

# The GenomeGraphs user's guide

Steffen Durinck\*and James Bullard†

October 13, 2014

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Creating a Ensembl annotation graphic</b>	<b>2</b>
2.1	Plotting a Gene . . . . .	2
2.2	Adding alternative transcripts . . . . .	3
2.3	Plotting a gene region . . . . .	4
<b>3</b>	<b>Adding Array data to the plot</b>	<b>6</b>
3.1	Array CGH and gene expression array data . . . . .	6
3.2	Exon array data . . . . .	7
3.3	Plotting Conservation Data . . . . .	9
<b>4</b>	<b>Odds and Ends</b>	<b>10</b>
4.1	Overlays . . . . .	12
4.2	GenomeGraphs Classes . . . . .	15

## 1 Introduction

Genomic data analyses can benefit from integrated visualization of the genomic information. The GenomeGraphs package uses the biomaRt package to do live queries to Ensembl and translates e.g. gene/transcript structures to viewports of the grid graphics package, resulting in genomic information plotted together with your data. Possible genomics datasets that can be plotted are: Array CGH data, gene expression data and sequencing data.

---

\*sdurinck@gmail.com

†bullard@stat.berkeley.edu

```
> library(GenomeGraphs)
```

## 2 Creating a Ensembl annotation graphic

To create an Ensembl annotation graphic, you need to decide what you want to plot. Genes and transcripts can be plotted individually using the **Gene** and **Transcript** objects respectively. Or one can plot a gene region the forward strand or reverse strand only or both. In this section we will cover these different graphics.

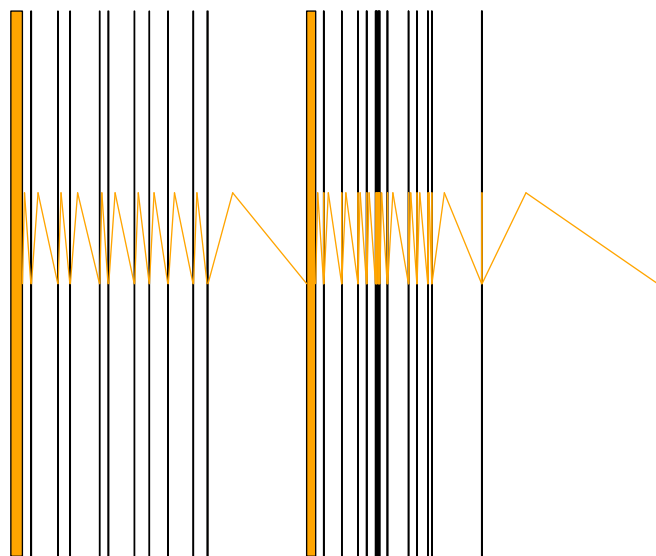
### 2.1 Plotting a Gene

If one wants to plot annotation information from Ensembl then you need to connect to the Ensembl BioMart database using the useMart function of the biomaRt package.

```
> mart <- useMart("ensembl", dataset="hsapiens_gene_ensembl")
```

Next we can retrieve the gene structure of the gene of interest.

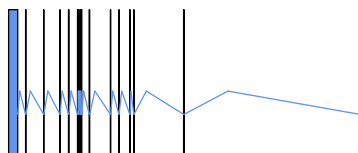
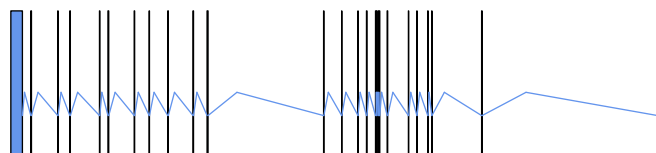
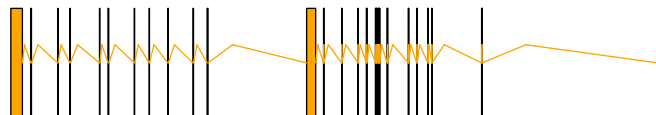
```
> gene <- makeGene(id = "ENSG00000095203", type="ensembl_gene_id", biomaRt = mart)
> gdPlot(gene)
```



## 2.2 Adding alternative transcripts

To add alternative transcripts you first have to create a **Transcript** object. Note that the order of the objects in the list determines the order in the plot.

