The quantroSim user's guide

Stephanie C. Hicks shicks@jimmy.harvard.edu
Rafael A. Irizarry rafa@jimmy.harvard.edu

Modified: November 24, 2014. Compiled: November 24, 2014

Contents

1 Introduction

This quantroSim package is the supporting data simulation package for the R/Bioconductor package quantro. This R package is designed to simulate gene expression and DNA methylation data. This document describes the classes, functions and tools available in the quantroSim package.

The features in this package include:

1. Simulate gene expression samples based on microarrays
2. Simulate DNA methylation samples based on microarrays
3. Control the proportion of differences (pDiff) between K groups
4. Vary the magnitude of technical variation observed in samples

2 Getting Started

To install the package, you can check out the Github repository https://github.com/stephaniehicks/quantroSim and install from source or use the devtools R package:

```
library(devtools)
install_github(repo = "quantroSim", username = "stephaniehicks")
```

After installation, load the package in R using

```
library(quantroSim)
```

The quantroSim package depends the MASS, quantro, minfi and affy R-packages and suggests the knitr R-package.

3 DNA Methylation

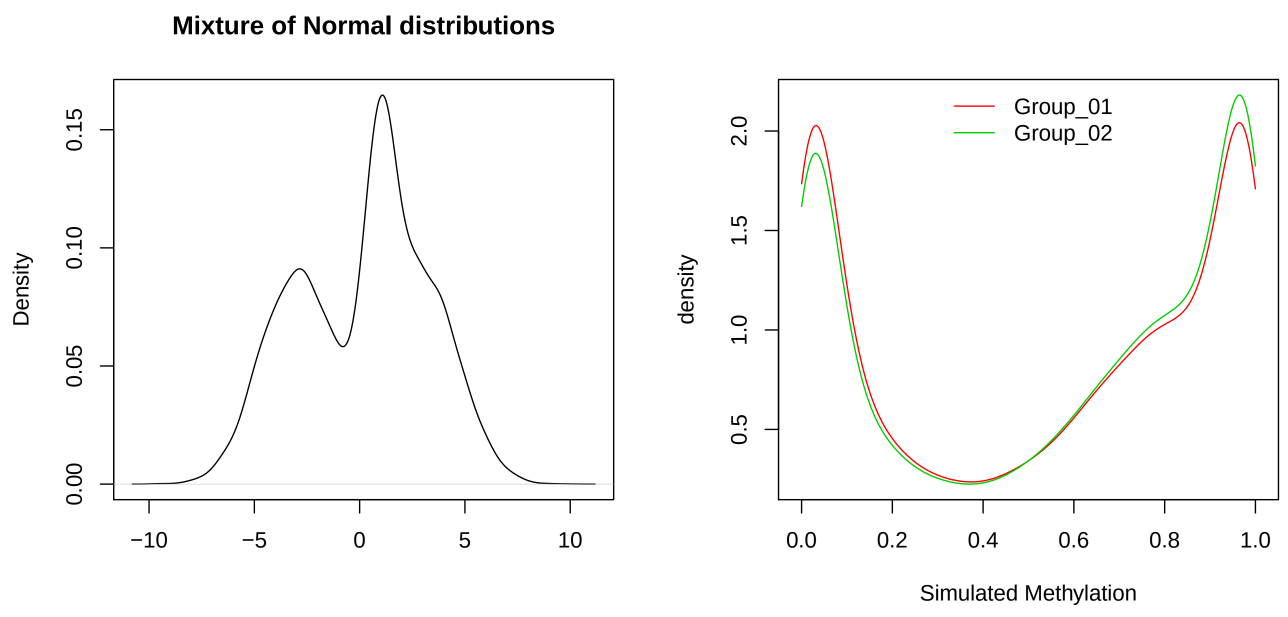
There are two main functions used to generate simulated DNA methylation data: simulateMethTruth and simulateMeth. The first function (simulateMethTruth) generates the true DNA methylation without any consideration for a platform technology. The second function (simulateMeth) simulates observed DNA methylation based on:

1. the platform technology
2. the magnitude of technical variation

3.1 Quick Start

To simulate the true level DNA methylation for a set of 2 groups, use the simulateMethTruth function.

```
set.seed(999)
methTruth <- simulateMethTruth(nProbes = 2e4, nGroups = 2,
                               pDiff = 0.05, pUp = 0.80)

## [quantroSim]:  Simulating a mixture of 3 Normal distributions
##              with mean (-3, 1, 3) and standard deviation (3, 0.4, 3)

plotMethTruth(methTruth)
```



pDiff is percent of probes different relative to Group 1. If nGroups = 1, pDiff should be 0. If nGroups > 1, the length of pDiff should be equal to nGroups - 1. The default for nGroups is 2 and the default for pDiff is 0.05.

Similarly, pUp is proportion of pDiff probes that are methylated relative to Group 1. If nGroups = 1, pUp is ignored. If nGroups > 1, the length of pUp should be equal to nGroups - 1. The default for nGroups is 2 and the default for pUp is 0.80.

The main output will be a matrix (methRange) of dimension nProbes × nGroups.

```
dim(methTruth$methRange)

## [1] 20000     2
```

The correlation between the two groups is given by:

```
cor(methTruth$methRange)

##            Group_01  Group_02
## Group_01 1.0000000 0.9040011
## Group_02 0.9040011 1.0000000
```

If pDiff was given, there will be pDiff × nProbes differences between the two groups. A boolean vector referring to which probes are different is in the methTruth object called methDiffInd. Here we list the indicies of which probes are different between the groups:

