@MPRA package vignette

Dandi Qiao

2019-01-30

Contents

1 Introduction

The analysis toolset for MPRA data (@MPRA) includes functions for simulating and analyzing MPRA data, and for power calculations of MPRA experiments. This tutorial briefly introduces the functions provided by the @MPRA package, using the example data included in the package.

We can load the library using:

```
library(atMPRA)
```

```
## Warning: package 'DESeq2' was built under R version 3.5.2
```

```
## Warning: package 'BiocParallel' was built under R version 3.5.2
```

We can do a quick power calculation:

```
nsim = 10
ntag = 10

result = getPower(nsim = nsim, ntag = ntag, nrepIn = 3, nrepOut = 3,
    slope = c(rep(1, ntag * nsim), rep(2, ntag * nsim)), method = c("MW",
        "mpra_lm"), scenario = "fixTotalDepth")
```

```
## Warning in getPower(nsim = nsim, ntag = ntag, nrepIn = 3, nrepOut = 3, slope = c(rep(1, : The input
```

```
## [1] 0.6
```

```
result$Power
```

```
## [1] 0.6
```

2 Data available in the package

The estimated distributional parameters of the MPRA data (GSE70531 in GEO database) was obtained using the `estimateMPRA` function in this package. The basic parameters include \ `inputProp`: The proportion of counts per tag among all tags in the library\ `transEff`: The distribution of transfection efficiencies (normalized RNA/DNA ratio) across tags\ `dispFunc_input`: The dispersion function of the input tag counts across replicates as a function of the mean \ `dispFunc_output`: The dispersion function of the output tag counts across replicates as a function of the mean\ \ The estimation was done using DESeq2. This distribution will be the default distribution for simulating MPRA data in this package if not specified otherwise. The data is loaded with the package automatically.

```
GSE70531_params
```

```
## $dispFunc_input
## function (means)
## exp(predict(fit, data.frame(logMeans = log(means))))
## <bytecode: 0x7fa235a5c418>
## <environment: 0x7fa247392ff8>
## attr(,"fitType")
## [1] "local"
## attr(,"varLogDispEsts")
## [1] 0.4812829
## attr(,"dispPriorVar")
## [1] 0.25
##
## $dispFunc_output
## function (q)
## coefs[1] + coefs[2]/q
## <bytecode: 0x7fa235a079e8>
## <environment: 0x7fa235a41638>
## attr(,"coefficients")
## asymptDisp  extraPois
##  0.5183263 12.7233435
## attr(,"fitType")
## [1] "parametric"
## attr(,"varLogDispEsts")
## [1] 0.8172103
## attr(,"dispPriorVar")
## [1] 0.3268525
##
## $inputProp
## function (v)
## .approxfun(x, y, v, method, yleft, yright, f)
## <bytecode: 0x7fa235a04c60>
## <environment: 0x7fa235a08270>
##
## $transEff
## function (v)
## .approxfun(x, y, v, method, yleft, yright, f)
## <bytecode: 0x7fa233e60e78>
## <environment: 0x7fa235a056a8>
##
## $sizeFactor_input
## K562_minP_DNA1 K562_minP_DNA2
##      1.2060454      0.8291562
##
## $sizeFactor_output
## K562_CTRL_minP_RNA1 K562_CTRL_minP_RNA2 K562_CTRL_minP_RNA3
##           0.6613516           0.4404059           1.4904598
## K562_CTRL_minP_RNA4 K562_CTRL_minP_RNA5 K562_CTRL_minP_RNA6
##           1.2337356           1.8488591           1.3372992
```

3 Functions

3.1 Simulating MPRA data

There are multiple ways to simulate MPRA data in this package:\ 1. Simulating MPRA data using default distribution.\ 2. Simulating MPRA data using estimated distributions from observed data\ 3. Simulating MPRA data by specifying parameters in the model.\

The parameters for simulating a MPRA dataset includes:\ 1. number of SNPs in the data (`nsim`)\ 2. number of tags per SNP (`ntag`)\ 3. number of replicates in the input and output (`nrepIn`, `nrepOut`)\ 4. RNA/DNA ratio for all tags (`slope`)\ 5. Total depth for one replicate (`fixTotalD`) or mean depth per tag (`fixMeanD`)\

