Supporting File 1: Transcriptome analysis of chronic lymphoid leukemia reveals isoform regulation associated with mutations in SF3B1

Alejandro Reyes, Carolin Blume, Vicent Pelechano, Petra Jakob, Lars Steinmetz, Thorsten Zenz, Wolfgang Huber

2013

Contents

Abstract

This document contains a documented R session with all the code used to analyse the RNA-seq data. It also describes the code used to generate the figure templates from the manuscript. Readers are welcomed to reproduce the code.

1 Preparation

In order to reproduce the code from this document, the Bioconductor data package CLL.SF3B1 should be installed. This package contains input files

that resulted from a first round of data preprocessing that are needed to reproduce the results. Therefore, we first load the package and the data:

```
suppressMessages( library("CLL.SF3B1") )
data("ecsSF3B1")
```

If you don't have the package installed, you can install it by typing in your R session

```
biocLite("CLL.SF3B1")
```

Also, modify the variable "cores" specifying the number of CPUs available in your machine. This will allow to distribute the computationally expensive jobs into many cores.

```
cores <- 15
```

2 Testing for differential exon usage

```
ecsSF3B1 <- estimateSizeFactors( ecsSF3B1 )
formulaDispersion <- ~ sample + ( phenotype + sf3b1) * exon
ecsSF3B1 <- estimateDispersions( ecsSF3B1, formula=formulaDispersion, nCores=cores)
ecsSF3B1 <- fitDispersionFunction( ecsSF3B1 )
formula0 = ~sample +  sf3b1 +  exon
formula1 = ~sample + sf3b1 * exon
ecsSF3B1 <- testForDEU( ecsSF3B1, formula0=formula0, formula1=formula1, nCores=cores )
```

We found 50 exons to be differentially used between the mutated samples and the normal samples

```
table( fData(ecsSF3B1)$padjust < 0.1 )
```

```
 FALSE   TRUE
253106     50
```

These were distributed along 41 genes

```
genes <- unique( geneIDs(ecsSF3B1)[which( fData(ecsSF3B1)$padjust < 0.1 )] )
length(genes)
```

```
[1] 41
```

3 Reactome pathway enrichment analysis

Then, we download the file from the database reactome that maps uniprot gene identificators to annotated pathways and read it as a data frame. After download, we reformat the data frame to make it easier to access.

```
reactome <- read.delim(
    url(
     "http://www.reactome.org/download/current/uniprot_2_pathways.txt"
),
    header=FALSE)

rownames(reactome) <- as.character( reactome$V1 )

processes <- gsub(
    "^\\[\\d+\\sprocesses\\]: ",
    "",
    as.character( reactome$V3 ),
    perl=TRUE)

processes <- strsplit( processes, "; ")

names( processes ) <- rownames( reactome )
processesDF <- lapply(
    seq_along( processes ),
    function(x){
data.frame(
    uniprot=names(processes)[x],
    process=processes[[x]]
    )
    })
processesDF <- do.call( rbind, processesDF )
head( processesDF )
```

```
  uniprot                                              process
1  E9Q414 Binding and Uptake of Ligands by Scavenger Receptors
2  E9Q414                      Scavenging by Class A Receptors
3  E9Q414                      Scavenging by Class B Receptors
4  G5EF96                                        Axon guidance
5  G5EF96                   DCC mediated attractive signaling
6  G5EF96                               Developmental Biology
```

The pathways in reactome are based on uniprot IDs, therefore we use biomaRt to map our ensembl gene identificators with uniprot identificators. We do the same to translate ensembl gene IDs to gene names.

```
library(biomaRt)
mart <- useMart("ensembl", dataset="hsapiens_gene_ensembl")
bm <- getBM(
    attributes=
      c("ensembl_gene_id", "uniprot_swissprot_accession"),
    filter="ensembl_gene_id",
    values=
      as.character(unique(geneIDs(ecsSF3B1))),
    mart=mart )
uniprots <- bm$‘uniprot_swissprot_accession‘
names( uniprots ) <- bm$‘ensembl_gene_id‘
uniprots <- uniprots[uniprots != ""]

bm <- getBM(
   c("ensembl_gene_id", "external_gene_id"),
   "ensembl_gene_id",
   values=as.character(unique(geneIDs(ecsSF3B1))),
   mart=mart)

geneName <- bm$‘external_gene_id‘
names(geneName) <- bm$‘ensembl_gene_id‘
```

