The GenomeGraphs user's guide

Steffen Durinck*and James Bullard†

August 6, 2008

Contents

1 Introduction

Genomic data analyses can benifit 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.

*steffen@stat.berkeley.edu
†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.

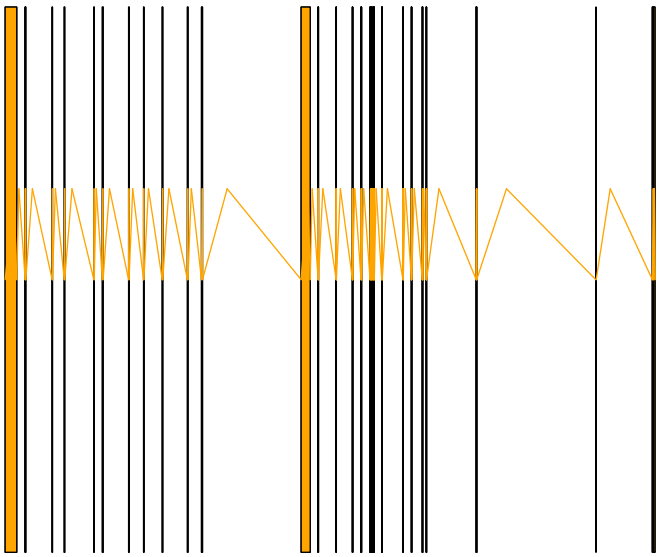
2.1 Plotting a Gene

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.

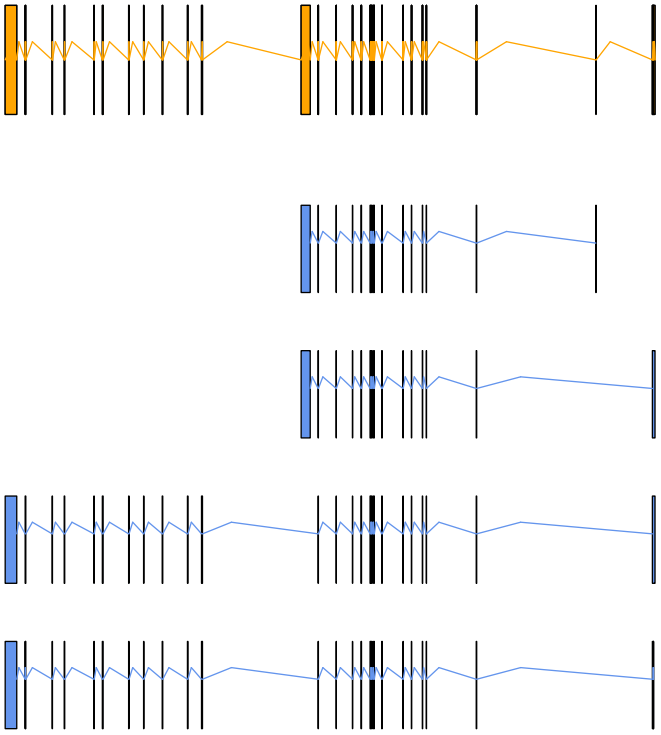
```
> gene = new("Gene", id = "ENSG00000095203",
+     type = "ensembl_gene_id", biomart = mart)
> gdPlot(list(gene), minBase = 110974000,
+     maxBase = 111122900)
```



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.

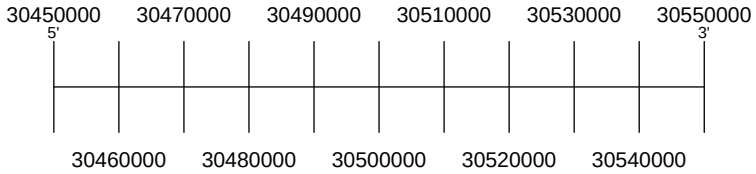
```
> transcript = new("Transcript", id = "ENSG00000095203",
+     type = "ensembl_gene_id", biomart = mart)
> gdPlot(list(gene, transcript), minBase = 110974000,
+     maxBase = 111122900)
```

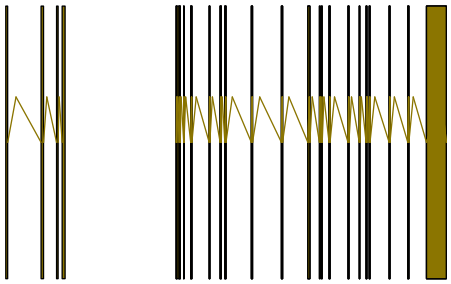


2.3 Plotting a gene region

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.