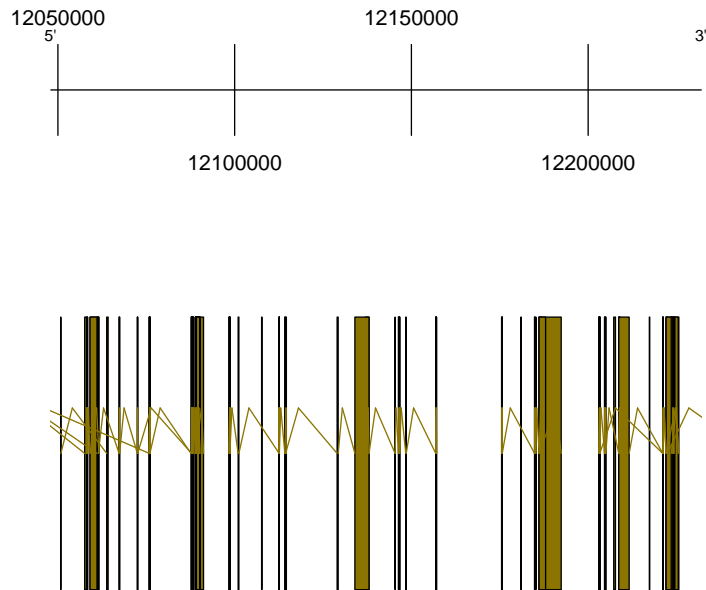
```
> transcript <- makeTranscript(id = "ENSG00000095203", type="ensembl_gene_id", biomaRt = biomaRt)
> gdPlot(list(gene, transcript))
```



## 2.3 Plotting a gene region

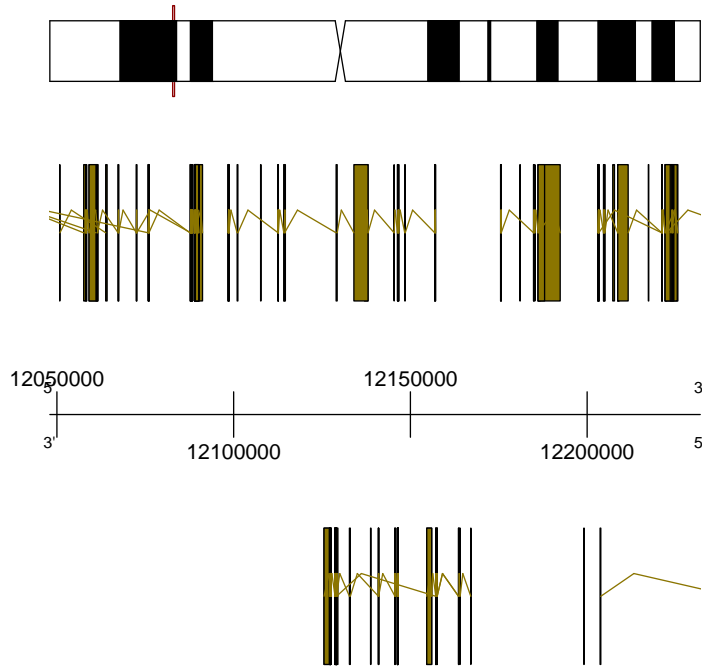
If you're interested in not just plotting one gene but a whole gene region the you should create a **GeneRegion** object. Note that a **GeneRegion** object is strand specific. In the example below we will retrieve the genes on the forward (+) strand only and add a genomic axis as well to give us the base positions.

```
> plusStrand <- makeGeneRegion(chromosome = 19, start = 12050000, end = 12230000, str
> genomeAxis <- makeGenomeAxis(add53 = TRUE)
> gdPlot(list(genomeAxis, plusStrand))
```