Now we can test for over-representation of the genes with isoform regulation associated to the mutation in SF3B1 compared to our all the genes that contain at least 600 counts in all the samples. We do this in order to to avoid biases associated to expression strength.

```
library(DESeq2)
data("dseSF3B1")

foreground <- uniprots[names(uniprots) %in% genes]
toTest <- unique(
    processesDF[processesDF[,"uniprot"]
%in% foreground,"process"] )

expressed <- rownames(counts(dseSF3B1))[rowSums( counts(dseSF3B1) ) > 600]
```

```
background <-  uniprots[names(uniprots) %in% expressed]

testForReactome <- function( toTest, foreground, background ){
  pvals <- mclapply( toTest, function(x){
 df2 <- processesDF[processesDF[,"process"] %in% x,]
 a <- sum( df2[,"uniprot"] %in% foreground )
 b <- sum( df2[,"uniprot"] %in% background )
 c <- df2[,"uniprot"] %in% foreground
 c <- unique( df2[,"uniprot"][which(c)] )
 c <- paste(c, collapse=",")
 c(a, length(foreground) - a)
 mat <- t( data.frame(
     fore=c(a, length(foreground)-a),
     back=c(b, length( background)-b ) )
  )
 colnames(mat) <- c("in", "out")
 ft <- fisher.test( mat , alternative="greater" )
 ft$estimate
 list( genes=c,
      numbers=c( foreground=mat[1,],
  background=mat[2,],
  ft$estimate,
  pval=ft$p.value ))
 }, mc.cores=cores)
  names(pvals) <- toTest
  againstMM <- pvals

  for(i in seq_along( againstMM )){
    sepGenes <- unlist( strsplit( againstMM[[i]]$genes, "," ) )
    againstMM[[i]]$geneNames <- paste(
unique( geneName[names( uniprots[uniprots %in% sepGenes] )] ),
collapse=",")
  }

  table <- t( sapply( againstMM, "[[", "numbers" ) )
  table <- as.data.frame( table )
  table$genes <- sapply( againstMM, "[[", "geneNames" )
  table$padj <- p.adjust( table$pval, method="BH" )
  table
}
```

```
allGenes <- testForReactome(toTest, uniprots[names(uniprots) %in% genes], background )
enriched <- allGenes[allGenes$padj < 0.1,c("foreground.in", "pval", "genes")]
rownames(enriched)
```

```
Antigen Presentation: Folding, assembly and peptide loading of class I MHC
Antigen processing-Cross presentation
Cytokine Signaling in Immune system
Endosomal/Vacuolar pathway
Immunoregulatory interactions between a Lymphoid and a non-Lymphoid cell
Interferon Signaling
Interferon gamma signaling
Interferon alpha/beta signaling
Processing of Capped Intron-Containing Pre-mRNA
mRNA Processing
mRNA Splicing
mRNA Splicing - Major Pathway
Acyl chain remodelling of PG
Acyl chain remodelling of PI
Acyl chain remodelling of PS
Cap-dependent Translation Initiation
Eukaryotic Translation Initiation
GTP hydrolysis and joining of the 60S ribosomal subunit
Translation
3' -UTR-mediated translational regulation
L13a-mediated translational silencing of Ceruloplasmin expression
```

4 Figures

We define the colors for each sample, load the data from the supplementary materials presented by Furney et al, and load the information from the pfam domains.

```
suppressMessages( library(ggbio ) )
suppressMessages(library(AnnotationDbi))
suppressMessages(library(GenomicRanges))
suppressMessages(library(GenomicFeatures))
suppressMessages(library(Biostrings))
colorSamples <- c("#238B45", "#238B45", "#0C2C84", "#0C2C84", "#0C2C84", "#0C2C84")
colorConditions <- c("#238B45", "#0C2C84")
```

```
path <- system.file(package="CLL.SF3B1", "extdata")
um <- read.delim( list.files( path, pattern="^furney", full.names=TRUE ) )
umRanges <- GRanges(um$chr, IRanges( start=um$start, end=um$end ), um$strand )

data("domains")
```

```
[1] "Antigen Presentation: Folding, assembly and peptide loading of class I MHC"
 [2] "Antigen processing-Cross presentation"
 [3] "Cytokine Signaling in Immune system"
 [4] "Endosomal/Vacuolar pathway"
 [5] "Immunoregulatory interactions between a Lymphoid and a non-Lymphoid cell"
 [6] "Interferon Signaling"
 [7] "Interferon gamma signaling"
 [8] "Interferon alpha/beta signaling"
 [9] "Processing of Capped Intron-Containing Pre-mRNA"
[10] "mRNA Processing"
[11] "mRNA Splicing"
[12] "mRNA Splicing - Major Pathway"
[13] "Acyl chain remodelling of PG"
[14] "Acyl chain remodelling of PI"
[15] "Acyl chain remodelling of PS"
[16] "Cap-dependent Translation Initiation"
[17] "Eukaryotic Translation Initiation"
[18] "GTP hydrolysis and joining of the 60S ribosomal subunit"
[19] "Translation"
[20] "3' -UTR-mediated translational regulation"
[21] "L13a-mediated translational silencing of Ceruloplasmin expression"
```

We also need to create a transcript database object based on the annotation file. We first download from ENSEMBL the reference fasta files and the annotation file in the gtf format. We need both of this in order to create our transcript database. This is done in a command line, not in an R session:

```
wget \
ftp://ftp.ensembl.org/pub/release-68/gtf/homo_sapiens/Homo_sapiens.GRCh37.68.gtf.gz

gunzip Homo_sapiens.GRCh37.68.gtf.gz

wget \
ftp://ftp.ensembl.org/pub/release-68/fasta/homo_sapiens/dna/Homo_sapiens.GRCh37.68.dna
gunzip Homo_sapiens.GRCh37.68.dna.toplevel.fa.gz
```

We now can create the transcript database in our R session based on the files that we downloaded:

```
fastq <- readDNAStringSet("Homo_sapiens.GRCh37.68.dna.toplevel.fa")

df <- data.frame(
    chrom=sapply( strsplit( names(fastq), " " ), "[[", 1),
    length=width(fastq),
    is_circular=rep(FALSE, length(fastq)))

transcriptDb <- makeTranscriptDbFromGFF(
    "Homo_sapiens.GRCh37.68.gtf",
    format="gtf",
    exonRankAttributeName="exon_number",
    chrominfo=df,
    dataSource=paste("ensembl human release 68"),
    species="Homo sapiens"
    )


saveDb(transcriptDb, file="transcriptDb.sqlite")
```

We load the transcriptDb object

```
library(GenomicFeatures)
transcriptDb <- loadDb("transcriptDb.sqlite")
```

Below is the code that was used to create the templates for each figure, afterwards they were merged, modified and adapted to the journal requirements using inkscape.

4.1 Figure 1

We use the package h5vc in order to generate this figure, this package is designed to work with genomic DNA sequencing. Here we tricked h5vc and use it with RNA-seq data in order to see the expression of the SF3B1 K700E variant.

