# PowerExplorer Manual

*Xu Qiao, Laura Elo*

*2018-04-10*

## Contents

# Abstract

This vignette demonstrates R package `PowerExplorer` as a power and sample size estimation tool for RNA-Seq and quantitative proteomics data.

`PowerExplorer` contains the following main features:

- Estimation of power based on the current data
- Prediction of power corresponding to the increased sample sizes
- Result visualizations

# Introduction

Power and sample size estimation is one of the important principles in designing next-generation sequencing experiments to discover differential expressions. `PowerExplorer` is a power estimation and prediction tool currently applicable to RNA-Seq and quantitative proteomics experiments.

The calculation procedure starts with estimating the distribution parameters of each gene or protein. With the obtained prior distribution of each feature, a specified amount of simulations are executed to generate data (read counts for RNA-Seq and protein abundance for proteomics) repetitively for each entry based on null and alternative hypotheses. Furthermore, the corresponding statistical tests (t-test or Wald-test) are performed and the test statistics are collected. Eventually the statistics will be summarized to calculate the statistical power.

## Input Data Preparation

For both RNA-Seq (gene expression levels) and quantitative proteomics (protein abundance levels) datasets, the data matrix should be arranged as genes/proteins in rows and samples in columns. Here we show a RNA dataset as an example:

```
library(PowerExplorer)
data("exampleProteomicsData")
head(exampleProteomicsData$dataMatrix)
#>           Sample_A_1 Sample_A_2 Sample_A_3 Sample_A_4 Sample_A_5
#> Protein_1     888390     939871    1040976     668450    1008080
#> Protein_2    1159451    1171040     806536     808084     775754
#> Protein_3     996873    1041867     851849     872192     889652
#> Protein_4     730004     976224    1102569    1471498    1202674
#> Protein_5    1075502     832203    1006920    1412591    1282106
#> Protein_6    1274303    1136278    1146456    1028072     915348
#>           Sample_B_1 Sample_B_2 Sample_B_3 Sample_B_4 Sample_B_5
#> Protein_1    3093632    1837451    2760696    2012186    4359111
#> Protein_2    3052050    1079945    5479397     714079    1517852
#> Protein_3    3180328    3265779    2329892    4604491    4050628
#> Protein_4    2123667    1797917    1768735    1652113    2300376
#> Protein_5    1345427    2115544     776261    2704702    2035877
#> Protein_6    3957101    2438872    2844246    1134957    1366432
#>           Sample_C_1 Sample_C_2 Sample_C_3 Sample_C_4 Sample_C_5
#> Protein_1    3303614    4113386    3974813    3723468    2570479
#> Protein_2    4213595    9508269    4342755    5278913    4058501
#> Protein_3    5350451    5616083    5167265    5666823    3950354
#> Protein_4    4557161    1607928    2938020    3850669    3699732
#> Protein_5    3708197    2707825    2818278    3078232    3234728
#> Protein_6    1837780    3608012    4149520    2328815    8972975
```

A grouping vector indicating the sample groups to which all the samples belong should also be created, for example:

```
show(exampleProteomicsData$groupVec)
#>  [1] "A" "A" "A" "A" "A" "B" "B" "B" "B" "B" "C" "C" "C" "C" "C"
```

The sample groups corresponding to the data:

```
colnames(exampleProteomicsData$dataMatrix)
#>  [1] "Sample_A_1" "Sample_A_2" "Sample_A_3" "Sample_A_4" "Sample_A_5"
#>  [6] "Sample_B_1" "Sample_B_2" "Sample_B_3" "Sample_B_4" "Sample_B_5"
#> [11] "Sample_C_1" "Sample_C_2" "Sample_C_3" "Sample_C_4" "Sample_C_5"
```

Note that the grouping vector length should be equal to the column number of the data matrix.

## Power Estimation

Here we use a randomly generated Proteomics dataset `exampleProteomicsData` as an example to estimate the current power of the dataset. The input dataset is named as `dataMatrix` and the grouping vector as `groupVec`.

