

Sushi: An R/Bioconductor package for visualizing genomic data

Douglas H. Phanstiel*, Alan P. Boyle, Carlos L. Araya, and Mike Snyder

February 27, 2014

Contents

1	Introduction	1
2	Data	1
2.1	Data types	1
2.2	Example datasets	2
3	Functions	2
3.1	Functions overview	2
3.2	Non-Sushi Functions	3
3.3	plotBedgraph	3
3.4	plotHic	9
3.5	plotBedpe	10
3.6	plotBed	13
3.7	plotManhattan	16
3.8	Zoom functions	18
3.9	SushiColors	21
3.9.1	Color functions	21
3.9.2	opaque	22
3.9.3	mapcolors	23

1 Introduction

Sushi is an R package for plotting genomic data stored in multiple common genomic formats including bed, bedpe, bedgraph format. The package was designed to be very flexible to allow for combinations of plots into multipanel figures that can include plots made by Sushi, R basecode, or other R packages. Sushi allows for simple flexible plotting of gene structures, transcript structures, sequencing tracks, ChIP-seq peaks, chromatin interactions, GWAS results and

*Corresponding contact: doug.phanstiel@gmail.com

other common genomic data types. This vignette shows some examples of the functions included in Sushi to get you started with plotting these diverse data types.

2 Data

2.1 Data types

Sushi accepts 4 types of genomic data as input. These include:

- bed format: 3-6 columns (chromosome, start, stop, name, score, strand)
- bedpe format: 6-10 columns (chromosome1, start1, stop1, chromosome2, start2, stop2, name, score, strand1, strand2)
- bedgraph format: 4 columns (chromosome, start, stop, score)
- interaction matrix: This is matrix in which row and column names are genomic coordinates and matrix values are some type of interaction score.

** strands are represented as 1 or -1 (instead of the standard "+" and "-").

** Some functions may require additional information depending on the plot and features desired.

2.2 Example datasets

To illustrate how Sushi works, we have included several publicly available data sets in the package Sushi. The data types include RNA-seq, ChIP-seq, ChIA-PET, and HiC data and can be loaded using the following commands:

```
> library('Sushi')
> Sushi_data = data(package = 'Sushi')
> data(list = Sushi_data$results[,3])
```

To see which data sets are loaded

```
> Sushi_data$results[,3]

[1] "Sushi_5C.bedpe"                "Sushi_ChIAPET_pol2.bedpe"
[3] "Sushi_ChIPSeq_CTCF.bedgraph"   "Sushi_ChIPSeq_pol2.bed"
[5] "Sushi_ChIPSeq_pol2.bedgraph"   "Sushi_ChIPSeq_severalfactors.bed"
[7] "Sushi_DNaseI.bedgraph"         "Sushi_GWAS.bed"
[9] "Sushi_HiC.matrix"             "Sushi_RNASeq_K562.bedgraph"
[11] "Sushi_genes.bed"              "Sushi_hg18_genome"
[13] "Sushi_transcripts.bed"
```

3 Functions

3.1 Functions overview

Sushi functions can be broken down into 3 categories: plotting, annotating, zooming, and coloring. Plotting functions generate a basic plot object using the data. Annotating functions add information to the plots such as an x-axis labeling the genomic region or a legend describing the values represented by different colors. Zooming functions allow for highlighting and zooming of genomic regions, which are of particular use for multipanel plots generated with base R functions `mfrow()` or `layout()`. The coloring functions provide simple tools for generating R colors and palettes.

- Plotting functions: `plotBed()`, `plotBedgraph()`, `plotBedpe()`, `plotgenes()`, `plotHiC()`, and `plotManhattan()`
- Annotating functions: `labelgenome()` and `addlegend()`
- Zooming functions: `zoomsregion()` and `zoombox()`
- Coloring functions: `maptocolors()`, `SushiColors()`, and `opaque()`

3.2 Non-Sushi Functions

An important characteristic of Sushi plots is their compatibility with all base R functions and their ability to be combined into complex multipanel figures. Two of the most useful base R functions for creating multipanel figures are `layout()` and `mfrow()`. Basic R plotting functions such as `axis()`, `mtext()`, and `legend()` are also particularly well suited to combine with Sushi plots. A familiarity with these functions will greatly improve your ability to create Sushi plots.

3.3 plotBedgraph

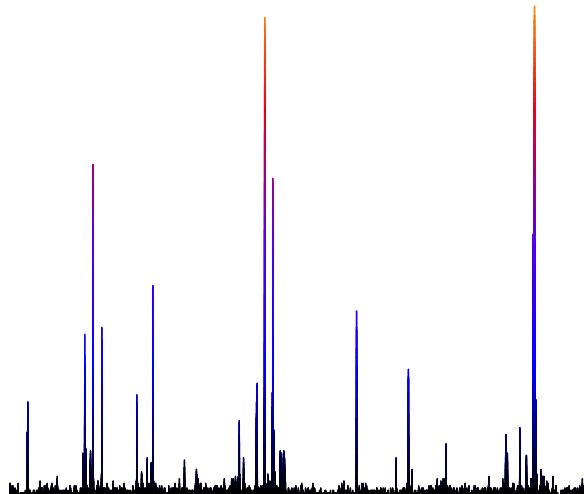
Signal tracks can be plotted using `plotBedgraph()`. The input requires data in bedgraph format. We will demonstrate this using bedgraph data representing a DNaseI hypersensitivity experiment in K562 cells.

```
> head(Sushi_DNaseI.bedgraph)

  chrom start  end value
1 chr11 77224 77244     1
2 chr11 77244 77384     2
3 chr11 96704 96724     1
4 chr11 96724 96844     3
5 chr11 96844 96884     2
6 chr11 97904 97924     3
```

