**Introduction**

The software available here is an implementation of our proposed method described in the paper “A Zero-Inflated Poisson Model for Insertion Tolerance Analysis of Genes Based on Tn-seq Data”. Our proposed method utilizes read count information obtained from transposon insertion sequencing to identify insertion tolerance of genes. We propose a zero-inflated Poisson (ZIP) model for analyzing the Tn-seq data that are high-dimensional and with an excess of zeros. Maximum likelihood estimates of model parameters are obtained using an expectation-maximization (EM) algorithm, and pseudogenes are utilized to construct appropriate statistical tests for the transposon insertion tolerance of normal genes of interest. Then we use a multiple testing procedure that categorizes genes into each of the three states, hypo-tolerant, tolerant, and hyper-tolerant, while controlling the false discovery rate.

**Instructions**

**Software Requirement**

1. R 3.1.0 (or later) software needs to be installed. R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. Here is the link (<https://cran.r-project.org/>) on how to download and install R software in local computers. For convenience, RStudio can be freely installed and serves as a powerful and productive user interface for R. We recommend at least 4GB of available RAM on the computer.
2. Download custom R scripts for our proposed method from the website: <http://github.com/ffliu/TnSeq>, put all the R scripts and also the input file in the same folder.

To run the R script, there are two ways,

1. Simply run R script from command line.

Type the following command: *nohup R CMD BATCH ./test.r &*

Where "test.r" is an example file name.

1. Open R software, copy and paste all the R scripts to the R Console.

Below the [input file format](http://saclab.tamu.edu/essentiality/README.html#format) and the output format are described and a detailed guidance about how to run the R scripts is also elucidated below.

**Input File Format**

The input file we need should have the read count information of all possible transposon insertion sites in the genome (i.e. with or without insertions). In addition, the read data must be stored in a tab-delimited text-file (say, in TXT, CSV, DAT format). The table below describes each of the columns necessary for the input file:

|  |  |
| --- | --- |
| **Column Header** | **Column Definition** |
| Gene | Gene ID. |
| count | Read count observed at this particular insertion site. |
| gene\_type | If the gene is pseudogene, gene\_type=TRUE or 1; otherwise gene\_type=FALSE or 0. |

Notice that more columns can be added to the input data file if there is addition information that might have effect on the insertion probability. Below is an example of an input file in this format:

Gene count gene\_type

Rv0001 0 FALSE

Rv0001 0 FALSE

Rv0001 0 FALSE

Rv0001 0 FALSE

Rv0001 0 FALSE

Rv0001 0 FALSE

If the read counts for all possible insertion sites within a gene are all zeros, this means the inactivation of the gene by transposon insertion completely suppresses the growth and obviously the gene is hypo-tolerant to transposon insertion. For simplicity, we can exclude the information for the gene from the input file.

Worth to mention, ARTIST user manual (Text S1 of Section Supporting Information in Pritchard *et al.*, 2014) provides very detailed instruction on how to obtain the mapped read counts at every possible insertion site in the genome in the MATLAB environment. The required data are pre-aligned reads from a transposon library (in SAM format), FASTA sequence files for each of the chromosomes mapped the reads to, Genome annotation file (in GTF or GFF2 format), which contains the name, start and ends sites for each annotated gene. Therefore, you could look at this user manual as reference to generate the required input file format.

**Output Format**

The results can contain the following information:

* The names for normal genes of interest.
* Maximum likelihood estimate for the parameters corresponding to the mean abundance for each normal gene.
* Estimated standard deviation for the estimated difference between the mean abundance for each of the normal genes of interest and that for pseudogenes.
* The p-values corresponding to the hypothesis testing for each normal gene with null hypothesis testing “the mean abundance of mutants for each normal gene is equal to that of the pseudogenes”.
* The tolerance state of transposon insertion for normal genes of interest.

0 = the gene is tolerant of disruption; 1 = the gene is hypo-tolerant of transposon insertion; 2 = the gene is hypo-tolerant of transposon insertion.

In addition, if possible covariates that may affect the transposon insertion rate are included in our zero-inflated Poisson model, then the results of the estimated regression coefficients corresponding to covariates can be used to check if the covariates are important factors. Below is the output format we could obtain.

|  |  |
| --- | --- |
| **Column Header** | **Column Definition** |
| est | Maximum likelihood estimate for regression coefficients corresponding to covariates. |
| sd | Standard deviation of parameter estimates |
| p-value | P values to test if the coefficient is zero or not |

**Guide to Run R scripts**

We describe how to run R scripts for two separate scenarios using example input files and example R codes.