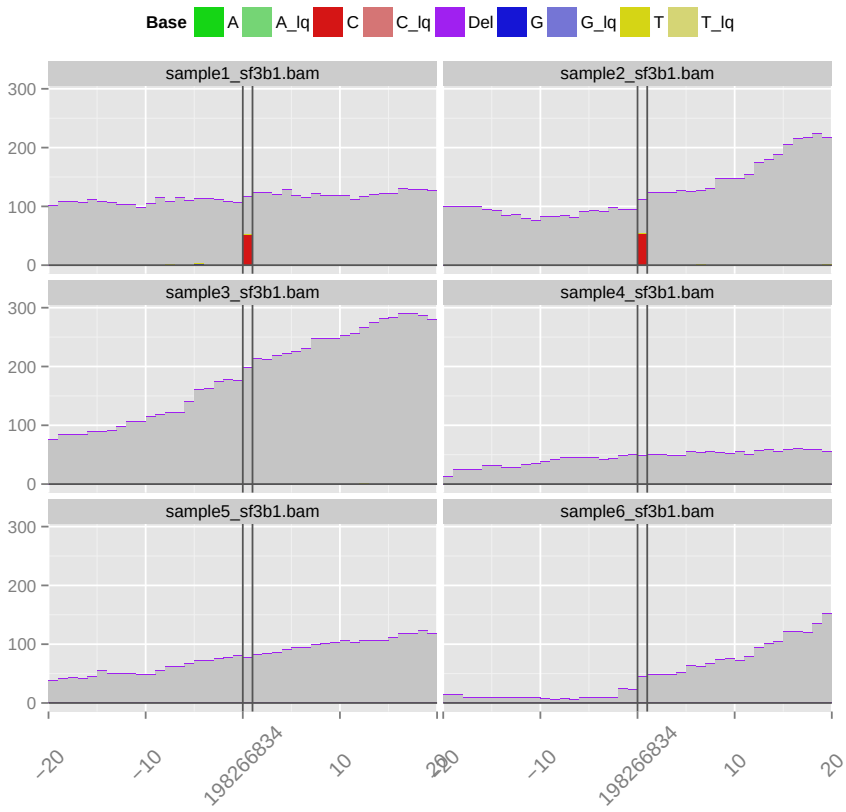
```
library(CLL.SF3B1)

path <- system.file( package="CLL.SF3B1" )
```

```
path <- file.path( path, "bam")
bamFiles <- list.files( path, pattern="bam$" )

suppressPackageStartupMessages(library(h5vc))
suppressPackageStartupMessages(library(rhdf5))
suppressPackageStartupMessages(library(deepSNV))

chrom <- "2"
study <- "/SF3B1"

tallyFile <- file.path(".", "SF3B1.tally.hfs5")
if (file.exists(tallyFile)) {
    file.remove(tallyFile)
}

h5createFile(tallyFile)
group <- paste(study, chrom, sep = "/")
h5createGroup(tallyFile, study)
h5createGroup(tallyFile, group)

end <- 198299815
start <- 198256698
dim4 <- end +1000

h5createDataset(tallyFile,
    paste(group, "Counts", sep = "/"), dims = c(12, 6,
    2, dim4), storage.mode = "integer", level = 9)

h5createDataset(tallyFile,
    paste(group, "Coverages", sep = "/"), dims = c( 6,
    2, dim4), storage.mode = "integer", level = 9)

h5createDataset(tallyFile,
    paste(group, "Deletions", sep = "/"), dims = c(6,
    2, dim4), storage.mode = "integer", level = 9)

h5createDataset(tallyFile,
   paste(group, "Reference", sep = "/"), dims = c(dim4),
   storage.mode = "integer", level = 9)
```

```
sample <- sapply( strsplit( bamFiles, "_sf3B1"), "[[", 1)
names( sample ) <- c("1", "2", "6", "5", "4", "3")

sampleData <- data.frame(
   Sample = sample, Column=0:5,
   Patient=names(sample),
   Type = c("CLL", "CLL", "CLL", "CLL", "healthy", "healthy"),
   stringsAsFactors = FALSE)

sampleData

setSampleData(tallyFile, group, sampleData)
getSampleData(tallyFile, group )

Counts <- lapply(file.path(path, bamFiles), function(bamf){
   bam2R( file=bamf, chr=chrom, start=start, stop=end )
})

Coverages <- lapply(Counts, function(count) matrix(c(rowSums(count[, c("A",
    "C", "G", "T", "DEL")]), rowSums(count[, c("a", "c", "g", "t", "del")])),
    ncol = 2, byrow = FALSE, dimnames = list(NULL, c("Fwd", "Rev"))))
Deletions <- lapply(Counts, function(count) count[, c("DEL", "del")])
Counts <- lapply(Counts, function(count) count[, c("A", "C", "G", "T", "a",
    "c", "g", "t")])
ref <- apply(Counts[[1]][, 1:4] +
  Counts[[1]][5:8] + Counts[[2]][, 1:4] +
  Counts[[2]][5:8],
    1, which.max)

for( j in 1:6){
  for (i in seq(length(ref))) {
    Counts[[j]][i, ref[i]] <- 0
    Counts[[j]][i, (ref[i] + 4)] <- 0
  }
}


Reference <- ref - 1
h5ls(tallyFile)
```

```
for( sample in 1:6 ){
    h5write(t(Counts[[sample]][, 1:4])+t(Counts[[sample]][, 5:8]),
       tallyFile, paste(group, "Counts", sep = "/"),
index = list(5:8, sample, 1, start:end))
    h5write(Coverages[[sample]][, "Fwd"] + Coverages[[sample]][, "Rev"],
       tallyFile, paste(group, "Coverages",
sep = "/"), index = list(sample, 1, start:end))
}

h5write(Reference, tallyFile,
   paste(group, "Reference", sep = "/"),
    index =list( start:end))


position <- 198266834
windowsize <- 20

data <- h5dapply(filename = tallyFile,
    group = group, blocksize = 1e+08,
    range = c(position -
       windowsize, position + windowsize))[[1]]
sampledata <- getSampleData(tallyFile, group)
samples <- sampledata$Sample

[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
             Sample Column Patient    Type
1 sample1_sf3b1.bam      0       1     CLL
2 sample2_sf3b1.bam      1       2     CLL
6 sample3_sf3b1.bam      2       6     CLL
5 sample4_sf3b1.bam      3       5     CLL
4 sample5_sf3b1.bam      4       4 healthy
3 sample6_sf3b1.bam      5       3 healthy
  Column Patient            Sample    Type
```

