Supplementary manual. Sequential design of RNA-seq experiments

Camille Stephan-Otto Attolini, Victor Peña, David Rossell

This manual explains how to use `casper` to help design RNA-seq experiments, for the main manual please load the package and type `vignette('casper')` at the command prompt. `casper` provides tools to design RNA-seq isoform expression experiments, both single and multiple sample studies. The former may *e.g.* estimate relative isoform expression within a gene or perform de novo isoform discovery, whereas the latter usually search for isoforms that are differentially expressed across sample groups (*e.g.* healthy vs. sick). When designing such an experiment researchers face numerous decisions, including the sequencing setup (*e.g.* number of reads, read length), sample preparation (*e.g.* insert sizes) and the sample size (in multiple sample studies). Choosing the optimal strategy is challenging, as it depends on the structure of the isoforms, their (unknown) absolute and relative expressions, the extent to which they differ across groups and on non-trivial interactions with the sequencing technology and sample preparation.

Because it is impossible to anticipate all these issues, `casper` adopts a sequential strategy. Based on preliminary data, it simulates RNA-seq data under several experimental setups and evaluates their relative merits using Bayesian decision theory. The approach is sequential in that, whenever new experimental data is observed it can be added to the preliminary data to refine the predictions. More formally, predictions are based on posterior predictive draws that incorporate the uncertainty on all unknown quantities and condition on all data observed so far.

The performance (utility) of each considered experimental setup is evaluated with default criteria related to estimation error (single sample studies) or operating characteristics (multiple sample studies). An advantage of the proposed simulation-based approach is that the experimenter may easily change or modify these default criteria. The best design can then be chosen either informally or by maximizing posterior expected utility (Savage, 1954). Simulated data are returned as `ExpressionSet` objects or `.bam` files, so that alternative analysis strategies within Bioconductor (Gentleman et al., 2004)

or third-party software can be easily integrated.

1 One sample experiments

1.1 Quick mean absolute error calculation with simMAE

The goal is to assess which sequencing settings are expected to better characterize isoform expression in a single sample. That is, we need to decide the number of paired-end reads (N), read length (r) and average insert size (f). `casper` provides functions to simulate RNA-seq data (based on the model introduced in Rossell et al. (2014)) and to evaluate the accuracy in isoform expression estimation for different combinations of (N, r, f). We note that the desired number of reads N differs from the actual number of reads $\tilde{N}$ in a random fashion that depends on read mappability and deviations from the target read yield in the sequencing facility. `casper` accounts for this uncertainty by generating a different $\tilde{N}$ in each simulation (see help for `simMAE` or `simMultSamples` for details). Our examples here assume that the goal is to estimate isoform expression at a sufficiently high precision, but one can easily consider alternative goals (*e.g.* determining the dominant isoform for each gene, the probability of detecting previously unknown isoforms) by running any desired software on our generated .bam files.

As illustration we now design a one sample study to estimate isoform expression in LCL transformed cells. Although `casper` works best when based on RNA-seq pilot data related to the experiment that is being designed, Section 1.3 describes default RNA-seq human and mouse data and Section 1.4 how to use any expression data formatted as an `ExpressionSet` (*e.g.* microarray data from GEO). We downloaded a bam file from the 1000 Genomes project (http://www.1000genomes.org/data) corresponding to an RNA-seq experiment on LCL cells, and processed it with function `wrapKnown` (see the main `casper` manual for details). We also formatted the human genome hg19 in the format required by `casper` by running

```
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> hg19DB <- procGenome(TxDb.Hsapiens.UCSC.hg19.knownGene,
    genome='hg19')
```

To consider another organism simply use the corresponding Bioconductor package, usually replacing `hg19` for the corresponding genome *e.g.* `mm10` for mouse, `dm3` for drosophila melanogaster, *etc.* Transcriptomes arising from *de novo* isoform predictions are also supported, see `help(procGenome)` for

details on how to import .gtf files. For convenience here we load directly a pre-computed `hg19DB` and the pilot data after running `wrapKnown` (which contains estimated read start and insert size distributions, expression levels, and path counts). The pilot data are available at `https://sites.google.com/site/rosselldavid/home/myfiles`, file `oneT.casper.RData` (see Section 1.3 for details).

```
> options(width=60,digits=3,continue="  ")
> library(casper)
> load("hg19DB.RData")
> load("oneT.casper.RData")
```

Suppose we wish to compare the following 12 experimental setups, which arise from considering total sequenced bp=4,10,16 billions, read lengths $r = 76, 101$ and mean insert sizes $f = 200, 300$.

```
> bp=rep(c(4,10,16) *1e9, 4)
> r=c(76, 101, 750)[gl(2, length(bp)/4, length=length(bp))]
> n <- round(bp/(r*2))
> f=c(rep(200, length(bp)/2),rep(300, length(bp)/2))
> f=rep(c(200,300), each=length(bp))
> cbind(n,r,f)
```

```
             n   r   f
 [1,] 2.63e+07  76 200
 [2,] 6.58e+07  76 200
 [3,] 1.05e+08  76 200
 [4,] 1.98e+07 101 200
 [5,] 4.95e+07 101 200
 [6,] 7.92e+07 101 200
 [7,] 2.63e+07  76 200
 [8,] 6.58e+07  76 200
 [9,] 1.05e+08  76 200
[10,] 1.98e+07 101 200
[11,] 4.95e+07 101 200
[12,] 7.92e+07 101 200
[13,] 2.63e+07  76 300
[14,] 6.58e+07  76 300
[15,] 1.05e+08  76 300
[16,] 1.98e+07 101 300
[17,] 4.95e+07 101 300
[18,] 7.92e+07 101 300
[19,] 2.63e+07  76 300
[20,] 6.58e+07  76 300
[21,] 1.05e+08  76 300
[22,] 1.98e+07 101 300
```

```
[23,] 4.95e+07 101 300
[24,] 7.92e+07 101 300
```

Throughout we assume that paired-end sequencing is used. However, single-end experiments can also be considered in `casper` by setting $f = 2r$, so that the two ends overlap and we effectively have a single-end experiment. For instance, 1,500bp single-end reads can be emulated by setting $r = 750$ and $f = 1500$.

The wrapper function `simMAE` simulates `nsim` RNA-seq experiments for each experimental setting, obtains relative isoform expression estimates for each of them and evaluates the mean absolute error (MAE) of these estimates. Alternatively, each of these steps can be performed separately, in particular .sam files can be generated with function `simReads` (see Section 1.2). The `simMAE` call to obtain `nsim=5` simulations is below.

```
> sims <- simMAE(nsim=5, nreads=n, readLength=r,
fragLength=f, pc=oneT$pc, distr=oneT$distr, readLength.pilot=101,
retTxsError=FALSE, genomeDB=hg19DB, mc.cores.int=4,
mc.cores=5, verbose=TRUE)
```

Before proceeding, we make some remarks regarding the `simMAE` arguments (for further details see the help).

- By default `simMAE` considers all isoforms in the given genome, but one can also selected a subset of genes with argument `islandid`.
- If `retTxsError` is `TRUE`, the function returns posterior expected MAE for each individual isoform. Else the output is a `data.frame` with (overall) MAE across all isoforms.
- `pc` contains path counts in the pilot data, alternatively one may provide an `ExpressionSet` via argument `eset.pilot`.
- The pilot data is assumed to be from a related experiment rather than the current tissue of interest (`usePilot=FALSE`). Hence, the pilot data is used to simulate new RNA-seq data but not to estimate its expression. However, in some cases we may be interested in re-sequencing the pilot sample at deeper depth, in which case one would want to combine the pilot data with the new data to obtain more precise estimates. This can be achieved by setting `usePilot=TRUE`.
- `mc.cores` and `mc.cores.int` indicate the number of cores to be used in the simulations. Setting them to values greater than 1 can speed up computations, but also be quite memory hungry.