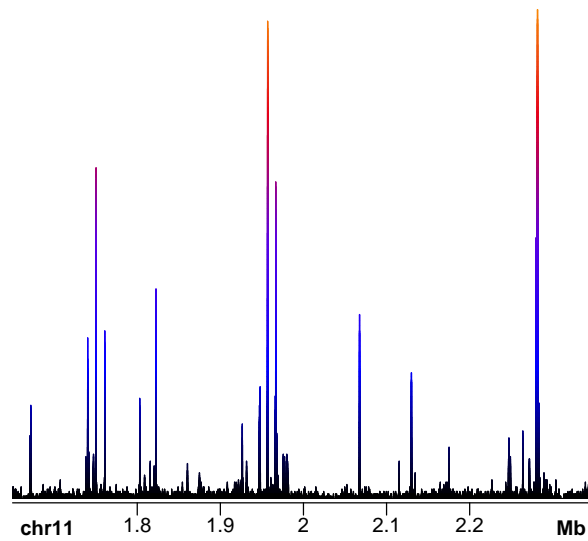
The `plotBedgraph()` function is used to plot the data. As with most Sushi functions the basic required arguments include the data to be plotted, the chromosome, and a start and stop position.

```
> chrom          = "chr11"
> chromstart     = 1650000
> chromend       = 2350000
> plotBedgraph(Sushi_DNaseI.bedgraph,chrom,chromstart,chromend,colorbycol= SushiColors(5))
```



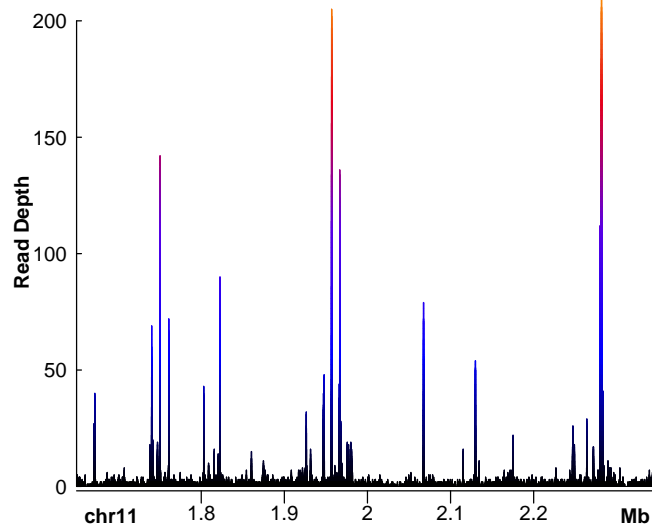
To annotate the genome position we use the `labelgenome()` function. We use `n = 4` to specify the desired number of tickmarks. The scale is set to `Mb` (other options are `Kb` or `bp`).

```
> labelgenome(chrom,chromstart,chromend,n=4,scale="Mb")
```



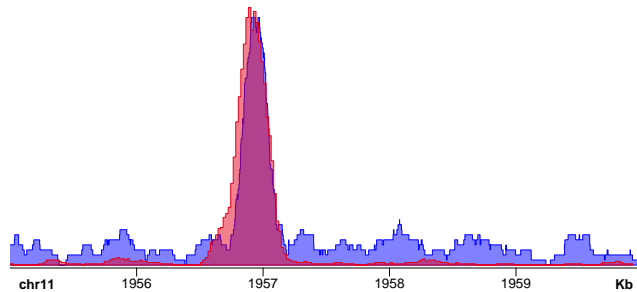
The y-axis can be added using basic R functions `mtext()` and `axis()`.

```
> mtext("Read Depth",side=2,line=1.75,cex=1,font=2)  
> axis(side=2,las=2,tcl=.2)
```



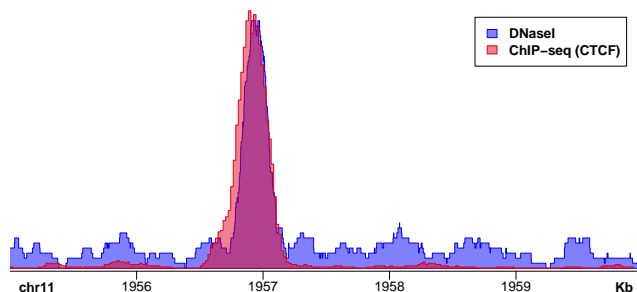
Multiple bedgraph tracks can be plotted on the same plot by setting `overlay=TRUE`. Transparencies can be added for easier viewing by adjusting the transparency value. The second plot can be rescaled to the maximum of the first plot by setting `rescaleoverlay=TRUE`.

```
> chrom          = "chr11"
> chromstart     = 1955000
> chromend       = 1960000
> plotBedgraph(Sushi_ChIPSeq_CTCF.bedgraph,chrom,chromstart,chromend,
               transparency=.50,color=SushiColors(2)(2)[1])
> plotBedgraph(Sushi_DNaseI.bedgraph,chrom,chromstart,chromend,
               transparency=.50,color=SushiColors(2)(2)[2],overlay=TRUE,
               rescaleoverlay=TRUE)
> labelgenome(chrom,chromstart,chromend,n=3,scale="Kb")
```