3.1.1 Simulating MPRA data using estimated distributions from observed data

We have simulated the MPRA using defulat settings above. Now we want to demonstrate how to simulate MPRA using estimated distribution from observed data. Here we will use the parameters we estimated for GSE70531.

```r
totalDepth = 2e+05

ntag = 10

nsim = 10

nrepIn = 5

nrepOut = 5

inputProp = GSE70531_params[[3]](runif(ntag * nsim * 2))

slopel = GSE70531_params[[4]](runif(nsim * 2))

inputDispFunc = GSE70531_params[[1]]

outputDispFunc = GSE70531_params[[2]]

slope = rep(slopel, each = ntag)

datt = sim_fixDepth(inputProp, ntag, nsim, nrepIn, nrepOut, slope,
    inputDispFunc = inputDispFunc, outputDispFunc = outputDispFunc,
    sampleDepth = totalDepth)

datt[1:10, ]
```

```
##    allele simN input_rep1 input_rep2 input_rep3 input_rep4 input_rep5
## 1     Ref    1       1195       1278       1109       1034       1114
## 2     Ref    1        141        179        133        173        164
## 3     Ref    1        145        100        109        109         85
## 4     Ref    1       1047       1085       1101       1109       1263
## 5     Ref    1         17         14         17          9         16
## 6     Ref    1       3717       3950       4016       4150       3790
## 7     Ref     1         242         240         223         262         293
## 8     Ref     1          13          10          17           8          14
## 9     Ref     1         100         113         109         107          84
## 10    Ref     1         190         203         260         237         214
##    output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
## 1         1338         591         344         294         722
## 2          268          27          39          15          12
## 3          100          55          46          38          49
## 4         1575         175        5217        1677        1173
## 5            0           0          29           3          32
## 6         3022        3828        3105        2027        2712
## 7          140         328         188         218         511
## 8            2          73          17          16           1
## 9           34          33          35          55          50
## 10          72         267          90         194         195
```

If we would like to simulate data based on an observed MPRA dataset, we can estimate the parameters using the function `estimateMPRA`.

```
rnaCol=8

new_params=estimateMPRA(datt, nrepIn, rnaCol, nrepOut, nsim, ntag)

new_params
```

```
## $dispFunc_input
## function (q)
## coefs[1] + coefs[2]/q
## <bytecode: 0x7fa24568a830>
## <environment: 0x7fa24568aec0>
## attr(,"coefficients")
##  asymptDisp   extraPois
## 0.001355058 2.835324647
## attr(,"fitType")
## [1] "parametric"
## attr(,"varLogDispEsts")
## [1] 0.7014079
## attr(,"dispPriorVar")
## [1] 0.25
##
## $dispFunc_output
## function (q)
## coefs[1] + coefs[2]/q
## <bytecode: 0x7fa24568a830>
## <environment: 0x7fa247dc43f0>
## attr(,"coefficients")
## asymptDisp  extraPois
##  0.5755493 12.1054886
## attr(,"fitType")
## [1] "parametric"
## attr(,"varLogDispEsts")
## [1] 1.199697
## attr(,"dispPriorVar")
## [1] 0.5547631
##
```

```
## $inputProp
## function (v)
## .approxfun(x, y, v, method, yleft, yright, f)
## <bytecode: 0x7fa24ea423a0>
## <environment: 0x7fa2343e3f68>
##
## $transEff
## function (v)
## .approxfun(x, y, v, method, yleft, yright, f)
## <bytecode: 0x7fa24ea423a0>
## <environment: 0x7fa23525e350>
##
## $sizeFactor_input
## input_rep1 input_rep2 input_rep3 input_rep4 input_rep5
##  0.9894536  1.0039159  1.0005035  1.0093590  1.0007959
##
## $sizeFactor_output
## output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
##   1.1236062   1.0737031   0.9246672   1.0506186   1.0670299
```

Then we can simulate new MPRA data using these parameters:

```
datt = sim_fixDepth(inputProp = new_params[[3]](runif(ntag * nsim * 2)),
    ntag, nsim, nrepIn, nrepOut, slope, inputDispFunc = new_params[[1]],
    outputDispFunc = new_params[[2]], sampleDepth = totalDepth)

datt[1:10, ]
```

```
##    allele simN input_rep1 input_rep2 input_rep3 input_rep4 input_rep5
## 1     Ref    1        351        438        464        510        402
## 2     Ref    1       1266       1245       1234       1342       1303
## 3     Ref    1       1116       1137       1384       1108       1199
## 4     Ref    1        204        227        244        272        242
## 5     Ref    1       1249       1058       1177       1285       1143
## 6     Ref    1       3260       3444       3399       3084       2873
## 7     Ref    1       1194       1300       1119       1257       1243
## 8     Ref    1        860        947        928        800        869
## 9     Ref    1       1122       1175       1117       1277       1184
## 10    Ref    1        218        236        284        254        283
##    output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
## 1          123          52          34          77         872
## 2         1384         313         350        2381         371
## 3          106         106         367         230        1540
## 4          170          57         126         158         245
## 5          819         537        1219        1009         914
## 6         1189        1877        6407        5602         748
## 7         1056         180         476         310         435
## 8          666         992         487        1664         699
## 9         1427        1111         842         426         793
## 10          73         367          21         471         133
```