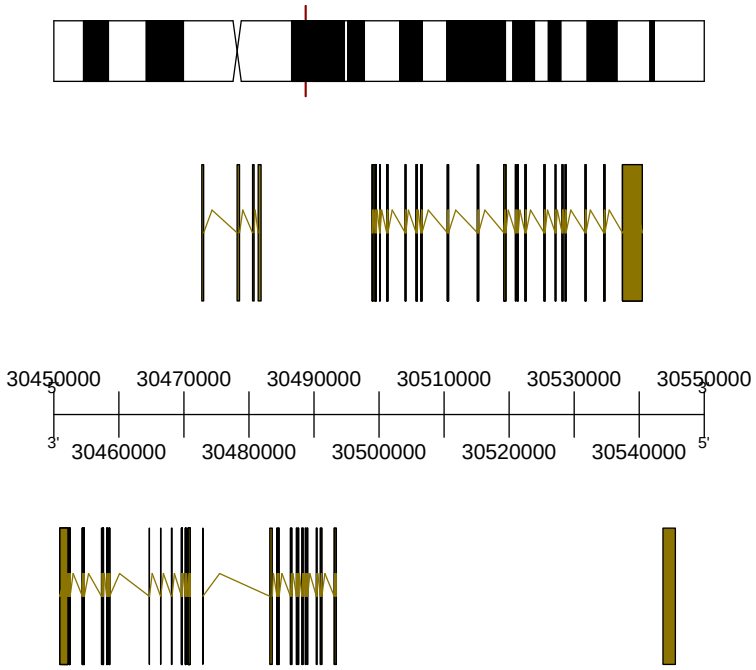
```
> plusStrand = new("GeneRegion", chromosome = "17",
+     start = 30450000, end = 30550000,
+     strand = "+", biomart = mart)
> genomeAxis = new("GenomeAxis", add53 = TRUE)
> gdPlot(list(genomeAxis, plusStrand), minBase = 30450000,
+     maxBase = 30550000)
```





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 = new("GeneRegion", chromosome = "17",
+     start = 30450000, end = 30550000,
+     strand = "-", biomart = mart)
> ideogram = new("Ideogram", chromosome = "17")
> genomeAxis = new("GenomeAxis", add53 = TRUE,
+     add35 = TRUE)
> gdPlot(list(ideogram, plusStrand, genomeAxis,
+     minStrand), minBase = 30450000, maxBase = 30550000)
```



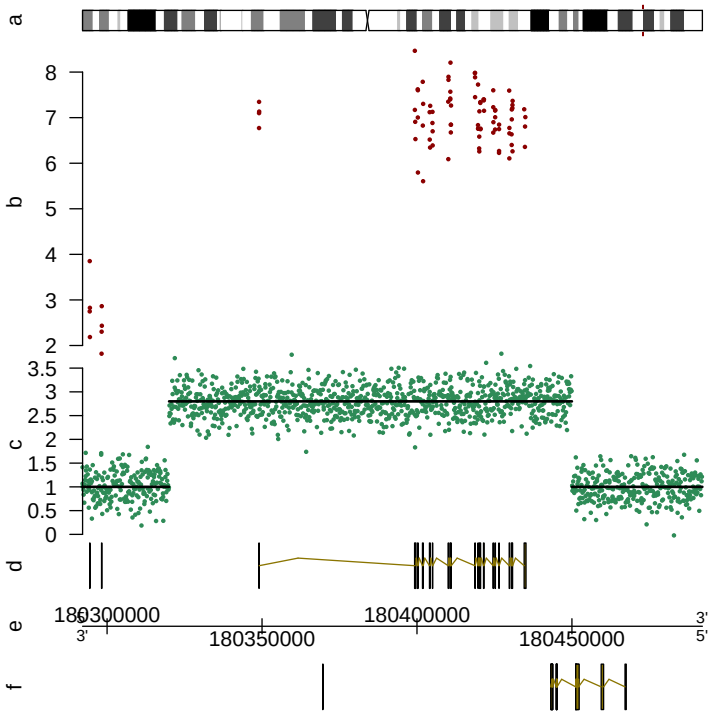
3 Adding Array data to the plot

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 t he 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("dummyData", package = "GenomeGraphs")
> minbase <- 180292097
> maxbase <- 180492096
> genesplus <- new("GeneRegion", start = minbase,
+     end = maxbase, strand = "+", chromosome = "3",
+     biomart = mart)
> genesmin <- new("GeneRegion", start = minbase,
```

```
+     end = maxbase, strand = "-", chromosome = "3",
+     biomart = mart)
> seg <- new("Segmentation", segments = segments,
+     segmentStart = segStart, segmentEnd = segEnd,
+     dp = DisplayPars(color = "black",
+         lwd = 2, lty = "solid"))
> cop <- new("GenericArray", intensity = cn,
+     probeStart = probestart, segmentation = seg,
+     dp = DisplayPars(size = 3, color = "seagreen",
+         type = "dot"))
> ideog <- new("Ideogram", chromosome = "3")
> expres <- new("GenericArray", intensity = intensity,
+     probeStart = exonProbePos, dp = DisplayPars(color = "darkred",
+         type = "point"))
> genomeAxis <- new("GenomeAxis", add53 = TRUE,
+     add35 = TRUE)
> gdPlot(list(a = ideog, b = expres, c = cop,
+     d = genesplus, e = genomeAxis, f = genesmin),
+     minBase = minbase, maxBase = maxbase)
```