To run the estimation, apart from the input, we still need to specify the following parameters:

- `isLogTransformed`: FALSE; the input data is not log-transformed.
- `dataType`: "Proteomics"; the datatype can be declared as "Proteomics" or "RNA-Seq".
- `minLFC`: 0.5; the threshold of Log2 Fold Change, proteins with lower LFC will be discarded.
- `enableROTS`: TRUE; Using Reproducibility-Optimized Test Statistic (ROTS) as the statistical model.
- `paraROTS`: the parameters to be passed to ROTS (if enabled). Check ROTS documentation for further details on the parameters.
- `alpha`: 0.05; the controlled false positive (Type I Error) rate.
- `ST`: 50; the simulation of each gene will be run 50 times (ST>50 is recommended).
- `seed`: 345; optional, a seed value for the random number generator to maintain the reproducibility.
- `showProcess`: FALSE; no detailed processes will be shown, set to TRUE if debug is needed.
- `saveSimulatedData`: FALSE; if TRUE, save the simulated data in ./savedData directory.

The results will be summaried in barplot, boxplot and summary table.

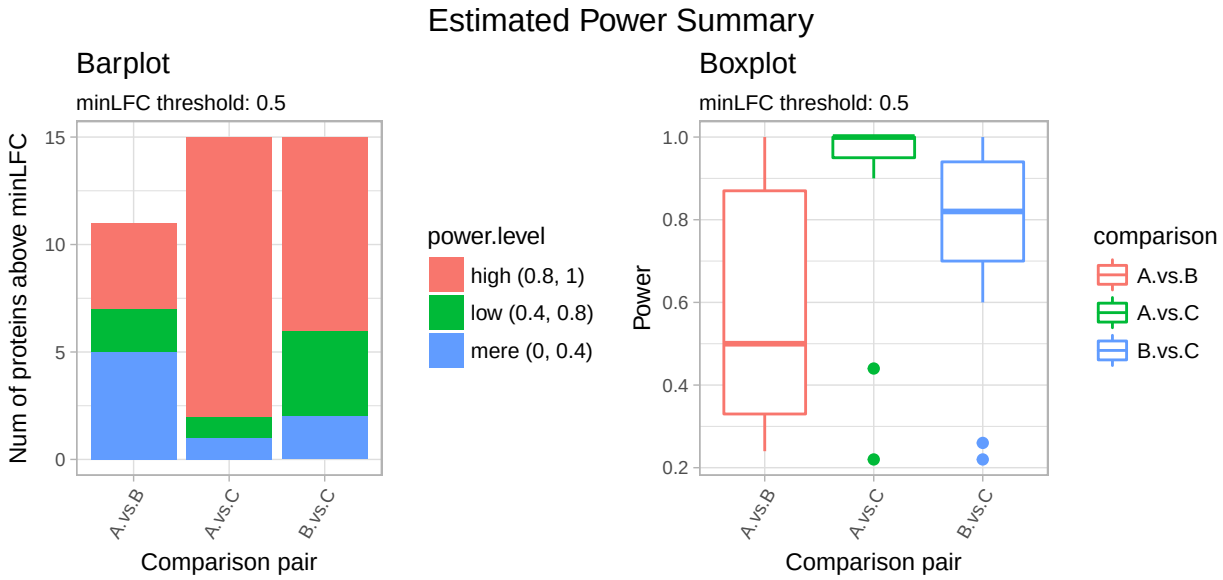
```
library(PowerExplorer)
data("exampleProteomicsData")
res <- estimatePower(inputObject = exampleProteomicsData$dataMatrix,
                     groupVec = exampleProteomicsData$groupVec,
                     isLogTransformed = FALSE,
                     dataType = "Proteomics",
                     minLFC = 0.5,
                     enableROTS = TRUE,
                     paraROTS = list(B = 1000, K = NULL),
                     alpha = 0.05,
                     ST = 50,
                     seed = 345,
                     showProcess = FALSE,
                     saveResultData = FALSE
                     )
#> ##------ Tue Apr 10 15:10:57 2018 ------##
#> Sample groups:            A, B, C
#> Num. of replicates:       5, 5, 5
#> Num. of simulations:      50
#> Min. Log Fold Change:         0.5
#> False Postive Rate:       0.05
#> Log-transformed:      FALSE
#> ROTS enabled:             TRUE
#> Parallel:             FALSE
#>
#> 0 of 110 entries are filtered due to excessive zero counts
#> [vsn] Transforming data...
#>
#>   ------ <A.vs.B> ------
#>
#> Log2 Fold Change Quantiles:
#>   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
#> 0.00 0.01 0.06 0.10 0.14 0.18 0.24 0.31 0.40 0.50 1.82
```

```
#>
#> [ROTS] Estimating statistics optimizing parameters...
#> [ROTS] optimization parameters:
#> a1 = 0.22, a2 = 1
#>
#> 11 of 110 proteins are over minLFC threshold 0.5.
#>
#>   ------ <A.vs.C> ------
#>
#> Log2 Fold Change Quantiles:
#>   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
#> 0.00 0.03 0.07 0.11 0.15 0.20 0.27 0.34 0.43 0.64 2.39
#>
#> [ROTS] Estimating statistics optimizing parameters...
#> [ROTS] optimization parameters:
#> a1 = 1, a2 = 1
#>
#> 15 of 110 proteins are over minLFC threshold 0.5.
#>
#>   ------ <B.vs.C> ------
#>
#> Log2 Fold Change Quantiles:
#>   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
#> 0.00 0.03 0.07 0.13 0.15 0.18 0.23 0.29 0.42 0.60 1.59
#>
#> [ROTS] Estimating statistics optimizing parameters...
#> [ROTS] optimization parameters:
#> a1 = 2.4, a2 = 1
#>
#> 15 of 110 proteins are over minLFC threshold 0.5.
#>
#> Simulation in process, it may take a few minutes...
#>
#> Power Estimation between groups A.vs.B:
#>
#> OVERALL ESTIMATED POWER: 0.5945
#>
#>
#> Simulation in process, it may take a few minutes...
#>
#> Power Estimation between groups A.vs.C:
#>
#> OVERALL ESTIMATED POWER: 0.8973
#>
#>
#> Simulation in process, it may take a few minutes...
#>
#> Power Estimation between groups B.vs.C:
#>
#> OVERALL ESTIMATED POWER: 0.76
```