3.1.2 Simulating MPRA data by specifying parameters in the model.

We can also simulate MPRA data by specifying parameters. For example, we may want to specify different mean input counts across tags for allele A and allele B. We can check if methods are biased by the allelic imbalance in the input distribution later using this simulated data.

```
inputDist = GSE70531_params[[3]](runif(nsim * ntag * 2))

datt = sim_fixInputMean(mean_A = 10, mean_B = 100, ntag = ntag, nsim = nsim,
    nrepIn = nrepIn, nrepOut = nrepOut, slope = slope, inputDist = inputDist,
    inputDispFunc = inputDispFunc, outputDispFunc = outputDispFunc)
```

```
## converting counts to integer mode
## converting counts to integer mode
```

```
datt[c(1:5, 101:105), ]
```

```
##     allele simN input_rep1 input_rep2 input_rep3 input_rep4 input_rep5
## 1      Ref    1          3         19          7         13         16
## 2      Ref    1         15         28         13         27         23
## 3      Ref    1         21         16         14         18         18
## 4      Ref    1          0          0          1          0          0
## 5      Ref    1          4          1          1          8          3
## 101    Mut    1        273        284        249        210        183
## 102    Mut    1          2          4          4          0          3
## 103    Mut    1          3          1          2         10          8
## 104    Mut    1          0          5          0          7          0
## 105    Mut    1        136        128         89        101        130
##     output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
## 1             1          22           7          17          23
## 2            12          10          15           8          76
## 3             7          15          10           5           8
## 4             0           0           0           0           0
## 5             1           1           1           0           7
## 101         393         254         123         378         157
## 102           3          30           0           0           0
## 103           8          14           2          30           9
## 104           0           0           0           1           0
## 105         101         116         194          95         169
```

Note the distribution of counts are very different between the two alleles `Ref` and `Mut`.

3.2 Analyze MPRA data

We provide a list of methods to analyze MPRA data. The input MPRA data frame should have `nsim*ntag*2` rows and `2+nrepIn+nrepOut` columns. The first column should be named 'allele', and the second column should be named 'simN'. The 'allele' columns should contain only two possible values 'Ref' and 'Mut' to refer to the two versions of alleles for each SNP.

A list of the methods that are available is here:

Test	singleReplicate	OptionName
Mann-Whitney	YES	MW
Matching	YES	Matching
Adaptive	YES	Adaptive
QuASAR-MPRA	YES	QuASAR
Fisher's Exact Test	YES	Fisher
T-test	NO	T-test
mpralm using mean	NO	mpralm

Test	singleReplicate	OptionName
mpralm using sum	NO	mpralm
edgeR	NO	edgeR
DESeq2	NO	DESeq2

To analyze a formmated MPRA data:

```
datt[1:10, ]
```

```
##    allele simN input_rep1 input_rep2 input_rep3 input_rep4 input_rep5
## 1     Ref    1          3         19          7         13         16
## 2     Ref    1         15         28         13         27         23
## 3     Ref    1         21         16         14         18         18
## 4     Ref    1          0          0          1          0          0
## 5     Ref    1          4          1          1          8          3
## 6     Ref    1          0          0          0          3          0
## 7     Ref    1          2          8         10          6          4
## 8     Ref    1         18         13          3         21          9
## 9     Ref    1         33         10         14         15         27
## 10    Ref    1          9         10          2          9          1
##    output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
## 1            1          22           7          17          23
## 2           12          10          15           8          76
## 3            7          15          10           5           8
## 4            0           0           0           0           0
## 5            1           1           1           0           7
## 6            0           0           0           1           0
## 7            2           7           9           4          15
## 8            0           1          10           0           8
## 9            0          23           8           8           1
## 10           0           3           0           0           9
```

```
results = analyzeMPRA(datt, nrepIn, rnaCol, nrepOut, nsim, ntag, method = c("MW",
    "Adaptive", "QuASAR", "T-test", "mpralm", "DESeq2"), cutoff = 0, cutoffo = 0)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
results
```

```
##    simN    resInput       resMW resMatching resAdaptive     QuASAR
## 1     1 0.446967893 0.133300136   0.7850000 0.133300136 0.70556915
## 2     2 0.011655012 0.965400612   0.8000000 0.800000000 0.21797602
## 3     3 0.015220074 0.742986425   0.5555556 0.555555556 0.39468110
## 4     4 0.002997003 0.093406593   1.0000000 1.000000000 0.75114684
## 5     5 0.015540016 0.408245349   0.9375000 0.937500000 0.91195738
## 6     6 0.094719522 0.315378120   1.0000000 0.315378120 0.24724308
## 7     7 0.954645355 0.004795205   0.0200000 0.004795205 0.49637433
## 8     8 0.015540016 0.572603867   1.0000000 1.000000000 0.27658795
## 9     9 0.002756067 0.931427396   0.6000000 0.600000000 0.65749897
## 10   10 0.002879473 0.853428305   0.8400000 0.840000000 0.06888803
##    ttest_paired      ttest mpralm_mean mpralm_sum     DESeq2
## 1    0.49600889 0.44695352  0.13947732 0.17692214 0.41145687
## 2    0.62283991 0.60933897  0.74079292 0.58523620 0.54655056
## 3    0.67165489 0.57520972  0.87891246 0.21255610 0.56512818
## 4    0.15584428 0.08185680  0.05774892 0.06173449 0.02402666
## 5    0.12586559 0.14520206  0.40745486 0.08727867 0.06398433
## 6    0.12791239 0.16463032  0.65767993 0.39718596 0.24811849
## 7    0.18530027 0.18304483  0.03245511 0.26855007 0.08682319
## 8    0.03619947 0.02659521  0.41057972 0.07743880 0.03366014
## 9    0.83524321 0.83505842  0.28909495 0.36536318 0.80565785
## 10   0.66773593 0.65763577  0.32731427 0.84686548 0.65469411
```

