Supporting File 1: Mutated SF3B1 is associated with transcript isoform changes of the genes UQCC and RPL31 both in CLLs and uveal melanomas

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

2013

Contents

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 welcome 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 all the genes that contain at least 600 counts across all the samples. We do this in order to to avoid biases associated to expression strength.

```
library(DESeq2)

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
Processing of Capped Intron-Containing Pre-mRNA
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
```

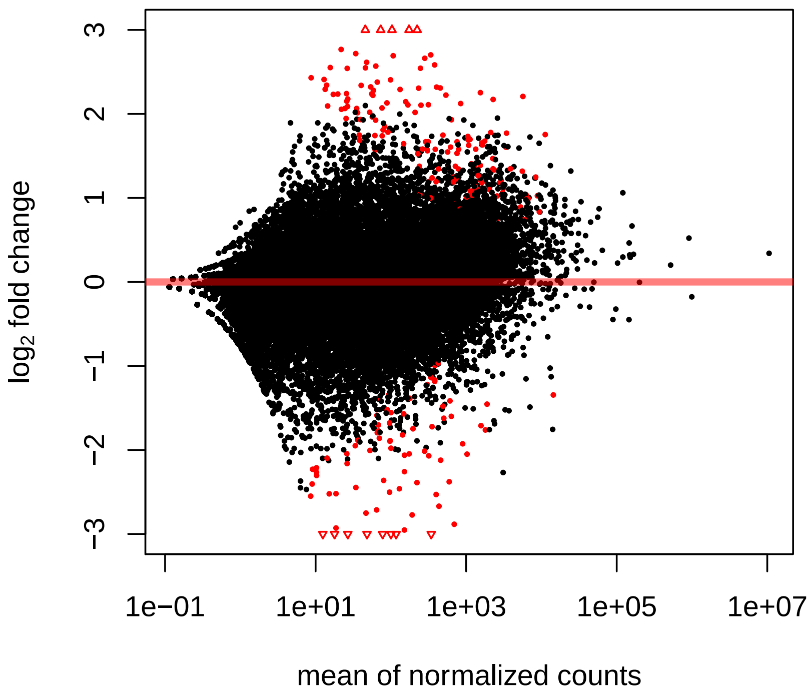
4 Differential expression

We tested for differential expression between the samples with mutated SF3B1 and the samples with wt SF3B1.

```
data("dseSF3B1")
dseSF3B1 <- DESeq(dseSF3B1)
res <- results(dseSF3B1)
upregulated <-
  rownames(res)[
    which( res$padj < 0.1 & res$log2FoldChange > 0 )]
downregulated <-
  rownames(res)[
    which( res$padj < 0.1 & res$log2FoldChange < 0 )]

table( res$padj < 0.1 )
```

```
  estimating size factors
 estimating dispersions
 gene-wise dispersion estimates
 mean-dispersion relationship
 final dispersion estimates
 fitting model and testing

 FALSE  TRUE
 14315   228
```

```
plotMA(dseSF3B1, ylim=c(-3, 3))
```



Set of upregulated genes on SF3B1 mutated genes

```
geneName[names(geneName) %in% downregulated]
```

```
ENSG00000168675 ENSG00000156755 ENSG00000199691 ENSG00000202058 ENSG00000077782
      "LDLRAD4"     "IGKV1OR-2"     "RN7SKP173"      "RN7SKP80"         "FGFR1"
ENSG00000111331 ENSG00000241666 ENSG00000224295 ENSG00000202077 ENSG00000252481
         "OAS3"   "RP3-455J7.4"   "AC087380.14"      "RNU1-60P"      "SCARNA13"
ENSG00000104938 ENSG00000199879 ENSG00000203799 ENSG00000112242 ENSG00000261040
       "CLEC4M"     "RNU1-120P"      "CCDC162P"          "E2F3" "CTD-2319I12.1"
ENSG00000148926 ENSG00000161513 ENSG00000252010 ENSG00000251495 ENSG00000175893
          "ADM"          "FDXR"       "SCARNA5"       "PPIAP11"       "ZDHHC21"
ENSG00000166510 ENSG00000137628 ENSG00000188886 ENSG00000158050 ENSG00000159256
       "CCDC68"         "DDX60"          "ASTL"         "DUSP2"         "MORC3"
ENSG00000123739 ENSG00000244405 ENSG00000198642 ENSG00000170734 ENSG00000117862
```

```
     "PLA2G12A"          "ETV5"         "KLHL9"          "POLH"       "TXNDC12"
ENSG00000100596 ENSG00000133103 ENSG00000145864 ENSG00000198440 ENSG00000257151
       "SPTLC2"          "COG6"        "GABRB2"        "ZNF583"         "PWAR6"
ENSG00000207133 ENSG00000160551 ENSG00000149485 ENSG00000149054 ENSG00000128482
   "SNORD116-7"         "TAOK1"         "FADS1"        "ZNF215"        "RNF112"
ENSG00000133835 ENSG00000101577 ENSG00000139116 ENSG00000166710 ENSG00000248302
      "HSD17B4"         "LPIN2"        "KIF21A"           "B2M"     "Z95704.4"
ENSG00000211553 ENSG00000006831 ENSG00000069329 ENSG00000152767 ENSG00000171208
   "AC118278.1"       "ADIPOR2"         "VPS35"         "FARP1"         "NETO2"
ENSG00000102349 ENSG00000121101 ENSG00000164296 ENSG00000111266 ENSG00000102053
         "KLF8"         "TEX14"         "TIGD6"        "DUSP16"       "ZC3H12B"
ENSG00000122877 ENSG00000203668 ENSG00000135698 ENSG00000089682 ENSG00000136810
         "EGR2"          "CHML"      "MPHOSPH6"         "RBM41"           "TXN"
ENSG00000146731 ENSG00000163873 ENSG00000136866 ENSG00000185885 ENSG00000108557
        "CCT6A"         "GRIK3"         "ZFP37"        "IFITM1"          "RAI1"
ENSG00000152926 ENSG00000213462 ENSG00000146757 ENSG00000239961 ENSG00000252835
       "ZNF117"        "ERV3-1"         "ZNF92"        "LILRA4"      "SCARNA21"
ENSG00000197852 ENSG00000136848 ENSG00000204099 ENSG00000134242 ENSG00000216490
      "FAM212B"        "DAB2IP"          "NEU4"        "PTPN22"         "IFI30"
ENSG00000163564
       "PYHIN1"
```

