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

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

September 6, 2011

1 Introduction

PoissonSeq is an R package that implements all methods described in Li *et al.* ([2]). Like edgeR ([3]) and DESeq ([1]), PoissonSeq 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 available on CRAN, to install it, just type on the R command:

```
install.packages("PoissonSeq")
```

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

```
library("PoissonSeq")
```

The main/unique features of PoissonSeq are:

1. 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), and data with quantitative outcome (eg. patients with different blood pressures).

2. 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 the violation of distributional assumptions, and more robust to gene-gene correlation.

3. It implements a new method for estimating the sequencing depth. This method has been shown to be more accurate than existing methods ([2]). Sequencing depth serves

as a scaling factor between experiments, and has been shown to be of critical importance to many inference problems based on sequencing data.

4. It can handle both Poisson-type data and overdispersed (such as negative binomial) data. In both cases, it is able to estimate the false discovery rate accurately.

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

Differences from Version 1.0

On Sep 6, 2011, PoissonSeq has been updated to Version 1.1. The following changes have been made:

1. In the new vesion, only two functions (`PS.Main` and `PS.Est.Depth`) are available to users to make this package easy to use. It is very unlikely that users will use other functions.

2. The output of `PS.Main` is now a data frame (table) instead of a list. This change is also to make this package easy to use.

3. The estimated FDRs are always monotone non-decreasing now.

4. We now filter out genes with too small number of reads across experiments. In Version 1.0, this needs to be done beforehand by the user.

5. The estimated sequencing depths have a different scale than before. In Version 1.0, sum(depth)=1; now we have product(depth)=1. However, their relative values do not change. For example, if the depths estimated by Version 1.0 are 0.1, 0.2, 0.3, and 0.4, then they will be 0.452, 0.904, 1.355, 1.807 by Version 1.1.

2 Usage

Compared to Version 1.0, Version 1.1 simplifies the output. Now, only two functions in the package are available to the users:

2.1 `PS.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. Transform the data so that it looks more like Poisson.

3. Estimate the sequencing depth, which is used to normalize the data implicitly.

4. Calculate the score statistics, 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.

2.1.1 `dat`

`dat` contains the following elements:

1. `n`: 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 PoissonSeq. 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`: 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`: `twoclass`, `multiclass`, or `quant`. No other values are accepted.

4. `pair`: paired data or not. Default value: `FALSE`. Only takes effect for twoclass data. Note that for paired data, the two classes are still denoted by `1` and `2`, and `y` must be like `1, 2, 1, 2, 1, 2, ..., 1, 2`. The first `1` is paired with the first `2`, the second `1` is paired with the second `2`, et al.

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

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`, `pair` should be `FALSE`, 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, pair=pair, gname=gname)`.

2.1.2 para

`para` contains the following elements:

1. `trans`: to tranform the data using the order transformation or not to transform it. default value: `TRUE`. As most sequencing datasets are overdispersed, trans should be set as `TRUE`. In case you are very sure that your dataset is Poisson, you can set it as `FALSE`. However, even in this case, leaving it as `TRUE` does not hurt much.

2. `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).

3. `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.

4. `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`.

5. `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`.

6. `div`: the number of divisions of genes for estimating theta. default value: `10`.

7. `pow.file`: the file to store the power transform curve (`mean(log(mu)) ~ 1/theta`). default value: `pow.txt`.

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.

2.2 PS.Est.Depth

This function estimates the sequencing depths of the experiments. We understand that some users may not want to use PoissonSeq to find differentially expressed genes, but just want to estimate the sequencing depth so that they can normalize the data for classification or clustering, so we provide this function. This function has four parameters:

1. `n`: The data matrix.
2. `iter`: Number of iterations used. Default value: `5`. The default value is usually a good choice.
3. `ct.sum`: a cutoff. If the total number of reads of a gene across all experiments <= ct.sum, this gene will not be considered for estimating sequencing depth. Default value: `5`.
4. `ct.mean`: a cutoff. If the mean number of reads of a gene across all experiments <= ct.mean, this gene will not be considered for estimating sequencing depth. Default value: `0.5`.

Usually, default values for `iter`, `ct.sum` and `ct.mean` are appropriate, and only `n` needs to be specified.

3 Examples

Here I use 't Hoen data ([4]) 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"
pair <- FALSE
dat <- list(n=n, y=y, type=type, pair=pair, 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
   `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 PoissonSeq:
   If you use the default `para`, type
   `res <- PS.Main(dat=dat)`
   If you have re-set `para`, type
   `res <- PS.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")`

5. If you want to estimate the sequencing depths, you can type
   `depth <- PS.Est.Depth(n)`

References

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

[2] 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.

[3] 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.

[4] 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.