Let's now add the genes on the negative strand as well and an ideogram of chromosome 17, highlighting the region we are looking at.

```
> minStrand <- makeGeneRegion( chromosome = 19, start = 12050000, end = 12230000, str
> ideogram <- makeIdeogram(chromosome = 19)
> genomeAxis <- makeGenomeAxis(add53=TRUE, add35=TRUE)
> gdPlot(list(ideogram, plusStrand, genomeAxis, minStrand))
```



### 3 Adding Array data to the plot

#### 3.1 Array CGH and gene expression array data

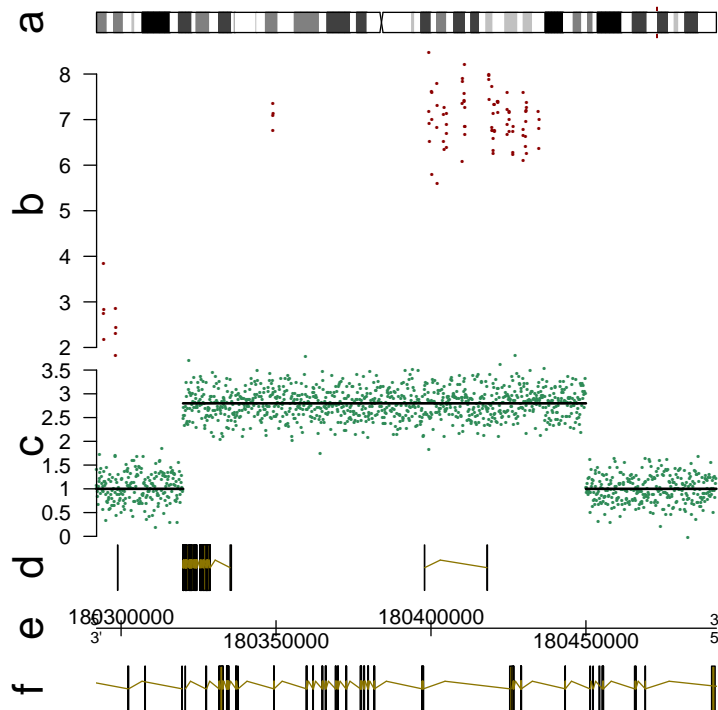
The `Generic Array` object enables plotting of expression and CGH array data together with segments if available. The array intensity data should be given as a matrix, with in the rows the different probes and in the columns the different samples. For each probe the start location should be given using the `probeStart` argument. This should be a one column matrix. Lets load some dummy data.

```
> data("exampleData", package="GenomeGraphs")
> minbase <- 180292097
> maxbase <- 180492096
> genesplus <- makeGeneRegion(start = minbase, end = maxbase,
+                             strand = "+", chromosome = "3", biomaart=mart)
> genesmin <- makeGeneRegion(start = minbase, end = maxbase,
+                             strand = "-", chromosome = "3", biomaart=mart)
```

```

> seg <- makeSegmentation(segStart[[1]], segEnd[[1]], segments[[1]],
+                         dp = DisplayPars(color = "black", lwd=2,lty = "solid"))
> cop <- makeGenericArray(intensity = cn, probeStart = probestart,
+                         trackOverlay = seg, dp = DisplayPars(size=3, color = "seagreen"))
> ideog <- makeIdeogram(chromosome = 3)
> expres <- makeGenericArray(intensity = intensity, probeStart = exonProbePos,
+                         dp = DisplayPars(color="darkred", type="point"))
> genomeAxis <- makeGenomeAxis(add53 = TRUE, add35=TRUE)
> gdPlot(list(a=ideog,b=expres,c=cop,d=genesplus,e=genomeAxis,f=genesmin),
+       minBase = minbase, maxBase =maxbase, labelCex = 2)