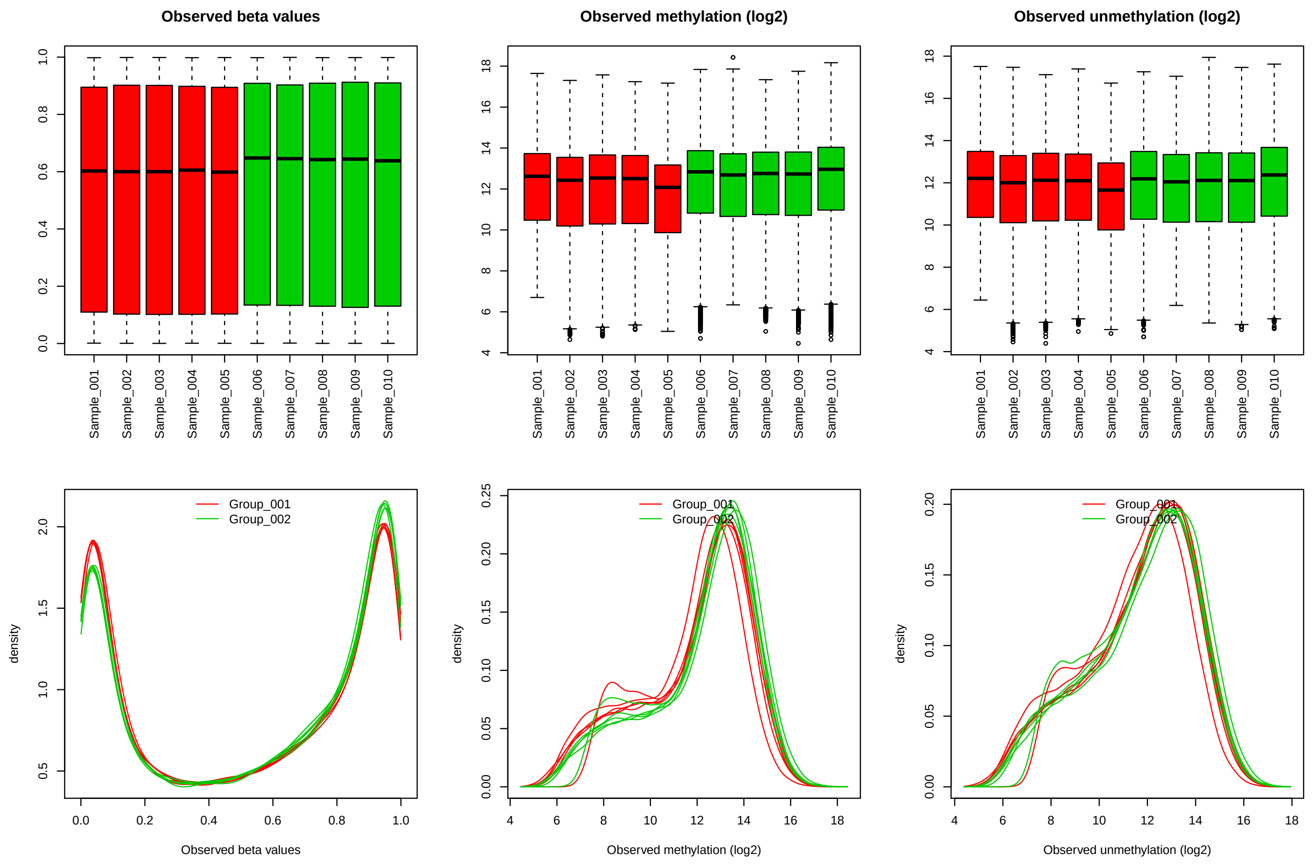
```
head(which(methTruth$methDiffInd))

## [1]  53  70  86 180 186 196
```

To simulate observed DNA methylation data based on a specific technology platform, use the simulateMeth function. First, a platform from list.meth.platforms must be selected:

```
list.meth.platforms()

## [1] "methArrays"
```

Once a platform has been selected,

```
set.seed(999)
simMeth <- simulateMeth(methTruth,  meth.platform = "methArrays",
                        nSamps = 5, nMol = 1e6)

## Simulating DNA methylation samples using the meth.platform:  methArrays

plotMeth(simMeth)
```

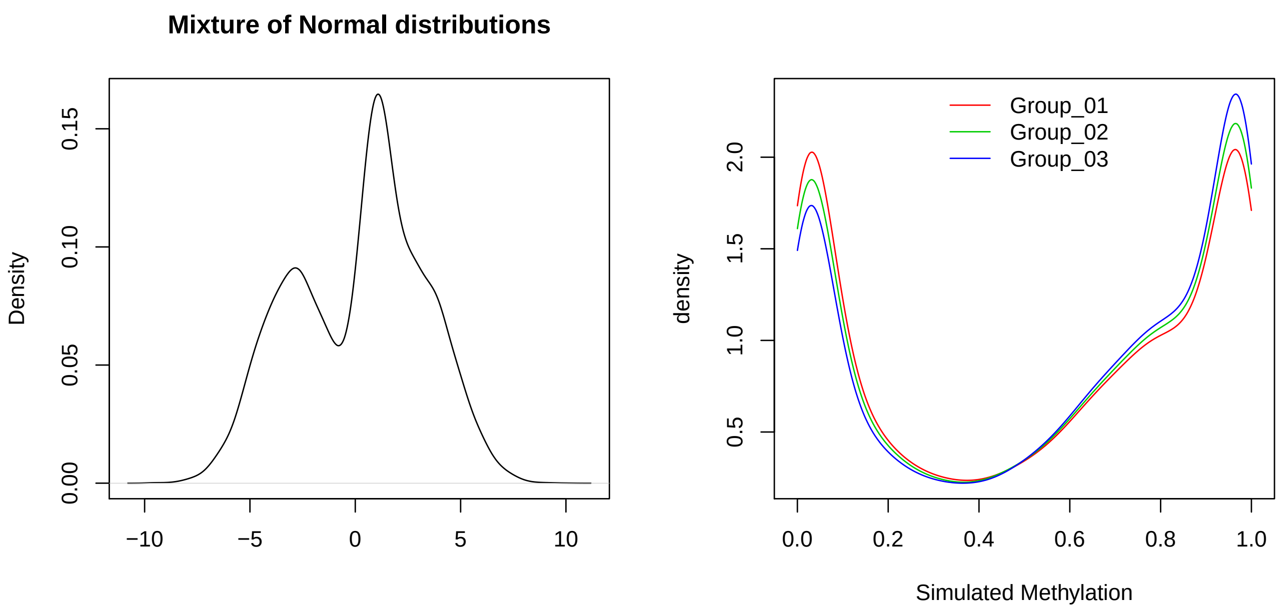


```
summary(simMeth$meth)

##    Sample_001        Sample_002        Sample_003        Sample_004        Sample_005    
##  Min.   :   104   Min.   :    25   Min.   :    28   Min.   :    35   Min.   :    33  
##  1st Qu.:  1424   1st Qu.:  1171   1st Qu.:  1253   1st Qu.:  1270   1st Qu.:   934  
##  Median :  6293   Median :  5507   Median :  5934   Median :  5817   Median :  4329  
##  Mean   : 10007   Mean   :  8692   Mean   :  9398   Mean   :  9207   Mean   :  6786  
##  3rd Qu.: 13572   3rd Qu.: 11944   3rd Qu.: 12963   3rd Qu.: 12702   3rd Qu.:  9239  
##  Max.   :204706   Max.   :161666   Max.   :194935   Max.   :154970   Max.   :147551  
##    Sample_006        Sample_007        Sample_008        Sample_009        Sample_010    
##  Min.   :    26   Min.   :    81   Min.   :    33   Min.   :    22   Min.   :    25  
##  1st Qu.:  1809   1st Qu.:  1619   1st Qu.:  1721   1st Qu.:  1678   1st Qu.:  2006
##  Median :  7322   Median :  6586   Median :  6912   Median :  6800   Median :  7972
##  Mean   : 10916   Mean   :  9978   Mean   : 10459   Mean   : 10450   Mean   : 12273
##  3rd Qu.: 14979   3rd Qu.: 13488   3rd Qu.: 14250   3rd Qu.: 14307   3rd Qu.: 16797
##  Max.   :234445   Max.   :352368   Max.   :165850   Max.   :220224   Max.   :294473
```

