# @MPRA package vignette

Dandi Qiao

2020-07-21

## 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:

```r
library(atMPRA)
```

We can do a quick power calculation:

```r
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] 0.9
```

```r
result$Power
```

```
## [1] 0.9
```

#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\ \

We assume that the tag counts across replicates follow a Negative Binomial distribution with mean $\mu$ and dispersion $\sigma^2$. Then the variance is $\mu + \sigma^2\mu^2$.

However, it is observed that the dispersion parameter is not constant in RNA-Seq data. In DESeq2, it was assumed that:

$$log(\sigma^2) \sim N(log(a + b/\mu), \sigma_d^2)$$

We will use the dispersion function estimated by DESeq2 to generate count data here.

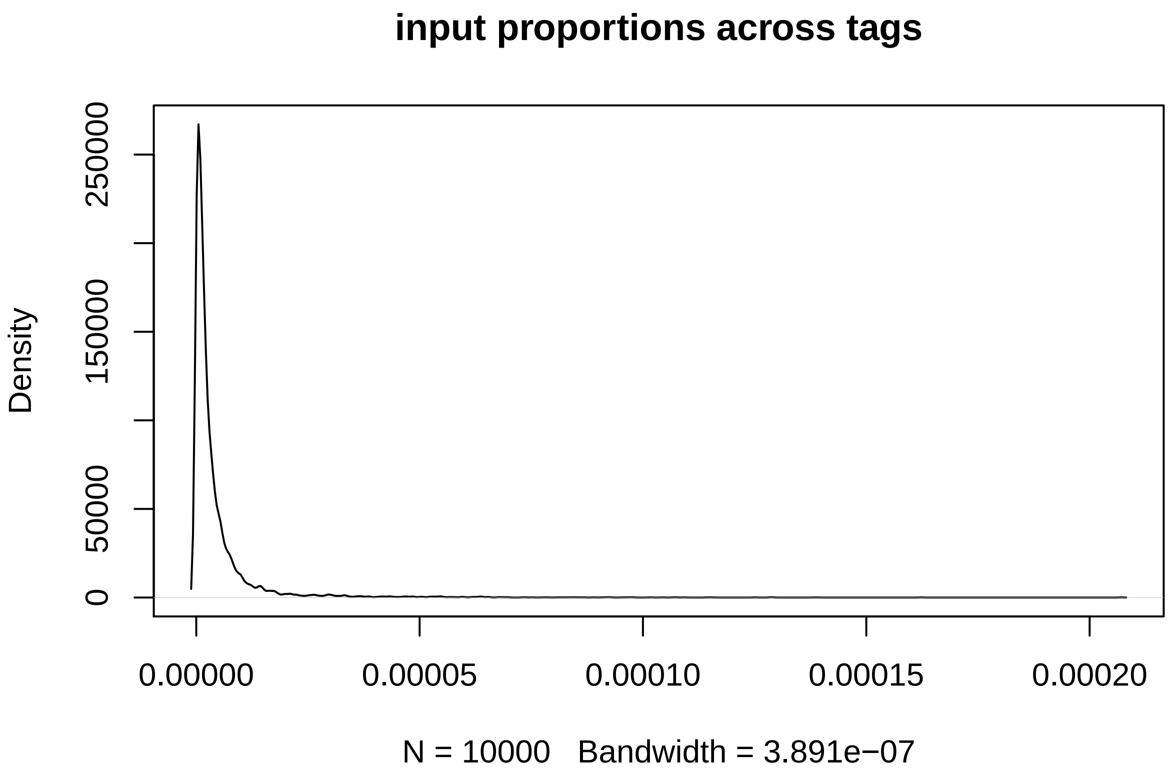
The estimation parameters for GSE70531 was done using the `estimateMPRA` function which used DESeq2 package. This distribution will be the default distribution for simulating MPRA data in this package if not specified otherwise. The data is loaded with the package automatically.

```
GSE70531_params
```

```
## $dispFunc_input
## function (means)
## exp(predict(fit, data.frame(logMeans = log(means))))
## <bytecode: 0x7fa04c8215d0>
## <environment: 0x7fa038a8bcf8>
## attr(,"fitType")
## [1] "local"
## attr(,"varLogDispEsts")
## [1] 0.4812829
## attr(,"dispPriorVar")
## [1] 0.25
##
## $dispFunc_output
## function (q)
## coefs[1] + coefs[2]/q
## <bytecode: 0x7fa03da52a18>
## <environment: 0x7fa038aecce0>
## 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: 0x7fa03da50238>
## <environment: 0x7fa03da53230>
##
## $transEff
## function (v)
## .approxfun(x, y, v, method, yleft, yright, f)
## <bytecode: 0x7fa03da51af0>
## <environment: 0x7fa03da50cb8>
##
## $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 K562_CTRL_minP_RNA4
##           0.6613516           0.4404059           1.4904598           1.2337356
## K562_CTRL_minP_RNA5 K562_CTRL_minP_RNA6
##           1.8488591           1.3372992
```