- `distr` are the read length and fragment length (insert size) distributions in the pilot data, as returned by `wrapKnown` or `getDistrs`
- `readLength.pilot` is the read length in the pilot data.

Because we set `retTxsError=FALSE`, the output of `simMAE` is a table with estimated mean absolute error (MAE) for each simulation. Since `nsim` was set to 5, there are 5 replicates for each of the experimental setups we are interested in. Standard errors can be obtained as SD over `sqrt(nsim)`, as usual. There is little variability in the MAE for each experimental setting, hence `nsim=5` already provides very low SE in this example.

```
> head(sims)
     MAE   Nreads ReadLength frLength
1 0.0387 26315789         76      200
2 0.0385 26315789         76      200
3 0.0383 26315789         76      200
4 0.0386 26315789         76      200
5 0.0390 26315789         76      200
6 0.0311 65789474         76      200
> e <- paste('N=',sims[,2],'_r=',sims[,3],'_f=',sims[,4],sep='')
> tapply(sims$MAE, e, mean)
N=105263158_r=76_f=200 N=105263158_r=76_f=300
                0.0274                 0.0264
N=19801980_r=101_f=200 N=19801980_r=101_f=300
                0.0411                 0.0394
 N=26315789_r=76_f=200  N=26315789_r=76_f=300
                0.0386                 0.0374
N=49504950_r=101_f=200 N=49504950_r=101_f=300
                0.0329                 0.0314
 N=65789474_r=76_f=200  N=65789474_r=76_f=300
                0.0309                 0.0298
N=79207921_r=101_f=200 N=79207921_r=101_f=300
                0.0291                 0.0276
> se <- tapply(sims$MAE, e, sd)/sqrt(5)
> se
N=105263158_r=76_f=200 N=105263158_r=76_f=300
             9.59e-05               3.31e-05
N=19801980_r=101_f=200 N=19801980_r=101_f=300
             9.65e-05               1.51e-04
 N=26315789_r=76_f=200  N=26315789_r=76_f=300
             1.06e-04               1.94e-04
N=49504950_r=101_f=200 N=49504950_r=101_f=300
```

```
                1.97e-04                 6.81e-05
 N=65789474_r=76_f=200  N=65789474_r=76_f=300
                7.68e-05                 1.23e-04
N=79207921_r=101_f=200 N=79207921_r=101_f=300
                1.56e-04                 5.22e-05
```

These results can be easily summarized with a plot (Figure 1). For any given coverage, the configuration with the best performance is read length $r = 76$ and fragment length $f = 300$. We expect that increasing the coverage from 30X to 80X will improve the accuracy of the estimation noticeably, but the expected increase in precision from 80X to 125X is not as substantial.

```
> totl <- sum(hg19DB@txLength)
> resall <- as.data.frame(sims)
> resall$bp <- resall$Nreads*resall$ReadLength*2
> resall$cov <- resall$bp/totl
> library(ggplot2)
> resall$exp.setup = factor(paste(resall$ReadLength,resall$frLength))
> resall$cov = as.factor(round(resall$cov))
> p = ggplot(aes(x=cov, y=MAE, fill=exp.setup), data=resall)

> print(p+geom_boxplot()+xlab("Coverage"))
```

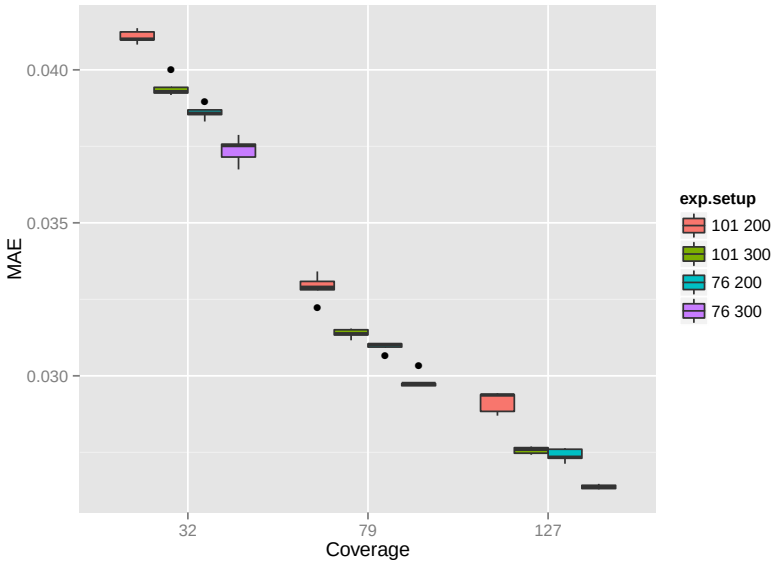

Figure 1: Estimated Mean Absolute Error (MAE) for 3 different choices of coverage (approx. 30X, 80X, 125X), 2 of read length (76bp and 101bp), and 2 of fragment length (200bp and 300bp).

Lastly, one can check the goodness-of-fit of the model using `simMAEcheck`.

This function compares the vector that contains the number of reads that have been aligned to each gene (in the pilot data) with posterior predictive simulations. The output of `simMAEcheck` is a list with 2 entries

- `U`: contains MAE estimates based on synthetic replications of the experimental data.
- `mod_check`: shows the expected number of islands for which the observed data lie in the range of the simulations (assuming they have the same distribution), and the actual results in the posterior predictive simulations. If the model fits the data, the expected value and observed value should be similar.

In our example, setting the number of posterior predictive simulations to 5:

```
checks <- simMAEcheck(nsim=5, pc=oneT$pc, distr=oneT$distr, readLength.pilot=101
              retTxsError=FALSE, genomeDB=hg19DB, mc.cores.int=2, mc.cores=1, ve
```

Note that the arguments of `simMAEcheck` are a subset of those used for `simMAE`. The output of the function is:

```
> checks
$U
     MAE   Nreads ReadLength frLength
1 0.0449 16770752        101      290
2 0.0448 16770752        101      290
3 0.0457 16770752        101      290
4 0.0452 16770752        101      290
5 0.0487 16770752        101      290

$mod_check
  Expected Observed
1    14387    10252
```

1.2 Generate a sam file with simulated reads

In order to facilitate integration with other software, function `simReads` provides the option to write simulated (aligned) reads to a sam file. The following code writes a simulation run to the file 'sims.sam':

First we need to get number of reads in each gene island in the pilot data. Rather than simulating reads for all genes, for computational speed here we select only the first 10 non-empty gene islands.

```
> load('oneT.casper.RData')
> nReads <- getNreads(oneT$pc)
> head(nReads)
   1    2    3    4    5    6
   0    0 1762    4    0 1894
> nSimReads <- nReads[nReads>10][1:10]
> nSimReads
   3    6    7    8    9   10   11   12   13   14
1762 1894   68  695  995  161  379  232  703  493
> isls <- names(nSimReads)
```