Then we can use the base R function `legend()` to add a legend to the plot. First we need to use the `rgb` function to add transparency to the colors in order to match our plot.

```
> legend("topright",inset=0.025,legend=c("DNaseI", "ChIP-seq (CTCF)"),
      fill=opaque(SushiColors(2)(2)),border=SushiColors(2)(2),text.font=2,
      cex=1.0)
```



Setting `flip=TRUE` is another method that can be used to compare tracks. First, we will use `mflow` to divide the plotting device into two vertically stacked regions.

```
> par(mfrow=c(2,1),mar=c(1,4,1,1))
```

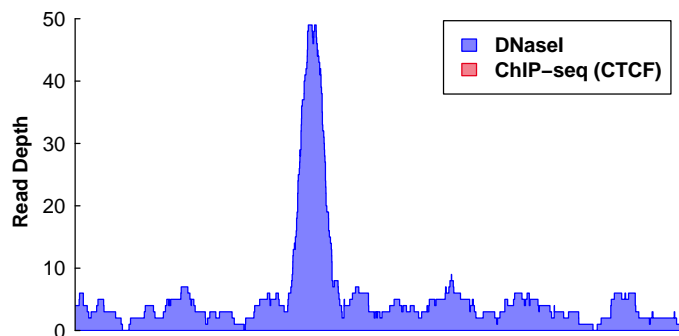
Next, we plot the first plot. We set the transparency of the plot to 0.5. We will also add the legend.

```
> plotBedgraph(Sushi_ChIPSeq_CTCF.bedgraph,chrom,chromstart,chromend,transparency=.50,
      color=SushiColors(2)(2)[1])
```

```

> axis(side=2,las=2,tcl=.2)
> mtext("Read Depth",side=2,line=1.75,cex=1,font=2)
> legend("topright",inset=0.025,legend=c("DNaseI", "ChIP-seq (CTCF)",
      fill=opaque(SushiColors(2)(2)),border=SushiColors(2)(2),text.font=2,
      cex=1.0)

```

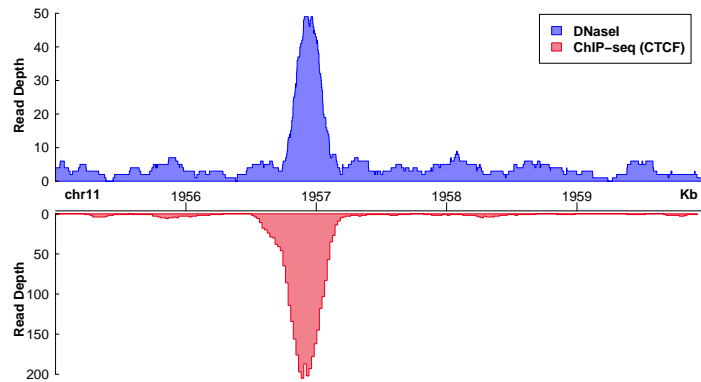


Finally, we add the second plot with `flip=TRUE`. We will also label the x-axis using `labelgenome()` and label the y-axis using `mtext()` and `axis()`.

```

> plotBedgraph(Sushi_DNaseI.bedgraph, chrom, chromstart, chromend,
      transparency=.50, flip=TRUE, color=SushiColors(2)(2)[2])
> labelgenome(chrom,chromstart,chromend,side=3,n=3,scale="Kb")
> axis(side=2,las=2,tcl=.2,at=pretty(par("yaxp")[c(1,2)]),
      labels=-1*pretty(par("yaxp")[c(1,2)]))
> mtext("Read Depth",side=2,line=1.75,cex=1,font=2)

```

3.4 plotHic

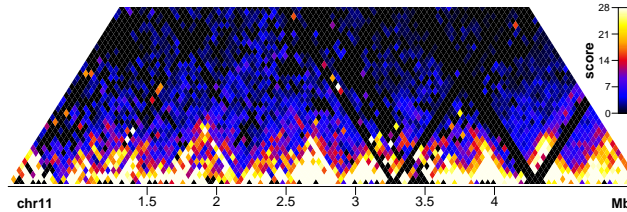
HiC interaction plots can be plotted given an interaction matrix in which row and column names are genomic coordinates and matrix values are some type of interaction score.

```
> Sushi_HiC.matrix[100:105,100:105]

      3980000  4020000  4060000  4100000  4140000  4180000
3980000  0.000000  50.087965  49.689032  22.89760  7.438259  2.219527
4020000  50.087965  40.469337  33.922805  24.07214  12.652542  3.620466
4060000  49.689032  33.922805  26.998026  30.17873  21.879022  6.850893
4100000  22.897599  24.072145  30.178735  54.47335  48.570924  11.379299
4140000  7.438259  12.652542  21.879022  48.57092  45.265394  26.369969
4180000  2.219527  3.620466  6.850893  11.37930  26.369969  11.413106
```