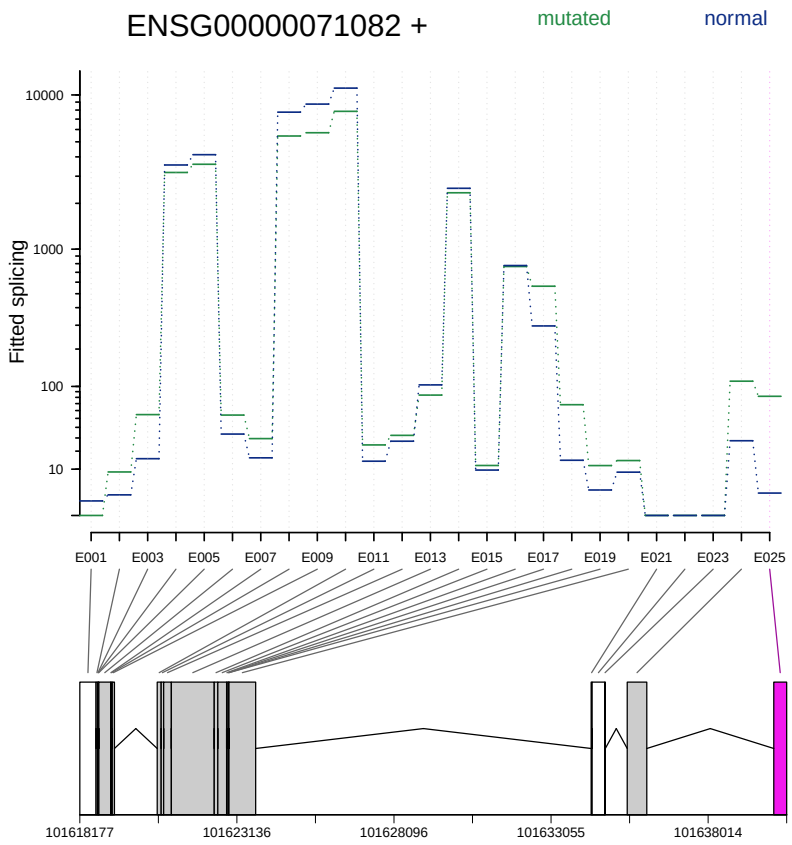
```
1      1       1 sample1_sf3b1.bam     CLL
2      2       2 sample2_sf3b1.bam     CLL
3      3       6 sample3_sf3b1.bam     CLL
4      4       5 sample4_sf3b1.bam     CLL
5      5       4 sample5_sf3b1.bam healthy
6      6       3 sample6_sf3b1.bam healthy
     group      name       otype  dclass                    dim
0        /     SF3B1   H5I_GROUP
1   /SF3B1         2   H5I_GROUP
2 /SF3B1/2    Counts H5I_DATASET INTEGER 12 x 6 x 2 x 198300815
3 /SF3B1/2 Coverages H5I_DATASET INTEGER      6 x 2 x 198300815
4 /SF3B1/2 Deletions H5I_DATASET INTEGER      6 x 2 x 198300815
5 /SF3B1/2 Reference H5I_DATASET INTEGER              198300815
 Attaching package bit
package:bit (c) 2008-2012 Jens Oehlschlaegel (GPL-2)
creators: bit bitwhich
coercion: as.logical as.integer as.bit as.bitwhich which
operator: ! & | xor != ==
querying: print length any all min max range sum summary
bit access: length<- [ [<- [[ [[<-
for more help type ?bit

Attaching package: 'bit'

The following object is masked from 'package:base':

    xor


Attaching package: 'bit64'

The following object is masked from 'package:Biobase':

    cache

library(ggplot2)
p <- mismatchPlot(data, sampledata,
    samples, windowsize, position) + facet_wrap(
    ~ Sample, ncol = 2)
print(p)
```



Note that this plot was generated from the coverage calculated to the "+" strand and SF3B1 is on the minus strand. Therefore this plot was mirrored afterwards with inkscape so that it reflected the coverage for the "-" strand.