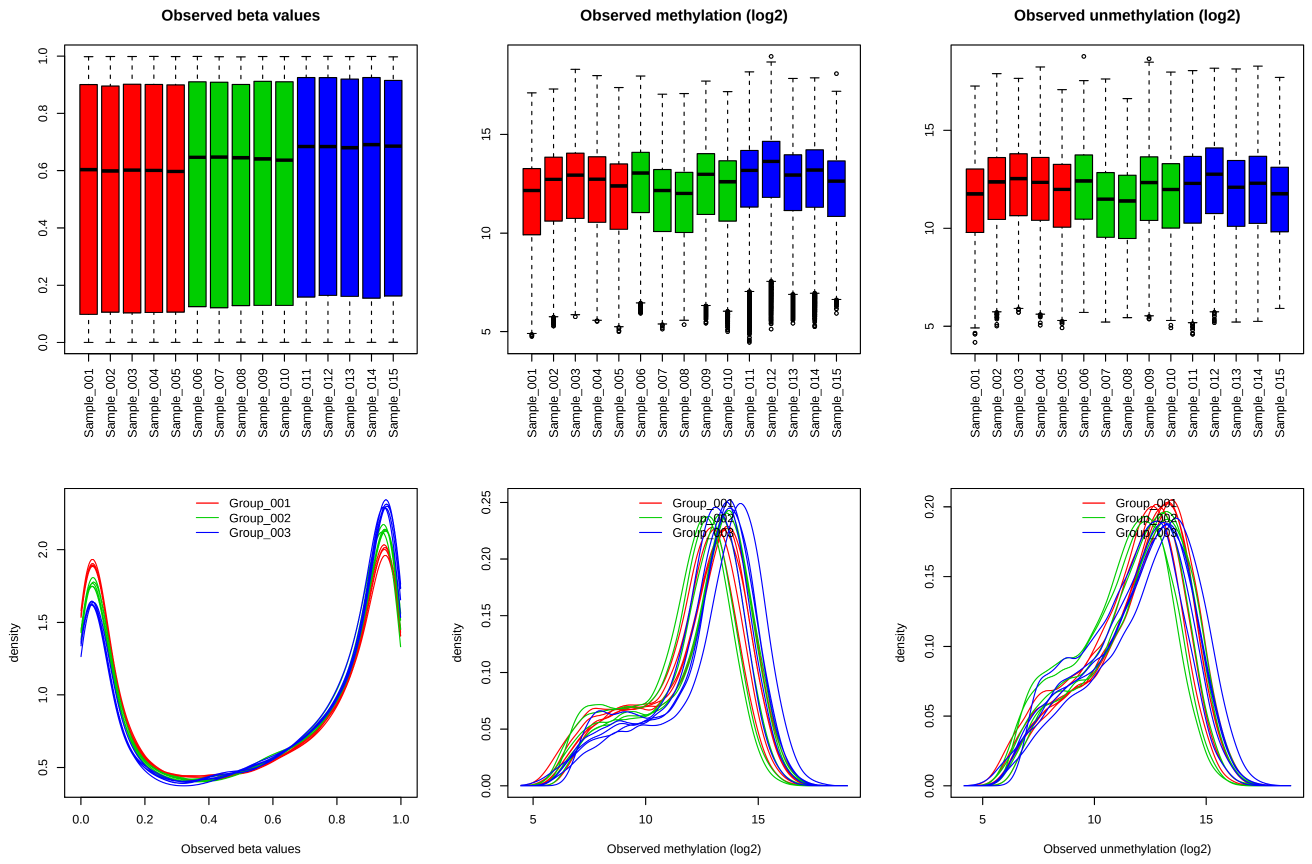
3.2 Simulating 2 or more groups

To simulate the true level DNA methylation for a set of 2 or more groups, again use the the same `simulateMethTruth` function, but change `nGroup` and the length of `pDiff` and `pUp`

```
set.seed(999)
methTruth <- simulateMethTruth(nProbes = 2e4, nGroups = 3,
                               pDiff = c(0.05, 0.10), pUp = c(0.80, 0.80))

## [quantroSim]:  Simulating a mixture of 3 Normal distributions
##              with mean (-3, 1, 3) and standard deviation (3, 0.4, 3)

plotMethTruth(methTruth)
```



```
set.seed(999)
simMeth <- simulateMeth(methTruth,  meth.platform = "methArrays",
                        nSamps = 5, nMol = 1e6)

## Simulating DNA methylation samples using the meth.platform:  methArrays

plotMeth(simMeth)
```



3.3 Exporting DNA Methylation arrays to the `minfi` R-package

To export the simulated DNA methylation object to `mini`, use the `getMethylSet` function.

```
mset <- getMethylSet(simMeth)
class(mset)

## [1] "MethylSet"
## attr(,"package")
## [1] "minfi"

head(minfi::getBeta(mset))

##      Sample_001  Sample_002 Sample_003 Sample_004 Sample_005 Sample_006 Sample_007
## [1,] 0.00107200 0.005681153 0.02104874 0.01329519 0.00737188 0.41431311  0.7955727
## [2,] 0.60861911 0.276913189 0.70446647 0.07069987 0.37629124 0.56325955  0.2163530
## [3,] 0.07482797 0.095685249 0.01816717 0.05761364 0.09079338 0.03481271  0.1938120
## [4,] 0.70322125 0.720787152 0.83043679 0.63708213 0.54945055 0.59193881  0.8029639
## [5,] 0.99486574 0.993225380 0.99319632 0.99639552 0.97999524 0.99397809  0.9817677
## [6,] 0.23069554 0.070011669 0.02212173 0.11058865 0.05687072 0.24056654  0.3724013
##      Sample_008 Sample_009 Sample_010 Sample_011  Sample_012  Sample_013  Sample_014
## [1,]  0.7364314 0.91680635 0.77712990 0.02488748 0.002076259 0.005362042 0.004826758
## [2,]  0.8182779 0.52998079 0.39585037 0.17606363 0.120002272 0.663642288 0.300044929
## [3,]  0.0851244 0.01245816 0.06346039 0.02830494 0.207956104 0.036411478 0.004687271
## [4,]  0.3784451 0.55584783 0.76532300 0.64324741 0.754493350 0.759291671 0.719616451
## [5,]  0.9953099 0.99766246 0.96271244 0.99657460 0.996118589 0.991937581 0.991931844
## [6,]  0.1335530 0.19840104 0.08672005 0.09554162 0.531417351 0.059207410 0.167966442
##       Sample_015
## [1,] 0.01559584
## [2,] 0.20763109
## [3,] 0.03816986
## [4,] 0.74108434
## [5,] 0.98429578
## [6,] 0.35855504
```

Functions in the `minfi` R/Bioconductor package such as `getBeta`, `getM`, `getCN` can be used after creating a `MethylSet` with the function `getMethylSet`.