Set of downregulated genes on SF3B1 mutated genes

```
geneName[names(geneName) %in% upregulated]
```

```
ENSG00000241781 ENSG00000247982 ENSG00000167702 ENSG00000253475 ENSG00000167306
   "AL161626.1"     "LINC00926"         "KIFC2" "RP11-110G21.2"         "MYO5B"
ENSG00000215440 ENSG00000163590 ENSG00000184441 ENSG00000108819 ENSG00000227039
       "NPEPL1"         "PPM1L"    "AP001062.7"       "PPP1R9B"     "ITGB2-AS1"
ENSG00000068831 ENSG00000128872 ENSG00000197549 ENSG00000211934 ENSG00000185522
      "RASGRP2"         "TMOD2"       "PRAMENP"       "IGHV1-2"      "C11orf35"
ENSG00000105655 ENSG00000234902 ENSG00000211945 ENSG00000141577 ENSG00000076344
       "ISYNA1"    "AC007879.2"      "IGHV1-18"          "AZI1"         "RGS11"
ENSG00000160014 ENSG00000047644 ENSG00000169682 ENSG00000130758 ENSG00000125347
        "CALM3"          "WWC3"         "SPNS1"       "MAP3K10"          "IRF1"
ENSG00000224796 ENSG00000225783 ENSG00000197146 ENSG00000174996 ENSG00000125534
      "RPL32P1"          "MIAT"    "AL133458.1"          "KLC2"         "PPDPF"
ENSG00000188599 ENSG00000248275 ENSG00000155158 ENSG00000162877 ENSG00000051128
```

```
       "NPIPP1"    "TRIM52-AS1"        "TTC39B"        "PM20D1"        "HOMER3"
ENSG00000168071 ENSG00000076928 ENSG00000105663 ENSG00000231925 ENSG00000105063
      "CCDC88B"       "ARHGEF1"         "KMT2B"         "TAPBP"        "PPP6R1"
ENSG00000266677 ENSG00000063169 ENSG00000105373 ENSG00000131584 ENSG00000103249
 "RP11-258F1.1"       "GLTSCR1"       "GLTSCR2"         "ACAP3"         "CLCN7"
ENSG00000251301 ENSG00000228727 ENSG00000188185 ENSG00000146285 ENSG00000151651
 "RP11-81H14.2"        "SAPCD1"     "LINC00265"         "SCML4"         "ADAM8"
ENSG00000214021 ENSG00000063245 ENSG00000064547 ENSG00000182379 ENSG00000196668
        "TTLL3"          "EPN1"         "LPAR2"         "NXPH4"     "LINC00173"
ENSG00000163704 ENSG00000124496 ENSG00000244486 ENSG00000099910 ENSG00000136819
        "PRRT3"        "TRERF1"        "SCARF2"        "KLHL22"       "C9orf78"
ENSG00000180096 ENSG00000005844 ENSG00000137216 ENSG00000008710 ENSG00000127419
        "SEPT1"         "ITGAL"       "TMEM63B"          "PKD1"       "TMEM175"
ENSG00000109113 ENSG00000135596 ENSG00000177084 ENSG00000204681 ENSG00000127415
        "RAB34"        "MICAL1"          "POLE"        "GABBR1"          "IDUA"
ENSG00000128284 ENSG00000144283 ENSG00000139668 ENSG00000107742 ENSG00000149499
        "APOL3"          "PKP4"         "WDFY2"        "SPOCK2"          "EML3"
ENSG00000129355 ENSG00000122707 ENSG00000160326 ENSG00000252438 ENSG00000101493
       "CDKN2D"          "RECK"        "SLC2A6"       "SNORD45"        "ZNF516"
ENSG00000124570 ENSG00000158526 ENSG00000161618 ENSG00000169994 ENSG00000123933
     "SERPINB6"          "TSR2"      "ALDH16A1"         "MYO7B"          "MXD4"
ENSG00000148384 ENSG00000143793 ENSG00000005379 ENSG00000100321 ENSG00000122515
       "INPP5E"       "C1orf35"        "BZRAP1"        "SYNGR1"         "ZMIZ2"
ENSG00000196642 ENSG00000104154 ENSG00000105698 ENSG00000077044 ENSG00000136286
        "RABL6"       "SLC30A4"          "USF2"          "DGKD"         "MYO1G"
ENSG00000142173 ENSG00000153443 ENSG00000160799 ENSG00000100351 ENSG00000160796
       "COL6A2"        "UBALD1"        "CCDC12"         "GRAP2"        "NBEAL2"
ENSG00000135318 ENSG00000104960 ENSG00000168264 ENSG00000170476 ENSG00000265735
         "NT5E"         "PTOV1"       "IRF2BP2"          "MZB1"       "RN7SL5P"
ENSG00000004777 ENSG00000182087 ENSG00000101400 ENSG00000185989 ENSG00000142583
     "ARHGAP33"       "TMEM259"         "SNTA1"         "RASA3"        "SLC2A5"
ENSG00000177483 ENSG00000071575 ENSG00000157570 ENSG00000153551 ENSG00000174944
        "RBM44"         "TRIB2"       "TSPAN18"         "CMTM7"        "P2RY14"
ENSG00000164574 ENSG00000182195 ENSG00000185920 ENSG00000133275 ENSG00000154134
      "GALNT10"         "LDOC1"         "PTCH1"       "CSNK1G2"         "ROBO3"
ENSG00000072071 ENSG00000104897 ENSG00000260054 ENSG00000135736 ENSG00000104814
        "LPHN1"         "SF3A2"  "RP11-611L7.1"      "CCDC102A"        "MAP4K1"
ENSG00000137571 ENSG00000184164 ENSG00000132718 ENSG00000007264 ENSG00000100258
      "SLCO5A1"        "CRELD2"         "SYT11"          "MATK"          "LMF2"
ENSG00000106780 ENSG00000099331 ENSG00000167280 ENSG00000185864 ENSG00000143320
```

```
        "MEGF9"         "MYO9B"        "ENGASE"        "NPIPB4"        "CRABP2"
ENSG00000119608 ENSG00000198910 ENSG00000139899 ENSG00000105639 ENSG00000134250
        "PROX2"         "L1CAM"         "CBLN3"          "JAK3"        "NOTCH2"
ENSG00000125648 ENSG00000168280 ENSG00000075826 ENSG00000163386 ENSG00000145020
     "SLC25A23"         "KIF5C"        "SEC31B"        "NBPF10"           "AMT"
ENSG00000198816 ENSG00000182179
       "ZNF358"          "UBA7"
```