Next we estimate expression on the pilot data. Notice that by setting `rpkm=FALSE`, `calcExp` will return relative rather than absolute isoform expression (*i.e.* adding up to 1 for each gene island).

```
> readLength=101
> tmp <- calcExp(distrs=oneT$distr, genomeDB=hg19DB,
  pc=oneT$pc, islandid=isls, rpkm=FALSE, readLength=readLength)
> pis <- exprs(tmp)[,1]
> names(pis) <- rownames(tmp)
> head(pis)
NM_001145277 NM_001145278    NM_018090    NM_013943
      0.0273       0.0109       0.9618       1.0000
NM_001195683    NM_003243
      0.2268       0.2275
```

Finally, we call `simReads` to simulate reads and write to file 'sims.sam'.

```
> sims <- simReads(islandid=isls, nSimReads=nSimReads, pis=pis,
  rl=readLength, writeBam=TRUE, distrs=oneT$distr,
  genomeDB=hg19DB, bamFile='sims', seed=1)
Formatting input
Simulating fragments
10 % of fragments simulated
20 % of fragments simulated
30 % of fragments simulated
40 % of fragments simulated
50 % of fragments simulated
60 % of fragments simulated
70 % of fragments simulated
80 % of fragments simulated
90 % of fragments simulated
100 % of fragments simulated
Splitting counts
```

To convert the sam file to the binary bam format, sort and index the file we can use the software samtools (Li et al. (2009)) with the following command:

```
> system("samtools view -Sb sims.sam > sims.bam &&
  samtools sort sims.bam sims.sorted && samtools index sims.sorted.bam")
```

1.3 Default human and mouse datasets

We provide two example datasets that can be used as default pilot data, one for human and one for mouse, which can be downloaded from `https://sites.google.com/site/rosselldavid/home/myfiles` (files oneT.casper.RData micebladder.rep1.RData). These files can be used to generate simulations with parameter combinations different to those considered in our paper.

As default pilot human data we downloaded fasta files for sample `ERS185276` from the 1000 Genomes project (`http://www.1000genomes.org/data`). After aligning with `Tophat` (Trapnell et al., 2009) and default parameters, we used function `wrapKnown` to generate the object `oneT`. The read length for this experiment is 101bp. This named list contains all necessary information to simulate reads with `simMAE` or `simReads` (Sections 1.1-1.2).

- `exp`: An `ExpressionSet` with estimated log-RPKM expression
- `distr`: An object of class readDistrs
- `pc`: An object of class pathCounts

```
> load("oneT.casper.RData")
> names(oneT)
[1] "pc"    "distr" "exp"
```

Note that these objects provide transcript expressions in RPKM, but currently `simMAE` only supports relative isoform, hence we need to compute relative expressions from the pilot data. This is easy since we have the `pathCounts` and `readDistr` objects, so we can simply set the argument `rpkm=FALSE` in `calcExp`.

```
> readLength=101
> pilot <- calcExp(distrs=oneT$distr, genomeDB=hg19DB,
  pc=oneT$pc, islandid=isls, rpkm=FALSE, readLength=readLength)
> pis <- exprs(pilot)[,1]
> names(pis) <- rownames(pilot)
```

We now describe the default RNA-seq mouse data. We downloaded bam files for mouse bladder sample `wgEncodeEM003062` of the Encode project

(http://www.noncode.org). We used function wrapKnown to generete the object rep1. The read length for this sample is 101bp.

```
> load("micebladder.rep1.RData")
> names(rep1)
[1] "pc"    "distr" "exp"
```

1.4 ExpressionSet as pilot data

The earlier sections used RNA-seq data formatted as .bam files for pilot data (processed with wrapKnown), which ideally come from a related study (same organism, related tissue or experimental conditions). Although we strongly recommend that such pilot data be used whenever available, simMAE also allows using any ExpressionSet object as pilot data by leaving argument pc missing and specifying argument eset.pilot instead. For instance, pilot data can be easily obtained from Gene Expression Omnibus (Edgar et al., 2002) using function getGEO from package GEOquery, and could even come from microarray or other technologies. Importantly, simMAE assumes that eset.pilot contains normalized log2 expression, *i.e.* gene expression is assumed proportional to 2^exprs(eset.pilot). Isoform expression within a gene is then generated from a symmetric Dirichlet distribution with parameter $1/I_g$, where I_g is the number of isoforms in gene g (see details in help(simMAE)). Within the package casper we also provide default read start and insert size distributions in data 'distrsGSE37704' (see Section 2).

As illustration we use an example where the pilot data contains 10 samples and measures expression for 100 genes. For computational speed here we create eset.pilot with random expression levels, but of course in practice it would contain actual experimental data. We observe that, as expected, MAE decreases with sequencing depth.

```
> exprsx <- matrix(rnorm(1000),nrow=100,ncol=10)
> eset.pilot <- new("ExpressionSet", exprs=exprsx)
> data("distrsGSE37704")
> distr <- distrsGSE37704[[1]]
> n <- c(10^6,2*10^6)
> r <- rep(101,length(n))
> f <- rep(300,length(n))

> sims <- simMAE(nsim=2, nreads=n, readLength=r,
  fragLength=f, distr=distr, eset.pilot=eset.pilot,
  genomeDB=hg19DB, verbose=TRUE)
Simulating pilot data...
Generating posterior samples j = 1
```

```
Obtaining expression estimates...
Average MH acceptance rate 0.827451
Formatting output...
Running simulations for j = 1
Formatting input
Simulating fragments
10 % of fragments simulated
20 % of fragments simulated
30 % of fragments simulated
40 % of fragments simulated
50 % of fragments simulated
60 % of fragments simulated
70 % of fragments simulated
80 % of fragments simulated
90 % of fragments simulated
100 % of fragments simulated
Splitting counts
Finished simulations
Formatting input
Simulating fragments
10 % of fragments simulated
20 % of fragments simulated
30 % of fragments simulated
40 % of fragments simulated
50 % of fragments simulated
60 % of fragments simulated
70 % of fragments simulated
80 % of fragments simulated
90 % of fragments simulated
100 % of fragments simulated
Splitting counts
Finished simulations
Generating posterior samples j = 2
Obtaining expression estimates...
Average MH acceptance rate 0.827402
Formatting output...
Running simulations for j = 2
Formatting input
Simulating fragments
10 % of fragments simulated
20 % of fragments simulated
30 % of fragments simulated
40 % of fragments simulated
```

```
50 % of fragments simulated
60 % of fragments simulated
70 % of fragments simulated
80 % of fragments simulated
90 % of fragments simulated
100 % of fragments simulated
Splitting counts
Finished simulations
Formatting input
Simulating fragments
10 % of fragments simulated
20 % of fragments simulated
30 % of fragments simulated
40 % of fragments simulated
50 % of fragments simulated
60 % of fragments simulated
70 % of fragments simulated
80 % of fragments simulated
90 % of fragments simulated
100 % of fragments simulated
Splitting counts
Finished simulations
```

```
> sims
     MAE Nreads ReadLength frLength
1 0.0711  1e+06        101      300
2 0.0698  1e+06        101      300
3 0.0562  2e+06        101      300
4 0.0559  2e+06        101      300
```

2 Multiple sample problem

Multiple sample problems are more challenging in that, additional to the sequencing setup (depth, read length and fragment size) one must also decide on the sample size. For a given monetary cost, one may choose to either sequence a few samples at high depth or more samples at lower depth. While adding samples increases the statistical power as usual, a higher sequencing depth increases the estimation precision within each sample and the probability of observing reads from low expression isoforms. As the adequate strategy depends on the specifics of the study (*e.g.* amount of differential expression, number of low expression isoforms), `casper` implements a sequential

strategy. It uses pilot data to learn these characteristics and simulates future data taking into account the inherent uncertainty (*e.g.* the pilot data provides imperfect expression and fold change estimates).