Note: there is no manifest and no method was used to preprocess the simulated data. Therefore, these functions from `minfi` will not work.

```
getManifest(mset)
preprocessMethod(mset)
```

3.4 Additional options for `simulateMeth`

3.4.1 Controlling level of technical variation

We use the Langmuir model to simulate chemical saturation observed using microarrays. Our model to simulate raw methylation and unmethylation value for the j^{th} probe from the i^{th} sample in the k^{th} group is given by

$$M_{ijk} = o_{ijk} + d_{ijk} + a_{ijk}(\frac{x^m_{jk}}{x^m_{jk} + b_{ijk}})\epsilon_{ijk}$$

$$U_{ijk} = o_{ijk} + d_{ijk} + a_{ijk}(\frac{x^u_{jk}}{x^u_{jk} + b_{ijk}})\epsilon_{ijk}$$

where x^m_{jk} and x^u_{jk} are the expected number of methylated and unmethylated molecules at j^{th} probe in the k^{th} group and the rest are parameters simulated from a log Normal distribution with a given set of hyperparameters. For example, $a_{ijk} = a_{ik} * a_j$ represents the florescence intensity from the scanner. We define $a_{ijk} = a_{ik} * a_j$ and let both parameters a_{ik} (sample-level noise) and a_j (probe-level noise) each have their own hyperparameters to allow for global shifts:

$$\log_2(a_{ik}) \sim N(16, 0.1)$$

$$\log_2(a_j) \sim N(0, 0.01)$$

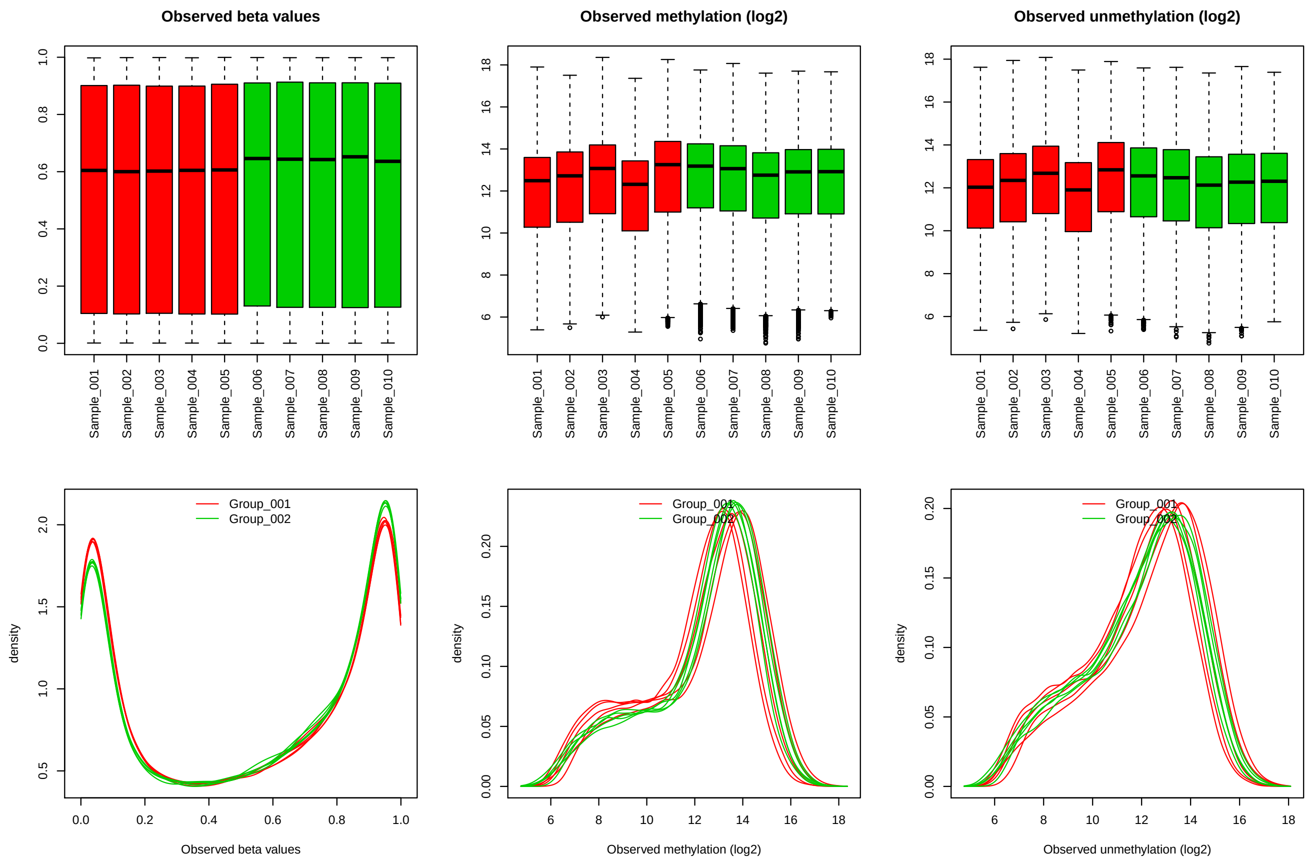
Similarly, $b_{ijk} = b_{ik} * b_j$ and $o_{ijk} = o_{ik} * o_j$ (optical noise) where the sample-level noise is simulated using

$$\log_2(b_{ik}) \sim N(22, 0.1)$$

$$\log_2(o_{ik}) \sim N(5, 1)$$
$$\log_2(d_{ijk}) \sim N(5, 1)$$
$$\log_2(\epsilon_{ijk}) \sim N(0, 1)$$

For efficiency, we simulate the parameters from a multivariate normal distribution for all 10 arrays (=5 samples per group * 2 groups). In the above example, covariance matrices would be given by:

```
set.seed(999)
siga = sigb = 0.1 * diag(10)
sigOpt = 1 * diag(10)

methTruth <- simulateMethTruth(nProbes = 2e4, nGroups = 2,
                               pDiff = 0.05, pUp = 0.80)

## [quantroSim]:  Simulating a mixture of 3 Normal distributions
##             with mean (-3, 1, 3) and standard deviation (3, 0.4, 3)

simMeth <- simulateMeth(methTruth,  meth.platform = "methArrays",
                        nSamps = 5, nMol = 1e6,
                        siga = siga, sigb = sigb, sigOpt = sigOpt)

## Simulating DNA methylation samples using the meth.platform:  methArrays

plotMeth(simMeth)
```