```



### 3.2 Exon array data

The example below plots probe level exon array data and is useful in relating alternative splicing with known transcript structures.

```

> data("unrData", package="GenomeGraphs")
> title <- makeTitle(text = "ENSG00000009307", color = "darkred")

```

```

> exon <- makeExonArray(intensity = unrData, probeStart = unrPositions[,3],
+                       probeEnd=unrPositions[,4], probeId = as.character(unrPositions[,1]),
+                       nProbes = unrNProbes, dp = DisplayPars(color = "blue", mapColor = "dodg
+                       displayProbesets=FALSE)
> affyModel.model <- makeGeneModel(start = unrPositions[,3], end = unrPositions[,4])
> affyModel <- makeAnnotationTrack(start = unrPositions[,3], end = unrPositions[,4],
+                                 feature = "gene_model", group = "ENSG00000009307",
+                                 dp = DisplayPars(gene_model = "darkblue"))
> gene <- makeGene(id = "ENSG00000009307", biomaRt = mart)
> transcript <- makeTranscript( id ="ENSG00000009307" , biomaRt = mart)
> legend <- makeLegend(c("affyModel","gene"), fill = c("darkgreen","orange"))
> rOverlay <- makeRectangleOverlay(start = 115085100, end = 115086500, region = c(3,5
+                                     dp = DisplayPars(alpha = .2, fill = "olivedrab1"))
> gdPlot(list(title, exon, affyModel, gene, transcript, legend),
+         minBase = 115061061, maxBase=115102147, overlay = rOverlay)

```





### 3.3 Plotting Conservation Data

The UCSC genome browser offers downloadable conservation data for a variety of species. Here we show how you can plot that conservation data along with annotation.

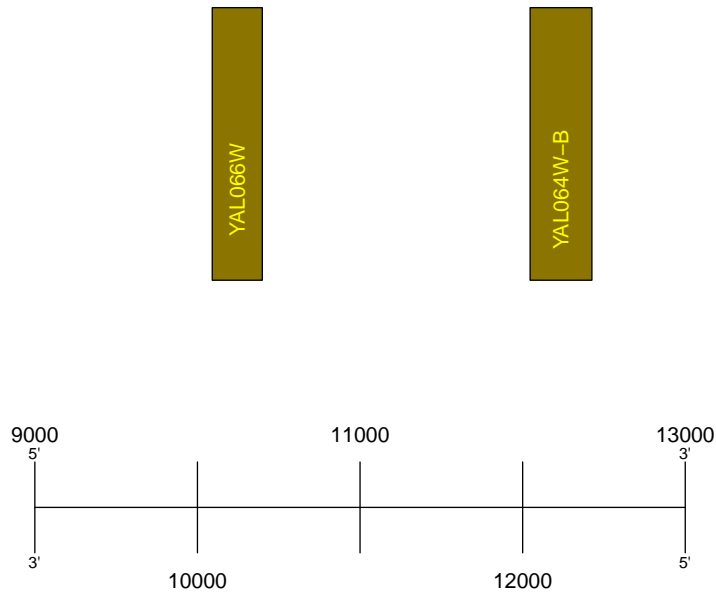
```
> yeastMart <- useMart("ensembl", dataset = "scerevisiae_gene_ensembl")
> minB <- 10000
> maxB <- 20000
> chrRoman <- as.character(as.roman(1))
> grP <- makeGeneRegion(start = minB, end = maxB, strand = "+",
+                       chromosome = chrRoman, biomart = yeastMart)
> grM <- makeGeneRegion(start = minB, end = maxB, strand = "-",
+                       chromosome = chrRoman, biomart = yeastMart)
> gaxis <- makeGenomeAxis(add53 = TRUE, add35 = TRUE)
> conserv <- yeastCons1[yeastCons1[,1] > minB & yeastCons1[,1] < maxB, ]
> s1 <- makeSmoothing(x = lowess(conserv[,1], conserv[,2], f = .01)$x,
+                    y = lowess(conserv[,1], conserv[,2], f = .01)$y,
+                    dp = DisplayPars(lwd = 3, color = "green"))
> s2 <- makeSmoothing(x = lowess(conserv[,1], conserv[,2], f = .1)$x,
+                    y = lowess(conserv[,1], conserv[,2], f = .1)$y,
+                    dp = DisplayPars(lwd = 3, color = "purple"))
> consTrack <- makeBaseTrack(base = conserv[, 1], value = conserv[,2],
+                            dp = DisplayPars(lwd=.2, ylim = c(0, 1.25),
+                            color = "darkblue"), trackOverlay = list(s1, s2))
> gdPlot(list(grP, gaxis, grM, "conservation" = consTrack))
```