5 Figures

We 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")
```

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/fasta/homo_sapiens/dna/Homo_sapiens.GRCh37.68.dna

gunzip Homo_sapiens.GRCh37.68.dna_sm.primary_assembly.fa.gz

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

```
gunzip Homo_sapiens.GRCh37.68.gtf.gz
perl -ne 'if( $_ !~ /^(HS|\S+PATCH|HG)/){ print $_; }' Homo_sapiens.GRCh37.68.gtf \
> Homo_sapiens.GRCh37.68.filtered.gtf
```

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_sm.primary_assembly.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.filtered.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.

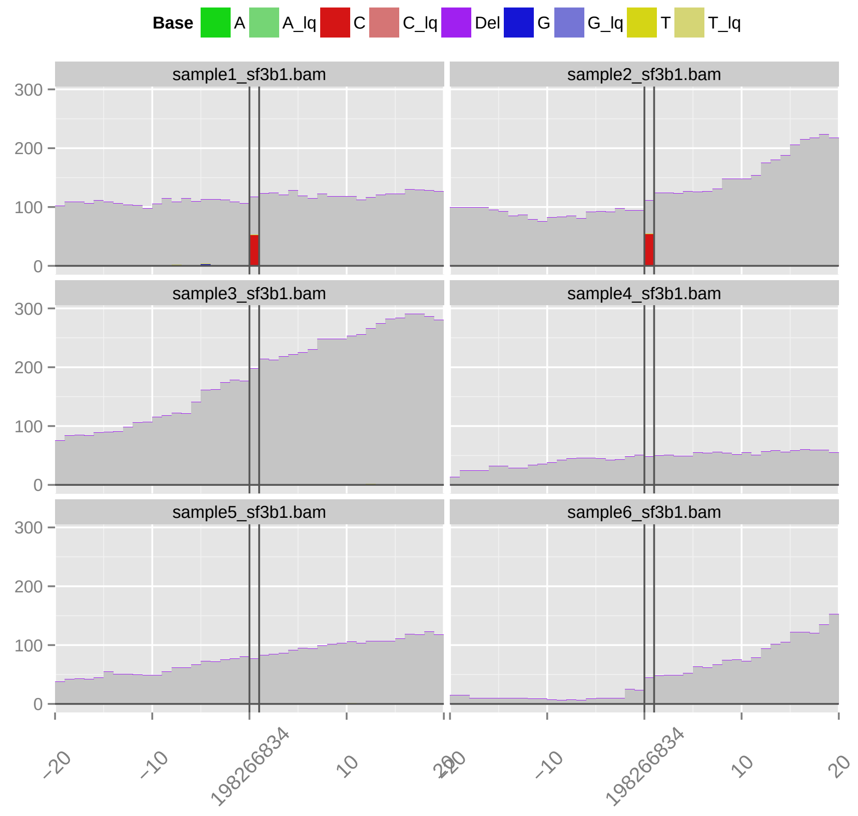
5.1 Figure 1

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
```