The `plotHic()` function is used to plot the data while the `labelgenome()` function is used to add the genome labels to the x-axis. `plotHic()` returns an object indicating the color palette and data range that can be fed into `addlegend()` to create a legend.

```
> chrom          = "chr11"
> chromstart     = 500000
> chromend       = 5050000
> phic = plotHic(Sushi_HiC.matrix, chrom, chromstart, chromend, max_y = 20,
                zrange=c(0,28), palette=SushiColors(7))
> addlegend(phic[[1]], palette=phic[[2]], title="score", side="right",
            bottominset=0.4, topinset=0, xoffset=-.035, labelside="left",
            width=0.025, title.offset=0.035)
> labelgenome(chrom, chromstart, chromend, n=4, scale="Mb",
            edgeblankfraction=0.20)
```

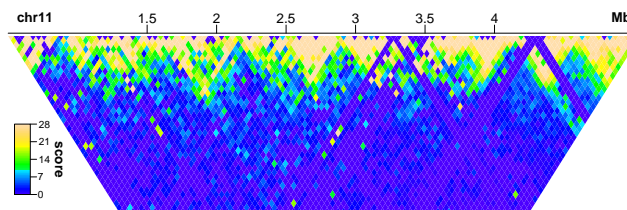


`plotHic()` has a number of customizable options. The plot can be flipped over the x-axis by setting `flip = TRUE`. The color palette can be changed by the `palette` argument.

`addlegend()` also has customizable features. The legend can be moved to the left side of the plot by setting `side = "left"` and the labeling can be moved to the right side of the legend by setting `labelside = "right"`. The vertical position of the legend can be adjusted by changing the `topinset` and `bottominset`.

Finally, the x-axis label can be moved to the top of the plot by setting `side = 3` in the `labelgenome()` function.

```
> chrom          = "chr11"
> chromstart     = 500000
> chromend       = 5050000
> phic = plotHic(Sushi_HiC.matrix,chrom,chromstart,chromend,max_y = 20,
                 zrange=c(0,28),flip=TRUE,palette=topo.colors)
> addlegend(phic[[1]],palette=phic[[2]],title="score",side="left",bottominset=0.1,
            topinset=0.5,xoffset=-.035,labelside="right",width=0.025,title.offset=0.035)
> labelgenome(chrom,chromstart,chromend,side=3,n=4,scale="Mb",edgeblankfraction=0.20)
```



3.5 plotBedpe

`plotBedpe()` allows for data in bedpe format to be plotted in multiple fashions. To illustrate this we will use 5C data formatted in the following way.

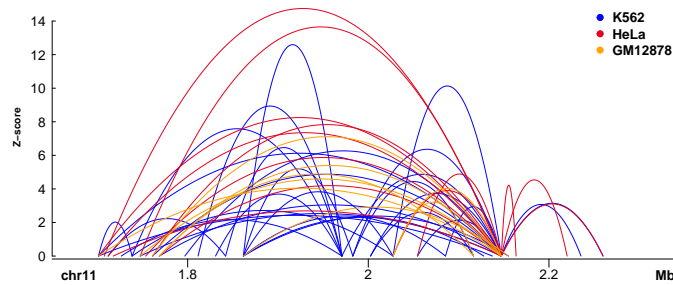
```
> head(Sushi_5C.bedpe)
```

	chrom1	start1	end1	chrom2	start2	end2	name	score	strand1
1	chr2	234208447	234223064	chr2	234156762	234159135	NA	44.39862	.
2	chr15	41711734	41718116	chr15	41802421	41808201	NA	20.62534	.
3	chr11	64172456	64183193	chr11	64068878	64079209	NA	16.91630	.
4	chr2	234208447	234223064	chr2	234163674	234170252	NA	12.34501	.
5	chr6	41755186	41769245	chr6	41435903	41452283	NA	11.63480	.
6	chr11	64159283	64172456	chr11	64068878	64079209	NA	11.13098	.