## 4 Odds and Ends

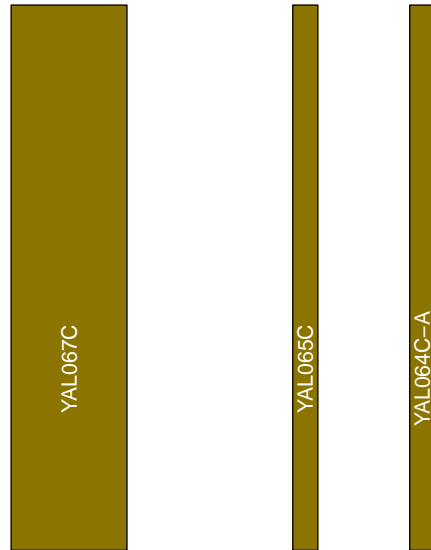
In addition to plotting the genes we can enable the plotting of names of genes.

```
> plotGeneRegion <- function(chr = 1, minB = 9000, maxB = 13000, rot = 0, col = "green",
+   chrRoman <- as.character(as.roman(1:17)[chr]),
+   grP <- makeGeneRegion(start = minB, end = maxB,
+                         strand = "+", chromosome = chrRoman, biomart = yeastMart,
+                         dp = DisplayPars(plotId = TRUE, idRotation = rot,
+                         idColor = col))
+   gaxis <- makeGenomeAxis( add53 = TRUE, add35 = TRUE, littleTicks = FALSE)
+   gdPlot(list(grP, gaxis), minBase = minB, maxBase = maxB)
+ }
> plotGeneRegion(col = "yellow", rot=90)
```



Finally, if you are interested in seeing how things look you can just plot the object without the list, or without the *minBase*, *maxBase* arguments.

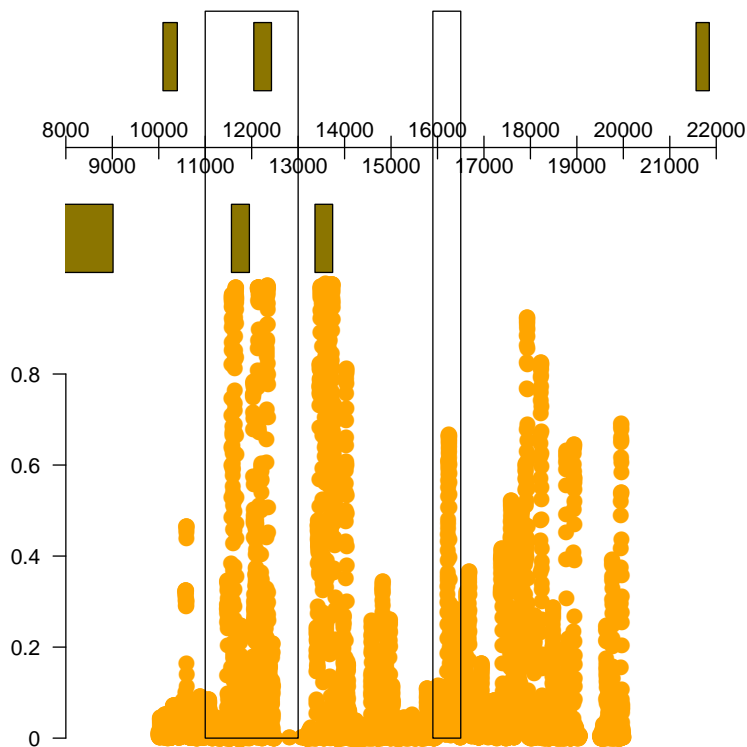
```
> gdPlot(makeGeneRegion(start = 9000, end = 15000, biomart = yeastMart,
+                        strand = "-", chromosome = "I",
+                        dp = DisplayPars(plotId=TRUE)))
```