As an illustration, we consider the Homo sapiens RNA-seq study described in Trapnell et al. (2013). The data are available at Gene Expression Omnibus (`http://www.ncbi.nlm.nih.gov/geo`) under accession GSE37704. Shortly, the study aims to compare expression between a knock-down (KO) and a scramble group. Three samples per group were sequenced using MiSeq technology, and three further independent samples using HiSeq. As MiSeq is a cheaper desktop sequencing technology yielding a relatively low number of sequences, we use the MiSeq data as our pilot data to assess the potential advantages of performing a follow-up HiSeq study. Because actual HiSeq data is available, we can then evaluate the quality of the predictions provided by `casper`.

The first step was to download the MiSeq and HiSeq fasta files, align the reads and obtain expression estimates with `wrapKnown`. The processed data are available at `https://sites.google.com/site/rosselldavid/home/myfiles`, files `gse37704_miseq.RData` and `gse37704.RData`. Before proceeding to the analysis we indicate the commands needed to obtain the processed data. We aligned the reads with TopHat (Trapnell et al., 2009) using default parameters to the human genome version hg19. The code to import the resulting BAM files into Bioconductor and obtain expression estimates with function `wrapKnown` is below (see `vignette("casper")` for details). The example is for a single sample (SRR493372), and also shows how to format the transcriptome as needed by `casper` with function `procGenome`.

```
> library(GenomicFeatures)
> genDB <- makeTranscriptDbFromUCSC("hg19",tablename="refGene")
> hg19DB <- procGenome(genDB, "hg19")

> bamFile <- 'SRR493372/sorted_hits.bam'
> SRR493372 <- wrapKnown(bamFile, genomeDB=hg19DB, readLength=101,
  keep.multihits=FALSE)
```

We used `wrapKnown` on the remaining MiSeq samples (SRR493373 to SRR493376). The slot `"exp"` in the output of `wrapKnown` contains an `ExpressionSet` with the estimated isoform expressions, and the slot `"distrs"` the estimated insert size and read start distributions. We used function `mergeExp` to combine the expressions in a single `ExpressionSet`. The `keep` option indicates to save transcript and gene ids, as well as the number of aligned reads per gene in the `featureData`. We stored the insert sizes and read start distributions in a list.

```
> sampleNames <- paste("SRR4933",72:76,sep="")
> gse37704.miseq <- mergeExp(SRR493372$exp,SRR493373$exp,SRR493374$exp,
  SRR493375$exp,SRR493376$exp,SRR493376$exp, sampleNames=sampleNames,
  keep=c('transcript','gene','explCnts'))
> gse37704.miseq$group <- factor(rep(c('Scramble','HOXA1KD'),each=3))
> gse37704.miseq <- quantileNorm(gse37704.miseq)

> distrsGSE37704 <- list(SRR493372$distr,SRR493373$distr,SRR493374$distr,
  SRR493375$distr,SRR493376$distr,SRR493377$distr)
```

Finally, we applied quantile normalization to `gse37704.miseq` using `quantileNorm`. We repeated the same process for the HiSeq data (samples SRR493367 to SRR493371).

We now proceed to the analysis. We first load the MiSeq data and annotated genome (available at `https://sites.google.com/site/rosselldavid/home/myfiles`).

```
> load('hg19DB.RData')
> load('gse37704_miseq.RData')
> gse37704.miseq
ExpressionSet (storageMode: lockedEnvironment)
assayData: 40892 features, 6 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: SRR493372 SRR493373 ... SRR493377 (6
    total)
  varLabels: group
  varMetadata: labelDescription
featureData
  featureNames: NM_032291 NM_001145277 ... NM_012312
    (40892 total)
  fvarLabels: transcript island_id ... readCount (16
    total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
> pData(gse37704.miseq)
             group
SRR493372 Scramble
SRR493373 Scramble
SRR493374 Scramble
SRR493375  HOXA1KD
```

```
SRR493376  HOXA1KD
SRR493377  HOXA1KD
```

The main step is to simulate future HiSeq data via posterior predictive simulation, *i.e.* conditional on the MiSeq data, which is implemented in `simMultSamples`. By default `simMultSamples` uses the log-normal normal with modified variance model (LNNMV, Yuan and Kendziorski (2006)) to simulate the true expression levels in each sample (argument `model='LNNMV'`), which is a hierarchical Normal-Normal model. As an alternative setting `model='GaGa'` generates data from the GaGa model (Rossell, 2009), a similar hierarchical model where expression levels and underlying parameters are assumed to be Gamma distributed. Essentially the GaGa model is preferrable when observed expression levels are positively skewed, although typically both models give fairly similar results when the pilot sample has $\geq$ 2 samples per group. We indicate to use the MiSeq pilot data with `x=gse37704.miseq`, and that there's a variable `'group'` in `pData(x)` containing group labels with `groups='group'`. We simulate 3 HiSeq samples per group (`nsamples=c(3,3)`) with a target of 12 million (`nreads`) 101bp (`readLength=101`) read pairs in each sample, with an average insert size of `fragLength=200`. We set the simulation in this manner to resemble the actual HiSeq data in GSE37704, which had 12-16 million 101bp aligned read pairs with an average insert size close to 200bp.

`simMultSamples` generates reads and insert sizes to reflect experimental biases (*e.g.* 3' end bias). Ideally these are estimated from pilot data with `wrapKnown` or `getDistrs` (`casper` does not impose any parametric assumptions to be as realistic as possible). For convenience, we also provide default distributions estimated from the GSE37704 MiSeq data in the dataset `distrsGSE37704`, which we observed is not too different from what estimates we obtained in various HiSeq datasets. `distrs` may be a list with estimates from several samples, and in this case `simMultSamples` randomly chooses one of them for each simulation, *i.e.* incorporates the uncertainty regarding the actual insert size and read distribution for future samples. To save computation time here we only obtain `nsim=5` simulations, but in practice we recommend more to better assess uncertainty (`nsim=20` seemed to suffice in most examples we considered). The argument `mc.cores` can be used for parallel processing.

```
> data(distrsGSE37704)
> nreads <- 12*10^6
> gse37704.new <- simMultSamples(5, nsamples=c(3,3), nreads=nreads,
+ readLength=101, fragLength=200, x=gse37704.miseq, groups='group',
+ distrs=distrsGSE37704, genomeDB=hg19DB, mc.cores=6)
```

```
Fitting NNGV model...
Obtaining 5 simulations (6 samples with 11833598 reads each -- some will be non-mappabl
......
......
......
......
......
```