## Visualization

The estimated results can be summarized using `plotEstPwr`, the only input needed is the `estimatedPower`, which should be the estimated power object returned from `estimatePower`.

```
plotEstPwr(res)
```




| Comp. | Protein Num. | Avg. Power | H (0.8, 1) | L (0.4, 0.8) | M (0, 0.4) |
|---|---|---|---|---|---|
| A.vs.B | 11 | 0.59 | 4 (36%) | 2 (18%) | 5 (45%) |
| A.vs.C | 15 | 0.90 | 13 (87%) | 1 (7%) | 1 (7%) |
| B.vs.C | 15 | 0.76 | 9 (60%) | 4 (27%) | 2 (13%) |

The graph contains 3 plots, the `barplot` vertically shows the number of genes/proteins above the minLFC threshold, columns indicates the comparison pairs, each column presents the proportions of three power levels in three colours as indicated in the legend `power.level`; The boxplot shows the overall power distribution of each comparsion; And the summary table summarize the power in a numerical way with the same information shown in the previous two plots.

## Result Summary

With the result `PowerExplorerStorage` object, summarized information can be shown by `show` method.

```
res
#> ##--Parameters--##
#> -dataType: Proteomics
#> -original repNum: 5, 5, 5
#> -Comparison groups: A, B, C
#> -False positive rate: 0.05
#> -LFC threshold: 0.5
#> -Simulations: 50
#>
#> ##--Log2 Fold Change Range--##
#>        A.vs.B A.vs.C B.vs.C
#> minLFC   0.00   0.00   0.00
#> maxLFC   1.82   2.39   1.59
#>
#> ##--Average Estimated Power--##
#>    A.vs.B    A.vs.C    B.vs.C
#> 0.5945455 0.8973333 0.7600000
#>
#> ##--Average Predicted Power--##
#> NA
```

If interested in specific genes/proteins or a ranking list, one can use `listEstPwr` with the following parameters:

- `inputObject`: A PowerExplorerStorage returned from PowerExplorer as input
- `decreasing`: logical; TRUE, decreasing order; FALSE, increasing order.
- `top`: numeric; the number of genes/proteins in the top list
- `selected`: default as NA; specify as a list of geneID or protein ID to show power of a list of interested genes/proteins.