## 4.1 Overlays

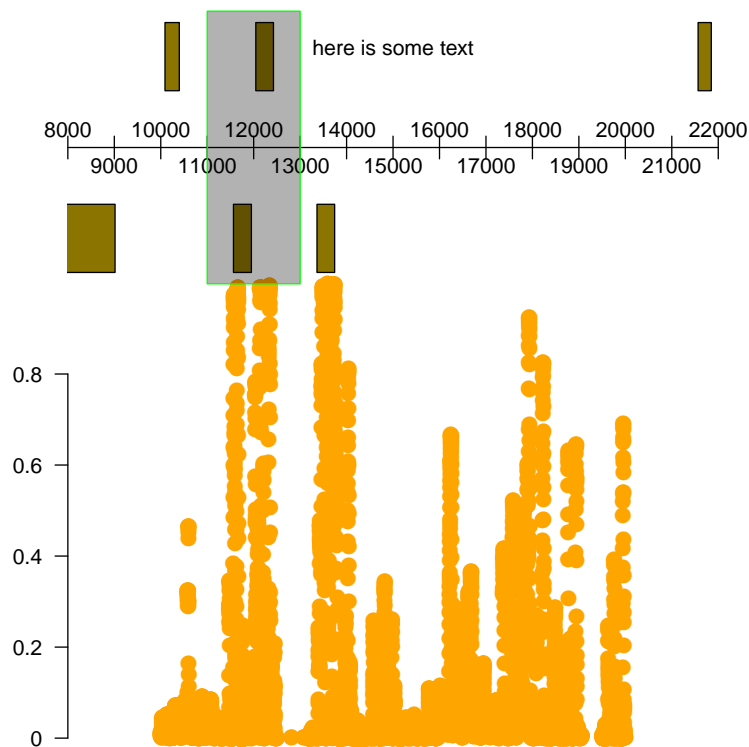
Overlays can be used to annotate different regions of the plot. Currently, we can draw boxes and write text on the plot.

```
> ga <- makeGenomeAxis()
> grF <- makeGeneRegion(start = 10000, end = 20000, chromosome = "I", strand = "+", b
> grR <- makeGeneRegion(start = 10000, end = 20000, chromosome = "I", strand = "-", b
> bt <- makeBaseTrack(base = yeastCons1[,1], value = yeastCons1[,2])
> hr1 <- makeRectangleOverlay(start = 11000, end = 13000)
> hr2 <- makeRectangleOverlay(start = 15900, end = 16500)
> gdPlot(list(grF, ga, grR, bt), overlays = list(hr1, hr2))
```