	strand2	samplenumber
1	.	1
2	.	1
3	.	1
4	.	1
5	.	1
6	.	1

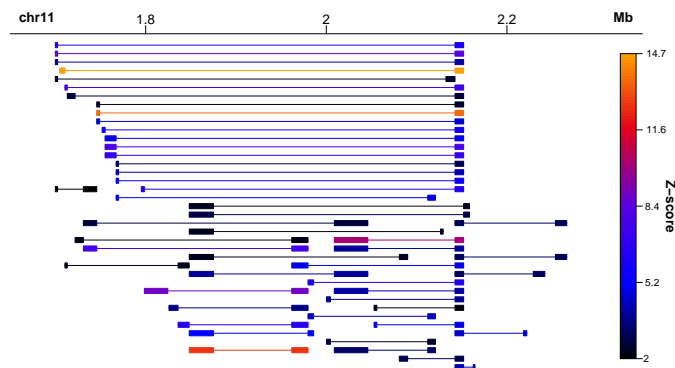
plotBedpe() can plot bedpe as arches. The height, linewidth, and color of each arch can be scaled to represent different aspects of the data. Here the height of the arches represents the Z-score of the 5C interaction, the color represents the cell line each interaction was detected in, and the line widths are kept constant (default lwd = 1).

```
> chrom = "chr11"
> chromstart = 1650000
> chromend = 2350000
> pbpe = plotBedpe(Sushi_5C.bedpe, chrom, chromstart, chromend,
  heights = Sushi_5C.bedpe$score, plotype="loops",
  colorby=Sushi_5C.bedpe$samplenumber,
  colorbycol=SushiColors(3))
> labelgenome(chrom, chromstart, chromend, n=3, scale="Mb")
> legend("topright", inset = 0.01, legend=c("K562", "HeLa", "GM12878"),
  col=SushiColors(3)(3), pch=19, bty='n', text.font=2)
> axis(side=2, las=2, tcl=.2)
> mtext("Z-score", side=2, line=1.75, cex=.75, font=2)
```



The plot can be flipped over the x-axis by setting `flip = TRUE`, Bedpe elements can be represented by boxes and straight lines by setting `plottype = "lines"`. And colors can be used to represent Z-scores by setting `colorby = "Sushi_5C.bedpe$score"`.

```
> chrom          = "chr11"
> chromstart     = 1650000
> chromend       = 2350000
> pbpe = plotBedpe(Sushi_5C.bedpe, chrom, chromstart, chromend, flip=TRUE,
  plottype="lines", colorby=Sushi_5C.bedpe$score,
  colorbycol=SushiColors(5))
> labelgenome(chrom, chromstart, chromend, side=3, n=3, scale="Mb")
> addlegend(pbpe[[1]], palette=pbpe[[2]], title="Z-score", side="right", bottominset=0.05,
  topinset=0.05, xoffset=-.035, labelside="right", width=0.025, title.offset=0.045)
```



3.6 plotBed

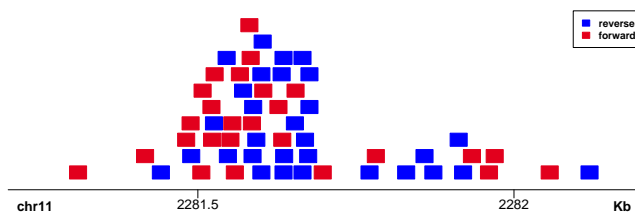
plotBed provides multiple different ways to represent genomic data stored in bed format. Below are the first six lines of a bed file detailing reads from Pol2 ChIP-Seq analysis of K562 cells.

```
> head(Sushi_ChIPSeq_pol2.bed)

  chrom  start    end          name score strand
1 chr11 2280543 2280570 GGGCTCTCTCCGGCTTCCCTGTCCCGT    63    -1
2 chr11 2288946 2288973 CCTTCCCATCCGCAGGGGCACCACATG   1000    -1
3 chr11 2272471 2272498 TGGGCATCAGTCAGGCTCCTTCCCCAG   1000    -1
4 chr11 2288939 2288966 ATCCGCAGGGGCACCACATGAGTCACC   1000    -1
5 chr11 2281534 2281561 TGTCTAGTGACAAGTGGCCGGAATTG    250    -1
6 chr11 2286805 2286832 GGTGAGGGCCAGCAGCTCCCTGGGGGG    250     1
```

Leaving row set to auto provides a pile-sup style plot. Here the colorby argument is used to color the bed elements by the strand.

```
> chrom          = "chr11"
> chromstart     = 2281200
> chromend       = 2282200
> plotBed(beddata = Sushi_ChIPSeq_pol2.bed, chrom = chrom, chromstart = chromstart,
          chromend = chromend, colorby = Sushi_ChIPSeq_pol2.bed$strand,
          colorbycol = SushiColors(2), row = "auto", wiggle=0.001)
> labelgenome(chrom, chromstart, chromend, n=2, scale="Kb")
> legend("topright", inset=0, legend=c("reverse", "forward"), fill=SushiColors(2)(2),
          border=SushiColors(2)(2), text.font=2, cex=0.75)
```



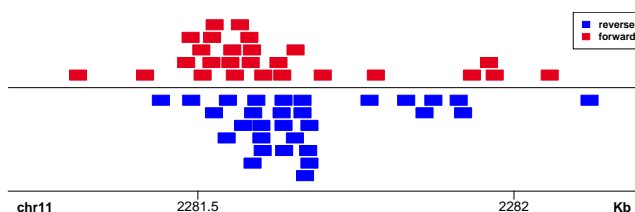
Setting splitstrand = TRUE plots reads from different strands in two separate vertical regions.

```
> chrom          = "chr11"
> chromstart     = 2281200
> chromend       = 2282200
> plotBed(beddata = Sushi_ChIPSeq_pol2.bed, chrom = chrom, chromstart = chromstart,
```

```

chromend = chromend, colorby = Sushi_ChIPSeq_pol2.bed$strand,
colorbycol = SushiColors(2), row = "auto", wiggle=0.001, splitstrand=TRUE)
> labelgenome(chrom, chromstart, chromend, n=2, scale="Kb")
> legend("topright", inset=0, legend=c("reverse", "forward"), fill=SushiColors(2)(2),
border=SushiColors(2)(2), text.font=2, cex=0.75)