To show the top 10 genes in an example result object `exampleObject` in decreasing order:

```
data(exampleObject)
listEstPwr(exampleObject, decreasing = TRUE, top = 10)
#>                    A.vs.B
#> ENSMUSG00000000402      1
#> ENSMUSG00000000958      1
#> ENSMUSG00000001473      1
#> ENSMUSG00000003477      1
#> ENSMUSG00000004341      1
#> ENSMUSG00000006154      1
#> ENSMUSG00000006403      1
#> ENSMUSG00000007035      1
#> ENSMUSG00000015484      1
#> ENSMUSG00000015852      1
```

To show the results of specific genes:

```
listEstPwr(exampleObject,
           selected = c("ENSMUSG00000000303",
                        "ENSMUSG00000087272",
                        "ENSMUSG00000089921"))
#>                    A.vs.B
#> ENSMUSG00000000303   0.34
```

```
#> ENSMUSG00000087272   0.25
#> ENSMUSG00000089921   0.04
```

## Power Predictions

With the same dataset, to run a prediction, a different parameter is needed:

- `rangeSimNumRep`: the power of replicate number 5 to 20 will be predicted.

Similar to the estimation process, however, the simulations will be excuted with each sample size specified in `rangeSimNumRep`. (Note: the term sample size in this vignette refers to the replicate number of each group/case)