You can remove tags with mean counts less than the cutoffs specified for the input and output.

3.3 Power calculation

We can compute power based on simulated MPRA data specified using the options described above. Addition parameters here include the correction method for multiple testing, and the significance level to be used.

```
nrepIn = 2
nrepOut = 2
slopel = GSE70531_params[[4]](runif(nsim))
slopel = c(slopel, slopel + 1)
slope = rep(slopel, each = ntag)
result2 = getPower(nsim, ntag, nrepIn, nrepOut, slope, scenario = "fixInputDist",
    method = c("MW", "T-test", "mpralm", "edgeR", "DESeq2"),
    fixInput = c(20, 100), inputDist = inputDist, inputDispFunc = inputDispFunc,
    outputDispFunc = outputDispFunc, cutoff = -1, cutoffo = -1)
```

```
##        resMW ttest_paired        ttest  mpralm_mean   mpralm_sum
##          0.1          0.0          0.0          0.2          0.0
##        edgeR       DESeq2
##          0.5          0.5
```

```
result2$Power
```

```
##        resMW ttest_paired        ttest  mpralm_mean   mpralm_sum
##          0.1          0.0          0.0          0.2          0.0
##        edgeR       DESeq2
##          0.5          0.5
```

```
result3 = getPower(nsim, ntag, nrepIn, nrepOut, slope = 1, scenario = "fixTotalDepth",
    method = c("MW", "Matching", "Adaptive", "Fisher", "QuASAR", "T-test",
        "mpralm", "edgeR", "DESeq2"), fixTotalD = 2e+05, inputDist = inputDist,
    inputDispFunc = inputDispFunc, outputDispFunc = outputDispFunc, cutoff = -1,
    cutoffo = -1)
```

```
##        resMW  resMatching  resAdaptive       QuASAR fisherPvalue
##          0.0          0.0          0.0          0.0          0.9
## ttest_paired        ttest  mpralm_mean   mpralm_sum        edgeR
##          0.0          0.0          0.0          0.0          0.0
##       DESeq2
##          0.0
```

```
result3$Power
```

```
##        resMW  resMatching  resAdaptive       QuASAR fisherPvalue
##          0.0          0.0          0.0          0.0          0.9
## ttest_paired        ttest  mpralm_mean   mpralm_sum        edgeR
##          0.0          0.0          0.0          0.0          0.0
##       DESeq2
##          0.0
```

3.4.Other methods included

`calEnrichment`: This function uses hypergeometric tests to compute enrichment in SNPs with significant allelic DNA imbalance among the SNPs with significant allele-specific effect defined by each method.

```
calEnrichment(results)
```

```
## Compute enrichment in allele-imbalanced SNPs among significant results...
```

```
##         method q m   enrichP
## 1        resMW 0 1 1.0000000
## 2  resMatching 0 1 1.0000000
## 3  resAdaptive 0 1 1.0000000
## 4       QuASAR 0 0 1.0000000
## 5 ttest_paired 1 1 0.7000000
## 6        ttest 1 1 0.7000000
## 7  mpralm_mean 0 1 1.0000000
## 8   mpralm_sum 0 0 1.0000000
## 9       DESeq2 2 2 0.4666667
```