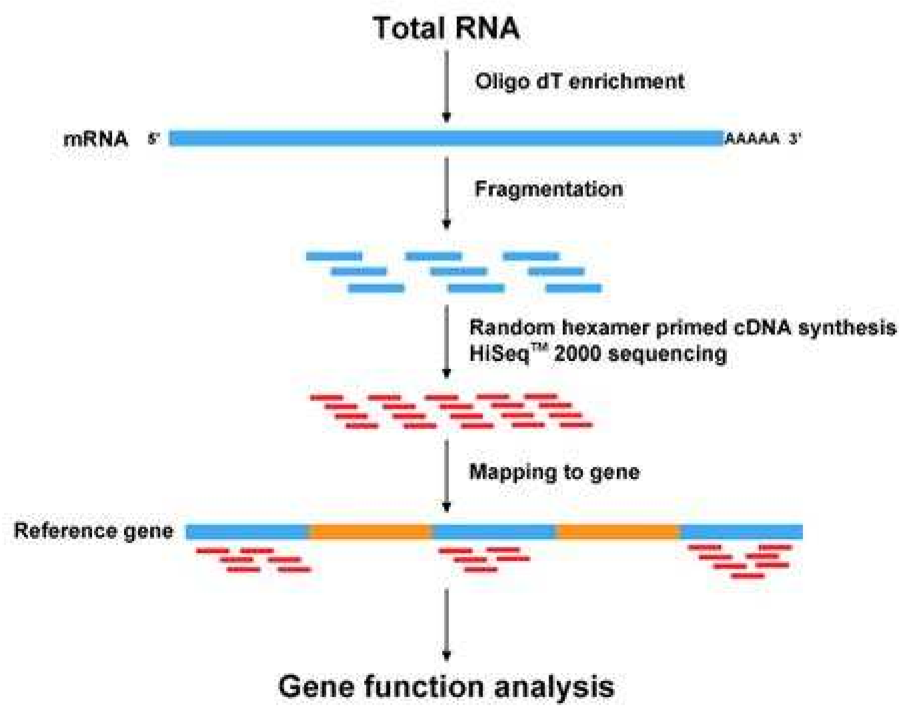
Fungi Net	SOP	Responsible Authors: Steffen Priebe, Jörg Linde, Reinhard Guthke HKI Jena Date: 2.11.2012

Processing of RNA-Seq Data

Contents

1. Next Generation Sequencing and RNA-Seq Data

RNA-Seq is a method to measure the transcript/gene expression by adaption of High-throughput sequencing technologies (Next Generation Sequencing – NGS). In contrast to microarrays where the measurement is based on a hybridization process, RNA-Seq sequences cDNA in order to get information about a sample's RNA content.



2. R packages for NGS data analysis

By now there are already several R packages for the analysis and handling of short read data. There are some other packages that are also very useful when dealing with NGS data. Here, a short overview:

Data and annotation I/O:

- *ShortRead* (http://www.bioconductor.org/help/bioc-views/release/bioc/html/ShortRead.html)
  and *Rsamtools* (http://www.bioconductor.org/help/bioc-views/release/bioc/html/Rsamtools.html)
  - Base classes, functions, and methods for representation of high-throughput, short-read sequencing data. File I/O, quality assessment, and high-level, general purpose data summary.

- *Iranges* (http://www.bioconductor.org/help/bioc-views/release/bioc/html/IRanges.html),
  GenomicRanges (http://www.bioconductor.org/help/bioc-views/release/bioc/html/GenomicRanges.html
  and *genomeIntervals* (http://www.bioconductor.org/help/bioc-views/release/bioc/html/genomeIntervals.html)

- range-based (e.g., chromosomal regions) calculation, data manipulation, and general-purpose data representation

- *Biostrings* (http://www.bioconductor.org/help/bioc-views/release/bioc/html/Biostrings.html)
  - alignment, pattern matching (e.g., primer removal), and data manipulation of large biological sequences or sets of sequences.