4.2 Figure 2: RPL31

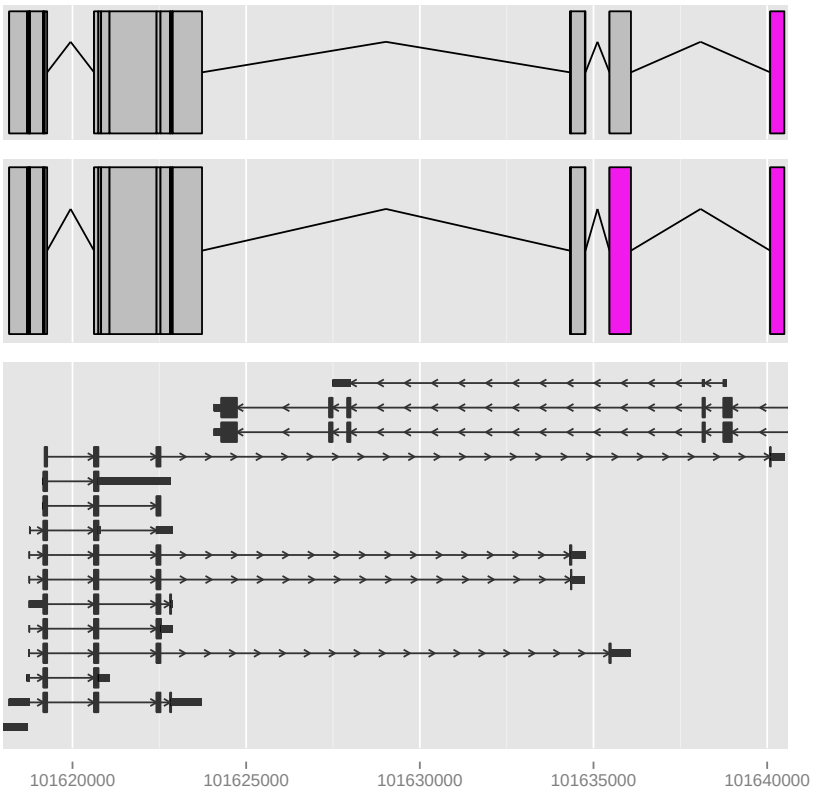
```
plotDEXSeq(ecsSF3B1, "ENSG00000071082",
   norCounts=FALSE, lwd=1.3, legend=TRUE, fitExpToVar="sf3b1",
   splicing=TRUE, expression=FALSE,
   cex.axis=1, color=c("#238B45", "#0C2C84"),
   color.samples=colorSamples)
```



zoomed region,

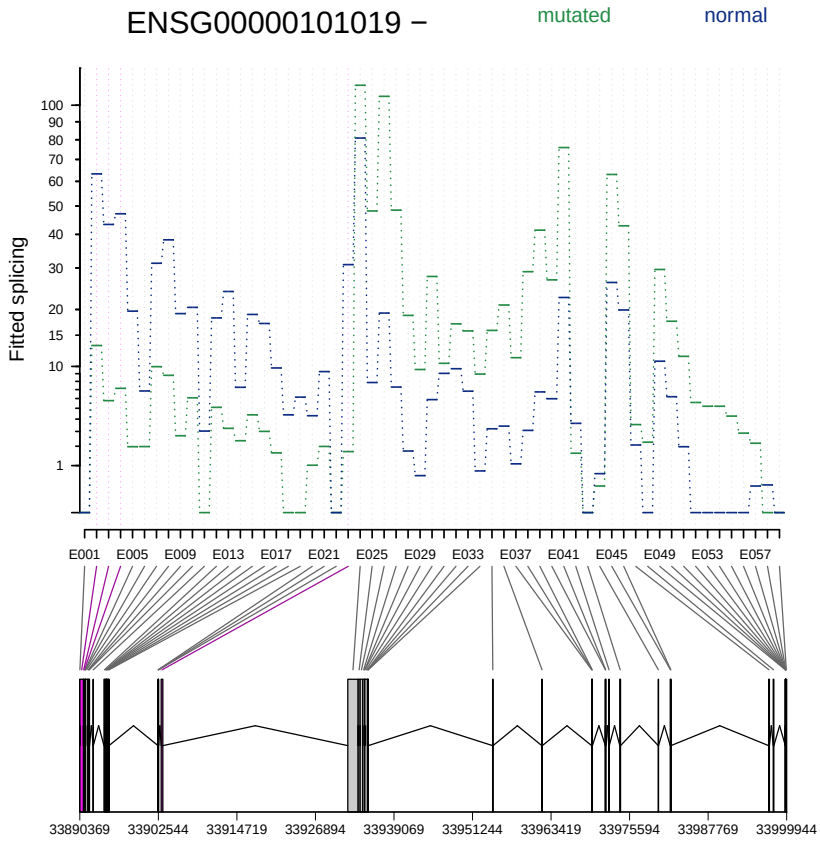
```
thisRange <- fData(ecsSF3B1)[geneIDs(ecsSF3B1)
    %in% "ENSG00000071082",
    c("chr", "start", "end", "strand", "padjust")]
exonRange <- GRanges( thisRange$chr,
  IRanges(
     start=thisRange$start,
     end=thisRange$end,
     names=rownames(thisRange)),
   thisRange$strand )

geneRange <- GRanges( 2, IRanges(start=101618000, end=101640594))
exonRange$significant <- as.numeric( thisRange$padjust < 0.1 )
exonRange$significant[is.na( exonRange$significant )] <- 0
overlap <- findOverlaps( exonRange, umRanges, type="equal" )
```

```
exonRange$significantUM <-
   as.numeric( names( exonRange ) %in%
    names( exonRange[queryHits(
       findOverlaps( exonRange, umRanges, type="equal" ) )] ) )
wh <- geneRange

tracks(
  autoplot( GRangesList( exonRange ),
     fill=ifelse(exonRange$significant == 1, "#F219ED", "gray"),
     colour=ifelse(exonRange$significant == 1, "black", "black")),
  autoplot( GRangesList( exonRange ),
     fill=ifelse(exonRange$significantUM == 1, "#F219ED", "gray"),
     colour=ifelse(exonRange$significant == 1, "black", "black")),
  autoplot( transcriptDb, wh, group.selfish=TRUE, names.expr=""),
xlim=wh, heights=c(1, 1, 2))
```