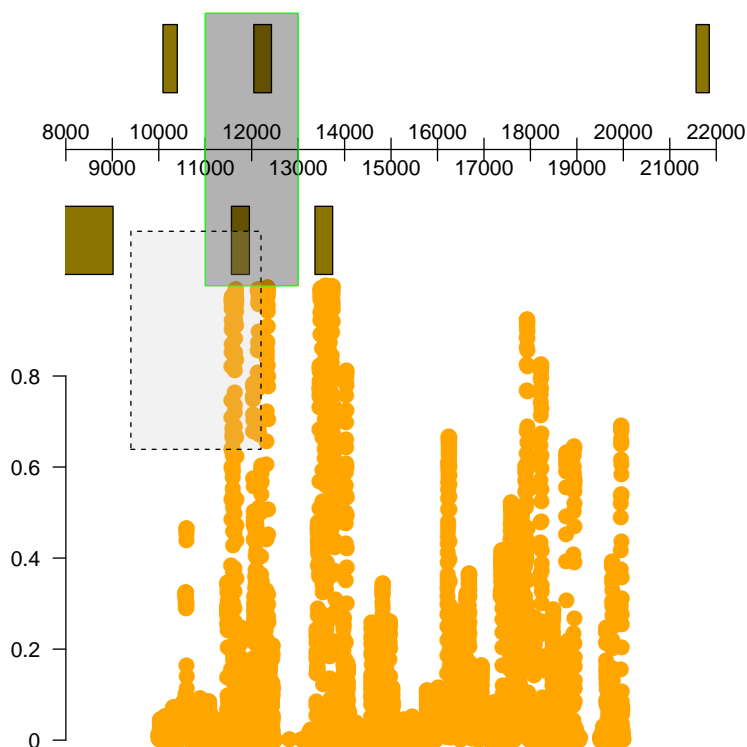
A little nifty feature is to allow alpha blending to make things slightly transparent. If the device you wish to plot on however, does not support transparency then you will get a warning.

```
> ro <- makeRectangleOverlay(start = 11000, end = 13000, region = c(1,3),
+                             dp = DisplayPars(color = "green", alpha = .3))
> to <- makeTextOverlay("here is some text", xpos = 15000, ypos = .95)
> gdPlot(list(grF, ga, grR, bt), overlay = c(ro, to))
```



Also, one can use "absolute" coordinates to specify a region just in case one wants to be a bit more precise.

```
> roR <- makeRectangleOverlay(start = .1, end = .3, coords = "absolute",
+                             dp = DisplayPars(fill = "grey", alpha = .2, lty = "dash",
+                             region = c(.4,.7))
> gdPlot(list(grF, ga, grR, bt), overlays = list(ro, roR))
```



## 4.2 GenomeGraphs Classes

```

> data("seqDataEx", package = "GenomeGraphs")
> str = seqDataEx$david[, "strand"] == 1
> biomart = useMart("ensembl", "scerevisiae_gene_ensembl")
> pList = list("-" = makeGeneRegion(chromosome = "IV", start = 1300000, end = 1310000
+                                   strand = "-", biomart = biomart,
+                                   dp = DisplayPars(plotId = TRUE, idRotation = 0,
+                                   makeGenomeAxis(dp = DisplayPars(size = 3)),
+                                   "+" = makeGeneRegion(chromosome = "IV", start = 1300000, end = 1310000
+                                   strand = "+", biomart = biomart,
+                                   dp = DisplayPars(plotId = TRUE, idRotation = 0,
+                                   "Nagalakshmi" = makeBaseTrack(base = seqDataEx$snyder[, "location"],
+                                   dp = DisplayPars(lwd = .3, color = "darkred"),
+                                   "David +" = makeGenericArray(probeStart = seqDataEx$david[str, "location"],
+                                   intensity = seqDataEx$david[str, "expression"],
+                                   dp = DisplayPars(pointSize = .5)),
+                                   "David -" = makeGenericArray(probeStart = seqDataEx$david[!str, "location"],
+                                   intensity = seqDataEx$david[!str, "expression"],
+                                   dp = DisplayPars(pointSize = .5))