- rtracklayer (http://www.bioconductor.org/help/bioc-views/release/bioc/html/rtracklayer.html)
  - Interacting with multiple genome browsers, manipulating/export/import annotation tracks in various formats (currently GFF, BED, bedGraph, BED15, and WIG built-in)
- biomaRt, Interface to BioMart databases (e.g. Ensembl, COSMIC ,Wormbase and Gramene)

- GenomicFeatures - Annotation of sequence features across common genomes

- BSgenome - Infrastructure for Biostrings-based genome data packages like BSgenome.Celegans.UCSC.ce6 of C.elegans or BSgenome.Mmusculus.UCSC.mm9 for mouse

Finding of differentially expressed genes (DEG):

- edgeR - Empirical analysis of digital gene expression data in R

- DESeq - Digital gene expresion analysis based on the negative binomial distribution

- baySeq - Empirical Bayesian analysis of patterns of differential expression in count data

- DEGSeq - Identify Differentially Expressed Genes from RNA-seq data

- MAIDscore - In the case of no replicates you can only uye the (log) foldchange as criterium. In this case us can use a constant cut-off, or use an exponential decreasing cutoff -as it is done with the MAID score.

Other useful things:

- gplots - Various R programming tools for plotting data, e.g. a nice heatmap (heatmap.2)

- multicore - Parallel processing of R code on machines with multiple cores or CPUs. Provides a parallel version of lapply called mclapply (very easy to use)

3. Data analysis

There are some basic steps in the analysis of NGS/RNA-Seq data. The first steps are probably the mapping and the quality assessment of the short reads.

3.1 Quality Control

Adapter and quality trimming

In case that there are the sequencing adapters still included in the FASTQ files or for the trimming of low quality read regions, one can use Btrim
http://seqanswers.com/wiki/Btrim

See the manual here: http://graphics.med.yale.edu/trim/readme

This will trim all reads if there is a window of 5bp detected with average quality of 15:

```
Btrim64 -t example.fq -o output.fq -q -w 5 -a 15
```

In case of paired-end each fastq file for each end needs to be trimmed separately. Afterwards, the
http://graphics.med.yale.edu/trim/paired_end_trim.pl
can be used to synchronize both trimmed files.

Trim FASTQ file "input_sequence.txt" using the adapters in "adapters.txt", write the output in "output.txt". Each line in "adapters.txt" contains two tab-delimited columns: the first is 5'-adapter, the second 3'-adapter:

```
Btrim64 -t input_sequence.txt -p adapters.txt -o output.txt
```

Read quality assessment

The ShortRead package provides a quality assessment function **qa.**
It can automatically create a HTML file with some plots and information about the qualities of your reads. If you use this function for several files, you can do this in parallel by loading the multicore package first. But be careful: This will need a lot of free memory (about twice of the file size of each file).

```
library(ShortRead) #installed on ilios for R212
#library(multicore) #uncomment this to use several cores
qa1 <- qa(dirPath=getwd(),pattern="reads.fastq",type="fastq")
qa2 <- qa(getwd(),pattern="export.txt",type="SolexaExport")
rpt <- report(qa1,destfile="qa1.html",type="html")
browseURL(rpt)
save(qa1,file="qa1.Rdata")
```

There is also a java based tool FASTQC
(http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/)
for quality control, which creates a quality report in HTML format.

3.2 Mapping

For this task the mapping tool *Bowtie* can be applied. For reads >50bp *Bowtie2* might give better results.

Note: Usually for mapping FASTQ files are required. If only BAM files are available that need to be remapped, it is possible to extract the reads from that again to create a FASTQ file with *bam2fastq* which is available at http://www.hudsonalpha.org/gsl/software/bam2fastq.php:

```
bam2fastq s_1.af1163.bowtie.25.sorted.bam
```

3.3 Mapping Quality

The tool *Samstats* produces visualization of the mapping results in a html file.

http://samstat.sourceforge.net/
```
ngsdata/ngs/samstat/src/samstat <file.sam>
```

It can also handle BAM files. For a (very) short summary samtools flagstat could be used.

```
samtools flagstat accepted_hits.bam
```

NOTE: *Tophat* (version 1 and 2) does not write unmapped reads into the sam file. For this reason, the statistics will always show 100% of reads being mapped. To find the number of reads which have not mapped do a wc -l of the fastq file minus wc -l of the sam file (without header)

3.4 Read in aligned reads

The *ShortRead* package is able to read several input formats, like the Bowtie output or Solexa-Export-files from the Illumina sequencing machine.

```
library(ShortRead)
aln<- readAligned(dirPath="/home/user/", pattern="bowtie_out",
type="Bowtie")
#get table of chromosome names:
table(chromosome(aln))
```