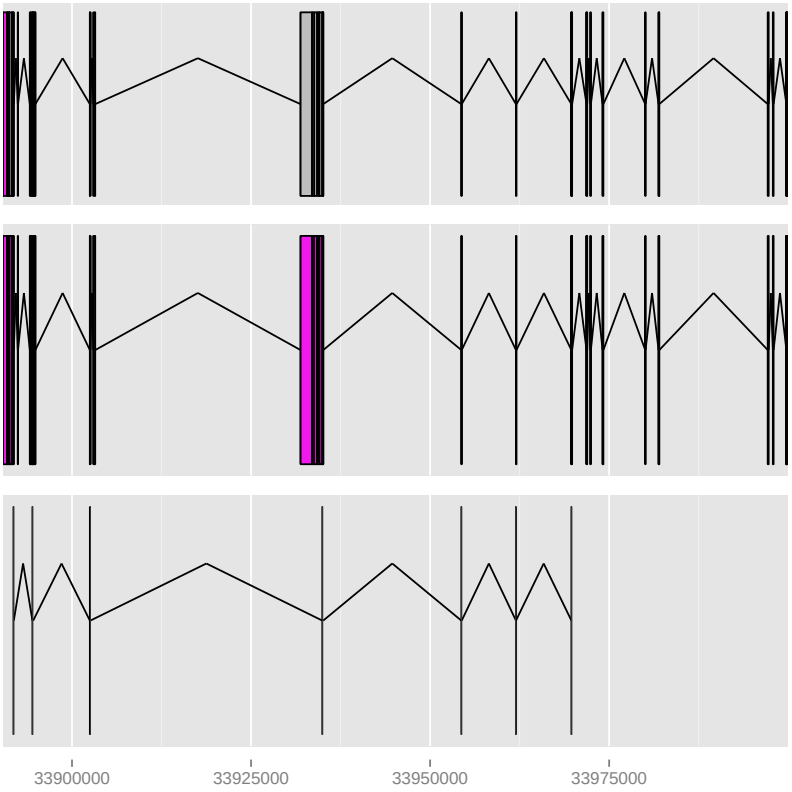
4.3 Figure 3: UQCC

```
plotDEXSeq(ecsSF3B1, "ENSG00000101019",
   norCounts=FALSE, lwd=1.3, legend=TRUE, fitExpToVar="sf3b1",
   splicing=TRUE, expression=FALSE,
   cex.axis=1, color=c("#238B45", "#0C2C84"),
   color.samples=colorSamples)
```



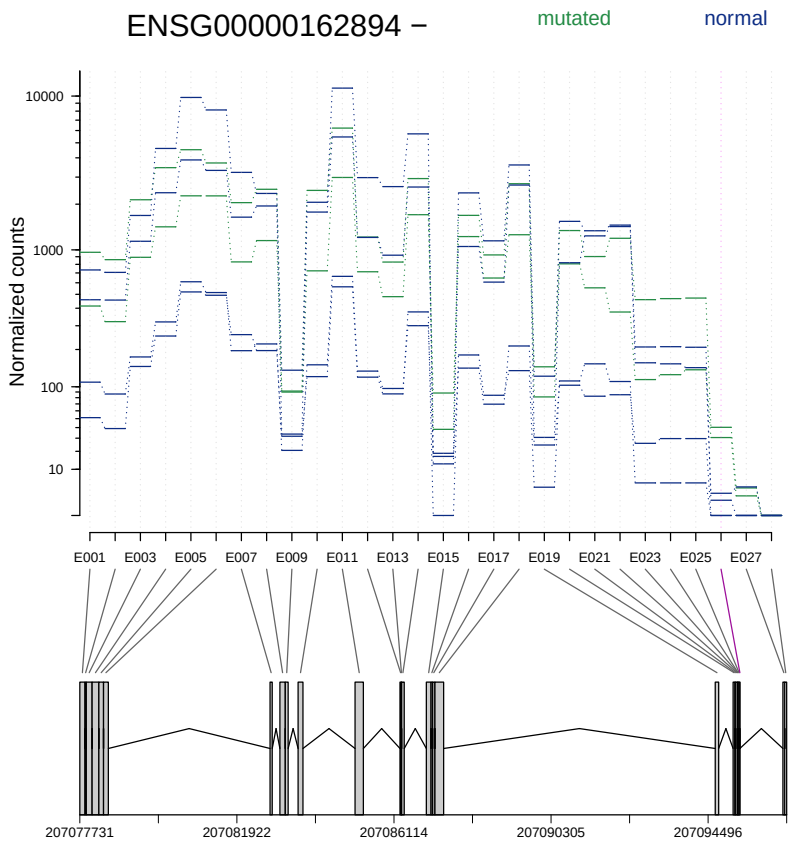

```
thisRange <- fData(ecsSF3B1)[
    geneIDs(ecsSF3B1) %in% "ENSG00000101019",
    c("chr", "start", "end", "strand", "padjust")]
exonRange <- GRanges( thisRange$chr,
  IRanges(
     start=thisRange$start,
     end=thisRange$end,
     names=rownames(thisRange)),
```

```
   thisRange$strand )

geneRange <- GRanges( 20, IRanges(start=33890369, end=33999944))
exonRange$significant <- as.numeric( thisRange$padjust < 0.1 )
exonRange$significant[is.na( exonRange$significant )] <- 0
overlap <- findOverlaps( exonRange, umRanges, type="equal" )
exonRange$significantUM <- as.numeric(
   names( exonRange ) %in% names(
      exonRange[queryHits(
findOverlaps( exonRange, umRanges, type="equal" ) )] ) )
wh <- geneRange

domainRange <- GRangesList(
   reduce( unique(
      domainRanges[
      subjectHits( findOverlaps( geneRange, domainRanges ) )] )
    ))


tracks(
  autoplot( GRangesList( exonRange ),
    fill=ifelse(exonRange$significant == 1, "#F219ED", "gray"),
    colour=ifelse(exonRange$significant == 1, "black", "black")),
  autoplot( GRangesList( exonRange ),
    fill=ifelse(exonRange$significantUM == 1, "#F219ED", "gray"),
    colour=ifelse(exonRange$significant == 1, "black", "black")),
  autoplot( domainRange ), heights=c(1, 1, 1), xlim=wh)
```