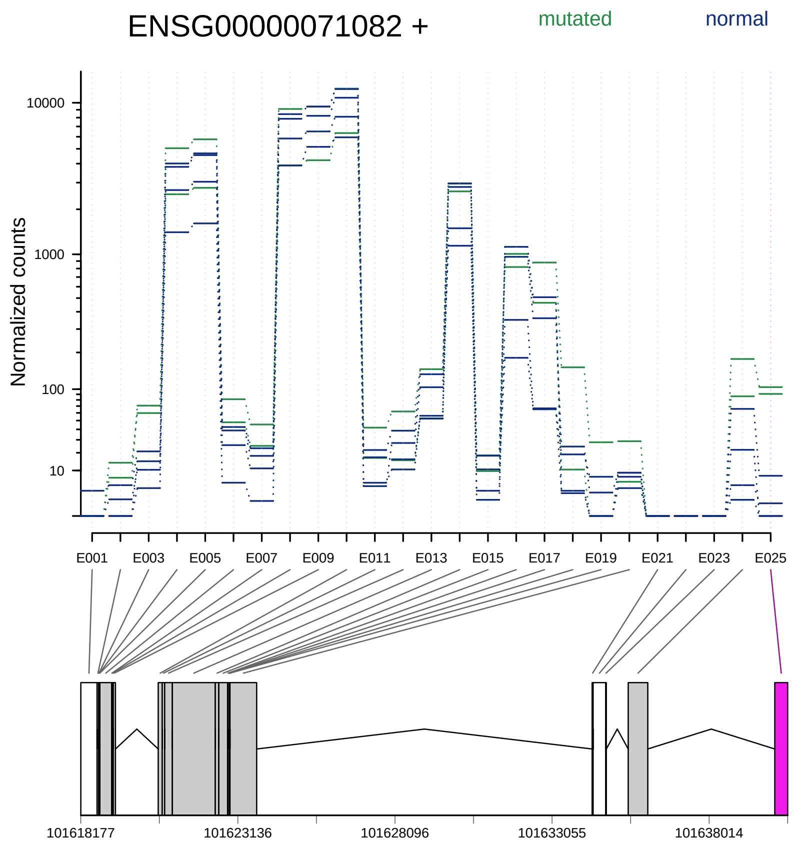
```
library(ggplot2)
p <- mismatchPlot(data, sampledata,
    samples, windowsize, position) + facet_wrap(
    ~ Sample, ncol = 2)
print(p)
```



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.