```



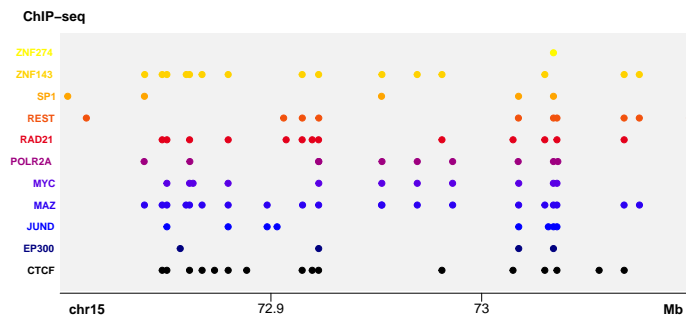
plotBed can also plot bed elements on different rows as specified by the user. First, we will use the Sushi function `maptocolors()` to assign a different color to each row.

```

> Sushi_ChIPSeq_severalfactors.bed$color =
  mapcolors(Sushi_ChIPSeq_severalfactors.bed$row,
col=SushiColors(6))

```

By providing row and color information `plotBed()` can be used to compare bed elements from different samples by plotting them on different rows.



```

> chrom = "chr15"
> chromstart = 72800000
> chromend = 73100000
> plotBed(beddata = Sushi_ChIPSeq_severalfactors.bed, chrom = chrom,
chromstart = chromstart, chromend = chromend,

```

```

rownumber = Sushi_ChIPSeq_severalfactors.bed$row, type = "circles",
color=Sushi_ChIPSeq_severalfactors.bed$color,row="given",
plotbg="grey95",rowlabels=unique(Sushi_ChIPSeq_severalfactors.bed$name),
rowlabelcol=unique(Sushi_ChIPSeq_severalfactors.bed$color),rowlabelcex=0.75)
> labelgenome(chrom,chromstart,chromend,n=3,scale="Mb")
> mtext("ChIP-seq",side=3, adj=-0.065,line=0.5,font=2)

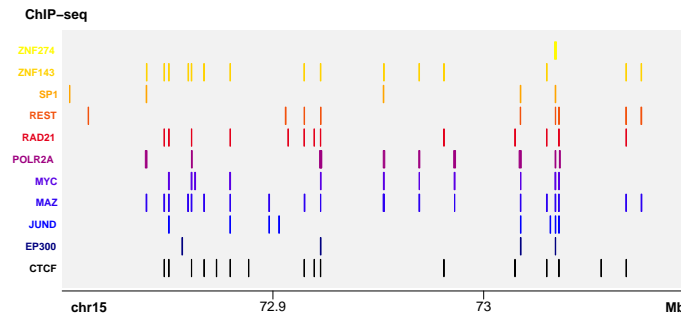
```

That same data can be represented by rectangles that depict the actual width of each bed element.

```

> plotBed(beddata = Sushi_ChIPSeq_severalfactors.bed,chrom = chrom,
chromstart = chromstart,chromend =chromend,
rownumber = Sushi_ChIPSeq_severalfactors.bed$row, type = "region",
color=Sushi_ChIPSeq_severalfactors.bed$color,row="given",
plotbg="grey95",rowlabels=unique(Sushi_ChIPSeq_severalfactors.bed$name),
rowlabelcol=unique(Sushi_ChIPSeq_severalfactors.bed$color),rowlabelcex=0.75)
> labelgenome(chrom,chromstart,chromend,n=3,scale="Mb")
> mtext("ChIP-seq",side=3, adj=-0.065,line=0.5,font=2)

```



plotBed() can also be used to plot heatmaps representing the density of bed elements. First, we will use the biomaRt function getBM() to get the gene information we require.

```

> chrom = "chr15"
> chromstart = 60000000
> chromend = 80000000
> chrom_biomaRt = gsub("chr","",chrom)
> mart=useMart(host='may2009.archive.ensembl.org', biomaRt='ENSEMBL_MART_ENSEMBL',
dataset='hsapiens_gene_ensembl')
> geneinfobed = getBM(attributes = c("chromosome_name","start_position","end_position"),
filters= c("chromosome_name","start","end"),
values=list(chrom_biomaRt,chromstart,chromend),mart=mart)
> geneinfobed[,1] = paste("chr",geneinfobed[,1],sep="")