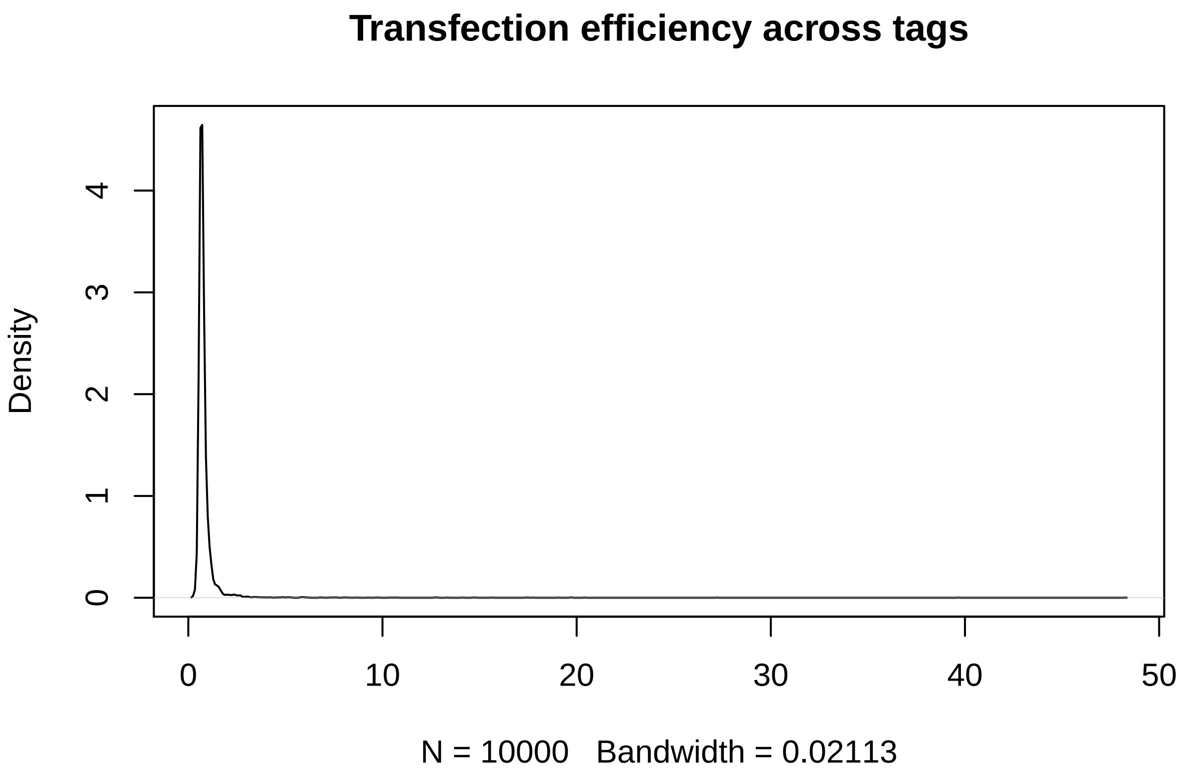
The input proportions are usually very skewed due to cloning and PCR.

```
plot(density(GSE70531_params[[3]](runif(10000))), main="input proportions across tags")
```



The transfection efficiency distribution for GSE70531 looks like this:

```
plot(density(GSE70531_params[[4]](runif(10000))), main="Transfection efficiency across tags")
```