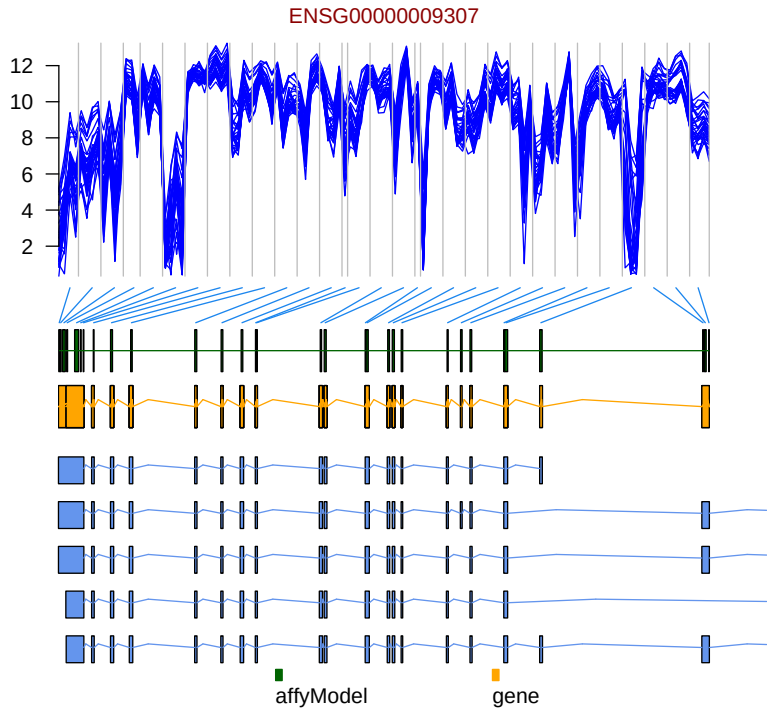


3.2 Exon array data

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

```
> data("unrData", package = "GenomeGraphs")
> title <- new("Title", title = "ENSG00000009307",
+     dp = DisplayPars(color = "darkred"))
> exon <- new("ExonArray", intensity = unrData,
+     probeStart = unrPositions[, 3], probeEnd = unrPositions[,
+         4], probeId = as.character(unrPositions[,
+         1]), nProbes = unrNProbes, dp = DisplayPars(color = "blue",
+         mapColor = "dodgerblue2"), displayProbesets = FALSE)
> affyModel <- new("GeneModel", exonStart = unrPositions[,
+     3], exonEnd = unrPositions[, 4])
> gene <- new("Gene", id = "ENSG00000009307",
+     biomart = mart)
> transcript <- new("Transcript", id = "ENSG00000009307",
```

```
+     biomart = mart)
> legend <- new("Legend", legend = c("affyModel",
+     "gene"), dp = DisplayPars(color = c("darkgreen",
+     "orange")))
> gdPlot(list(title, exon, affyModel, gene,
+     transcript, legend), minBase = min(exon@probeStart),
+     maxBase = max(exon@probeEnd))
```

