Package GenoView

Sharon Lee

October 13, 2014

Contents

1 Overview

The GenoView package helps a wide audience of users to visualize their genomic data. A main component of the package is the ability to plot user-specified mutations over the exons of a gene. The package also displays protein domain information specific to the hg19 human genome.

2 Setup and Initial Processing

The package uses existing datasets to provide location reference information on genes, exons, strands, and other plot components. All reference information must come from the same genome build for accurate results. This vignette will obtain all its reference information from the hg19 human genome.

```
> # Load relevant packages containing hg19 datasets
> require(TxDb.Hsapiens.UCSC.hg19.knownGene)
> require(biovizBase)
> require(grid)
```

Information from various genomic data types is required for quick plotting. The main function requires RNA transcript information in a TranscriptDb object, sequence lengths in an integer vector, and gene locations in a GRanges dataset object.

```
> # Extract the relevant data from published genomic datasets
> tx.db <- TxDb.Hsapiens.UCSC.hg19.knownGene
> seqs <- seqlengths(tx.db)
> data(genesymbol, package = "biovizBase")
```

3 mutExonPlot

The main function of the GenoView package is creating Mutations over Exons plots (mutExonPlot function). It accepts mut.gr, a GRanges object containing the mutation data. Data.frame objects containing mutation information can be converted to GRanges objects using the provided makeGR function.

The wrapper functions mepHuman and the interface (see Section 5) ensure that we create a GRanges object. Otherwise, we use the makeGR function to obtain a suitable GRanges object before plotting.

For example, we specify the gene as TP53 and then make a basic Mutations Over Exons Plot.

```
> # Select the location of TP53
> exon.int <- genesymbol["TP53"]
> # An example of a truncated plotting interval from 7575000 bp to 7579000 bp
> plot.int <- intSel(exon.int, new.start = 7575000,
+                    new.end = 7579000)
> gr <- biovizBase::crunch(tx.db, which=exon.int, type = "all")
```

Next we create a sample.mut.gr object which contains all mutations within the range of exon.int (TP53), and a sample.mut.trunc object that contains all mutations within the plot.int range (truncated region of TP53).

```
> data(sample.mut.df, package = 'GenoView')
> head(sample.mut.df)
```

```
     ids seqnames   start     end width strand midpoint
1 TP53_1    chr17 7575924 7575924     1      -  7575924
2 TP53_2    chr17 7578512 7578512     1      -  7578512
3 TP53_3    chr17 7577867 7577867     1      -  7577867
4 TP53_4    chr17 7575840 7575840     1      -  7575840
5 TP53_5    chr17 7579720 7579720     1      -  7579720
6 TP53_6    chr17 7579718 7579718     1      -  7579718
```

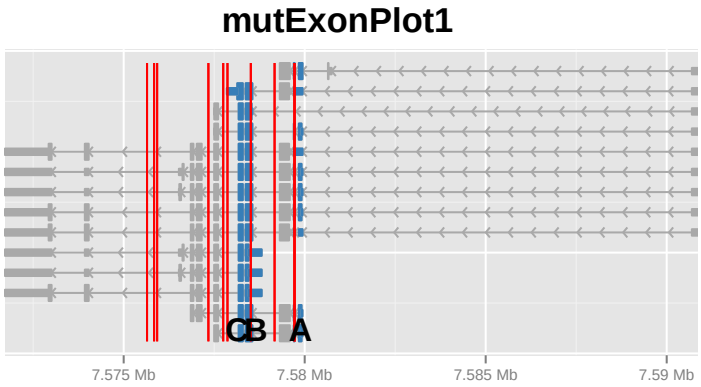
```
> # Using exon.int range (TP53)
> sample.mut.gr <- makeGR(sample.mut.df, chr.col = "seqnames",
+                         s.p.col = "start", e.p.col = "end",
+                         str.col = "strand", id.col = "ids",
+                         show.legend = TRUE, des.seqs = seqs,
+                         plot.int = exon.int)
```

```
> # Using plot.int range (truncated region of TP53)
> sample.mut.trunc <- makeGR(sample.mut.df, chr.col = "seqnames",
+                         s.p.col = "start", e.p.col = "end",
+                         str.col = "strand", id.col = "ids",
+                         show.legend = TRUE, des.seqs = seqs,
+                         plot.int = plot.int) # uses plot.int
```

Once we have a suitable mut.gr object, we can call the mutExonPlot function. This versatile function lets users alter many different aspects of the final plot.

```
> ml1 = mutExonPlot(mut.gr = sample.mut.gr, exon.int = exon.int,
+                   plot.int = exon.int, disp.track = 1, p.height = 4/8,
+                   d.height = 0, l.height = 4/8, plt.title = "mutExonPlot1",
+                   id.col = "IDs", tx.db = tx.db, gr = gr)
```

```
> grid.draw(ml1$mep)
```



IDs	Exon	Exonlabel
TP53_1	NA	NA
TP53_2	5	B
TP53_3	6	C
TP53_4	NA	NA
TP53_5	3	A
TP53_6	3	A
TP53_7	NA	NA
TP53_8	NA	NA
TP53_9	NA	NA
TP53_10	NA	NA

Figure 1: Mutations Over Exons Plot displaying all available RNA transcripts of TP53, created from the sample.mut.df dataset. The exons which contain mutations (red lines) are highlighted in blue and annotated in the legend.