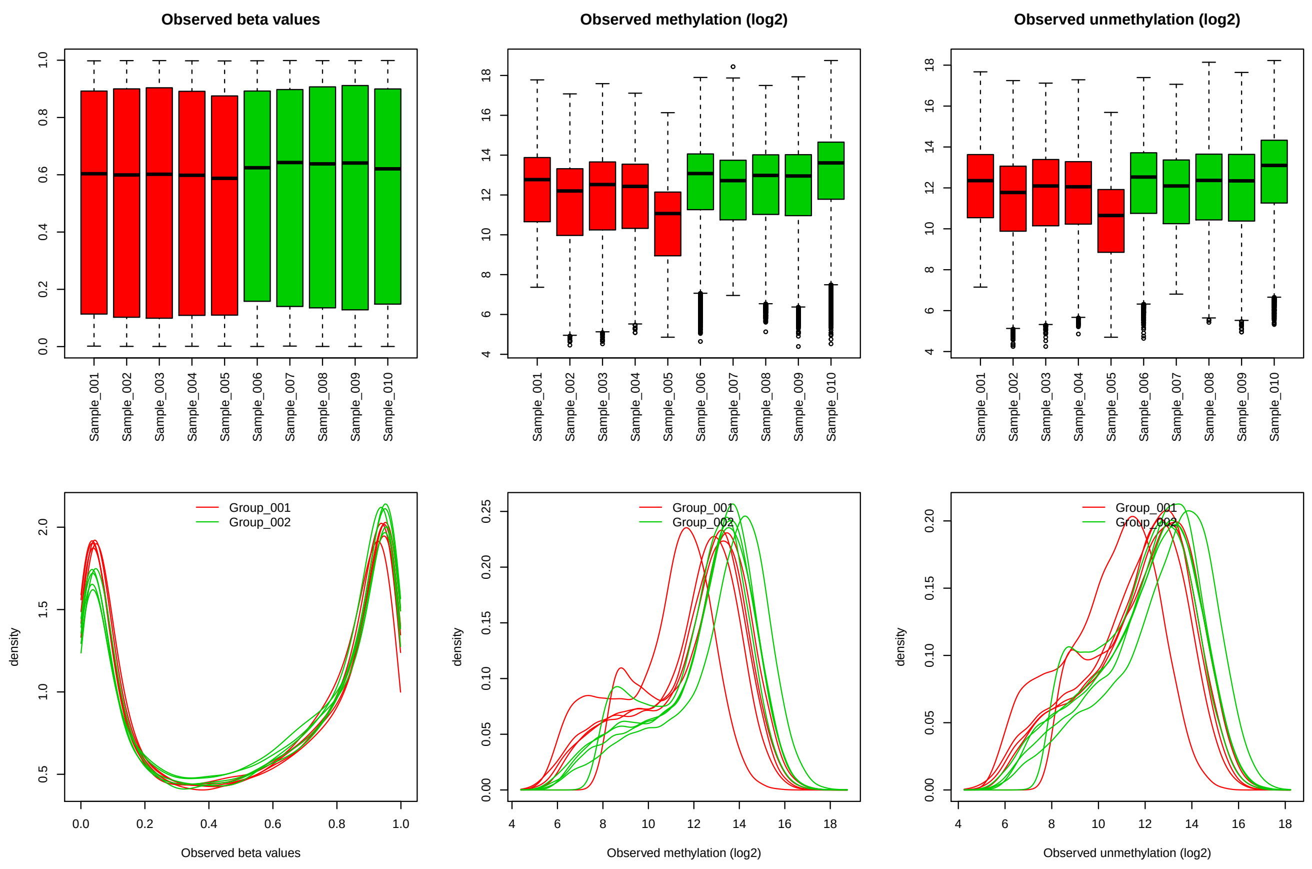


These are the default values for the (`siga`, `sigb` and `sibOpt`) parameters in the `simulateMeth` function.

To control how much technical variation is induced from the platform-technology, the variance hyperparameters from the sample-level noise (`siga`, `sigb` and `sibOpt`) can be controlled manually.

```
set.seed(999)
siga = sigb = 1 * diag(10)
```

```
sigOpt = 2 * diag(10)
simMeth <- simulateMeth(methTruth,  meth.platform = "methArrays",
                        nSamps = 5, nMol = 1e6,
                        siga = siga, sigb = sigb, sigOpt = sigOpt)

## Simulating DNA methylation samples using the meth.platform:  methArrays

plotMeth(simMeth)
```



4 Gene Expression

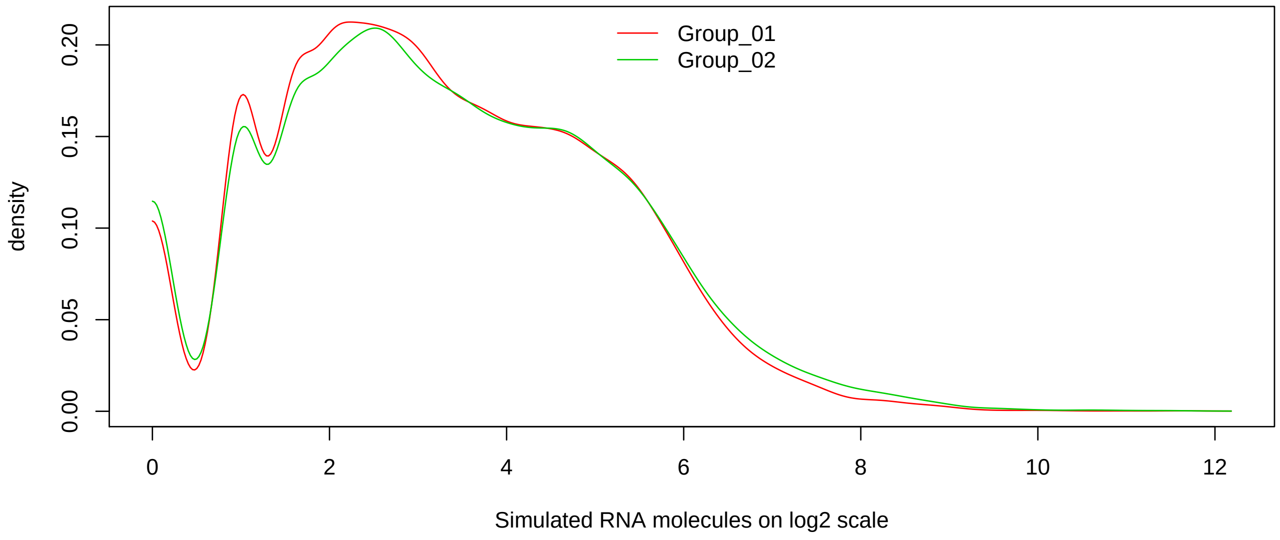
There are two main functions used to generate simulated gene expression data: `simulateGExTruth` and `simulateGEx`. The first function (`simulateGExTruth`) generates the true gene expression without any consideration for a platform technology. The second function (`simulateGEx`) simulates observed gene expression based on:

1. the platform technology
2. the magnitude of technical variation

4.1 Quick Start

To simulate the true level gene expression for a set of 2 groups, use the `simulateGExTruth` function.

```
set.seed(999)
geneTruth <- simulateGExTruth(nGenes = 2e4, nGroups = 2,
                              pDiff = 0.05, foldDiff = 5)

## [quantroSim]:  Simulating RNA transcript counts using a Poisson
##          distribution with mean parameters from 0.01 to 4662.66

plotGExTruth(geneTruth)
```



Similar to `simulateMethTruth`, `pDiff` is percent of probes different relative to Group 1. If `nGroups` = 1, `pDiff` should be 0. If `nGroups` > 1, the length of `pDiff` should be equal to `nGroups` - 1. The default for `nGroups` is 2 and the default for `pDiff` is 0.05.

`foldDiff` is the fold difference of gene differentially expressed in one group relative to Group 1. If `nGroups` = 1, `foldDiff` is ignored. If `nGroups` > 1, the length of `foldDiff` should be equal to `nGroups` - 1. The default for `nGroups` is 2 and the default for `foldDiff` is 5.