3.5 Read in aligned reads 2

Another way to read in all Tophat-mappings (BAM-files) at the same time automatically works with the yet unpublished RNA-Seq package File:RNAseq 1.0.55.tar.gz.zip (you have to unzip it first) of Nicolas Delhomme which I (smueller) got from him directly. This also takes care of producing counts and RPKM values as follows:

```
library(RNAseq)
count.tableobj <-
easyRNAseq(filesDirectory="/home/smueller/data/analysen/vito/raw_data
/tophat1163",
                              organism="afumigatusa1163_eg_gene",
                              chr.sizes=chr.size,
                              readLength=36L,
                              annotationMethod="env",
                         annotationObject=exon.range,
                              format="bam",
                         outputFormat="RNAseq",
                              count="transcripts",
                         validity.check=FALSE,
                              pattern=".bam$")
```

· chr.sizes can be obtained like the following:

```
genome <-
read.DNAStringSet("/opt/appz/ngs/bowtie/genomes/Aspergillus_fumigatus
a1163.CADRE.dna.toplevel.fa")
names(genome) <- substr(names(genome),1,8)
chr.size <- width(genome)
```

- The BAM files (and only the required ones!) should be in "filesDirectory". They can be produced by Tophat.
- The organism should be choosen according to BioMart, note,

We had trouble to get this to work for Aspergillus, I ended up to modify the getBmRange function (you have to adapt the bold part according to your critter):

```
getBmRange <- function (organism = character(1), ...)
{
   ensembl <- useMart("fungi_mart_10")
   if (organism == character(1)) {
        stop(paste("To use the biomaRt functionnalities, we need an
organism name. Set it using the organism() function."))
   }
   ensembl = useDataset(organism, ensembl)
   exon.annotation <- getBM(c("ensembl_gene_id", "strand",
"ensembl_transcript_id",
        "chromosome_name", "ensembl_exon_id", "exon_chrom_start",
        "exon_chrom_end"), mart = ensembl, ...)
   return(RangedData(IRanges(start =
exon.annotation"exon_chrom_start",
        end = exon.annotation"exon_chrom_end"), space =
exon.annotation"chromosome",
        strand = exon.annotation"strand", exon =
exon.annotation"ensembl_exon_id",
        transcript = exon.annotation"ensembl_transcript_id", gene =
exon.annotation"ensembl_gene_id",
        universe = organism))
}
```

- exon.range contains the gene annotation, see the Biomart SOP.

3.6 Coverage vectors

Aligned Read objects can be transformed into coverage objects. Therefore you will need a vector including all chromosomes and their length. For the most common organism you can just load this information using the BSgenome library for this critter. You can then create WIG (Wiggle) Files but only for single chromosomes, which can be viewed with several genome browsers. In advantage to BAM/SAM files their filesize is much smaller.

3.6.1 WIG

```
#aln is an alignedRead object
library(BSgenome.Mmusculus.UCSC.mm9)
chrSizes <-seqlengths(Mmusculus)
cov <- coverage(aln, width=chrSizes)
chrom <- "chr17"
gd <- GenomicData(findRange(start(cov[[chrom]]),cov[[chrom]]),
score=runValue(cov[[chrom]]), chrom=chrom)
export(gd, con="chromsome17.wig")
```

You can do this also for all chromosomes at once and append the wig files afterwards:

```
filenames <- ""
for (chrom in names(cov)) {
  gd <- GenomicData(findRange(start(cov[[chrom]]),cov[[chrom]]),
score=runValue(cov[[chrom]]), chrom=chrom)
  export(gd, con=paste("chr",chrom,".wig",sep=""))
  filenames <- paste(filenames, paste("chr",chrom,".wig",sep=""))
}
```

```
system(paste("cat", filenames, " > allchromosome.wig"))
#system(paste("rm", filenames))
```

In some cases this error was produced by the export command: (R version 2.13.0, rtracklayer_1.12.0)

```
Fehler in FUN(X[i], ...) :
The span must be uniform for Wiggle export. Consider bedGraph or
bigWig as alternatives.
```

We don't know what's the reason for this, but you can try to create a bigWig file (bw):

```
export(gd, con="chromsome8.bw", seqlengths=chrSizes)

#or like this for all contigs:
for (chrom in names(cov)) {
 i <- which(names(cov)==chrom)
 gdtmp <- GenomicData(findRange(start(cov[[chrom]]),cov[[chrom]]),
score=runValue(cov[[chrom]]), chrom=chrom)
 if (i==1) { gd <- gdtmp }
 else { gd <- append(gd,gdtmp) }
}
export(gd, con="all_chromsomes.bw", seqlengths=chrSizes)
```

