the 10,000 genes. If you do not know the gene names, you do not need to specify it, or set it as `NULL`–the program will re-set it as `1, 2, ..., 10000`.

After setting `n`, `y`, `type`, `pair` and `gname`, you just type

`dat <- list(n=n, y=y, type=type, gname=gname)`.

3.1.2 para

`para` contains the following elements:

1. `npermu`: number of permutations. default value: `100`. You may want to set them to larger values like `200`, `500`, or even `1000`, as larger `npermu` gives more stable results (under different seeds). However, the computational time also increases (proportionally).

2. `seed`: random seed to generate the permutation indexes. default value: `10`. Note that different seeds typically give slightly different estimated FDRs. The larger `npermu` is, the smaller the difference is.

3. `ct.sum`: if the total number of reads of a gene across all experiments <= ct.sum, this gene will not be considered for differential expression detection. Default value: `5`.

4. `ct.mean`: if the mean number of reads of a gene across all experiments <= ct.mean, this gene will not be considered for differential expression detection. Default value: `0.5`.

5. `nsam`: number of resamplings used to calculate the statistic. Default value: `20`. You may set it to a larger value, but do not set it to a smaller value if there is no specific reason.

6. `sam.meth`: resampling method: `1` for subsampling, `2` for Poisson sampling. Default value: `2`.

Note that all the above elements are optional. Feel free to set none, one, two, ..., or all of them. The ones that are not set will be given the default values by the program.

3.2 npSeq.Simu.Data

This function simulates sequencing data with twoclass, multiclass, quantitative, or survival outcomes. The simulation parameters are chosen so that the output data best mimic real datasets. The input parameters should be a list containing all simulation parameters. The output is also a list. It contains all simulation parameters as well as simulated data. More details are below. The input list includes the following parameters:

1. `type` (required): the types of outcome. It should be one of the following: `twoclass`, `multiclass`, `quant`, or `survi`.

2. `option` (required): the distribution of the simulated data. `1` for Poisson, `2` for negative binomial with dispersion 0.25, `3` for Poisson with outliers, `4` for negative binomial with outliers.

3. `NSAM` (required): number of samples. an integer for `quant` and `survi`, and a vector of integers for `twoclass` and `multiclass`.

4. `NGENE` (optional): number of genes. default value `20000`.

5. `psig` (optional): percentage of significant genes. default value `0.3`.

6. `up.perc` (optional): in the significant genes, how many percent are up-regulated. Default value: `0.8`.

The output list contains all elements required by `npSeq.Main` function, so it can be directly used by `npSeq.Main`.

4 Examples

Here I use 't Hoen data ([5]) as an example. The dataset is available from Gene Expression Omnibus under accession number GSE10782. An arranged version called `tHoen.txt` (zipped to `tHoen.zip`) can be downloaded from `http://www.stanford.edu/~junli07/research.html`. One can analyze the data using the following steps:

1. Read in the data from the file:

```
hdat <- read.table("tHoen.txt", header=T, stringsAsFactors=F)
gname <- hdat[, 1]
n <- as.matrix(hdat[, -1])
y <- c(1, 2, 1, 2, 1, 2, 1, 2)
```

```
type <- "twoclass"
dat <- list(n=n, y=y, type=type, gname=gname)
```

2. Set the parameter list. In this case (and in most other cases), we do not need to set it. Of course, you can set it if you want to. For example, if you want to increase the number of permutations to 200, simply type (does not make difference to the results in this dataset since the number of all distinct permutations is choose(8, 4) = 70.)
   `para <- list(npermu=200)`
   If you also want to change the random seed so that you get a slightly different result, type
   `para <- list(npermu=200, seed=100)`

3. Run npSeq:
   If you use the default `para`, type
   `res <- npSeq.Main(dat=dat)`
   If you have re-set `para`, type
   `res <- npSeq.Main(dat=dat, para=para)`

4. All the results has been stored in a data frame (i.e. a table) `res`. You can type
   `print(res[1:100, ])`
   to check the top 100 significant genes. Or you can type
   `write.table(res, file="res.txt", quote=F, row.names=F, col.names=T, sep="\t")`
   to write the results into a file.
   Or you can plot the FDR curve by typing
   `plot(res$nc, res$fdr, xlab="Number called", ylab="estimated FDR", type="l")`

References

[1] S. Anders and W. Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11:R106, 2010.

[2] J. Li and R. Tibshirani. Finding consistent patterns: a nonparametric approach for identifying di?erential expression in rna-seq data. in press, 2011.

[3] J. Li, D. M. Witten, I. Johnstone, and R. Tibshirani. Normalization, testing, and false discovery rate estimation for rna-sequencing data. *Biostatistics*, 2011. in press.

[4] M. D. Robinson, D. J. McCarthy, and G. K. Smyth. edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–40, 2010.

[5] P. A. C. 't Hoen, Y. Ariyurek, H. H. Thygesen, E. Vreugdenhil, R. H. Vossen, R. X. de Menezes, J. M. Boer, G. J. van Ommen, and J. T. den Dunnen. Deep sequencing-based expression analysis shows major advances in robustness, resolution and inter-lab portability over five microarray platforms. *Nucleic Acids Res*, 36(21):e141, 2008.

[6] V. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to transcriptional responses to ionizing radiation. *Proc. Natl. Acad. Sci. USA.*, 98:5116–5121, 2001.