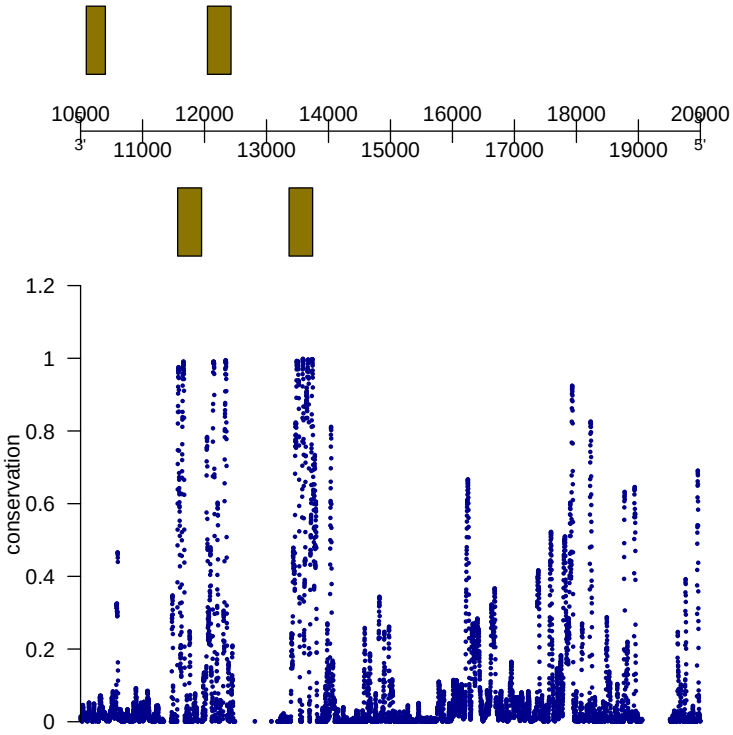


3.3 Plotting Conservation Data

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))
```

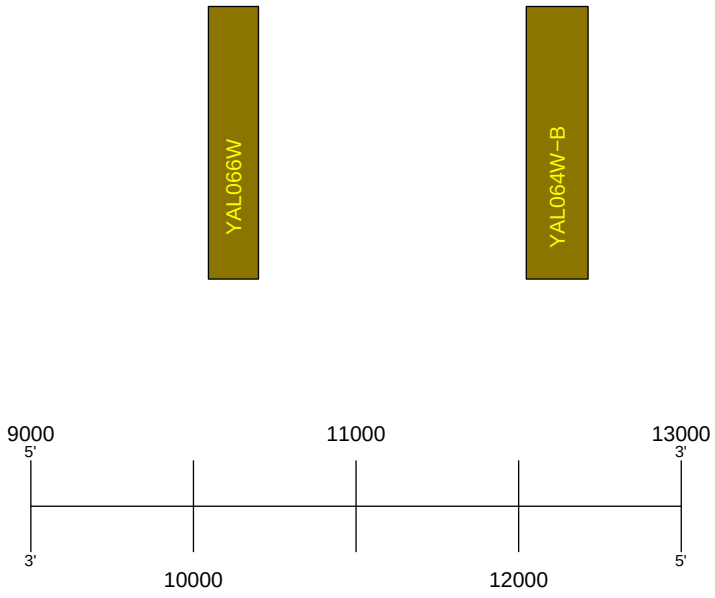
```
> grP <- new("GeneRegion", start = minB,
+     end = maxB, strand = "+", chromosome = chrRoman,
+     biomart = yeastMart)
> grM <- new("GeneRegion", start = minB,
+     end = maxB, strand = "-", chromosome = chrRoman,
+     biomart = yeastMart)
> gaxis <- new("GenomeAxis", add53 = TRUE,
+     add35 = TRUE)
> conserv <- yeastCons1[yeastCons1[, 1] >
+     minB & yeastCons1[, 1] < maxB, ]
> consTrack <- new("BaseTrack", base = conserv[,
+     1], value = conserv[, 2], dp = DisplayPars(lwd = 0.2,
+     ylim = c(0, 1.25), color = "darkblue"))
> gdPlot(list(grP, gaxis, grM, conservation = consTrack),
+     minBase = minB, maxBase = maxB)
```



4 Odds and Ends

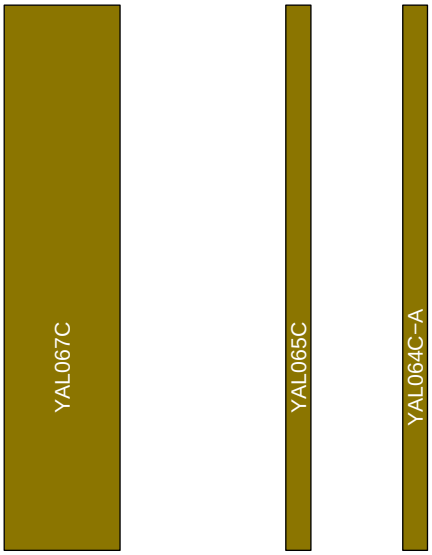
In addition to plotting the genes we can enable the plotting of names of genes.