```

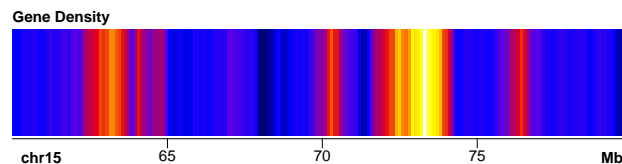
The data is in simple bed format with just three columns representing chromosome, start, and stop.

```
> head (geneinfobed)
```

	chromosome_name	start_position	end_position
1	chr15	73372069	73372334
2	chr15	64580642	64580710
3	chr15	63375442	63375557
4	chr15	72570353	72570422
5	chr15	60903209	60903293
6	chr15	70130646	70130724

Now we can make a gene density plot using the plotBed function.

```
> plotBed(beddata = geneinfobed[!duplicated(geneinfobed),],chrom = chrom,  
          chromstart = chromstart,chromend =chromend,row='supplied',  
          palettes = list(SushiColors(7)), type = "density")  
> labelgenome(chrom, chromstart, chromend, n=4,scale="Mb",edgeblankfraction=0.10)  
> mtext("Gene Density",side=3, adj=0,line=0.20,font=2)
```



3.7 plotManhattan

plotManhattan() differs from most other Sushi functions in that it can plot multiple chromosomes in a single plot. Because of this plotManhattan requires some additional inputs. It requires an object in bed format describing the location of data points as well as vector of p-values (typically one of the columns of the bed file). But it also requires an genome object that describes which chromosomes to plot and their sizes (in bp). The genome object is very similar to the genome files used for bedtools.

The bed data should look something like this:

```
> head(Sushi_GWAS.bed)
```


	chr.hg18	pos.hg18	pos.hg18.1	rsid	pval.GC.DBP	V6
1	chr1	1695996	1695996	rs6603811	0.003110	.
2	chr1	1696020	1696020	rs7531583	0.000824	.
3	chr1	1698661	1698661	rs12044597	0.001280	.
4	chr1	1711339	1711339	rs2272908	0.001510	.
5	chr1	1712792	1712792	rs3737628	0.001490	.
6	chr1	1736016	1736016	rs12408690	0.004000	.

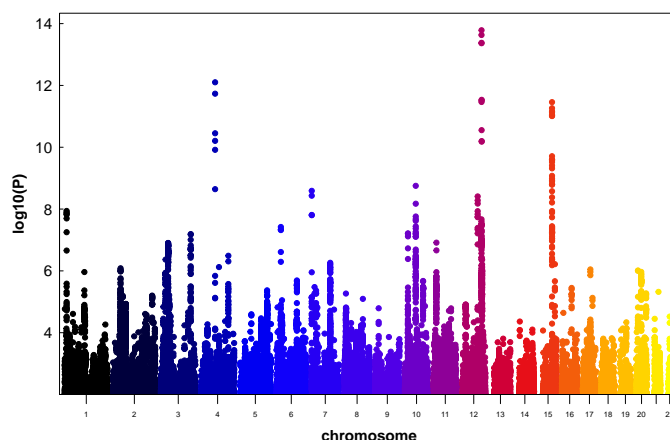
And the genome file should look like this:

```
> head(Sushi_hg18_genome)
```

	V1	V2
1	chr1	247249719
2	chr10	135374737
3	chr11	134452384
4	chr12	132349534
5	chr13	114142980
6	chr14	106368585

The `plotManhattan()` function is used to plot the data while the `labelgenome()` function is used to add the genome labels to the x-axis. The `labelgenome()` function also requires a genome object.

```
> plotManhattan(bedfile=Sushi_GWAS.bed,pvalues=Sushi_GWAS.bed[,5],col=SushiColors(6),
               genome=Sushi_hg18_genome,cex=0.75)
> labelgenome(genome=Sushi_hg18_genome,n=4,scale="Mb",edgeblankfraction=0.20,cex.axis=.5)
> axis(side=2,las=2,tcl=.2)
> mtext("log10(P)",side=2,line=1.75,cex=1,font=2)
> mtext("chromosome",side=1,line=1.75,cex=1,font=2)
```



3.8 Zoom functions

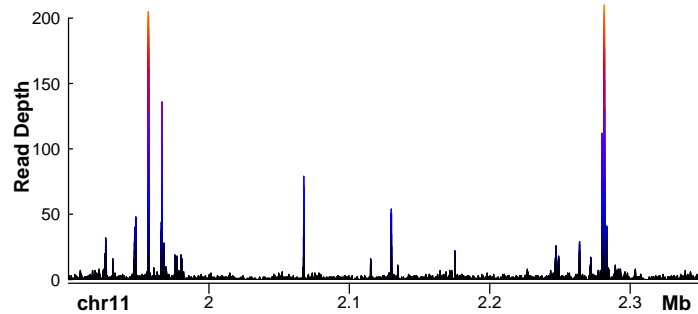
A critical characteristic of the Sushi package is its ability to create highly customizable, publication-ready, multi-panel figures. Here, we will create a basic three panel figure and demonstrate how the zoom functions work (`zoomsregion` and `zoombox`). To illustrate these feature we will use the `plotBedgraph()` function to plot bedgraph data representing a DNaseI hypersensitivity experiment in K562 cells.