The main output will be a matrix (`geneRange`) of dimension `nGenes` x `nGroups`.

```
dim(geneTruth$geneRange)

## [1] 20000     2
```

The correlation between the two groups is given by:

```
cor(geneTruth$geneRange)

##            Group_01  Group_02
## Group_01 1.0000000 0.8736777
## Group_02 0.8736777 1.0000000
```

If pDiff was given, there will be pDiff $\times$ nGenes differences between the two groups. A boolean vector referring to which genes are different is in the geneTruth object called genesDiffInd. Here we list the indicies of which genes are different between the groups:

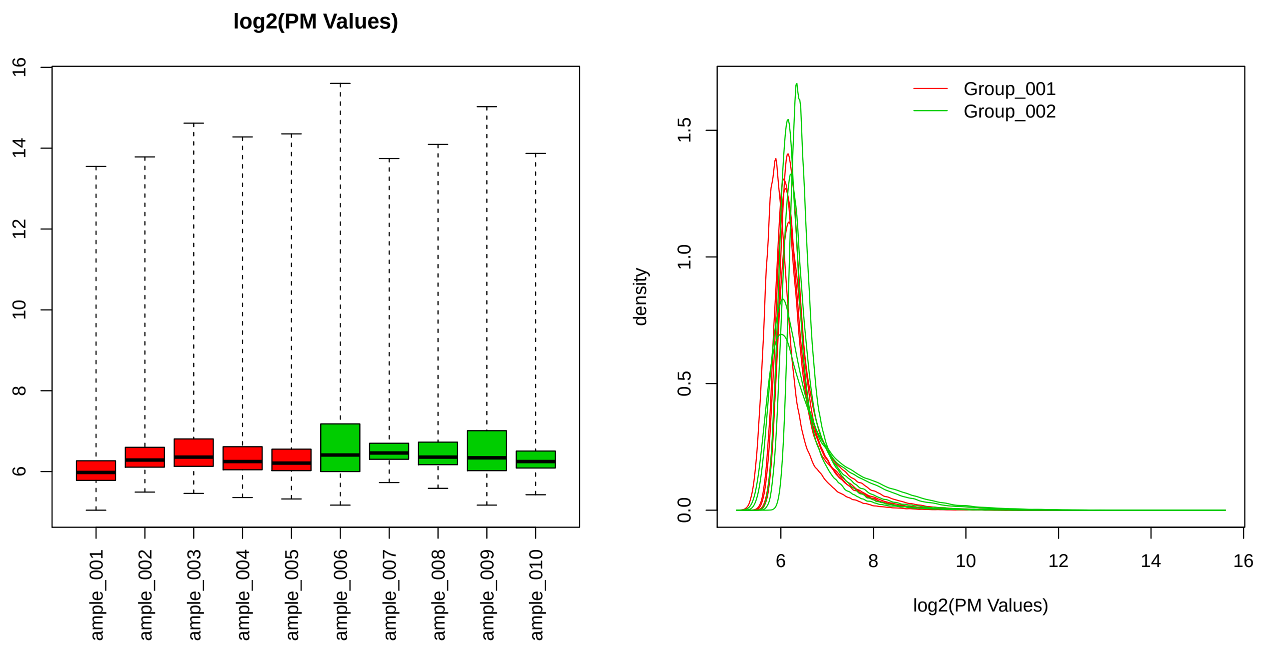
```
head(which(geneTruth$genesDiffInd))

## [1]  28  43  67  85 117 130
```

To simulate observed gene expression data based on a specific technology platform, use the simulateGEx function. First, a platform from list.GEx.platforms must be selected:

```
list.GEx.platforms()

## [1] "GExArrays"
```

Once a platform has been selected,

```
set.seed(999)
sim <- simulateGEx(geneTruth,  GEx.platform = "GExArrays", nSamps = 5)

## Simulating gene expression samples using the GEx.platform:  GExArrays
## No PCR amplification of RNA transcript counts.

plotGEx(sim)
```

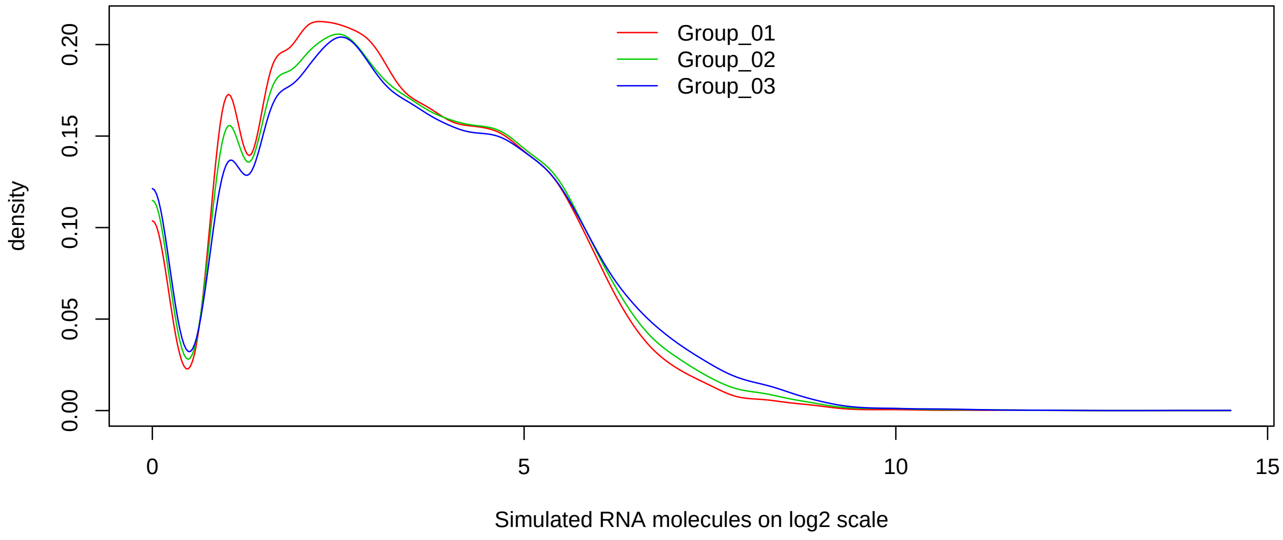


```
summary(simMeth$meth)

##    Sample_001        Sample_002        Sample_003        Sample_004        Sample_005
##  Min.   :   165   Min.   :    22   Min.   :    23   Min.   :    34   Min.   :   29
##  1st Qu.:  1614   1st Qu.:  1002   1st Qu.:  1214   1st Qu.:  1281   1st Qu.:  493
##  Median :  6986   Median :  4710   Median :  5884   Median :  5522   Median : 2146
##  Mean   : 11110   Mean   :  7431   Mean   :  9363   Mean   :  8658   Mean   : 3339
##  3rd Qu.: 15055   3rd Qu.: 10206   3rd Qu.: 12929   3rd Qu.: 11959   3rd Qu.: 4529
##  Max.   :224914   Max.   :138133   Max.   :197185   Max.   :141540   Max.   :71803
##    Sample_006        Sample_007        Sample_008        Sample_009        Sample_010
##  Min.   :    25   Min.   :   124   Min.   :    35   Min.   :    21   Min.   :    23
##  1st Qu.:  2455   1st Qu.:  1724   1st Qu.:  2081   1st Qu.:  1994   1st Qu.:  3531
##  Median :  8626   Median :  6742   Median :  8086   Median :  7930   Median : 12504
##  Mean   : 12532   Mean   : 10165   Mean   : 12157   Mean   : 12134   Mean   : 18862
##  3rd Qu.: 17090   3rd Qu.: 13719   3rd Qu.: 16570   3rd Qu.: 16652   3rd Qu.: 25760
##  Max.   :244585   Max.   :355651   Max.   :185149   Max.   :250245   Max.   :440915
```