```
> plotGeneRegion <- function(chr = 1, minB = 9000,
+     maxB = 13000, rot = 0, col = "green") {
+     chrRoman <- as.character(as.roman(1:17)[chr])
+     grP <- new("GeneRegion", start = minB,
+         end = maxB, strand = "+", chromosome = chrRoman,
+         biomart = yeastMart, dp = DisplayPars(plotId = TRUE,
+             idRotation = rot, idColor = col))
+     gaxis <- new("GenomeAxis", 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(new("GeneRegion", start = 9000,
+     end = 15000, biomart = yeastMart,
+     strand = "-", chromosome = "I", dp = DisplayPars(plotId = TRUE)))
```

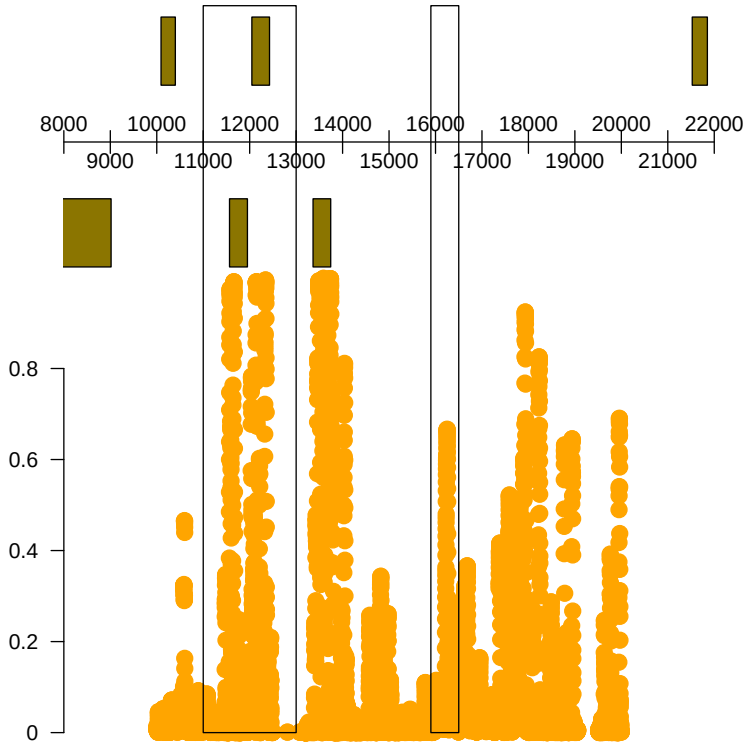


4.1 HighlightRegions

We can also highlight vertical areas of interest using `HighlightRegions`. This allows us to highlight certain areas of the plot. The key things to notice here are that the coordinates for the horizontal points are in genomic coordinates and the vertical coordinates represent the "tracks" that you wish to be highlighted.

```
> ga <- new("GenomeAxis")
> grF <- new("GeneRegion", start = 10000,
+     end = 20000, chromosome = "I", strand = "+",
+     biomart = yeastMart)
```

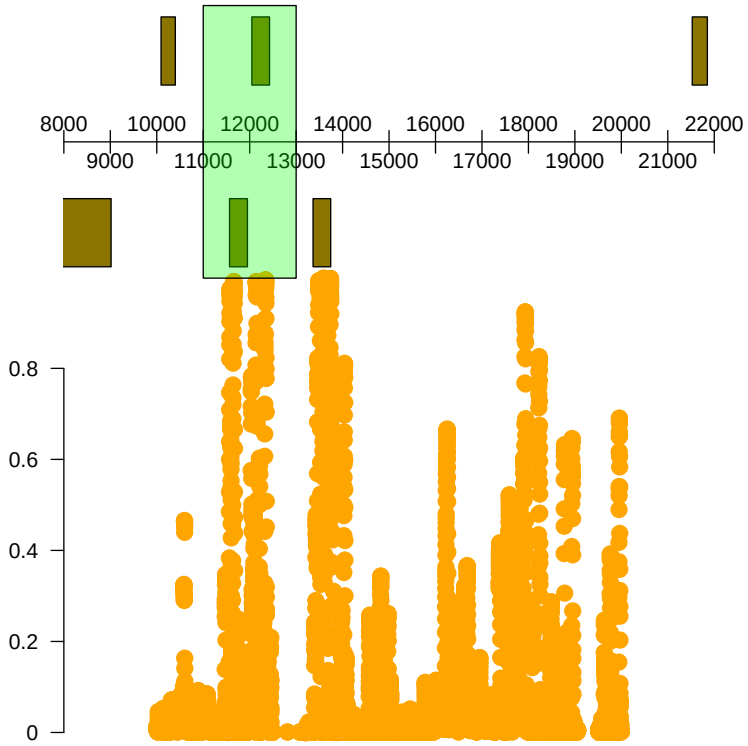
```
> grR <- new("GeneRegion", start = 10000,
+     end = 20000, chromosome = "I", strand = "-",
+     biomart = yeastMart)
> bt <- new("BaseTrack", base = yeastCons1[,
+     1], value = yeastCons1[, 2])
> hr1 <- new("HighlightRegion", start = 11000,
+     end = 13000)
> hr2 <- new("HighlightRegion", start = 15900,
+     end = 16500)
> gdPlot(list(grF, ga, grR, bt), highlightRegions = 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.