5.2 Figure 2: RPL31

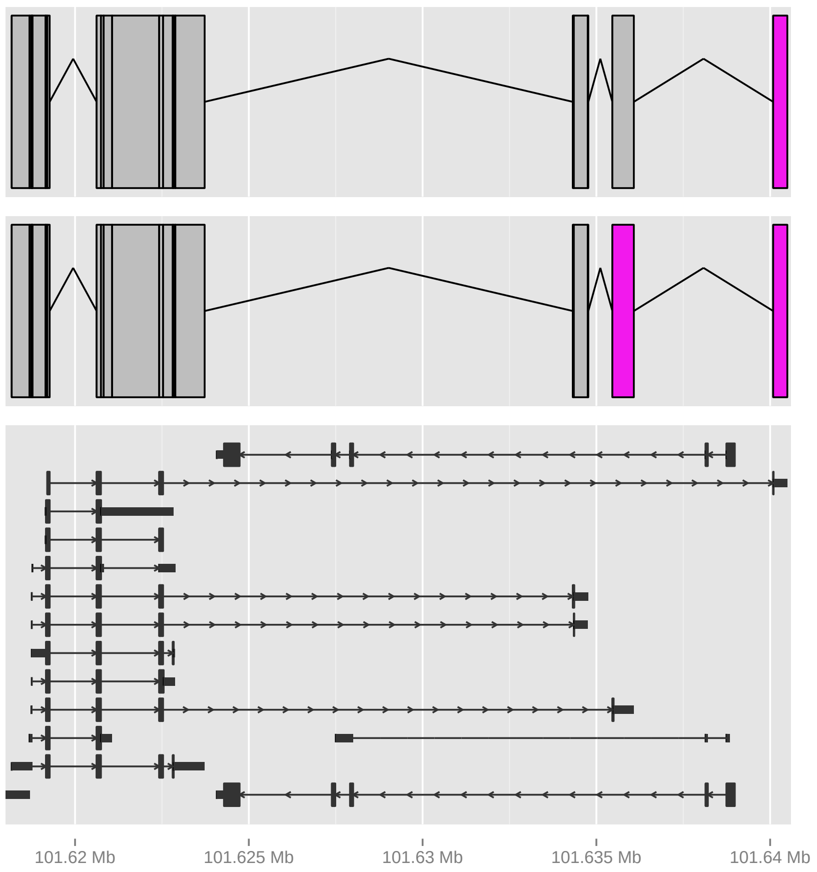
```
plotDEXSeq(ecsSF3B1, "ENSG00000071082",
   norCounts=TRUE, lwd=1.3, legend=TRUE, fitExpToVar="sf3b1",
   splicing=FALSE, expression=FALSE,
   cex.axis=1, color=c("#238B45", "#0C2C84"),
   color.samples=colorSamples)
```



zoomed region,

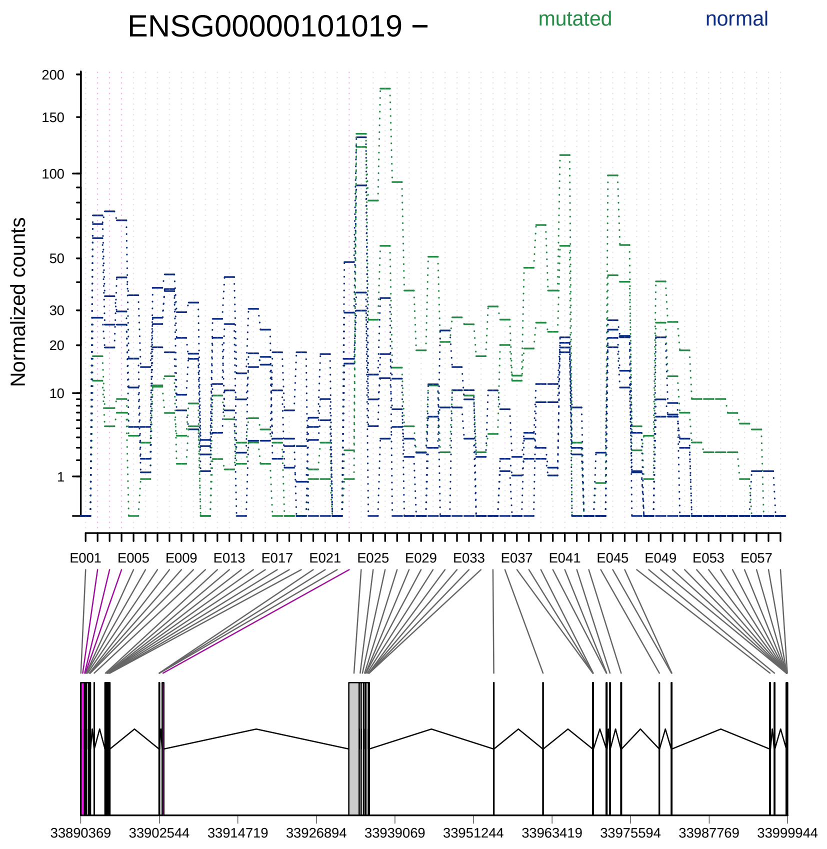
```
library(ggbio)
thisRange <- fData(ecsSF3B1)[geneIDs(ecsSF3B1)
    %in% "ENSG00000071082",
    c("chr", "start", "end", "strand", "padjust")]
exonRange <- GRanges( thisRange$chr,
  IRanges(
     start=thisRange$start,
     end=thisRange$end,
     names=rownames(thisRange)),
   thisRange$strand )

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))
```