There is also another way without using R. With this code you can create one WIG file with all the chromosomes which are contained in your sorted BAM file:

```
samtools pileup bamfile-sorted.bam | perl -ne 'BEGIN{print "track
type=wiggle_0 name=fileName description=fileName\n"};($c, $start,
undef, $depth) = split; if ($c ne $lastC) { print "variableStep
chrom=$c\n"; };$lastC=$c;next unless $. % 10 ==0;print
"$start\t$depth\n" unless $depth<3;'  > fileName.wig
```

3.6.2 bigwig

If the WIG file can not be displayed in a browser like IGV, there are other coverage formats like bigWig. BigWig files are in an indexed binary format, so for large data sets bigWig is considerably faster than regular WIG files.
Tools used: samtools (http://sourceforge.net/projects/samtools/files/), genomeCoverageBed (part of BEDtools (http://code.google.com/p/bedtools/downloads/list)), bedGraphToBigWig (http://hgdownload.cse.ucsc.edu/admin/exe/) They should all be installed on fengari in /opt/appz/ngs/
More Information : http://cancan.cshl.edu/labmembers/gordon/files/viz2.pdf

Here is the workflow to create a bigWig file from SAM/BAM format:

```
# 1. Convert SAM to BAM (see also the SAMtools article)
samtools view -S -b -o sample.bam sample.sam
# 2. Sort the BAM file
samtools sort sample.bam sample.sorted
# 3. Create BedGraph coverage file
# (chromsizes.txt should look like this:
chr1 \t 3893948
chr2 \t 3444324 ...
chrX \t  232433 )
genomeCoverageBed -bg -ibam sample.sorted.bam -g chromsizes.txt >
sample.bedgraph
# 4. Convert the BedGraph file to BigWig
bedGraphToBigWig sample.bedgraph chromsizes.txt sample.bw
```

The bigWig file (sample.bw) can now be loaded with the IGV.

3.7 Assignment to genes with R

First you have to select an annotation source. If you are dealing with common model critters you can directly download the UCSC or the Ensembl annotation using the *rtracklayer* or *biomaRt* package in R. Generally you must be sure to use an annotation which fits to the reference sequence which was used for the read mapping! This is important, otherwise you will assign the wrong stuff to your mapping!

3.7.1 Get annotation

Here an example for downloading the annotation for Mouse:

3.7.1.1 UCSC

UCSC provides the RefSeq gene annotation. Genes and transcript are in the RefSeq format, e.g. like this: NM_153661.

```
library(rtracklayer)
species="mm9" #the current Mouse genome version which was also used
for mapping
session <- browserSession()   #Creates a session
#head(ucscGenomes())          #List available genomes from UCSC
genome(session) <- species    #Set up a genome object
#head(trackNames(session),20)       #List available tracks
query <- ucscTableQuery(session,"RefSeq Genes")      #Generate a
query for UCSC
#tracktable <- getTable(query) #Retrieve the UCSC track as data.frame
(e.g. for csv export)
track_species <- track(query)  #this is the object you need
```

3.7.1.2 Ensembl

Annotation can also be downloaded from Biomart. Here is the code for downloading the gene, transcript and exon information for mouse. See also the biomart articel for more infos.

```
library(biomaRt)
ensembl <- useMart("ensembl",dataset="mmusculus_gene_ensembl")
exon.annotation<-getBM(c("ensembl_gene_id",
                        "external_gene_id",
                        "strand",
                        "ensembl_transcript_id",
                        "chromosome_name",
                        "ensembl_exon_id",
                        "exon_chrom_start",
                        "exon_chrom_end"),
                      mart=ensembl,
                      filters="chromosome_name",
                      values=c(1:19,"X","Y","MT"))
exon.annotation$chromosome <- paste("chr",
exon.annotation$chromosome_name, sep="") #adjust the chromosome names
to your reference seq.
exon.annotation$chromosome[exon.annotation$chromosome=="chrMT"] <-
"chrM"
exon.range <- RangedData(IRanges(
                         start=exon.annotation$exon_chrom_start,
                         end=exon.annotation$exon_chrom_end),
                         space=exon.annotation$chromosome,
                         strand=exon.annotation$strand,

transcript=exon.annotation$ensembl_transcript_id,

ensembl_gene_id=exon.annotation$ensembl_gene_id,
                         gene=exon.annotation$external_gene_id,
                         universe = NULL #"ce2" #not needed?
                         )
track_species <- exon.range
```