1. Use the Zip model without covariates

We have included the example input file “dat\_Mtb\_example.txt” to provide examples of real Tn-Seq data that have been processed by our proposed method. Details are in the main R script “EM\_cov\_example.r”. Comments are proceeded with "#" tags to describe the usage of some functions in the R script.

1. Use the ZIP model with covariates included

A simulated example input file is provided here to test the execution of the script. Details of how to obtain the simulated data and how to utilize our proposed method to identify insertion tolerance states for all the normal genes are in the main R script “EM\_cov\_example.r”. Comments are proceeded with "#" tags to describe the usage of some functions in the R script.

**Case Study**

This section provides a very detailed analysis of a subset of the IA3902 dataset generated using Tn5 transposon.The purpose of the analysis is to identify insertion tolerance states for all the normal genes.

1. Reading in the data.

> rm(list=ls())

> dat<-read.table("subdat\_Tn5.txt",header=TRUE)

> head(dat)

ID count GC\_percent match gene\_type

1 CJSA\_0016 0 34.1 0 0

2 CJSA\_0016 0 34.1 0 0

3 CJSA\_0016 0 34.1 0 0

4 CJSA\_0016 0 34.1 0 0

5 CJSA\_0016 0 34.1 0 0

6 CJSA\_0016 0 34.1 0 0

#calculate the number of normal genes

> N<-length(unique(dat[dat$gene\_type==0,1]))

#calculate the number of possible insertion locations for each normal genes.

> lengthN<- na.omit( as.numeric( tapply( dat[dat$gene\_type==0,1] , dat[dat$gene\_type==0,1] ,length)))

> as.numeric(lengthN)

[1] 675 846 1329 1791 297 657 1563 369 756 450 1248 363 1386 1128 630 1023 1128

[18] 510 1092 624 750 1095 987 351 750 801 546 900 729 234 960 1047 318 387

[35] 2076 1074 552 711 648 318 1770 2571 903 1164 924 1233 1083 498 738 528 771

[52] 723 1239 1626 282 1296 849 429 489 180 729 381 786 2589 1098 351 732 1398

[69] 888 819 1548 1611 483 498 1419 2265 1803 720 1194 651 990 1320 546 327 399

[86] 639 1074 198 765 1548 444 222 1503 1536 204 777 702 426 831 2559

#length\_ps: calculate the number of insertion sites for all pseudogenes.

> pos\_pseudo<-which(dat$gene\_type==1)

> length\_ps<-length(pos\_pseudo)

> length\_ps

[1] 24017

> #### add a column "ord" to denote normal genes ID and

> #### use the sampe ID for all the pseudogenes.

> dat$ord<- N+1

> dat$ord[dat$gene\_type==0]<-rep( 1:N ,lengthN)

> #### design matrix.

> X<-model.matrix(~as.factor(dat$match)+ dat$GC\_percent)

> head(X)

(Intercept) as.factor(dat$match)1 dat$GC\_percent

1 1 0 34.1

2 1 0 34.1

3 1 0 34.1

4 1 0 34.1

5 1 0 34.1

6 1 0 34.1

> y<-dat$count

> ### the length of coefficient regression.

> nbet<-ncol(X);nbet

[1] 3

1. Set initial parameter estimates for EM algorithm using following commands.

> # mu\_initis: initial estimate for mu

> mu\_inits = rep(0, N+1)

> y\_numn0\_gene<- tapply( y>0 , dat$ord ,sum)

> y\_sum\_gene<-tapply( y , dat$ord ,sum)

> for(i in 1:(N+1)){

+ score\_mug<-function(mu) {

+ s=y\_sum\_gene[i]

+ ss=y\_numn0\_gene[i]