#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.

```
totalDepth = 2e+05

ntag = 10

nsim = 10
```

```r
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         24          5          8         16         16
## 2     Ref    1       3418       3304       3272       3285       3158
## 3     Ref    1        437        455        537        452        507
## 4     Ref    1       1522       1466       1591       1522       1504
## 5     Ref    1        623        579        526        487        596
## 6     Ref    1        202        280        185        244        239
## 7     Ref    1        215        247        247        241        268
## 8     Ref    1         11          3         23         14          6
## 9     Ref    1       2172       2124       1874       1918       2247
## 10    Ref    1        213        198        192        211        252
##    output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
## 1            3           3           1           0           7
## 2         3965        1305        1947         998        1068
## 3           85         484         420         843         146
## 4          474         339         991        2241         640
## 5          357         159         524         442         124
## 6          325          77           4          53         144
## 7           44         176         327         207         137
## 8            0           0          16          23          13
## 9         2133          91        1548         988        1842
## 10         101          98          27         157          59
```

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

```r
rnaCol=8

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

new_params
```

```
## $dispFunc_input
## function (q)
## coefs[1] + coefs[2]/q
## <bytecode: 0x7fa04edd3c40>
## <environment: 0x7fa04edd0388>
## attr(,"coefficients")
##   asymptDisp    extraPois
## 0.0009491525 2.7615830416
## attr(,"fitType")
## [1] "parametric"
## attr(,"varLogDispEsts")
## [1] 0.6636758
## attr(,"dispPriorVar")
## [1] 0.25
##
## $dispFunc_output
## function (q)
## coefs[1] + coefs[2]/q
## <bytecode: 0x7fa04edd3c40>
## <environment: 0x7fa02bd39e98>
## attr(,"coefficients")
## asymptDisp  extraPois
##  0.5861787 18.4354124
## attr(,"fitType")
## [1] "parametric"
## attr(,"varLogDispEsts")
## [1] 0.8630406
## attr(,"dispPriorVar")
## [1] 0.25
##
## $inputProp
## function (v)
## .approxfun(x, y, v, method, yleft, yright, f, na.rm)
## <bytecode: 0x7fa051279ae8>
## <environment: 0x7fa051269a40>
##
## $transEff
## function (v)
## .approxfun(x, y, v, method, yleft, yright, f, na.rm)
## <bytecode: 0x7fa051279ae8>
## <environment: 0x7fa052597888>
##
## $sizeFactor_input
## input_rep1 input_rep2 input_rep3 input_rep4 input_rep5
##  1.0026323  1.0082816  0.9948114  1.0038644  1.0095945
##
## $sizeFactor_output
## output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
##   1.0587708   1.0368740   1.1565576   1.0232856   0.9580533
```

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        676        571        669        608        662
## 2     Ref    1         41         54         54         66         53
## 3     Ref    1        495        549        491        500        520
## 4     Ref    1       2110       2190       2201       2417       2057
## 5     Ref    1       3682       3532       3482       3878       3196
## 6     Ref    1       1268       1344       1356       1202       1303
## 7     Ref    1        953        907        958        874        993
## 8     Ref    1         13         12         14          6         11
## 9     Ref    1       1770       1885       1883       1963       1917
## 10    Ref    1          8         10          7          8         16
##    output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
## 1          491         162         390          38         153
## 2            3          77          10          74          27
## 3          166         125          75          41         145
## 4         1024        1925        1009         860        1119
## 5         1548        1623        2086         625        4684
## 6          210         959         751         185        2061
## 7         1371         536        2065          36          58
## 8            2           8           1           3           0
## 9          799        1157        2062         366         607
## 10           8          27           0           1          16
```

### 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          0          0          0          0          0
## 2      Ref    1          0          6          0          0          0
## 3      Ref    1         50         45         21         41         28
## 4      Ref    1         34         30         40         41         32
## 5      Ref    1          0          0          0          0          4
## 101    Mut    1        671        690        706        760        758
## 102    Mut    1         15         10          4          1          9
## 103    Mut    1         48         29         48         49         42
## 104    Mut    1         10          9         29         14         45
## 105    Mut    1          0          9          4          1         15
##     output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
## 1             0           0           0           0           0
## 2             0           0           0           0           0
## 3            29          18         148          19           6
## 4             3           3           0          15           5
## 5           0           0           1           0           0
## 101       943         143         510         167         353
## 102         2          23          26           6           0
## 103        25          59          11          28          39
## 104        22           1           4           7          21
## 105         3           1           0           1           6
```