It is possible to append the prediction results within the same object by using the same result object as an input.

```
data("exampleProteomicsData")
res <- predictPower(inputObject = res,
                    groupVec = exampleProteomicsData$groupVec,
                    isLogTransformed = FALSE,
                    dataType = "Proteomics",
                    minLFC = 0.5,
                    rangeSimNumRep = c(5, 10, 15, 20),
                    enableROTS = TRUE,
                    paraROTS = list(B = 1000, K = NULL),
                    alpha = 0.05,
                    ST = 50,
                    seed = 345)
#> ##------ Tue Apr 10 15:11:17 2018 ------##
#> Sample groups:    A, B, C
#> Replicates of prediction:     5, 10, 15, 20
#> Num. of simulations:  50
#> Min. Log Fold Change:     0.5
#> False Postive Rate:   0.05
#> Transformed:  FALSE
#> ROTS enabled:             TRUE
#> Parallel:             FALSE
#>
#> 0 of 110 entries are filtered due to excessive zero counts
#> [vsn] Transforming data...
#>
#>   ------ <A.vs.B> ------
#>
#> Log2 Fold Change Quantiles:
#>   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
#> 0.00 0.01 0.06 0.10 0.14 0.18 0.24 0.31 0.40 0.50 1.82
#>
#> [ROTS] Estimating statistics optimizing parameters...
#> [ROTS] optimization parameters:
#> a1 = 0.22, a2 = 1
#>
#> 11 of 110 proteins are over minLFC threshold 0.5.
#>
#>   ------ <A.vs.C> ------
#>
#> Log2 Fold Change Quantiles:
#>   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
#> 0.00 0.03 0.07 0.11 0.15 0.20 0.27 0.34 0.43 0.64 2.39
```

```
#>
#> [ROTS] Estimating statistics optimizing parameters...
#> [ROTS] optimization parameters:
#> a1 = 1, a2 = 1
#>
#> 15 of 110 proteins are over minLFC threshold 0.5.
#>
#>   ------ <B.vs.C> ------
#>
#> Log2 Fold Change Quantiles:
#>   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
#> 0.00 0.03 0.07 0.13 0.15 0.18 0.23 0.29 0.42 0.60 1.59
#>
#> [ROTS] Estimating statistics optimizing parameters...
#> [ROTS] optimization parameters:
#> a1 = 2.4, a2 = 1
#>
#> 15 of 110 proteins are over minLFC threshold 0.5.
#>
#> ##--Simulation with 5 replicates per group--##
#>
#> [repNum:5] Simulation in process, it may take a few minutes...
#>
#> [repNum:5] Power Estimation between groups A.vs.B:
#>
#> OVERALL ESTIMATED POWER: 0.6073
#>
#>
#> [repNum:5] Simulation in process, it may take a few minutes...
#>
#> [repNum:5] Power Estimation between groups A.vs.C:
#>
#> OVERALL ESTIMATED POWER: 0.8973
#>
#>
#> [repNum:5] Simulation in process, it may take a few minutes...
#>
#> [repNum:5] Power Estimation between groups B.vs.C:
#>
#> OVERALL ESTIMATED POWER: 0.6867
#>
#>
#> ##--Simulation with 10 replicates per group--##
#>
#> [repNum:10] Simulation in process, it may take a few minutes...
#>
#> [repNum:10] Power Estimation between groups A.vs.B:
#>
#> OVERALL ESTIMATED POWER: 0.8509
#>
#>
#> [repNum:10] Simulation in process, it may take a few minutes...
#>
```

```
#> [repNum:10] Power Estimation between groups A.vs.C:
#>
#> OVERALL ESTIMATED POWER: 0.968
#>
#>
#> [repNum:10] Simulation in process, it may take a few minutes...
#>
#> [repNum:10] Power Estimation between groups B.vs.C:
#>
#> OVERALL ESTIMATED POWER: 0.8667
#>
#>
#> ##--Simulation with 15 replicates per group--##
#>
#> [repNum:15] Simulation in process, it may take a few minutes...
#>
#> [repNum:15] Power Estimation between groups A.vs.B:
#>
#> OVERALL ESTIMATED POWER: 0.9036
#>
#>
#> [repNum:15] Simulation in process, it may take a few minutes...
#>
#> [repNum:15] Power Estimation between groups A.vs.C:
#>
#> OVERALL ESTIMATED POWER: 0.9867
#>
#>
#> [repNum:15] Simulation in process, it may take a few minutes...
#>
#> [repNum:15] Power Estimation between groups B.vs.C:
#>
#> OVERALL ESTIMATED POWER: 0.94
#>
#>
#> ##--Simulation with 20 replicates per group--##
#>
#> [repNum:20] Simulation in process, it may take a few minutes...
#>
#> [repNum:20] Power Estimation between groups A.vs.B:
#>
#> OVERALL ESTIMATED POWER: 0.9855
#>
#>
#> [repNum:20] Simulation in process, it may take a few minutes...
#>
#> [repNum:20] Power Estimation between groups A.vs.C:
#>
#> OVERALL ESTIMATED POWER: 0.9973
#>
#>
#> [repNum:20] Simulation in process, it may take a few minutes...
#>
```

```
#> [repNum:20] Power Estimation between groups B.vs.C:
#>
#> OVERALL ESTIMATED POWER: 0.9693
```