simMultSamples returns an object of class "simulatedSamples " with nsim simulated datasets and nsamples each. We can select subsets as usual, e.g. gse37704.new[1:3] returns the first three simulations and gse37704.new[,c(1,4)] returns samples 1 and 4 (the first sample in each group) from all nsim simulations. We can also recover the simulation truth for the differences between group means in each simulation with coef. Estimated isoform expressions (casper log-rpkm estimate) in each simulated data are returned in ExpressionSets, on which one may use any desired analysis strategy. Here we use mergeBatches to combine the simulated HiSeq with the observed MiSeq data, which performs quantile normalization, a linear-model batch effect adjustment, and returns a list of ExpressionSet objects ready for analysis.

```
> gse37704.new
simulatedSamples object with 5 simulated datasets (6 samples each)
- 'coef' gets true differences between group means (returns matrix)
- 'exprs' gets estimated expressions (returns list of ExpressionSets)
- 'mergeBatches' combines exprs with a given ExpressionSet (returns list of Expr
> #Select 2 simulations
> gse37704.new[1:2]
simulatedSamples object with 2 simulated datasets (6 samples each)
- 'coef' gets true differences between group means (returns matrix)
- 'exprs' gets estimated expressions (returns list of ExpressionSets)
- 'mergeBatches' combines exprs with a given ExpressionSet (returns list of Expr
> #Select 2 samples from each sim
> gse37704.new[,c(1,2,4,5)]
simulatedSamples object with 5 simulated datasets (4 samples each)
- 'coef' gets true differences between group means (returns matrix)
- 'exprs' gets estimated expressions (returns list of ExpressionSets)
- 'mergeBatches' combines exprs with a given ExpressionSet (returns list of Expr
> #Simulation truth
> logfc.true <- coef(gse37704.new)
> head(logfc.true)
```

```
               sim1   sim2    sim3    sim4    sim5
NM_032291    -0.854 -1.094 -0.7904 -0.9190 -0.9334
NM_001145277 -0.157 -0.251 -0.2433  0.1392 -0.0115
NM_001145278  1.113  0.282  0.6674  0.6239  0.5010
NM_018090     0.268  0.302  0.5607  0.2463  0.1211
NM_052998    -0.380 -0.310 -0.4127 -0.4313 -0.2583
NM_001080397 -0.301 -0.275 -0.0944 -0.0193 -0.2560
> #Simulated expressions
> #(casper log-rpkm estimates)
> gse37704.new[[1]]$simExpr
ExpressionSet (storageMode: lockedEnvironment)
assayData: 40892 features, 6 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: Sample 1 Sample 2 ... Sample 6 (6
    total)
  varLabels: group
  varMetadata: labelDescription
featureData
  featureNames: NM_032291 NM_001145277 ... NM_012312
    (40892 total)
  fvarLabels: explCnts.1 explCnts.2 ... readCount (7
    total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
> #Merge with pilot data
> xnewadj <- mergeBatches(gse37704.miseq,gse37704.new,mc.cores=2)
> length(xnewadj)
[1] 5
> class(xnewadj[[1]])
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

Before proceeding to the analysis, we show that this preprocessing is indeed necessary with the actual GSE37704 HiSeq data. (file `gse37704.RData` at `https://sites.google.com/site/rosselldavid/home/myfiles`). We combine the data using `mergeBatches` as before, and also combine the data with no adjustment.

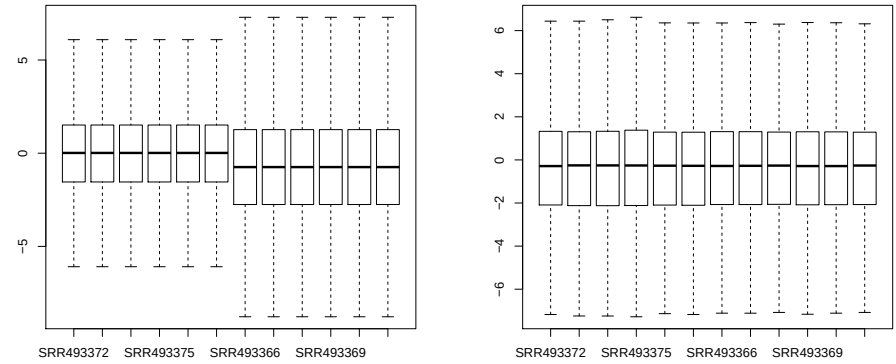

Figure 2: Boxplot with estimated isoform expressions with no adjustment (left) and the adjustment in `mergeBatches` (right)

```
> load('gse37704.RData')
> xadj <- mergeBatches(gse37704.miseq,gse37704)
.
> n <- featureNames(gse37704)
> xnoadj <- cbind(exprs(gse37704.miseq)[n,], exprs(gse37704))
```

When no adjustment is applied the range of estimated expressions in MiSeq data is substantially shorter than in HiSeq (Figure 2 left). That is, the higher HiSeq sequencing depth results in a broader dynamic range. As usual with expression studies, a suitable normalization must be applied to avoid systematic biases. The simple quantile normalization in `mergeBatches` seems to perform satisfactorily (Figure 2 right).

```
> boxplot(xnoadj,outline=FALSE)
> boxplot(exprs(xadj),outline=FALSE)
```

The boxplots discussed above inform about biases at the genome-wide level, but they cannot reveal biases at the transcript level. For this purpose we obtain PCA plots. In the unadjusted data (Figure 3, left) we observe that, while KO and Scramble samples are well separated within each batch, there are strong differences between batches. It should be noted that these differences remain even after applying quantile normalization to the unadjusted data (data not shown). The adjustment implemented in `mergeBatches` effectively removes batch effects (Figure 3, right). Here HiSeq samples exhibit lower variability than MiSeq, which would be consistent with a lower estimation error afforded by their higher sequencing depth.

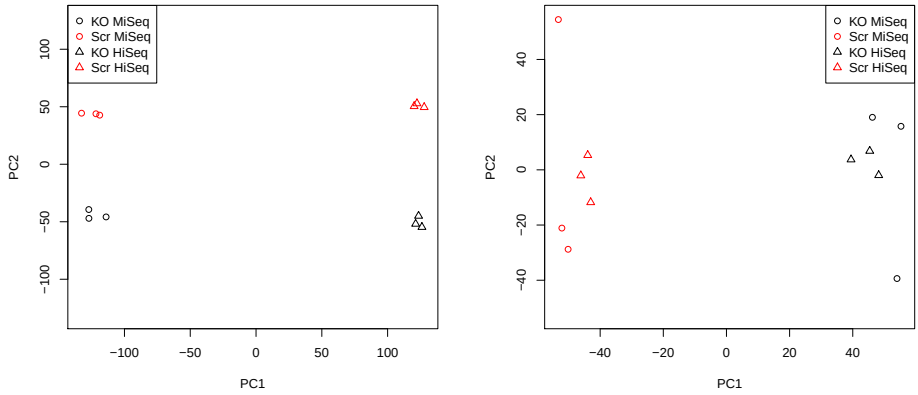

Figure 3: PCA plot for unadjusted data (left) and data adjusted with `mergeBatches` (right)