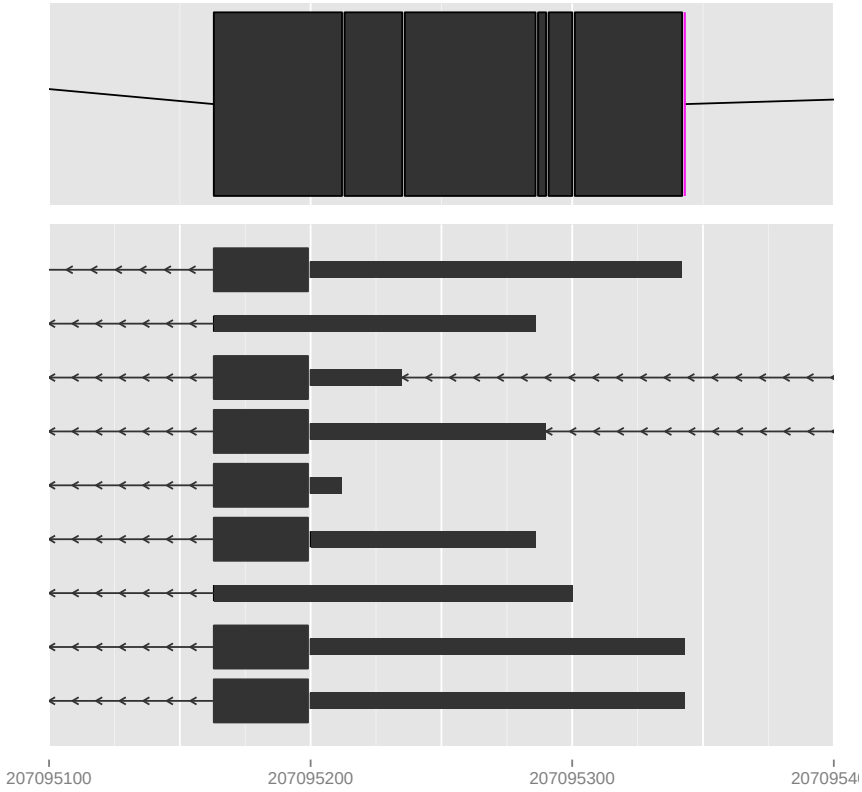
4.4 Figure 4: FAIM3

```
plotDEXSeq(ecsSF3B1, "ENSG00000162894", norCounts=TRUE,
    lwd=1, legend=TRUE, fitExpToVar="sf3b1", splicing=FALSE,
    expression=FALSE, cex.axis=1, color=colorConditions,
    color.samples=colorSamples,
    displayTranscripts=FALSE, names=FALSE)
```




```
thisRange <- fData(ecsSF3B1)[
    geneIDs(ecsSF3B1) %in% "ENSG00000162894",
    c("chr", "start", "end", "strand", "padjust")]
thisRange$padjust[is.na( thisRange$padjust )] <- 1
geneRange <- GRanges( thisRange$chr,
   IRanges(
     start=thisRange$start,
     end=thisRange$end,
     names=rownames(thisRange)),
  thisRange$strand )
geneRange$significant <- as.numeric( thisRange$padjust < 0.1 )
wr <- GRanges( "1", IRanges(start=207095100, end=207095400 ) )

tr1 <- autoplot( transcriptDb, which=wr, group.selfish=TRUE, names.expr=FALSE )
```

```
tr2 <- autoplot(
   GRangesList( geneRange ),
   colour=ifelse(geneRange$significant == 1,
     "#F219ED", "black"))

tracks( tr2, tr1, heights=c(1, 2), xlim=wr )
```