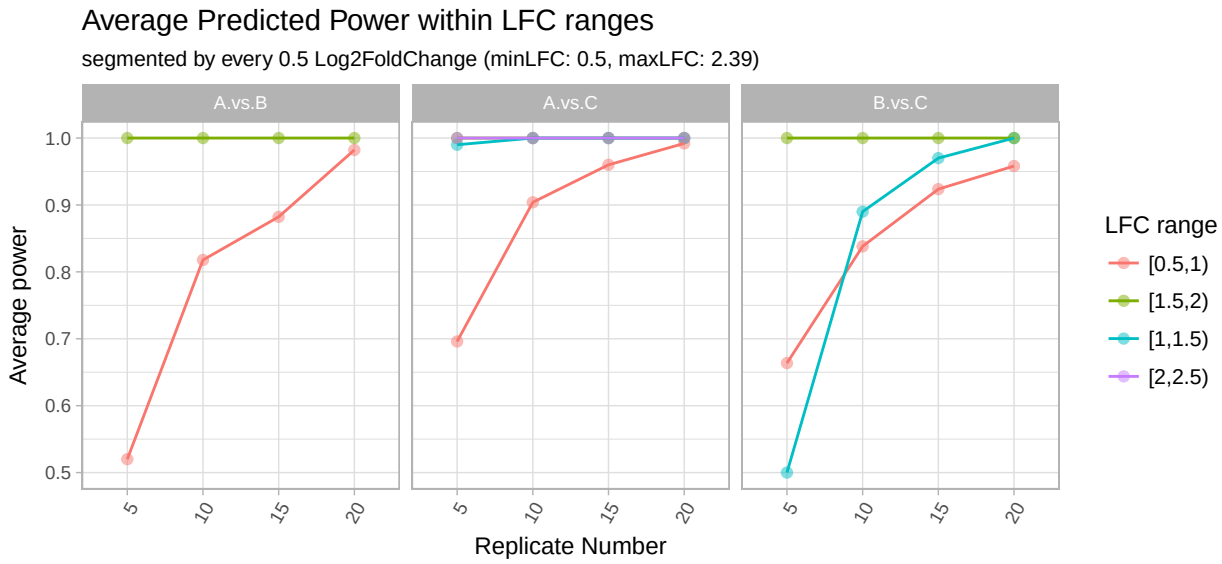
## Visualization

The predicted results can be summaried using `plotPredPwr`. The input should be the predicted power object returned from `predictPower`, the summary can be optionally visualized by setting the following parameters:

- `inputObject`: A PowerExplorerStorage returned from PowerExplorer as input
- `minLFC` and `maxLFC`: to observe power in a specific range of LFC
- `LFCscale`: to determine the LFC scale of the observation

Lineplot (LFCscale = 0.5):

```
plotPredPwr(res, LFCscale = 0.5)
```




| Comp. | repNum:5 | repNum:10 | repNum:15 | repNum:20 |
|---|---|---|---|---|
| A.vs.B | 0.61 | 0.85 | 0.9 | 0.99 |
| A.vs.C | 0.9 | 0.97 | 0.99 | 1 |
| B.vs.C | 0.69 | 0.87 | 0.94 | 0.97 |

The output figure contains a lineplot and a summary table. For each comparison, the lineplot shows the power tendency across every Log2 Fold Change segment resulted from a complete LFC list divided by a specified `LFCscale`. Each dot on the lines represents the average power (y-axis) of the genes/proteins at a certain sample size (x-axis) within different LFC ranges. In addition, a summary table below displays the average power of each comparison across the sample sizes.

For instance, the line plot here shows the average power at four different sample sizes (5 to 30, with increment of 5) in LFCscale of 0.5. The LFC ranges from 0 to 5, and within each LFC segment, the graph shows the average power of the genes/proteins. Here, the higher LFC shows higher power, the average power of each LFC range increases with the larger sample sizes, as expected.

## Result Summary

With the result `PowerExplorerStorage` object, summarized information can be shown by `show` method. Both estimated and predicted results can be summaried.

```
res
#> ##--Parameters--##
#> -dataType: Proteomics
#> -original repNum: 5, 5, 5
#> -Comparison groups: A, B, C
#> -False positive rate: 0.05
#> -LFC threshold: 0.5
#> -Simulations: 50
#>
#> ##--Log2 Fold Change Range--##
#>        A.vs.B A.vs.C B.vs.C
#> minLFC   0.00   0.00   0.00
#> maxLFC   1.82   2.39   1.59
#>
#> ##--Average Estimated Power--##
#>    A.vs.B    A.vs.C    B.vs.C
#> 0.5945455 0.8973333 0.7600000
#>
#> ##--Average Predicted Power--##
#> $`repNum: 5`
#>    A.vs.B    A.vs.C    B.vs.C
#> 0.6072727 0.8973333 0.6866667
#>
#> $`repNum: 10`
#>    A.vs.B    A.vs.C    B.vs.C
#> 0.8509091 0.9680000 0.8666667
#>
#> $`repNum: 15`
#>    A.vs.B    A.vs.C    B.vs.C
#> 0.9036364 0.9866667 0.9400000
#>
#> $`repNum: 20`
#>    A.vs.B    A.vs.C    B.vs.C
#> 0.9854545 0.9973333 0.9693333
```

If interested in specific genes/proteins or a ranking list of predicted powerat each sample size, one can use `listPrePwr` with the following parameters:

- `inputObject`: A PowerExplorerStorage returned from PowerExplorer as input
- `decreasing`: logical; TRUE, decreasing order; FALSE, increasing order.
- `top`: numeric; the number of genes/proteins in the top list
- `selected`: default as NA; specify as a list of geneID or protein ID to show power of a list of interested genes/proteins.