```
> pca <- prcomp(t(xnoadj))
> col <- c(gse37704.miseq$group,gse37704$group)
> pch <- rep(1:2,each=6)
> ylim <- range(pca$x[,1:2])
> plot(pca$x[,1:2], col=col, pch=pch, ylim=ylim)
> txt <- c('KO MiSeq','Scr MiSeq','KO HiSeq','Scr HiSeq')
> legend('topleft',txt,col=c(1,2,1,2),pch=c(1,1,2,2))
> pca <- prcomp(t(exprs(xadj)))
> col <- ifelse(xadj$group=='HOXA1KD',1,2)
> pch <- ifelse(xadj$batch=='batch1',1,2)
> ylim <- range(pca$x[,1:2])
> plot(pca$x[,1:2], col=col, pch=pch, ylim=ylim)
> legend('topright',txt,col=c(1,2,1,2),pch=c(1,1,2,2))
```

We now proceed to differential expression analysis. The basic idea is to apply any desired data analysis technique to the simulated data, the corresponding results giving an idea of what is expected in actual data. As an illustration, here we focus on finding isoforms where expression changes at least 3 fold between groups. More precisely, the null hypothesis for isoform i is $H_0 : |\mu_{i1} - \mu_{i2}| < \log(3)$ and the alternative $H_1 : |\mu_{i1} - \mu_{i2}| > \log(3)$. This is a test for equivalence rather than strict equality between groups, which aims to detect isoforms differentially expressed by a biologically meaningful margin, *e.g.* as discussed by McCarthy and Smyth (2009). We note that `casper` outputs continuous isoform expression estimates rather than gene-

level counts, hence data analysis methods targetting continuous response variables are more appropriate that analyses for categorical data.

`casper` implements an empirical Bayes framework to test equivalence in expression across groups in function `probNonEquiv`. Briefly, `probNonEquiv` computes the posterior probability $P(|\mu_{i1} - \mu_{i2}| \mid \mathbf{y})$, where $\mathbf{y}$ is all the available data. The probability is based on the LNNMV model implemented in package `EBarrays` (Yuan et al., 2007; Yuan and Kendziorski, 2006), which has a similar formulation as limma (Smyth, 2004) except that information is shared for group means as well as the residual variance. We first proceed to the analysis, and subsequently perform some model checks.

We apply `probNonEquiv` to the MiSeq data alone, the combined MiSeq + observed HiSeq and the combined Miseq + simulated HiSeq data. In all analyses we restrict attention to genes with $\geq$10 reads.

```
> ppt0 <- probNonEquiv(gse37704.miseq, groups='group',
    logfc=log(3), minCount=10)
> head(ppt0)
   NM_032291 NM_001145277 NM_001145278    NM_018090
    1.52e-01     2.19e-10     9.71e-02     1.08e-04
   NM_052998 NM_001080397
    6.97e-04     2.96e-06
> ppt1 <- probNonEquiv(xadj, groups='group',
    logfc=log(3), minCount=10)
> head(ppt1)
   NM_032291 NM_001145277 NM_001145278    NM_018090
    9.17e-03     8.20e-05     8.61e-04     1.27e-10
   NM_052998 NM_001080397
    3.26e-07     1.37e-13
> pp <- probNonEquiv(xnewadj, groups='group',
    logfc=log(3),minCount=10, mc.cores=2)
> head(pp)
                 sim1     sim2     sim3     sim4     sim5
NM_032291    5.17e-02 5.59e-02 5.01e-02 5.85e-03 1.08e-02
NM_001145277 1.16e-20 2.05e-21 2.71e-17 4.54e-17 3.18e-15
NM_001145278 4.25e-02 1.01e-02 3.86e-01 3.83e-03 2.25e-02
NM_018090    3.36e-05 3.31e-07 4.82e-08 2.10e-04 5.03e-07
NM_052998    6.86e-10 9.55e-05 3.59e-05 3.51e-05 4.19e-10
NM_001080397 3.77e-03 4.73e-06 4.20e-15 2.54e-07 3.85e-11
```

Suppose we declare as differentially expressed all genes with posterior probability >0.95, which guarantees that the posterior expected False Discovery Proportion (the Bayesian counterpart of the FDR) is below 0.05 (Müller

et al., 2004). The function `getRoc` computes operating characteristics for each simulation.

```
> n0obs <- sum(ppt0>.95,na.rm=TRUE)
> n1obs <- sum(ppt1>.95,na.rm=TRUE)
> n1sim <- colSums(pp>.95,na.rm=TRUE)
> n0obs
[1] 640
> n1obs
[1] 870
> mean(n1sim)
[1] 855
> n1sim
sim1 sim2 sim3 sim4 sim5
 859  857  862  855  842
> oc <- getRoc(abs(logfc.true)>log(3),pp>0.95)
> oc
      tp fp    tn   fn   p    fdr   pow
sim1 842 17 38687 1261 859 0.0198 0.400
sim2 836 21 38724 1241 857 0.0245 0.403
sim3 845 17 38711 1231 862 0.0197 0.407
sim4 836 19 38723 1239 855 0.0222 0.403
sim5 822 20 38709 1281 842 0.0238 0.391
> colMeans(oc)
      tp       fp       tn       fn        p      fdr
8.36e+02 1.88e+01 3.87e+04 1.25e+03 8.55e+02 2.20e-02
     pow
4.01e-01
```

There are 640 differential expression calls using the MiSeq data and 870 using the combined MiSeq and observed HiSeq data. The predicted number of calls based on the simulations is 855, in good agreement the observed data. The mean false discovery proportion in the posterior predictive simulations is 0.022, and the mean power 0.401. Additionally to the point estimates given by `colMeans(oc)`, we may portray the uncertainty with credibility intervals or straightforward plots. The following code produces Figure 4, which compares observed with posterior predictive number of DE calls (3 samples per group indicates MiSeq data alone was used, 6 samples per group corresponds to 3 Miseq + 3 HiSeq). The observed DE calls with 6 samples (black) falls within the range of corresponding posterior predictive draws (grey), suggesting that uncertainty was adequately portrayed in the simulation.

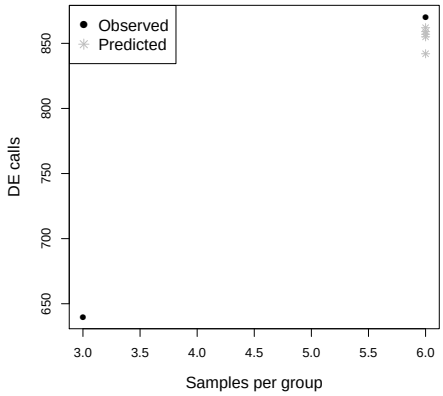

Figure 4: Number of isoforms with $P(|\mu_{i1} - \mu_{i2}| > \log(2) \mid \mathbf{y}) > 0.95$ for observed data (black) and posterior predictive simulations (grey).

```
> ylim <- range(c(n0obs,n1obs,n1sim))
> plot(c(3,6),c(n0obs,n1obs),ylim=ylim,pch=16,ylab='DE calls',
    xlab='Samples per group',cex.lab=1.25)
> points(rep(6,length(n1sim)),n1sim,col='gray',pch=8)
> legend('topleft',c('Observed','Predicted'),pch=c(16,8),
    col=c('black','gray'),cex=1.25)
```

Here we used `probNonEquiv`, but the posterior predictive simulations can help assess the results one would get with any other suitable analysis strategy. As an illustration, we now perform equivalence tests using limma one-side P-values. This test is implemented in function `treat` in package `limma` McCarthy and Smyth (2009). Given that `xadj` contains a list of `ExpressionSet` objects, one need only apply the desired analysis method to each element in `xadj`. For convenience, we included a wrapper `pvalTreat` in `casper` that calls `treat` and adjusts the resulting P-values using any method available in `p.adjust`.

```
> pvalt0 <- pvalTreat(gse37704.miseq, groups='group',
    logfc=log(3), minCount=10, p.adjust.method='BH')
> pvalt1 <- pvalTreat(xadj, groups='group', logfc=log(3),
    minCount=10, p.adjust.method='BH')
> pvals <- pvalTreat(xnewadj, groups='group', logfc=log(3),
    minCount=10, p.adjust.method='BH', mc.cores=2)
```

The limma two-one test procedure followed by Benjamini-Hochberg P-value adjustment gives only 52 calls on the MiSeq data. On the experi-

mental MiSeq + HiSeq data the calls increase drastically to 366, again in good agreement with the posterior predictive simulations. The posterior predictive operating characteristics suggest that the procedure is indeed quite conservative, with an almost 0 proportion of false discoveries.