+ s/mu - (1/(1- exp(-mu)))\* ss

+ }

+

+ if( y\_sum\_gene[i] <=5) mu\_inits[i]==0.01

+ else mu\_inits[i]<- uniroot( score\_mug ,interval=c(0.001,10000000))$root

+ }

#the initial values for mu

> mu\_inits

[1] 7.541467 5.033633 7.119245 187.862501 5.170454 41.500007

[7] 972.975000 33.285715 16.799999 14.999968 17.216203 8.164343

[13] 38.749998 23.357143 7.663077 10.190094 9.544778 6.434096

[19] 6.129480 14.083346 45.800000 10.384296 5.561898 5.221659

[25] 5.404159 6.658110 109.107143 6.599703 74.595216 39.909091

[31] 5.984901 12.999971 4.342803 30.941150 22.359375 6.129480

[37] 3.197036 26.882353 14.999968 3.562692 5.872292 4.573926

[43] 10.685478 21.367832 11.133144 10.666425 3.652825 5.984901

[49] 332.885246 11028.066668 7.930485 6.849877 3.920692 8.050064

[55] 21.000007 4.727012 10.249638 4.758853 11.812413 6.993586

[61] 7.997312 4.342803 5.516200 49.740001 6.539555 6.562144

[67] 7.328533 4.062194 7.302783 24.222223 6.916212 84.824984

[73] 105.846154 11847.125903 9.213351 5.120753 11.892776 33.611111

[79] 3.380976 5.244919 5.771411 8.598440 11.666567 4.505225

[85] 4.758853 14.266684 18.599991 1466.000003 4.836310 5.083699

[91] 3.144048 3.920692 8.266167 105.928571 9.499288 5.782098

[97] 7.997312 3.413193 6.260748 209.800000 7.057007

> # beta\_inits: initial estimate for beta

> p0 <- ( sum(y==0)- exp(- mu\_inits) %\*% c(lengthN, length\_ps) )/ length(y)

> beta\_inits <- c(log(p0/(1-p0)), 0,0)

> beta\_inits

[1] 3.398091 0.000000 0.000000

1. Obtain the maximum likelihood estimates for all the parameters using EM algorithm

### need to source the R code “EM\_algorithm\_withcov.r”.

> source("EM\_algorithm\_withcov.r")

> ## Run EM Algorithm

> output <- em.algo(mu\_inits,beta\_inits)

> output$mu

1 2 3 4 5 6 7

7.540551 5.018236 7.115892 187.862500 5.170826 41.500000 972.975000

8 9 10 11 12 13 14

33.285714 16.799998 14.999990 17.216216 8.163149 38.750000 23.357143

15 16 17 18 19 20 21

7.642872 10.189882 9.544856 6.431732 6.122102 14.083315 45.800000

22 23 24 25 26 27 28

10.384321 5.527187 5.209949 5.397423 6.651186 109.107143 6.601151

29 30 31 32 33 34 35

74.595238 39.909091 5.984965 12.999939 4.289641 30.941176 22.359375

36 37 38 39 40 41 42

6.108137 2.599324 26.882353 14.999992 3.049183 5.871720 4.576273

43 44 45 46 47 48 49

10.685539 21.367816 11.133004 10.666054 3.606610 5.949558 332.885246

50 51 52 53 54 55 56

11028.066667 7.928808 6.836658 2.987217 8.048370 21.000000 4.648021

57 58 59 60 61 62 63

10.249348 4.710792 11.812448 6.987459 7.995395 4.365477 5.512844

64 65 66 67 68 69 70

49.740000 6.540123 6.566321 7.326990 4.029927 7.293778 24.222222

71 72 73 74 75 76 77

6.916000 84.825000 105.846154 11847.125926 9.212572 5.077571 11.892731

78 79 80 81 82 83 84

33.611111 3.301646 5.227674 5.769168 8.597218 11.666294 4.504000

85 86 87 88 89 90 91

4.743306 14.266658 18.599999 1466.000000 4.799992 5.071823 2.927872

92 93 94 95 96 97 98

3.919826 8.263442 105.928571 9.498718 5.719022 7.992349 3.404771

99 100 101

6.238990 209.800000 7.054990

> output$beta

(Intercept) as.factor(dat$match)1 dat$GC\_percent

7.2158803 8.9026954 -0.1181193

1. Now we proceed to obtain estimated variance for the MLE of parameters.

> ## Obtain variance of parameter estimates

> var\_dif<-deriv\_SD\_diff(output$beta,output$mu)

> est\_dif<- as.numeric(output$mu[1:N])-output$mu[N+1]

1. Identify the insertion tolerance states of normal genes using the multiple testing procedure.

> #res: estimate, sd, pvalue, qvalue, type based on q value and estimate

> res<-matrix(0,N,5)

> res[,1]<-output$mu[1:N]

>

> #standard deviation

> res[,2]<- sqrt(var\_dif)

> #p value

> res[,3]<- 2\*pnorm(-abs(est\_dif/ sqrt(var\_dif)))

>

> # q value

> res[,4]<-round(p.adjust(res[,3], "BH"), 4)

>

> ## 1: hypo-tolerant of transposon insertion.