3.7.2 Assign annotation

Now you can assign your annotation track to the coverage, created from the alignedRead object:

```
    track_species_select <- track_species[names(track_species) %in%
    names(cov)] #to make sure to assign only chromosomes which are in
    both, the coverage and annotation object
    cov_select <- cov[names(cov) %in% names(track_species)]
    exon.coverage <-
    aggregate(cov_select[match(names(track_species_select),names(cov_sele
    ct))], ranges(track_species_select), sum)
    exon.coverage <- ceiling(exon.coverage/unique(width(aln2)))
```

Now the reads (coverage) is assigned to the exons. In order to get a gene/transcript table you need to summarize them:

```
    #for UCSC:
    "summarize.by.names" <- function(sample,annotation) {
      "func" <- function(chr,sample,annotation) {
         counts<-
    stats:::aggregate(sample[[chr]],list(transcript=annotation[chr]$name)
    ,sum)
      }
      genes <-
    do.call(rbind,lapply(names(sample),func,sample,annotation))
      colnames(genes)[2] <- "counts"
      return(genes)
    }
    genes_genome <- summarize.by.names(exon.coverage, track_species)
    #names are transcripts (RefseqIDs)


    #for Ensembl annotation:
    "summarize.by.transcripts" <- function(sample,annotation){
       transcripts <-
    do.call(rbind,lapply(names(sample),function(chr,sample,annotation){
         counts<-
    stats:::aggregate(sample[[chr]],list(transcript=annotation[chr]$trans
    cript),sum)
       },sample,annotation))
       colnames(transcripts)[2] <- "counts"
       return(transcripts)
    }
    "summarize.by.ensembl_gene_id" <- function(sample,annotation){
       genes <-
    do.call(rbind,lapply(names(sample),function(chr,sample,annotation){
         counts<-
    stats:::aggregate(sample[[chr]],list(transcript=annotation[chr]$ensem
    bl_gene_id),sum)
       },sample,annotation))
       colnames(genes) <- c("gene","counts")
       return(genes)
    }
    transcripts <- summarize.by.transcripts(exon.coverage,track_species)
    genes <- summarize.by.ensembl_gene_id(exon.coverage,track_species)
```

3.8 Assignement to annotation with Htseq

You can also combine your mapped reads directly with an annotation file (e.g. GTF) using the HTSeq python scripts from Simon Anders, available here: [1]. To use HTSeq, Phython (>=2.5) including the numpy package must be installed.

Run HTSeq:

```
htseq-count [options] <sam_file> <gff_file>
```

Options:

```
 -m MODE, --mode=MODE  mode to handle reads overlapping more than one
                       feature(choices: union, intersection-strict,
                       intersection-nonempty; default: union)
 -s STRANDED, --stranded=STRANDED
                       whether the data is from a strand-specific
assay.
                       Specify 'yes', 'no', or 'reverse' (default:
yes).
                       'reverse' means 'yes' with reversed strand
                       interpretation
 -a MINAQUAL, --minaqual=MINAQUAL
                       skip all reads with alignment quality lower
than the
                       given minimum value (default: 0)
 -t FEATURETYPE, --type=FEATURETYPE
                       feature type (3rd column in GFF file) to be
used, all
                       features of other type are ignored (default,
suitable
                       for Ensembl GTF files: exon)
 -i IDATTR, --idattr=IDATTR
                       GFF attribute to be used as feature ID
(default,
                       suitable for Ensembl GTF files: gene_id)
 -o SAMOUT, --samout=SAMOUT
                       write out all SAM alignment records into an
output SAM
                       file called SAMOUT, annotating each line with
its
                       feature assignment (as an optional field with
tag
                       'XF')
 -q, --quiet           suppress progress report and warnings
```

For more infos see: Counting reads in features with htseq-count
(http://www-huber.embl.de/users/anders/HTSeq/doc/count.html#usage)

3.9 Counting/RPKM normalization

- results of the alignment/mapping process are raw-counts, assigned to genes or exons
- not comparable, due to different library sizes or/and sequencing depth
- RPKM measure is a normalization for this effects (Peads Per Kilobase of exon model per Million mapped reads) [Mortazavi et al., 2008]:

<math>RPKM = \frac{10^9 * C}{N*L}</math>

```
RPKM = 10^9 * C / (N*L)
C = number of mappable reads to exons
N = total number of mappable reads
L = sum of exon lengths (in bp)
```

Using the RNAseq package above, counts and RPKM values can be obtained as follows:

```
count.table <- readCounts(count.tableobj)\[\[1\]\]
rpkmcad <- RPKM(count.tableobj,from=,"transcripts")
```

3.10 Finding of DEG

There are several packages to find differential expressed genes (DESeq, DEGseq, edgeR, BaySeq).
Generally you will need replicates for this test. Only DESeq (eVFmethod parameter) is able to find DEG without any replicates.