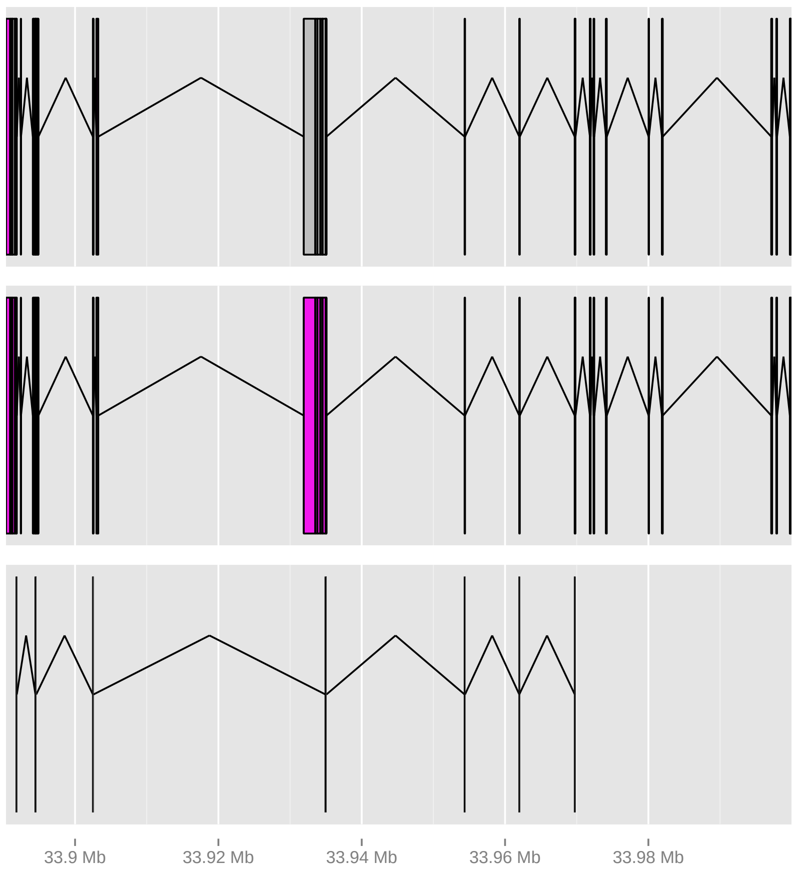
5.3 Figure 3: UQCC

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

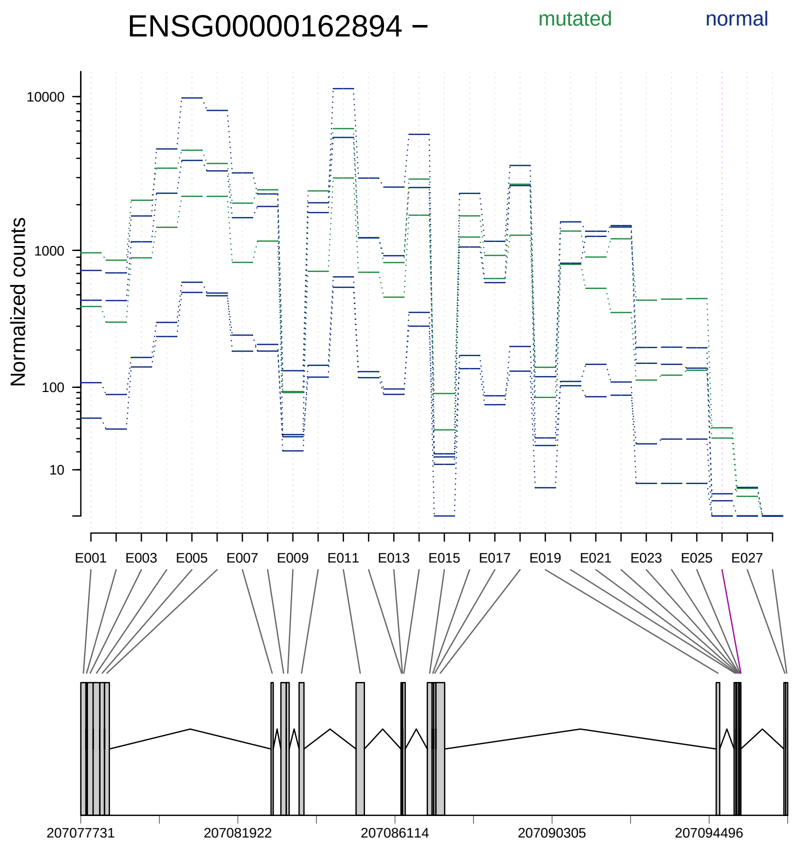


```
thisRange <- fData(ecsSF3B1)[
    geneIDs(ecsSF3B1) %in% "ENSG00000101019",
    c("chr", "start", "end", "strand", "padjust")]
exonRange <- GRanges( thisRange$chr,
  IRanges(
     start=thisRange$start,
     end=thisRange$end,
     names=rownames(thisRange)),
   thisRange$strand )

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)
```