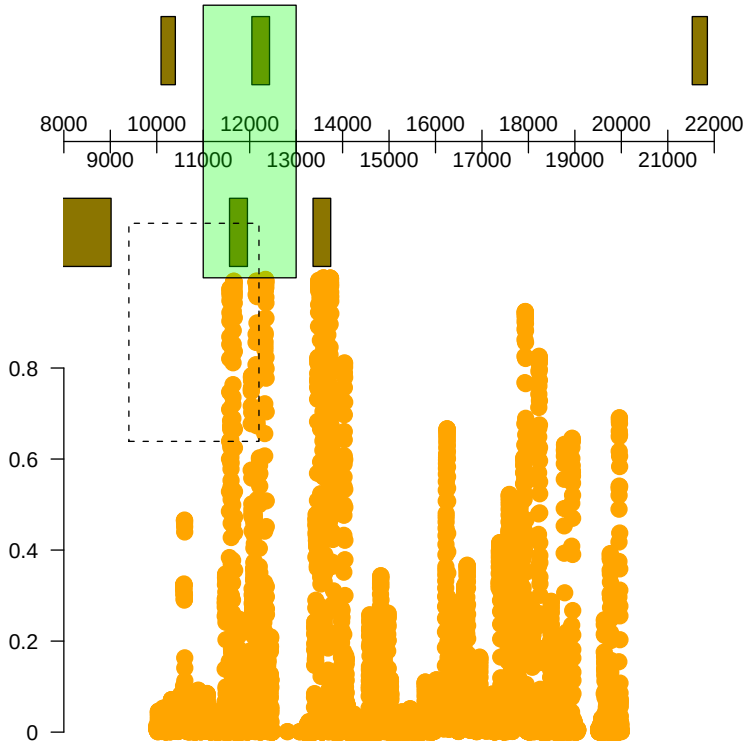
```
> hr <- new("HighlightRegion", start = 11000,
+     end = 13000, region = c(1, 3), dp = DisplayPars(color = "green",
```

```
+         alpha = 0.3))
> gdPlot(list(grF, ga, grR, bt), highlightRegions = hr)
```



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

```
> hrR <- new("HighlightRegion", start = 0.1,
+     end = 0.3, coords = "absolute", dp = DisplayPars(color = "grey",
+         alpha = 0, lty = "dashed"), region = c(0.4,
+         0.7))
> gdPlot(list(grF, ga, grR, bt), highlightRegions = list(hr,
+     hrR))
```



4.2 GenomeGraphs Classes

class	description
gdObject	The root class of the system, never directly instantiated.
Gene	A class representing a Gene.
GeneRegion	A class defining a region of a chromsome – generally a set of genetic elements (genes).
Transcript	A class defining a transcript
TranscriptRegion	A class defining a region of a chromsome – generally a set of genetic elements (transcripts).
Ideogram	An Ideogram
Title	A class to draw a title.
Legend	A class to draw a legend.
GenomeAxis	A class to draw a axis.
Segmentation	A class used to draw horizontal lines in various sets of data.
GenericArray	Used to draw data from microarrays.
ExonArray	Used to draw data from exon microarrays.
GeneModel	A class to draw custom gene models (intron-exon structures)
BaseTrack	Used to draw whatever kind of data at a given base.
MappedRead	A class to plot sequencing reads that are mapped to the genome
DisplayPars	A class which manages various plotting parameters.