4 Protein (PFAM) Domain Visualization

The genomic coordinates for the PFAM protein domains were taken from the UCSC Table Browser, using the ucscGenePfam.txt.gz file (last modified 21-July-2013).

In addition to this dataset, other functions, such as makePFAMObjs and updatePFAM, retrieve the hg19 PFAM protein domain information and store it in suitable data objects to be passed onto the mutExonPlot function.

```
> # Create PFAM objects using the wrapper function makePFAMObjs
> pfam.objs <- makePFAMObjs()
```

```
> pfam.desc <- pfam.objs$desc
> pfam.ids <- pfam.objs$ids
```

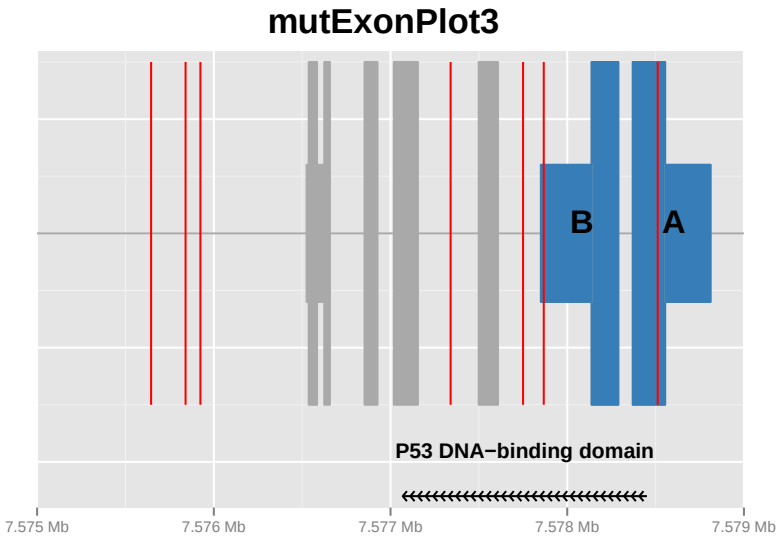
Next, we have to create the GRanges object which contains the genomic coordinates of all protein domains.

```
> # Load the pfam data.frame
> data(pfam.df, package = "GenoView", envir = environment())
> # Create the GRanges object using the pfam.df dataset
> pfam.gr <- GRanges(seqnames = as.character(pfam.df[,"chr"]),
+                         IRanges(start = pfam.df[,"start"],
+                                 end = pfam.df[,"end"]),
+                         strand = pfam.df[,"strand"])
> # Assign the column of domain information as values for pfam.gr
> values(pfam.gr) <- list(domain = as.character(as.matrix(pfam.df[,1])))

> ml2 = mutExonPlot(mut.gr = sample.mut.trunc, exon.int = exon.int,
+                   plot.int = plot.int, p.height = 2/4, d.height = 1/4,
+                   l.height = 1/4, plt.title = "mutExonPlot3",
+                   id.col = "IDs", tx.db = tx.db, pfam.gr = pfam.gr,
+                   pfam.desc = pfam.desc, pfam.ids = pfam.ids, gr = gr)
```

We can then combine this with our hg19 information to annotate the domains in a Mutations Over Exons plot.

```
> grid.draw(ml2$mep)
```



IDs	Exon	Exonlabel
TP53_1	NA	NA
TP53_2	5	A
TP53_3	6	B
TP53_4	NA	NA
TP53_7	NA	NA
TP53_9	NA	NA
TP53_10	NA	NA

Figure 2: Mutations Over Exons Plot displaying one track of exon information, created with the sample.mut.df dataset and hg19 protein domain annotation. The exons which contain mutations (red lines) are highlighted in blue and annotated in the legend.

5 Interfaces

The package is equipped with a graphical user interface and a text interface, in order to maximize ease of use and access for all audiences. Both interfaces provide plotting instructions and allow for repeated plotting, which is useful for reviewing and altering the plot.

5.1 GUI

The GUI is enabled in the mepHuman function by setting gui=TRUE, or it can be enabled directly using the mepGUI function.

```
> if(interactive()) {
+     mepHuman(m.data = sample.mut.df, gui = TRUE)
+ }
```

5.2 Text Interface

The text interface is the default option in the mepHuman function, and it can be enabled directly from the mepTxtInt function.

```
> if(interactive()) {
+     mepHuman(m.data = sample.mut.df, gui = FALSE)
+ }
```