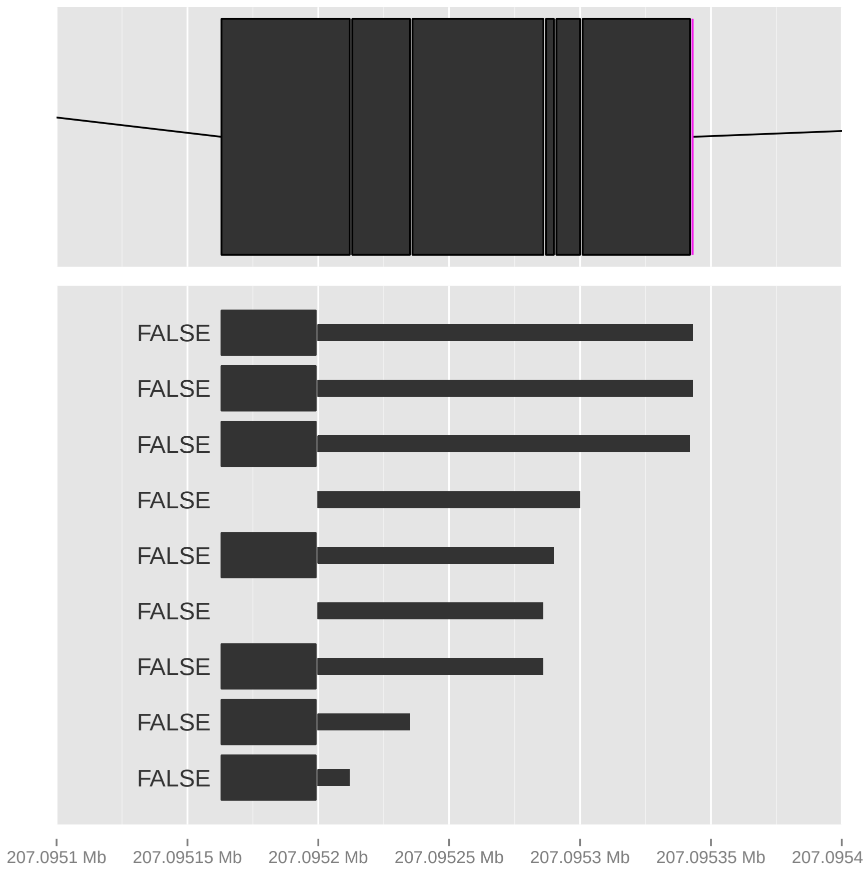


```
plotDEXSeq(ecsSF3B1, "ENSG00000162894", norCounts=TRUE,
    lwd=1, legend=TRUE, fitExpToVar="sf3b1", splicing=FALSE,
    expression=FALSE, cex.axis=1, color=colorConditions,
    color.samples=colorSamples,
    displayTranscripts=FALSE, names=FALSE)
```