3.10.1 edgeR

Robinson MD, McCarthy DJ and Smyth GK (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26, 139-140
Wheras for microarray data analysis with real valued expression intensities after logarithmization a normal distrubution is assumed and t-test is widely used for DEG identification, this is not the case for count data that are discrete and the their distribution is skewed. Thus, Negative Binomial distribution NB(E_g,i,D) is assumed with the mean E (specific for the gene g and the sample i) and the Dispersion D (http://de.wikipedia.org/wiki/Negative_Binomialverteilung). Empirical Bayes procedure is used to moderate (to shrink) the dispersion D across genes. Differential expression is assessed for each gene by a p-value using an exact test analogous to Fisher’s exact test, but adapted for overdispersed data. ("Overdispersion" means: the variance can be greater than the mean.)

```
"runedgeR" <- function(counts, conds, remove.zeros=FALSE) {
   dge <- DGEList(counts=counts, group=conds, remove.zeros =
remove.zeros)
   d   <- estimateCommonDisp(dge)
   de.common <- exactTest(d)
   res={}
   res$dge <- dge
   res$d   <- d
   res$de.common <- de.common
   res$padj <- p.adjust(de.common$table$p.value, method = "BH")
   return(res)
}
```

3.10.2 DESeq

Anders S, Huber W. Differential expression analysis for sequence count data. Genome Biol. 2010;11(10):R106. Epub 2010 Oct 27. PubMed PMID: 20979621; PubMed Central PMCID: PMC3218662.

Model: DESeq owes its basic idea to edgeR, yet differs in several aspects. Negative Binomial distribution and Fisher's Test is also used. However, the modeling of the Dispersion is different and uses some assumptions for estimation of the estimation of gene-specific dispersion D_g.

```
"runDESeq" <- function(genes, conds, cond1=unique(conds)[1],
cond2=unique(conds)[2], eVFmethod="normal") {
   cds=newCountDataSet(countData=genes, conditions=conds)
   cds <- estimateSizeFactors( cds )
   #sizeFactors( cds )
   cds <- estimateVarianceFunctions(cds, method=eVFmethod)
   res <- nbinomTest( cds, cond1, cond2 )
   return(res)

}
```

3.10.3 BaySeq

Hardcastle TJ, Kelly KA. baySeq: empirical Bayesian methods for identifying differential expression in sequence count data. BMC Bioinformatics. 2010 Aug 10;11:422. PubMed PMID: 20698981; PubMed Central PMCID: PMC2928208.

The method assumes a negative binomial distribution for the count data (as also in edgeR and DESeq), but derives an empirically determined prior distribution from the entire dataset.
BaySeq it the preferred method for experimental designs involving multiple sample groups (i.e. not only two, where edgeR and DESeq can be applied too)
Using the obove obtained count.table object, one could go like this (this is the quick'n dirty way):

```
library(baySeq)
library(snow)
cl <- makeCluster(16, "SOCK")
#calculating robust library sizes (note: there are other ways to do
that!)
libsizes <- apply(count.table,2, function(x) sum(x[x <=
quantile(x,0.75)]))
#here you have to specify your hypothesis (see the manual!):
groups <- list(NDE = c(1,1,1,1,1,1),DE=c(1,1,1,2,2,2))
CD <- new("countData", data = count.table, replicates =
c(1,1,1,2,2,2),libsizes = as.integer(libsizes), groups = groups)
CD@annotation <- data.frame(name = rownames(count.table))
plotMA.CD(CD, samplesA = 1:3, samplesB = 4:6)
CDP.NBML <- getPriors.NB(CD, samplesize = 20000, estimation = "QL",cl
= cl)
CDPost.NBML <- getLikelihoods.NB(CDP.NBML, pET = "BIC", cl = cl)
stopCluster(cl)
topCounts(CDPost.NBML, group = 2)
```