```

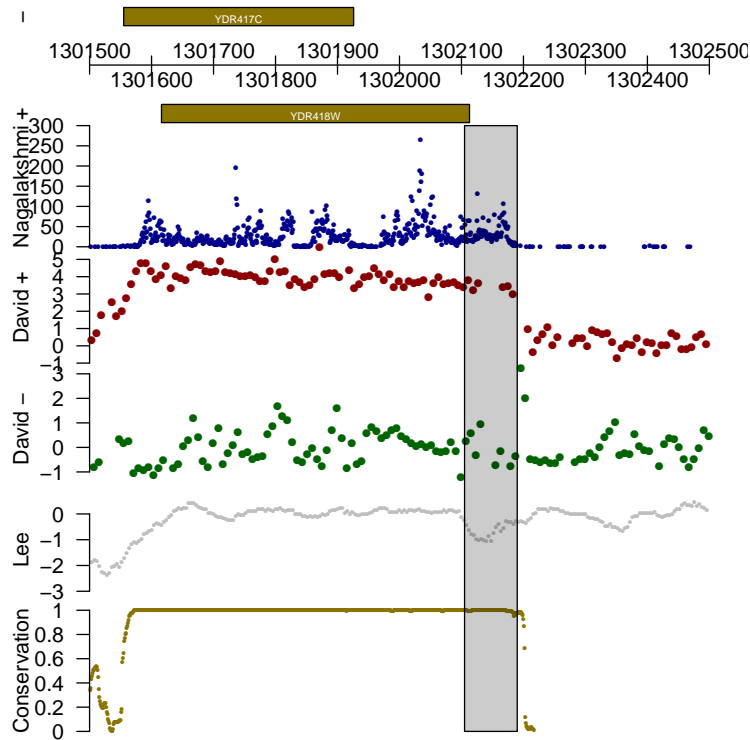
class	description
<code>gdObject</code>	the root class of the system, never directly instantiated
<code>Gene</code>	class representing a gene
<code>GeneRegion</code>	class defining a region of a chromosome, generally a set of genetic elements (genes)
<code>Transcript</code>	class defining a transcript
<code>TranscriptRegion</code>	class defining a region of a chromosome, generally a set of genetic elements (transcripts)
<code>Ideogram</code>	an ideogram
<code>Title</code>	class to draw a title
<code>Legend</code>	class to draw a legend
<code>GenomeAxis</code>	class to draw a axis
<code>Segmentation</code>	class to draw horizontal lines in various sets of data
<code>GenericArray</code>	class to draw data from microarrays.
<code>ExonArray</code>	class to draw data from exon microarrays.
<code>GeneModel</code>	class to draw custom gene models (intron-exon structures)
<code>BaseTrack</code>	class to draw whatever kind of data at a given base
<code>MappedRead</code>	class to plot sequencing reads that are mapped to the genome
<code>DisplayPars</code>	class managing various plotting parameters
<code>AnnotationTrack</code>	class used to represent custom annotation
<code>Overlay</code>	root class for overlays, never directly instantiated
<code>RectangleOverlay</code>	class to represent rectangular regions of interest
<code>TextOverlay</code>	class to draw text on plots



```

+                                     intensity = seqDataEx$david[!str, "expr"]
+                                     dp = DisplayPars(color = "darkgreen", pointSize = .5)),
+                                     "Lee" = makeBaseTrack(base = seqDataEx$nislow[, "location"],
+                                     value = seqDataEx$nislow[, "evaluate"], dp = Disp
+                                     "Conservation" = makeBaseTrack(base = seqDataEx$conservation[, "locat
+                                     value = seqDataEx$conservation[, "scor
+                                     dp = DisplayPars(color="gold4", lwd=.2
> gdPlot(pList, minBase = 1301500, maxBase = 1302500,
+         overlay = makeRectangleOverlay(start = 1302105, end = 1302190, region = c(4,

```



We can also employ different plotting types for the BaseTrack object.

```

> setPar(pList$Lee@dp, "type", "h")
> setPar(pList$Lee@dp, "color", "limegreen")
> setPar(pList$Lee@dp, "lwd", 2)
> gdPlot(pList, minBase = 1301500, maxBase = 1302500,
+         overlay = makeRectangleOverlay(start = 1302105, end = 1302190, region = c(4,
+         dp = DisplayPars(alpha = .2)))

```