> ## 0: tolerant of disruption.

> ## 2: hyper-tolerant of transposon insertion.

> res[,5]<-ifelse(res[,4]<=0.01 & est\_dif <=0, 1, 2)

> res[ res[, 4]> 0.01, 5]<- 0

> res<-cbind(as.character(unique(dat[dat$gene\_type==0,1])),res)

> colnames(res)<-c("GeneID","est","sd","pvalue","qvalue","essential")

#the first 20 rows of the result

> res[1:20,]

GeneID est sd pvalue qvalue essential

[1,] "CJSA\_0016" "7.54055078917553" "0.597827654951062" "0.416671953903724" "0.463" "0"

[2,] "CJSA\_0022" "5.01823572551679" "0.601018990935994" "0.000701901825696681" "0.0013" "1"

[3,] "CJSA\_0023" "7.11589189193606" "0.487141245585719" "0.900508597213297" "0.9096" "0"

[4,] "CJSA\_0041" "187.8625" "1.08964324123947" "0" "0" "2"

[5,] "CJSA\_0065" "5.17082608678603" "1.03550231263463" "0.0688252488552616" "0.093" "0"

[6,] "CJSA\_0066" "41.5" "2.04038735188134" "6.14528769980757e-64" "0" "2"

[7,] "CJSA\_0074" "972.975" "4.93331080976228" "0" "0" "2"

[8,] "CJSA\_0081" "33.2857142857143" "1.26420510967261" "1.25719892594831e-95" "0" "2"

[9,] "CJSA\_0109" "16.7999979814369" "1.06451046841726" "5.46353628341203e-20" "0" "2"

[10,] "CJSA\_0117" "14.9999902056289" "1.93990009477585" "4.21127636656926e-05" "1e-04" "2"

[11,] "CJSA\_0140" "17.2162156459722" "0.691725322466319" "7.50642925905399e-49" "0" "2"

[12,] "CJSA\_0161" "8.16314862172409" "1.17384766201261" "0.345149452008785" "0.3967" "0"

[13,] "CJSA\_0185" "38.75" "1.18199370435968" "2.16761781450948e-158" "0" "2"

[14,] "CJSA\_0189" "23.3571428559292" "1.29674514410118" "3.02563342936024e-36" "0" "2"

[15,] "CJSA\_0205" "7.64287158142996" "1.61697415434862" "0.716179939245443" "0.7539" "0"

[16,] "CJSA\_0208" "10.1898815219086" "0.706168091768089" "9.0252250171689e-06" "0" "2"

[17,] "CJSA\_0223" "9.54485639882733" "0.668782577627404" "0.000196882553956543" "4e-04" "2"

[18,] "CJSA\_0243" "6.43173215026727" "0.857655800611013" "0.467410171691768" "0.5136" "0"

[19,] "CJSA\_0249" "6.12210227083056" "0.676905318070957" "0.168152101427893" "0.2102" "0"

[20,] "CJSA\_0258" "14.0833154340286" "1.0894079496365" "1.10741733159361e-10" "0" "2"

1. Save the result to data format using either “write.csv” or “write.table”.

> write.csv(res,"EM\_res.csv",row.names=FALSE)

> write.table(res,"EM\_res.txt",row.names=FALSE)

**Running Time**

The time for the execution of the software will depend on the number of genes in the genome being analyzed, the number of possible insertion sites for each of the interested genes and the stopping criterion for the EM algorithm. In this paper, the stopping criterion we used was based on the maximum value of differences between the current and new estimates among all parameters. More specifically, when the maximum difference is smaller than a specified tolerance level (set to be10^(-6) in our implementation), the EM algorithm stops iterating. When we analyzed the IA3902 datasetin our paper (1631 genes with a total of 1,544,034 possible insertion sites), the EM algorithm stopped after 65 iterations and this took about 20 minutes using a Windows machine with a 3.07 GHz CPU and 8GB RAM.

Reference

Pritchard J. R., Chao M. C., Abel S., Davis B. M., Baranowski C., et al. (2014) ARTIST: High-Resolution Genome-Wide Assessment of Fitness Using Transposon-Insertion Sequencing. PLoS Genet. 10, e1004782.