Here is a summarized calling function for baySeq:

```
"run.baySeq" <- function(genetable, replicates, libsizes, groups,
seglens, samplesize = 1000, nb_cpu=1) {
   if (nb_cpu==1) {
     cl <- NULL
   }
   else {
     library(snow)
     cl <- makeCluster(nb_cpu, "SOCK")
   }
   simCount <- as.matrix(genetable)
   CD <- new("countData", data = simCount, replicates = replicates,
libsizes = as.integer(libsizes), groups = groups, seglens=seglens)
   #plotMA.CD(CD, samplesA = 1:5, samplesB = 6:10, col =
c(rep("red",100), rep("black", 900)))
   CD@annotation <- data.frame(name = rownames(genetable))
   #nba:
   CDP.NBML <- getPriors.NB(CD, samplesize = samplesize, estimation =
"QL", cl = cl)
   CDPost.NBML <- getLikelihoods.NB(CDP.NBML, pET = "BIC", cl = cl)
   CDPost.NBML@estProps
   #tC=topCounts(CDPost.NBML, group = 2, number=dim(CDPost.NBML)[1])
   #return(tC)
   return(CDPost.NBML)
}
```

3.10.4 NoiSeq

Noiseq is a novel tool for the prediction of DEGs. It is able to handle non-replicated data. It can only compare two groups of genes. For more information see NOTE: Noiseq is NOT working with a p-value (even though they call it pvalue). Instead it is

something like odds. See turtorial for details
http://bioinfo.cipf.es/noiseq/doku.php?id=tutorial A small wrapper function:

```
source("/ngsdata/ngs/noiseq.r")
"runnoiseq" <- function(counts, conds,long,rep.type="bio", k=0.5,
norm="rpkm",odds=0.9,lc=1) {
  #counts count matrix
  #cons vector of maximal two different conditions for each column of
count
  #long named vector of length of genes
  #other parameters see ?noiseq or tutorial
  count_cond1=counts[,which(conds==unique(conds)[1])]
  count_cond2=counts[,which(conds==unique(conds)[2])]
  myresults <- noiseq(count_cond1, count_cond2, repl = rep.type, k =
k, norm = norm, long = long, q =odds, lc = lc)
  res={}
  res$deg <-myresults$deg
  res$Ds   <- myresults$Ds
  res$Ms <- myresults$Ms
  res$Dn   <- myresults$Dn
  res$Mn <- myresults$Mn
  res$p<- myresults$probab
  res$oddscut<- odds
  return(res)
}
```

3.11 SNP calling

To detect SNP's or more general disagreements between the sequenced transcriptome and some Genome (given in Fasta-format), one can use samtools, which requires the mapped reads as BAM-format (see Tophat) and the genome sequence:

```
samtools pileup –cvf /opt/appz/ngs/bowtie/
genomes/Aspergillus_fumigatusa1163.CADRE.dna.toplevel.fa tophat1163_368.bam
> snpforR1163_368 &
```

The pilup-command as been flaged as depricated, but seems to generate the only output format supported by Rsamtools. Here is some information on the output: http://samtools.sourceforge.net/cns0.shtml The new way to go seem to be mpileup. Take a look at http://samtools.sourceforge.net/mpileup.shtml, but I don't know if the generated BCF/VCF format is supported by Rsamtools yet.
The pileup-output (snpforR1163_368) begins like this:

```
chr1.fa 1577    C       M       3       3       60      2       A,      EI
chr1.fa 4013    C       A       12      48      60      5       aAAAA
84444
chr1.fa 4542    A       G       3       10      60      6       ggggGG
)7)7))
```

For instance in the second line, there is a C in the genome-sequence on chromosome 1 at position 4013, which is contradicted by at 12 reads. This file can be read in R (requireing the *Rsamtools*-package):

```
snps <-
readPileup("/home/smueller/data/analysen/vito/raw_data/tophat1163_368/snpfo
rR1163_368", variant="SNP")
snps2 <- snps[elementMetadata(snps)$coverage > 3]
```

since most snps are caused my mismapping at the intron-exon border, they have to be excluded

```
bed368 <-
import.ucsc(con="../tophat1163_368/junctions1163_368.bed",subformat="bed",a
sRangedData=FALSE)1*
bed368s <- bed368[bed368@elementMetadata$score > 3,]
overlaps_snpjunc <- findOverlaps(query=snps2,subject=bed368s)
goodsnps <- snps2[-queryHits(overlaps_snpjunc),]
```