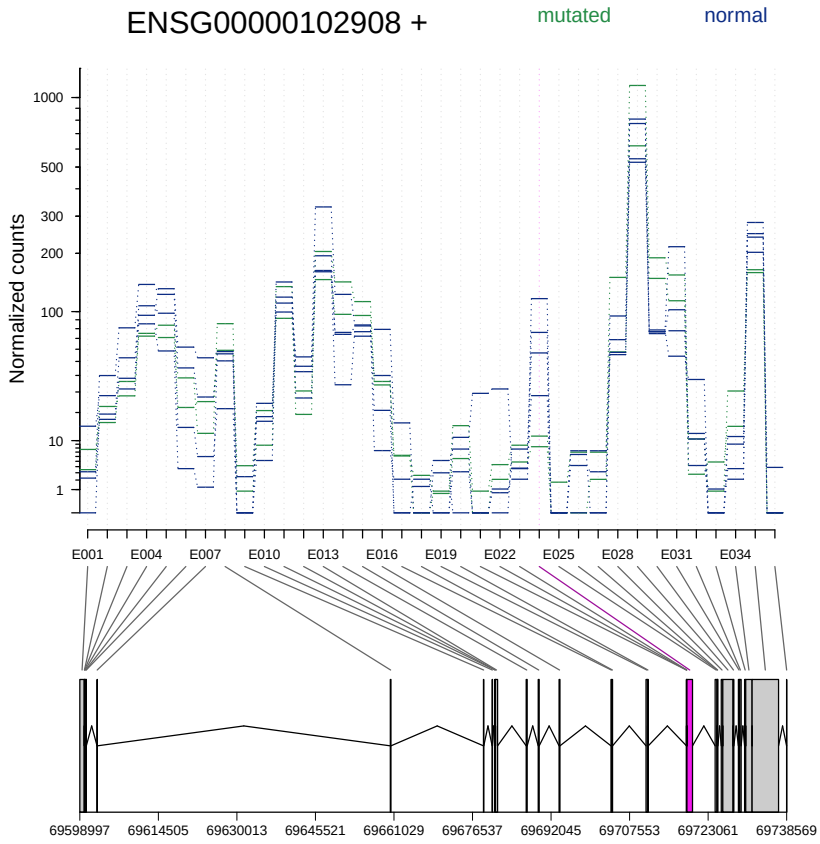


4.5 Figure 5: NFAT5

PLot NFAT5

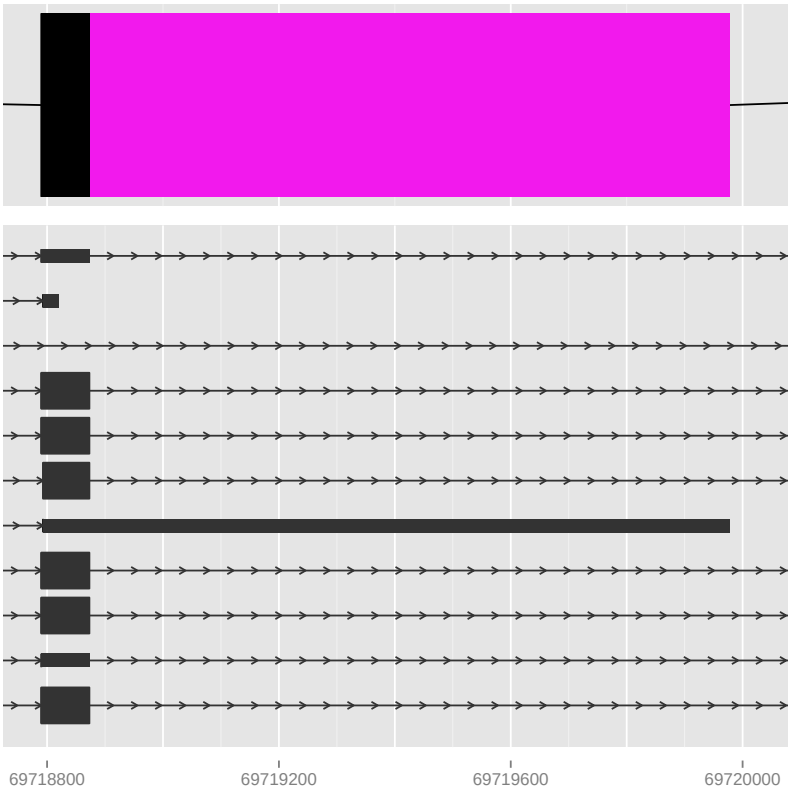
```
plotDEXSeq(ecsSF3B1, "ENSG00000102908",
  norCounts=TRUE, lwd=1, legend=TRUE,
  fitExpToVar="sf3b1", splicing=FALSE,
```

```
  expression=FALSE, cex.axis=1,
  color=colorConditions, color.samples=colorSamples)
```




```
thisRange <- fData(ecsSF3B1)[
     geneIDs(ecsSF3B1) %in% "ENSG00000102908",
     c("chr", "start", "end", "strand", "padjust")]
 thisRange$padjust[is.na( thisRange$padjust )] <- 1
 geneRange <- GRanges( thisRange$chr,
    IRanges(
      start=thisRange$start,
      end=thisRange$end,
      names=rownames(thisRange)),
   thisRange$strand )
 geneRange$significant <- as.numeric( thisRange$padjust < 0.1 )
```

```
prueba <- GRanges( "16", IRanges(start=69718874-150, end=69719978+100 ))

tr1 <- autoplot( transcriptDb, prueba, group.selfish=TRUE, names.expr="")
tr2 <- autoplot( GRangesList( geneRange ),
   fill=ifelse(geneRange$significant == 1, "#F219ED", "black"))
tracks( tr2, tr1, heights=c(1, 2), xlim=prueba )
```



5 Session information

```
sessionInfo()

R Under development (unstable) (2013-10-21 r64088)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
```

```
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
 [1] stats4    splines   parallel  stats     graphics  grDevices utils
 [8] datasets  methods   base

other attached packages:
 [1] bit64_0.9-2             bit_1.1-10              deepSNV_1.9.0
 [4] VariantAnnotation_1.9.3 VGAM_0.9-2              Rsamtools_1.15.2
 [7] rhdf5_2.7.0             h5vc_1.1.1              Biostrings_2.31.0
[10] GenomicFeatures_1.15.0  AnnotationDbi_1.25.0    ggbio_1.9.7
[13] ggplot2_0.9.3.1         biomaRt_2.19.0          CLL.SF3B1_0.0.1
[16] DESeq2_1.3.0            RcppArmadillo_0.3.920.1 Rcpp_0.10.5
[19] GenomicRanges_1.15.1    XVector_0.3.0           IRanges_1.21.2
[22] DEXSeq_1.9.1            Biobase_2.23.1          BiocGenerics_0.9.0
[25] BiocInstaller_1.13.1

loaded via a namespace (and not attached):
 [1] annotate_1.41.0    biovizBase_1.11.1  bitops_1.0-6       BSgenome_1.31.0
 [5] cluster_1.14.4     colorspace_1.2-4   DBI_0.2-7          dichromat_2.0-0
 [9] digest_0.6.3       genefilter_1.45.0  grid_3.1.0         gridExtra_0.9.1
[13] gtable_0.1.2       Hmisc_3.12-2       hwriter_1.3        labeling_0.2
[17] lattice_0.20-24    locfit_1.5-9.1     MASS_7.3-29        munsell_0.4.2
[21] plyr_1.8           proto_0.3-10       RColorBrewer_1.0-5 RCurl_1.95-4.1
[25] reshape_0.8.4      reshape2_1.2.2     rpart_4.1-3        RSQLite_0.11.4
[29] rtracklayer_1.23.1 scales_0.2.3       statmod_1.4.18     stringr_0.6.2
[33] survival_2.37-4    tools_3.1.0        XML_3.98-1.1       xtable_1.7-1
[37] zlibbioc_1.9.0
```