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

Another way to specify the dispersion function is through `inputDispParam` and `outputDispParam`. These parameters are required if `inputDispFunc` and `outputDispFunc` are not provided. Each of them should give the three parameter estimates $(a, b, \sigma_d^2)$ for the dispersion function of the DNA input or RNA output counts across replicates. The three parameters correspond to $a$, $b$, and $\sigma_d^2$, which specify that the dispersion parameter is a lognormal distribution with mean $log(a + b/\mu)$ and sd $\sigma_d^2$, where $\mu$ is the mean of RNA count across the replicates.

```r
datt = sim_fixDepth(inputProp = new_params[[3]](runif(ntag * nsim * 2)),
    ntag, nsim, nrepIn, nrepOut, slope, inputDispParam = c(0, 4.37, 0.25),
    outputDispParam = c(0.54, 12, 0.25), sampleDepth = totalDepth)
### Thesea are default dispersion parameters, if none was specified.
datt[1:10, ]
```

```
##    allele simN input_rep1 input_rep2 input_rep3 input_rep4 input_rep5
## 1     Ref    1        244        506        339        390        392
## 2     Ref    1        421        480        480        427        493
## 3     Ref    1       1076       1000       1022        884        950
## 4     Ref    1        224        204        242        203        183
## 5     Ref    1       1139       1269       1315       1101       1224
## 6     Ref    1        693        608        602        686        700
## 7     Ref    1          9         19         34         19         20
## 8     Ref    1        530        672        510        745        470
## 9     Ref    1        124        198        166         99        146
## 10    Ref    1         20          9         26         10          6
##    output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
## 1          163         254          12         159         444
## 2           42          83         366         216         444
## 3         1354         639         605        1236         261
## 4           68         127          48         167          36
## 5          225          89         675         381         270
## 6          135         188          29         225         216
## 7            0          53          13           3           7
## 8          319         599         379         439         727
## 9          186          98         292           0         397
## 10           7           6           3          12           9
```

## 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 |

| Test | singleReplicate | OptionName |
|---|---|---|
| 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 |
| 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        244        506        339        390        392
## 2     Ref    1        421        480        480        427        493
## 3     Ref    1       1076       1000       1022        884        950
## 4     Ref    1        224        204        242        203        183
## 5     Ref    1       1139       1269       1315       1101       1224
## 6     Ref    1        693        608        602        686        700
## 7     Ref    1          9         19         34         19         20
## 8     Ref    1        530        672        510        745        470
## 9     Ref    1        124        198        166         99        146
## 10    Ref    1         20          9         26         10          6
##    output_rep1 output_rep2 output_rep3 output_rep4 output_rep5
## 1          163         254          12         159         444
## 2           42          83         366         216         444
## 3         1354         639         605        1236         261
## 4           68         127          48         167          36
## 5          225          89         675         381         270
## 6          135         188          29         225         216
## 7            0          53          13           3           7
## 8          319         599         379         439         727
## 9          186          98         292           0         397
## 10           7           6           3          12           9
```

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

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