```
> n0obs <- sum(pvalt0<.05,na.rm=TRUE)
> n1obs <- sum(pvalt1<.05,na.rm=TRUE)
> n1sim <- colSums(pvals<.05,na.rm=TRUE)
> n0obs
[1] 52
> n1obs
[1] 366
> mean(n1sim)
[1] 335
> n1sim
sim1 sim2 sim3 sim4 sim5
 335  329  334  341  335
> getRoc(abs(logfc.true)>log(3),pvals<.05)
      tp fp    tn   fn   p fdr   pow
sim1 335  0 38704 1768 335   0 0.159
sim2 329  0 38745 1748 329   0 0.158
sim3 334  0 38728 1742 334   0 0.161
sim4 341  0 38742 1734 341   0 0.164
sim5 335  0 38729 1768 335   0 0.159
```

Both the LNNMV and limma models assume that individual expression estimates are normally distributed, plus some further (and less critical) hierarchical assumptions on the group means and residual standard deviation. Although such modelling assumptions are not expected to hold exactly, strong deviations would make the inference suspect. To address this issue, we perform several informal model checks. Function `qqnormGenomeWide` produces qq-normal plots for 1000 isoforms and overlays them on a single graph.

```
> qqnormGenomeWide(gse37704.miseq,ngenes=500)
> qqnormGenomeWide(xadj,ngenes=500)
```

Figure 5 shows the results for MiSeq data alone (left) and combined with HiSeq (right). No strong departures from a straight line are observed, therefore suggesting the normality assumption is reasonable. We note that some deviations are expected, *e.g.* isoforms with strong differential expression typically exhibit bimodality. A qq-normal plot on the residuals (*i.e.* deviations from group means) exhibits a similar pattern (data not shown).

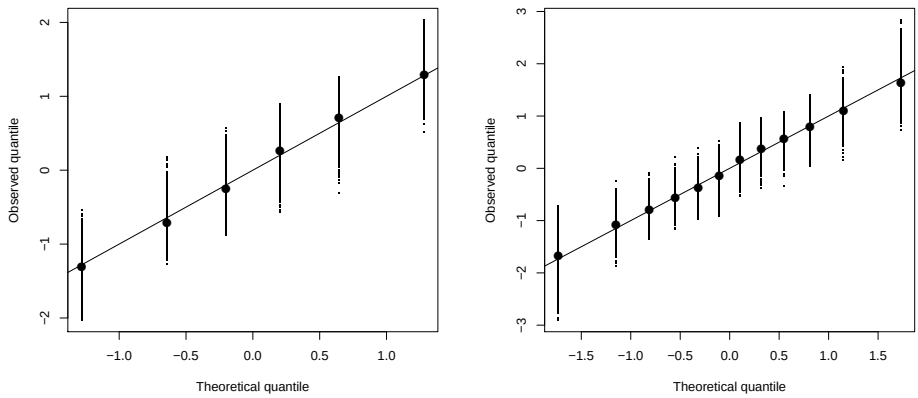

Figure 5: qq-normal plot for 500 isoforms. Left: MiSeq data; Right: combined MiSeq + HiSeq data

As a further assessment, we measure the degree of asymmetry in the data. The mean, median and inter-quartile range of the skewness coefficients are all centered around 0, suggesting that no strong asymmetries are present. Function `asymmetryCheck` produces a boxplot comparing observed asymmetry coefficients with those observed in data simulated under the Normality assumption (population means and variances are set equal to sample means and variances). Figure 6 shows the result.

```
> library(psych)
> sel <- fData(xadj)$readCount >= 10
> sk <- skew(t(exprs(xadj)))
> summary(sk)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -2.180  -0.305  -0.014   0.033   0.265   2.220
```

```
> asymmetryCheck(xadj)
```

`casper` also implements qq-gamma plots, which can help decide whether to use the LNNMV or the GaGa model wnen calling `simMultSamples`. Note that the Gamma distribution implies that values must be positive, a simple way to achieve that is to add an offset as shown below. Another option would be to take exponents, but we found that this option tends to produce outliers and that hence the offset option tends to produce more robust results.

```
> offset <- min(exprs(gse37704.miseq))
> exprs(gse37704.miseq) <- exprs(gse37704.miseq) - offset + 1
> qqgammaGenomeWide(gse37704.miseq,ngenes=500)
```

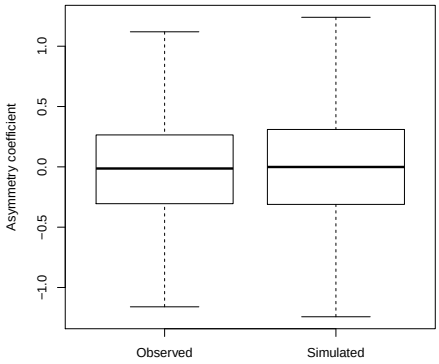

Figure 6: Output from `asymmetryCheck`

```
> offset <- min(exprs(xadj))
> exprs(xadj) <- exprs(xadj) - offset + 1
> qqgammaGenomeWide(xadj,ngenes=500)
```

Figure 7 shows the results for MiSeq data alone (left) and combined with HiSeq (right). The gamma distribution also seems reasonable as no strong departures from a straight line are observed. Although not shown here, we run `simMultSamples` setting `model='GaGa'` and obtained very similar results to those shown above with `model='LNNMV'`.

3 Session information and checksums

For reproducibility purposes, the session information is provided below. The md5 checksums for the files hosted at `https://sites.google.com/site/rosselldavid/home/myfiles` are

- gse37704.RData: 5043bc05bc76d0fdc8792b4a7b05604b
- gse37704_miseq.RData: 657880573413a08767ae6206d3db38a0
- hg19DB.RData: 8c45c52af4436db4a40a5b3109cbfcfe
- micebladder.rep1.RData: 724d3367a99901026d6f4e5e66dafcc1
- oneT.casper.RData: a55dbb7e0a15e39da1137de74a91f4fb

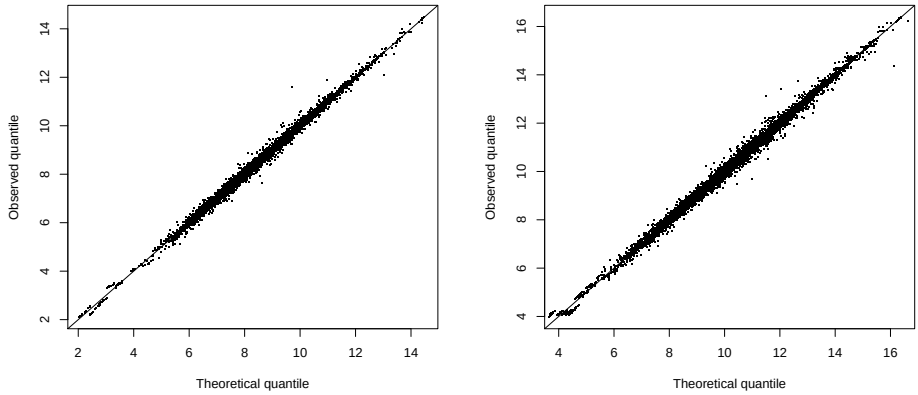

Figure 7: qq-gamma plot for 500 isoforms. Left: MiSeq data; Right: combined MiSeq + HiSeq data