To show the top 10 genes in an example result object `exampleObject` in decreasing order at each sample size:

```
listPredPwr(exampleObject, decreasing = TRUE, top = 10)
#> $`repNum: 10`
#>                     A.vs.B
#> ENSMUSG00000000402      1
#> ENSMUSG00000000958      1
```

```
#> ENSMUSG00000001473      1
#> ENSMUSG00000003477      1
#> ENSMUSG00000004341      1
#> ENSMUSG00000005553      1
#> ENSMUSG00000006154      1
#> ENSMUSG00000006403      1
#> ENSMUSG00000007035      1
#> ENSMUSG00000011263      1
#>
#> $`repNum: 15`
#>                     A.vs.B
#> ENSMUSG00000000402      1
#> ENSMUSG00000000958      1
#> ENSMUSG00000001473      1
#> ENSMUSG00000001493      1
#> ENSMUSG00000003477      1
#> ENSMUSG00000004341      1
#> ENSMUSG00000005553      1
#> ENSMUSG00000005681      1
#> ENSMUSG00000006154      1
#> ENSMUSG00000006403      1
#>
#> $`repNum: 20`
#>                     A.vs.B
#> ENSMUSG00000000402      1
#> ENSMUSG00000000958      1
#> ENSMUSG00000001473      1
#> ENSMUSG00000001493      1
#> ENSMUSG00000003477      1
#> ENSMUSG00000004341      1
#> ENSMUSG00000004709      1
#> ENSMUSG00000005553      1
#> ENSMUSG00000005681      1
#> ENSMUSG00000006154      1
#>
#> $`repNum: 30`
#>                     A.vs.B
#> ENSMUSG00000000402      1
#> ENSMUSG00000000958      1
#> ENSMUSG00000001473      1
#> ENSMUSG00000001493      1
#> ENSMUSG00000003355      1
#> ENSMUSG00000003477      1
#> ENSMUSG00000004341      1
#> ENSMUSG00000004709      1
#> ENSMUSG00000005553      1
#> ENSMUSG00000005677      1
```

To show the results of specific genes at each sample size:

```
listPredPwr(exampleObject,
            selected = c("ENSMUSG00000000303",
                         "ENSMUSG00000087272",
                         "ENSMUSG00000089921"))
```

```
#> $`repNum: 10`
#>                     A.vs.B
#> ENSMUSG00000000303   0.28
#> ENSMUSG00000087272   0.16
#> ENSMUSG00000089921   0.01
#>
#> $`repNum: 15`
#>                     A.vs.B
#> ENSMUSG00000000303   0.39
#> ENSMUSG00000087272   0.17
#> ENSMUSG00000089921   0.00
#>
#> $`repNum: 20`
#>                     A.vs.B
#> ENSMUSG00000000303   0.37
#> ENSMUSG00000087272   0.35
#> ENSMUSG00000089921   0.02
#>
#> $`repNum: 30`
#>                     A.vs.B
#> ENSMUSG00000000303   0.64
#> ENSMUSG00000087272   0.56
#> ENSMUSG00000089921   0.01
```

## Parallel computation

The calculation may take much longer time when an input dataset contains more than thousands of features, especially for the power prediction process. The computational time can be significantly shortened by using parallelised computation, and the simulations will be distributed to multiple cores. This can be done by loading Bioconductor pacakge BiocParallel and then set the following arguments of `estimatePower` and `predictPower`: `parallel=TRUE` and `BPPARAM=bpparam()`. This will distribute the jobs to all the available cores. One can register the number of cores to be used by setting `BPPARAM=MulticoreParam(4)`, for instance, distributing simulations (jobs) to 4 cores. However, `MulticoreParam()` only supports non-Windows platforms. For Windows platforms, one can use `SnowParam()` instead. For further details, please check the BiocParallel documentation.