```
results
```

```
## $MW
##    simN       res_MW
## 1     1 5.242590e-02
## 2     2 6.305289e-01
## 3     3 1.082509e-05
## 4     4 1.230055e-01
## 5     5 3.546299e-02
## 6     6 3.546299e-02
## 7     7 2.056767e-04
## 8      8 9.117972e-01
## 9      9 8.534283e-01
## 10    10 8.930698e-03
##
## $Matching
##    simN res_matching
## 1     1       0.1624
## 2     2       0.4515
## 3     3       0.0001
## 4     4       0.1976
## 5     5       0.6695
## 6     6       0.1574
## 7     7       0.0056
## 8     8       0.7269
## 9     9       0.0940
## 10   10       0.0804
##
## $Adaptive
##    combos res_adaptive
## 1       1 5.242590e-02
## 2       2 6.305289e-01
## 3       3 1.082509e-05
## 4       4 1.230055e-01
## 5       5 3.546299e-02
## 6       6 3.546299e-02
## 7       7 2.056767e-04
## 8       8 9.117972e-01
## 9       9 8.534283e-01
## 10     10 8.930698e-03
##
## $QuASAR
##    simN            Z   DNAprop      pvalue
## 2     1  0.075041434 0.1773892 0.940181745
## 4     2 -1.996806100 0.5661089 0.045846251
## 6     3  2.701267967 0.3428492 0.006907566
## 8     4  0.396336729 0.5668522 0.691856635
## 10    5 -0.007542702 0.4486909 0.993981852
## 12    6 -0.305665355 0.4490911 0.759859454
## 14    7  1.879770476 0.4863459 0.060139366
## 16    8  0.308372879 0.6239042 0.757798617
## 18    9 -0.630946413 0.8256984 0.528075563
## 20   10  2.264632300 0.4040725 0.023535253
##
## $T_test
##    simN ttest_paired        ttest
## 1     1 7.869952e-02 2.030177e-02
## 2     2 3.096812e-02 2.120233e-02
## 3     3 1.523131e-05 7.013296e-09
## 4     4 6.532243e-01 5.866101e-01
## 5     5 5.893858e-02 6.563350e-02
## 6     6 7.187900e-02 1.720496e-02
## 7     7 1.144461e-02 2.687863e-03
## 8     8 2.757534e-01 2.889281e-01
## 9     9 4.321435e-01 4.841897e-01
## 10   10 6.947869e-04 2.855830e-04
##
## $mpralm_mean
##         logFC    AveExpr          t      P.Value    adj.P.Val          B simN
## 3   4.0930594  1.4287221 13.4832311 4.546204e-12 4.546204e-11 17.7965725    3
## 10  1.4483103 -0.2429838  4.6427930 1.274874e-04 6.374371e-04  0.6332879   10
## 7   1.0663375 -0.6428265  3.6506658 1.420120e-03 4.733734e-03 -1.6756823    7
## 1  -0.6758440 -0.8738254 -3.0287100 6.200106e-03 1.428027e-02 -3.3724265    1
## 5  -0.8535956 -0.9021576 -2.9676657 7.140134e-03 1.428027e-02 -3.4596470    5
## 6  -0.5714637 -0.7782373 -2.4133123 2.464421e-02 4.107368e-02 -4.6696035    6
## 4   0.5231685 -1.2022048  1.8817239 7.326071e-02 1.046582e-01 -5.7218597    4
## 8  -0.3425102 -1.0736382 -0.9968414 3.297518e-01 3.882787e-01 -6.7138673    8
## 9   0.2465782 -1.2956969  0.8802937 3.882787e-01 3.882787e-01 -6.8251954    9
## 2   0.2396463 -0.8146115  0.8989807 3.784619e-01 3.882787e-01 -6.9842902    2
##
## $mpralm_sum
##         logFC     AveExpr          t      P.Value    adj.P.Val         B simN
## 3   4.1554451  1.89848125 19.1074694 7.968033e-16 7.968033e-15 26.297697    3
## 10  1.5977796  0.37375661  5.3396265 1.880092e-05 9.400462e-05  2.465553   10
## 6  -1.3580770 -0.46816484 -3.7462002 1.024429e-03 3.414765e-03 -1.469438    6
## 7   1.3440736 -0.09214839  3.5394784 1.708530e-03 4.271324e-03 -1.679263    7
## 2  -0.8579709 -0.34672234 -2.6786548 1.326511e-02 2.653022e-02 -3.872919    2
## 5  -0.8711105 -0.48310622 -2.4759766 2.089660e-02 2.985229e-02 -4.311064    5
## 1  -0.6108328 -0.45881583 -2.5215835 1.888799e-02 2.985229e-02 -4.409739    1
## 8  -0.5667890 -0.55921516 -1.4221417 1.681146e-01 2.101433e-01 -6.049639    8
## 9  -0.1703311 -0.83804930 -0.4465838 6.592613e-01 6.592613e-01 -6.794554    9
## 4  -0.1828624 -0.77570142 -0.6154234 5.441859e-01 6.046510e-01 -7.066168    4
##
## $DESeq2
## log2 fold change (MLE): group T vs A
## Wald test p-value: group T vs A
## DataFrame with 10 rows and 7 columns
##     baseMean log2FoldChange     lfcSE       stat       pvalue         padj
##    <numeric>      <numeric> <numeric>  <numeric>    <numeric>    <numeric>
## 1    7667.00       0.633226  0.216913   2.919267  