This can be used to make statistics on the frequencies of the nucleotide transitions (Latex ready table):

```
table(elementMetadata(goodsnps)$referenceBase,elementMetadata(goodsnps)$con
sensusBase)[1:4,1:10]
    A  C  G  T  M   R  W  S   Y  K
 A  0  4 26  2 43 128 44  0   0  1
 C 20  0  2 21 62   0  0 59  97  0
 G 10  5  0  7  0 102  0 51   0 69
 T  6 19  6  0  0   0 66  0 140 65
```

Fabian found a paper [2] which describes that apparently, SNPs can also be caused by systematic read errors for specific motifs (e.g. GGGT, with T beeing wrong). They developed a method (SysCall) to correct for identifying and correcting those errors. But I haven't got it running yet.

3.12 Discovery of Novel Fusion Transcripts

If you suspect fusion transcripts one can use TopHat-Fusion,which is an enhanced version of TopHat with the ability to align reads across fusion points, which results from the breakage and re-joining of two different chromosomes, or from rearrangements within a chromosome.
E.g. on fengari:

```
python /opt/appz/ngs/tophatfusion-0.1.0/src/tophat-fusion.py -o
tophat_fusion -p 16 -g 1 --solexa1.3-quals /opt/appz/ngs/bowtie/indexes/mm9
OE1643_skin_export.fastq&
```

4. Literature
4.1 Introduction

- Schuster. Next-generation sequencing transforms today’s biology. VOL.5 NO.1 | JANUARY 2008 | NATURE METHODS
- Shendure. The beginning of the end for microarrays? nature methods | VOL.5 NO.7 | JULY 2008

4.2 Overview of different next-generation-sequencing technologies

- Mardis. The impact of next-generation sequencing technology on genetics. Trends in Genetics Vol.24 No.3 2007
- Tucker et al. Massively Parallel Sequencing: The Next Big Thing in Genetic Medicine. The American Journal of Human Genetics 85, 142–154, August 14, 2009
- Harismendy et al. Evaluation of next generation sequencing platforms for population targeted sequencing studies. Genome Biology 2009, 10:R32

4.3 Data analysis

- SEQanswers - Forum all about NGS questions

- Advanced RNA-Seq and ChiP-Seq data analysis / Bioconductor
- Dave Tangs Blog about NGS/bioinformatics
- Horner et al. Bioinformatics approaches for genomics and post genomics applications of next-generation sequencing. BRIEFINGS IN BIOINFORMATICS 2009
  (nice overview of tools for the analysis of next-generation sequencing data in several application categories)
- Pop & Salzberg. Bioinformatics challenges of new sequencing technology. Trends Genet. 2008 March ; 24(3): 142–149.
- Trapnell & Salzberg. How to map billions of short reads onto genomes. nature biotechnology volume 27 number 5 may 2009
- Ruffalo et al.: Comparative analysis of algorithms for next-generation sequencing read alignment. Bioinformatics (2011) 27(20):2790-2796.
- Bias correction
- Peng et al.: Comprehensive analysis of RNA-Seq data reveals extensive RNA editing in a human transcriptome. Nat. Biotechnol. (2012) 30, 253–260.
- Meacham et al.: Identification and correction of systematic error in high-throughput sequence data. BMC Bioinformatics (2011) 12, 451.
- Jones et al.: A new approach to bias correction in RNA-Seq Bioinformatics (2012) 28(7), 947-954.

4.4 Comparison of microarrays and NGS/

<u>RNA-Seq</u>

- Bloom et al. Measuring differential gene expression by short read sequencing: quantitative comparison to 2-channel gene expression microarrays. BMC Genomics 2009, 10:221
- Marioni et al. RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays. 2008
- Sultan et al. A Global View of Gene Activity and Alternative Splicing by Deep Sequencing of the Human Transcriptome. 2008
- Sackton and Clark. Comparative profiling of the transcriptional response to infection in two species of Drosophila by short-read cDNA sequencing. 2009 (Validation of cDNA sequencing by comparison to published microarray data)

<u>TagSeq</u>

- Sasidharan et al. An approach to comparing tiling array and high throughput sequencing technologies for genomic transcript mapping. 2009
- Feng et al. Power of Deep Sequencing and Agilent Microarray for Gene Expression Profiling Study. 2010
- Roh et al. Comparing microarrays and next- generation sequencing technologies for microbial ecology research. 2010 (Review Paper)