4.2 Simulating 2 or more groups

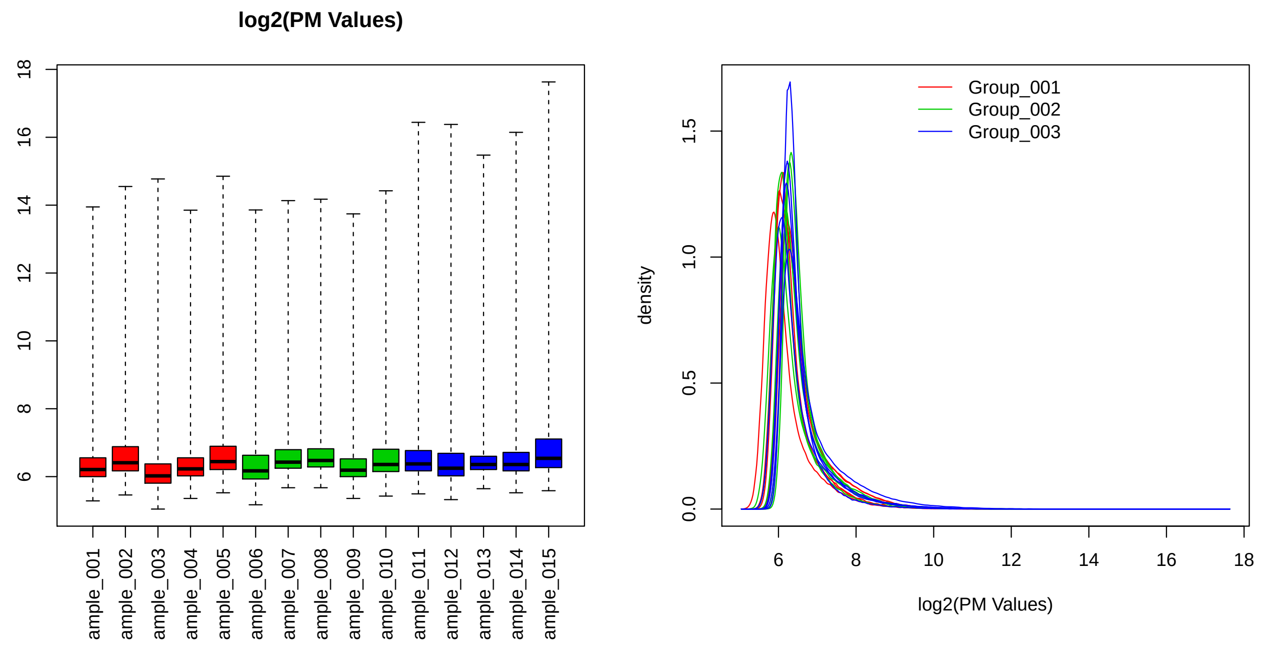
To simulate the true level gene expression for a set of 2 or more groups, again use the the same simulateGExTruth function, but change nGroup and the length of pDiff and foldDiff

```
set.seed(999)
geneTruth <- simulateGExTruth(nGenes = 2e4, nGroups = 3,
                              pDiff = c(0.05, 0.10), foldDiff = c(5,5))

## [quantroSim]:  Simulating RNA transcript counts using a Poisson
##           distribution with mean parameters from 0.01 to 4662.66

plotGExTruth(geneTruth)
```



```
set.seed(999)
sim <- simulateGEx(geneTruth,  GEx.platform = "GExArrays", nSamps = 5)

## Simulating gene expression samples using the GEx.platform:  GExArrays
## No PCR amplification of RNA transcript counts.
```

```
plotGEx(sim)
```



4.3 Additional options for `simulateGEx`

4.3.1 Controlling level of technical variation

We use the Langmuir model to simulate chemical saturation observed using microarrays. Our model to simulate raw Perfect Match (PM) value for the j^{th} probe from the i^{th} sample in the k^{th} group is given by

$$PM_{ijk} = o_{ijk} + d_{ijk} + a_{ijk}(\frac{x_{jk}}{x_{jk} + b_{ijk}})\epsilon_{ijk}$$

where x_{jk} is the number of RNA molecules at j^{th} probe in the k^{th} group and the rest are parameters simulated from a log Normal distribution with a given set of hyperparameters, similar to simulating DNA methylation:

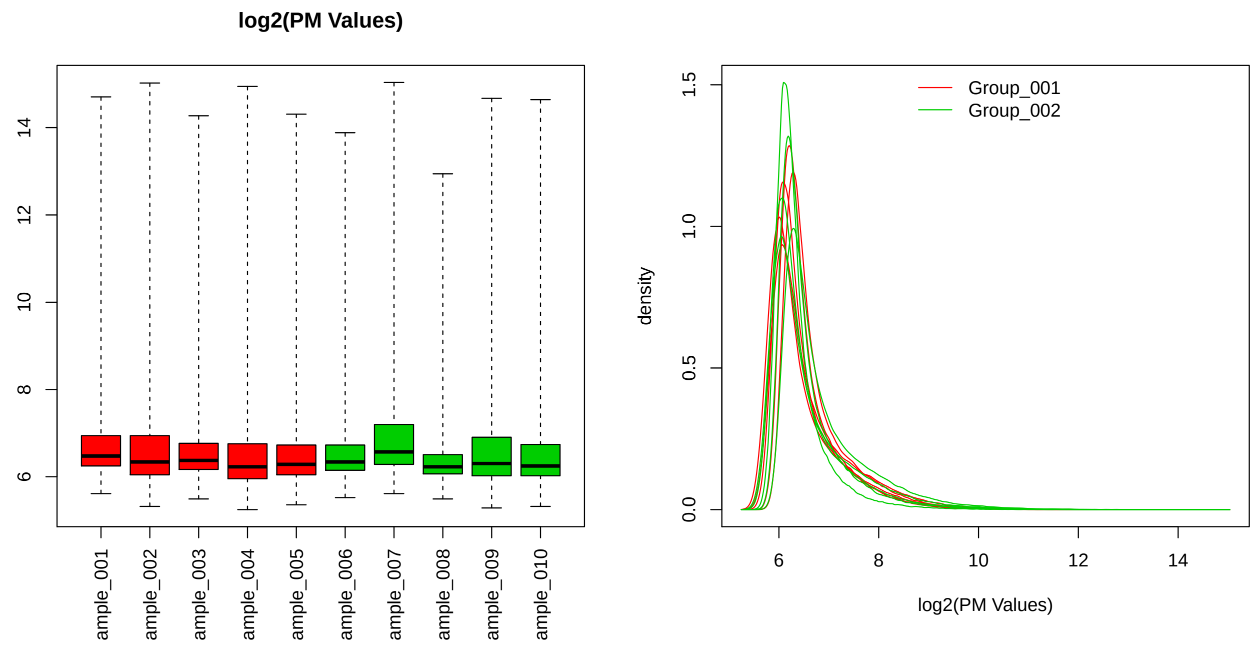
$$\log_2(a_{ik}) \sim N(20, 0.1)$$

$$\log_2(b_{ik}) \sim N(18, 0.1)$$
$$\log_2(o_{ik}) \sim N(5, 0.1)$$
$$\log_2(d_{ijk}) \sim N(5, 1)$$
$$\log_2(\epsilon_{ijk}) \sim N(0, 1)$$