```
> library(devtools)
> options(width = 120)
> session_info()
```

```
 setting  value
 version  R version 3.2.0 (2015-04-16)
 system   x86_64, darwin13.4.0
 ui       X11
 language (EN)
 collate  en_US.UTF-8
 tz       Europe/London

 package           * version  date       source
 AnnotationDbi       1.30.1   2015-04-26 Bioconductor
 Biobase           * 2.28.0   2015-04-17 Bioconductor
 BiocGenerics      * 0.14.0   2015-04-17 Bioconductor
 BiocParallel        1.2.2    2015-05-24 Bioconductor
 biomaRt             2.24.0   2015-04-17 Bioconductor
 Biostrings          2.36.1   2015-05-07 Bioconductor
 bitops              1.0-6    2013-08-17 CRAN (R 3.2.0)
 casper            * 2.3.1    2015-06-15 Bioconductor
 chron               2.3-45   2014-02-11 CRAN (R 3.2.0)
 cluster           * 2.0.1    2015-01-31 CRAN (R 3.2.0)
 coda                0.17-1   2015-03-03 CRAN (R 3.2.0)
```

```
colorspace          1.2-6    2015-03-11 CRAN (R 3.2.0)
curl                0.8      2015-06-06 CRAN (R 3.2.0)
DBI                 0.3.1    2014-09-24 CRAN (R 3.2.0)
devtools          * 1.8.0    2015-05-09 CRAN (R 3.2.0)
digest              0.6.8    2014-12-31 CRAN (R 3.2.0)
EBarrays            2.32.0   2015-04-17 Bioconductor
futile.logger       1.4.1    2015-04-20 CRAN (R 3.2.0)
futile.options      1.0.0    2010-04-06 CRAN (R 3.2.0)
gaga                2.15.1   2015-06-14 Bioconductor
GenomeInfoDb      * 1.4.0    2015-04-17 Bioconductor
GenomicAlignments   1.4.1    2015-04-24 Bioconductor
GenomicFeatures     1.20.1   2015-05-06 Bioconductor
GenomicRanges     * 1.20.4   2015-05-30 Bioconductor
ggplot2           * 1.0.1    2015-03-17 CRAN (R 3.2.0)
git2r               0.10.1   2015-05-07 CRAN (R 3.2.0)
gsubfn              0.6-6    2014-08-27 CRAN (R 3.2.0)
gtable              0.1.2    2012-12-05 CRAN (R 3.2.0)
gtools              3.5.0    2015-05-29 CRAN (R 3.2.0)
IRanges           * 2.2.3    2015-06-02 Bioconductor
labeling            0.3      2014-08-23 CRAN (R 3.2.0)
lambda.r            1.1.7    2015-03-20 CRAN (R 3.2.0)
lattice             0.20-31  2015-03-30 CRAN (R 3.2.0)
limma               3.24.7   2015-06-07 Bioconductor
magrittr            1.5      2014-11-22 CRAN (R 3.2.0)
MASS                7.3-40   2015-03-21 CRAN (R 3.2.0)
Matrix              1.2-1    2015-06-01 CRAN (R 3.2.0)
memoise             0.2.1    2014-04-22 CRAN (R 3.2.0)
mgcv                1.8-6    2015-03-31 CRAN (R 3.2.0)
mnormt              1.5-3    2015-05-25 CRAN (R 3.2.0)
munsell             0.4.2    2013-07-11 CRAN (R 3.2.0)
nlme                3.1-120  2015-02-20 CRAN (R 3.2.0)
plyr                1.8.2    2015-04-21 CRAN (R 3.2.0)
proto               0.3-10   2012-12-22 CRAN (R 3.2.0)
psych             * 1.5.4    2015-04-27 CRAN (R 3.2.0)
Rcpp                0.11.6   2015-05-01 CRAN (R 3.2.0)
RCurl               1.95-4.6 2015-04-24 CRAN (R 3.2.0)
reshape2            1.4.1    2014-12-06 CRAN (R 3.2.0)
Rsamtools           1.20.4   2015-06-01 Bioconductor
RSQLite             1.0.0    2014-10-25 CRAN (R 3.2.0)
rtracklayer         1.28.4   2015-05-28 Bioconductor
rversions           1.0.1    2015-06-06 CRAN (R 3.2.0)
```

```
S4Vectors         * 0.6.0    2015-04-17 Bioconductor
scales             0.2.4    2014-04-22 CRAN (R 3.2.0)
sqldf              0.4-10   2014-11-07 CRAN (R 3.2.0)
stringi            0.4-1    2014-12-14 CRAN (R 3.2.0)
stringr            1.0.0    2015-04-30 CRAN (R 3.2.0)
survival           2.38-1   2015-02-24 CRAN (R 3.2.0)
VGAM               0.9-8    2015-05-11 CRAN (R 3.2.0)
XML                3.98-1.2 2015-05-31 CRAN (R 3.2.0)
xml2               0.1.1    2015-06-02 CRAN (R 3.2.0)
XVector            0.8.0    2015-04-17 Bioconductor
zlibbioc           1.14.0   2015-04-17 Bioconductor
```

References

R. Edgar, V. Domrachev, and A.E. Lash. Gene expression omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research*, 30:207–210, 2002.

R.C. Gentleman, V.J. Carey, D.M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A.J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J.Y.H. Yang, and J. Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004. URL `http://genomebiology.com/2004/5/10/R80`.

H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup. The sequence alignment/map (SAM) format and SAMtools. *Bioinformatics*, 25(16):2078–9, 2009.

D.J. McCarthy and G.K. Smyth. Testing significance relative to a fold-change is a TREAT. *Bioinformatics*, 25(6):765–771, 2009.

P. Müller, G. Parmigiani, C. Robert, and J. Rousseau. Optimal sample size for multiple testing: the case of gene expression microarrays. *Journal of the American Statistical Association*, 99:990–1001, 2004.

D. Rossell. GaGa: a simple and flexible hierarchical model for differential expression analysis. *Annals of Applied Statistics*, 3:1035–1051, 2009.

D. Rossell, C. Stephan-Otto Attolini, M. Kroiss, and A. Stöcker. Quantifying alternative splicing from paired-end rna-seq data. *Annals of Applied Statistics*, 8(1):309–330, 2014.

L. J. Savage. *The Foundations of Statistics*. Dover Publications (1972 ed), 2 revised edition, 1954. ISBN 0486623491.

G.K. Smyth. Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 3:Number 1, Article 3, 2004.

C. Trapnell, L. Pachter, and S.L. Salzberg. TopHat: discovering splice junctions with RNA-seq. *Bioinformatics*, 25(9):1105–1111, 2009.

C. Trapnell, D.G. Hendrickson, M. Sauvageau, L. Goff, J.L. Rinn, and L. Pachter. Differential analysis of gene regulation at transcript resolution with RNA-seq. *Nature Biotechnology*, 31(1):46–53, 2013. doi: 10.1038/nbt.2450.

M. Yuan and C. Kendziorski. A unified approach for simultaneous gene clustering and differential expression identification. *Biometrics*, 62:1089–1098, 2006.

Ming Yuan, Michael Newton, Deepayan Sarkar, and Christina Kendziorski. *EBarrays: Unified Approach for Simultaneous Gene Clustering and Differential Expression Identification*, 2007. R package version 2.24.0.