3.50856e-03  6.62157e-03
## 2    9739.61       0.922484  0.285279   3.233623  1.22231e-03  3.05577e-03
## 3   30956.74      -4.146027  0.132663 -31.252430 2.06908e-214 2.06908e-213
## 4    7706.75       0.149553  0.276219   0.541428  5.88212e-01  6.33399e-01
## 5    8426.10       0.813312  0.363467   2.237653  2.52437e-02  3.60625e-02
## 6   10779.69       1.288785  0.447448   2.880302  3.97294e-03  6.62157e-03
## 7    6486.45      -1.284286  0.273520  -4.695396  2.66091e-06  8.86969e-06
## 8    5274.57       0.445096  0.449679   0.989808  3.22268e-01  4.02835e-01
## 9    8132.86       0.120636  0.252934   0.476949  6.33399e-01  6.33399e-01
## 10   7299.01      -1.573719  0.228740  -6.879933  5.98808e-12  2.99404e-11
##         simN
##    <integer>
## 1          1
## 2          2
## 3          3
## 4          4
## 5          5
## 6          6
## 7          7
## 8          8
## 9          9
## 10        10
##
## $resultAll
##    simN  resInput        resMW resMatching  resAdaptive      QuASAR
## 1     1 0.1654939 5.242590e-02      0.1624 5.242590e-02 0.940181745
## 2     2 0.7393644 6.305289e-01      0.4515 6.305289e-01 0.045846251
## 3     3 0.8534283 1.082509e-05      0.0001 1.082509e-05 0.006907566
## 4     4 0.4358722 1.230055e-01      0.1976 1.230055e-01 0.691856635
## 5     5 0.5787417 3.546299e-02      0.6695 3.546299e-02 0.993981852
## 6     6 1.0000000 3.546299e-02      0.1574 3.546299e-02 0.759859454
## 7     7 0.6842105 2.056767e-04      0.0056 2.056767e-04 0.060139366
## 8     8 0.9705125 9.117972e-01      0.7269 9.117972e-01 0.757798617
## 9     9 0.7393644 8.534283e-01      0.0940 8.534283e-01 0.528075563
## 10   10 0.7959363 8.930698e-03      0.0804 8.930698e-03 0.023535253
##    ttest_paired        ttest  mpralm_mean   mpralm_sum        DESeq2
## 1  7.869952e-02 2.030177e-02 6.200106e-03 1.888799e-02  3.508560e-03
## 2  3.096812e-02 2.120233e-02 3.784619e-01 1.326511e-02  1.222307e-03
## 3  1.523131e-05 7.013296e-09 4.546204e-12 7.968033e-16 2.069077e-214
## 4  6.532243e-01 5.866101e-01 7.326071e-02 5.441859e-01  5.882123e-01
## 5  5.893858e-02 6.563350e-02 7.140134e-03 2.089660e-02  2.524372e-02
## 6  7.187900e-02 1.720496e-02 2.464421e-02 1.024429e-03  3.972940e-03
## 7  1.144461e-02 2.687863e-03 1.420120e-03 1.708530e-03  2.660907e-06
## 8  2.757534e-01 2.889281e-01 3.297518e-01 1.681146e-01  3.222679e-01
## 9  4.321435e-01 4.841897e-01 3.882787e-01 6.592613e-01  6.333986e-01
## 10 6.947869e-04 2.855830e-04 1.274874e-04 1.880092e-05  5.988076e-12
```

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)
```

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

```
##        resMW ttest_paired        ttest  mpralm_mean   mpralm_sum        edgeR
##          0.2          0.0          0.0          0.4          0.3          0.5
##       DESeq2
##          0.6
```

```
result2$Power
```

```
##        resMW ttest_paired        ttest  mpralm_mean   mpralm_sum        edgeR
##          0.2          0.0          0.0          0.4          0.3          0.5
##       DESeq2
##          0.6
```

```
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)
```

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

```
##        resMW  resMatching  resAdaptive       QuASAR fisherPvalue ttest_paired
##          0.0          0.0          0.0          0.0          0.8          0.0
##        ttest  mpralm_mean   mpralm_sum        edgeR       DESeq2
##          0.0          0.0          0.0          0.0          0.0
```

```
result3$Power
```

```
##        resMW  resMatching  resAdaptive       QuASAR fisherPvalue ttest_paired
##          0.0          0.0          0.0          0.0          0.8          0.0
##        ttest  mpralm_mean   mpralm_sum        edgeR       DESeq2
##          0.0          0.0          0.0          0.0          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$resultAll)
```

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

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