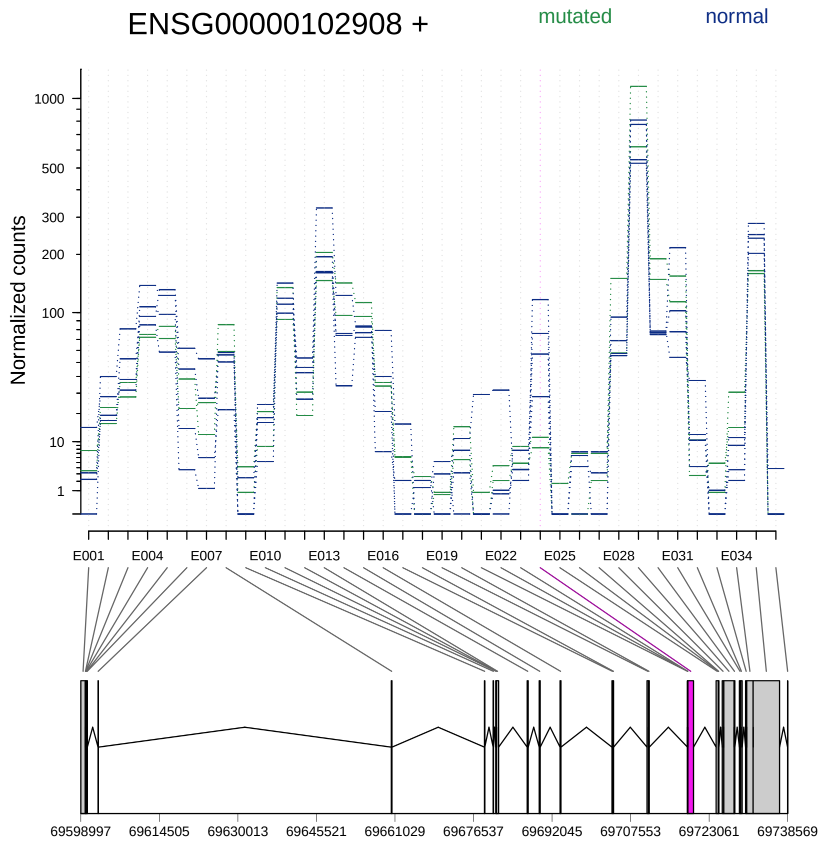
```
thisRange <- fData(ecsSF3B1)[
    geneIDs(ecsSF3B1) %in% "ENSG00000162894",
    c("chr", "start", "end", "strand", "padjust")]
thisRange$padjust[is.na( thisRange$padjust )] <- 1
geneRange <- GRanges( thisRange$chr,
   IRanges(
     start=thisRange$start,
     end=thisRange$end,
     names=rownames(thisRange)),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



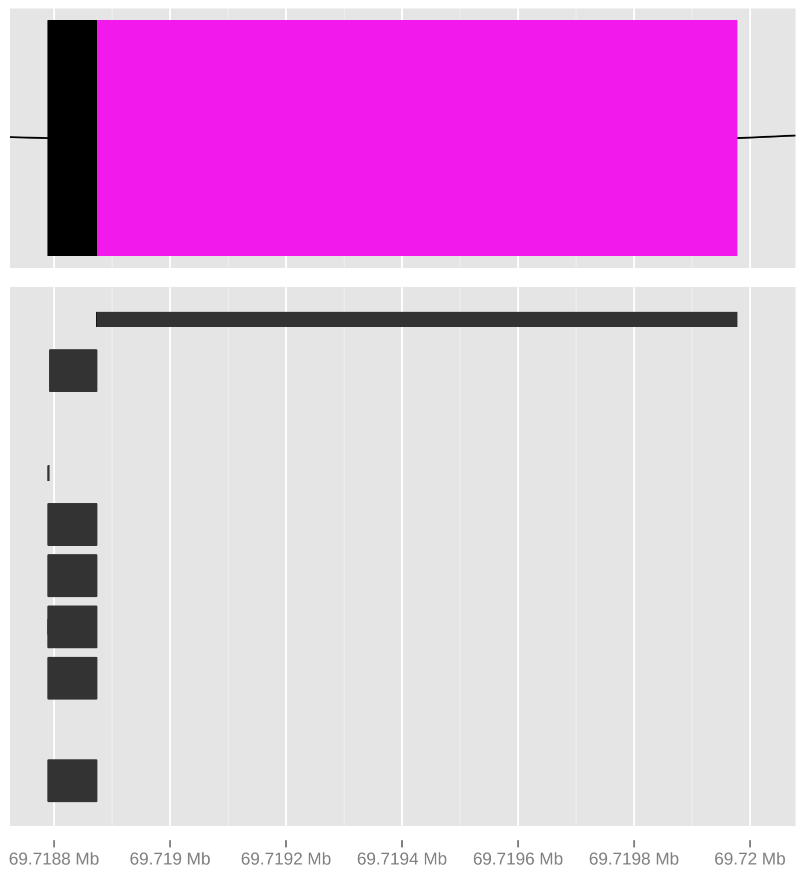
PLot NFAT5

```
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 )
```



6 Session information

```
sessionInfo()
```

```
R version 3.0.3 (2014-03-06)
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] deepSNV_1.8.0            VariantAnnotation_1.8.13 VGAM_0.9-4
 [4] Rsamtools_1.14.3         bit64_0.9-4              bit_1.1-12
 [7] rhdf5_2.6.0              h5vc_1.0.0               Biostrings_2.30.1
[10] GenomicFeatures_1.14.5   AnnotationDbi_1.24.0     ggbio_1.10.16
[13] ggplot2_1.0.0            biomaRt_2.18.0           CLL.SF3B1_0.0.1
[16] DESeq2_1.2.10            RcppArmadillo_0.4.300.0  Rcpp_0.11.1
[19] GenomicRanges_1.14.4     XVector_0.2.0            IRanges_1.20.7
[22] DEXSeq_1.8.0             Biobase_2.22.0           BiocGenerics_0.8.0
[25] BiocInstaller_1.12.1

loaded via a namespace (and not attached):
 [1] annotate_1.40.1     biovizBase_1.10.8   bitops_1.0-6
 [4] BSgenome_1.30.0     cluster_1.15.2      colorspace_1.2-4
 [7] DBI_0.2-7           dichromat_2.0-0     digest_0.6.4
[10] Formula_1.1-1       genefilter_1.44.0   grid_3.0.3
[13] gridExtra_0.9.1     gtable_0.1.2        Hmisc_3.14-4
[16] hwriter_1.3         labeling_0.2        lattice_0.20-29
[19] latticeExtra_0.6-26 locfit_1.5-9.1      MASS_7.3-33
[22] munsell_0.4.2       plyr_1.8.1          proto_0.3-10
[25] RColorBrewer_1.0-5  RCurl_1.95-4.1      reshape_0.8.5
[28] reshape2_1.4        RSQLite_0.11.4      rtracklayer_1.22.7
[31] scales_0.2.4        statmod_1.4.18      stringr_0.6.2
[34] survival_2.37-7     tools_3.0.3         XML_3.98-1.1
[37] xtable_1.7-3        zlibbioc_1.8.0
```