For efficiency, we simulate the parameters from a multivariate normal distribution for all 10 arrays (=5 samples per group * 2 groups). In the above example, covariance matrices would be given by:

```
set.seed(999)
siga = sigb = 0.1 * diag(10)
sigOpt = 0.1 * diag(10)
```

```
geneTruth <- simulateGExTruth(nGenes = 2e4, nGroups = 2,
                              pDiff = 0.05, foldDiff = 5)

## [quantroSim]:  Simulating RNA transcript counts using a Poisson
##           distribution with mean parameters from 0.01 to 4662.66

sim <- simulateGEx(geneTruth,  GEx.platform = "GExArrays", nSamps = 5,
                   siga = siga, sigb = sigb, sigOpt = sigOpt)

## Simulating gene expression samples using the GEx.platform:  GExArrays
## No PCR amplification of RNA transcript counts.

plotGEx(sim)
```

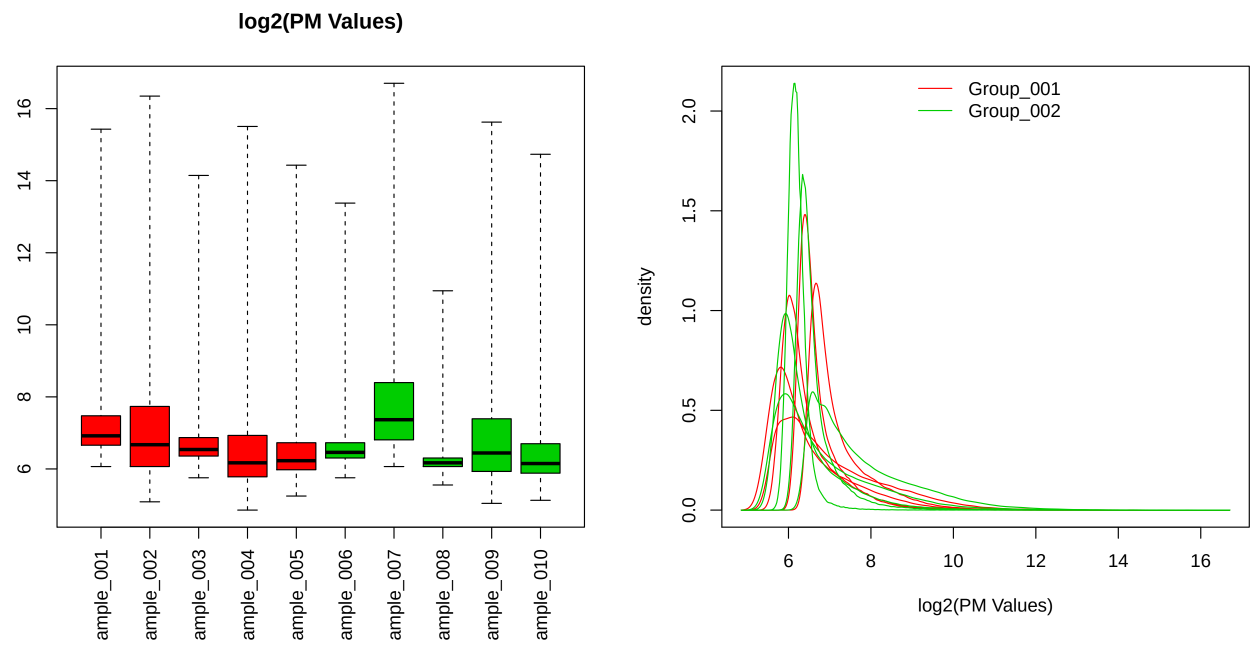


These are the default values for the (`siga`, `sigb` and `sibOpt`) parameters in the `simulateGEx` function.

To control how much technical variation is induced from the platform-technology, the variance hyperparameters from the sample-level noise (`siga`, `sigb` and `sibOpt`) can be controlled manually.

```
set.seed(999)
siga = sigb = 1 * diag(10)
sigOpt = 1 * diag(10)

geneTruth <- simulateGExTruth(nGenes = 2e4, nGroups = 2,
                              pDiff = 0.05, foldDiff = 5)

## [quantroSim]:  Simulating RNA transcript counts using a Poisson
##           distribution with mean parameters from 0.01 to 4662.66

sim <- simulateGEx(geneTruth,  GEx.platform = "GExArrays", nSamps = 5,
                   siga = siga, sigb = sigb, sigOpt = sigOpt)
```

```
## Simulating gene expression samples using the GEx.platform:  GExArrays
## No PCR amplification of RNA transcript counts.

plotGEx(sim)
```



5 Getting Help

For more help, open the HTML help file:

```
help(package = 'quantroSim', help_type = 'html')
```

6 SessionInfo

```
sessionInfo()

## R version 3.1.2 (2014-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] quantroSim_0.0.1    Biobase_2.26.0      BiocGenerics_0.12.1 knitr_1.8
##
```

```
## loaded via a namespace (and not attached):
##  [1] annotate_1.44.0       AnnotationDbi_1.28.1  base64_1.1
##  [4] beanplot_1.2          BiocStyle_1.4.1       Biostrings_2.34.0
##  [7] bumphunter_1.6.0      codetools_0.2-9       colorspace_1.2-4
## [10] DBI_0.3.1             digest_0.6.4          doParallel_1.0.8
## [13] doRNG_1.6             evaluate_0.5.5        foreach_1.4.2
## [16] formatR_1.0           genefilter_1.48.1     GenomeInfoDb_1.2.3
## [19] GenomicRanges_1.18.3  ggplot2_1.0.0         grid_3.1.2
## [22] gtable_0.1.2          highr_0.4             illuminaio_0.8.0
## [25] IRanges_2.0.0         iterators_1.0.7       lattice_0.20-29
## [28] limma_3.22.1          locfit_1.5-9.1        MASS_7.3-35
## [31] matrixStats_0.10.3    mclust_4.4            minfi_1.12.0
## [34] multtest_2.22.0       munsell_0.4.2         nlme_3.1-118
## [37] nor1mix_1.2-0         pkgmaker_0.22         plyr_1.8.1
## [40] preprocessCore_1.28.0 proto_0.3-10          quadprog_1.5-5
## [43] quantro_1.0.0         R.methodsS3_1.6.1     RColorBrewer_1.0-5
## [46] Rcpp_0.11.3           registry_0.2          reshape_0.8.5
## [49] reshape2_1.4          rngtools_1.2.4        RSQLite_1.0.0
## [52] S4Vectors_0.4.0       scales_0.2.4          siggenes_1.40.0
## [55] splines_3.1.2         stats4_3.1.2          stringr_0.6.2
## [58] survival_2.37-7       tools_3.1.2           XML_3.98-1.1
## [61] xtable_1.7-4          XVector_0.6.0         zlibbioc_1.12.0
```