In order to make a multipanel figure we will use the R function `layout`. `Layout` divides the device into rows and columns according to a matrix you provide. The matrix also tells it which plots will appear on which parts of the plotting device. Below we make a 2 by 2 matrix. The entire top row will be used to plot the first plot while the bottom row will contain two plots. For more info on `layout` try `?layout`.

```
> layout(matrix(c(1,1,2,3),2, 2, byrow = TRUE))
> par(mar=c(3,4,1,1))
```

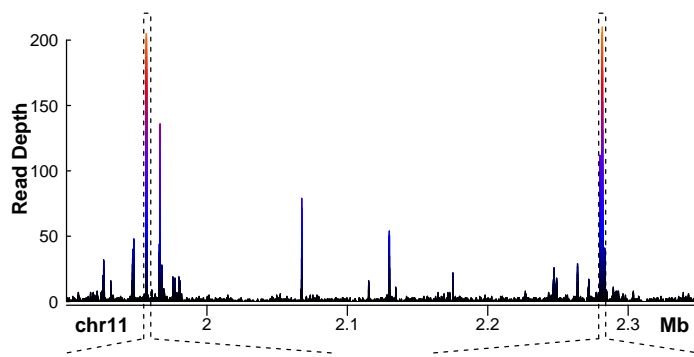
Next we will add the first plot

```
> chrom          = "chr11"
> chromstart     = 1900000
> chromend       = 2350000
> plotBedgraph(Sushi_DNaseI.bedgraph,chrom,chromstart=chromstart,chromend=chromend,colorbyco
> labelgenome(chrom,chromstart=chromstart,chromend=chromend,n=4,scale="Mb")
> mtext("Read Depth",side=2,line=1.75,cex=1,font=2)
> axis(side=2,las=2,tcl=.2)
```



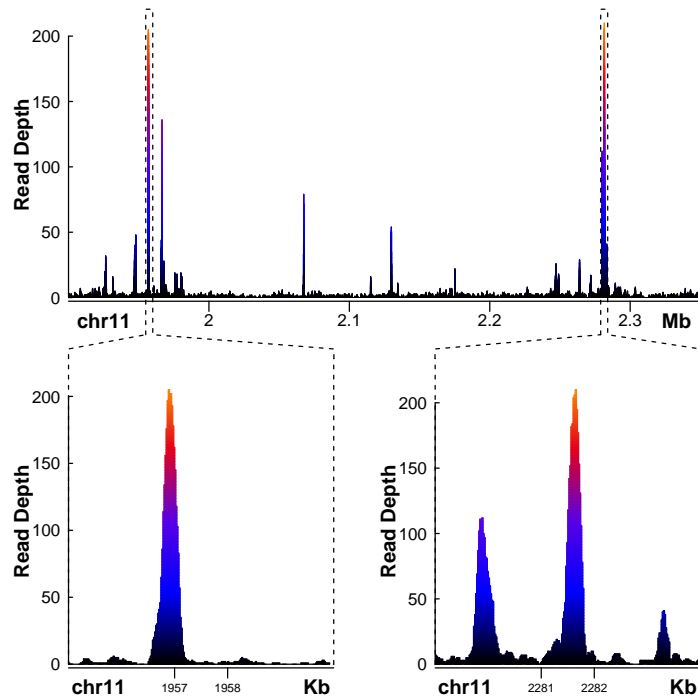
Next we will add the zoom regions using the function `zoomsregion()`. The argument `offsets` is used to precisely position the left and right edges of the widest part of the zoom.

```
> zoomregion1      = c(1955000,1960000)
> zoomregion2      = c(2279000,2284000)
> zoomsregion(zoomregion1,extend=c(0.01,0.13),wideextend=0.05,offsets=c(0,0.580))
> zoomsregion(zoomregion2,extend=c(0.01,0.13),wideextend=0.05,offsets=c(0.580,0))
```



Then we can add each of the zoomed inset regions. For, each region we need execute the `zoombox` function in order to draw the lines around the new plots.

```
> plotBedgraph(Sushi_DNaseI.bedgraph,chrom,chromstart=zoomregion1[1],
               chromend=zoomregion1[2],colorbycol= SushiColors(5))
> labelgenome(chrom,chromstart=zoomregion1[1],chromend=zoomregion1[2],
               n=4,scale="Kb",edgeblankfraction=0.2,cex.axis=.75)
> zoombox()
> mtext("Read Depth",side=2,line=1.75,cex=1,font=2)
> axis(side=2,las=2,tcl=.2)
> plotBedgraph(Sushi_DNaseI.bedgraph,chrom,chromstart=zoomregion2[1],
               chromend=zoomregion2[2],colorbycol= SushiColors(5))
> labelgenome(chrom,chromstart=zoomregion2[1],chromend=zoomregion2[2],
               n=4,scale="Kb",edgeblankfraction=0.2,cex.axis=.75)
> zoombox()
> mtext("Read Depth",side=2,line=1.75,cex=1,font=2)
> axis(side=2,las=2,tcl=.2)
```



3.9 SushiColors

Sushi includes three functions to assist in the generating of R colors and color palettes: `SushiColors()`, `maptocolors()`, `opaque()`.

3.9.1 Color functions

`SushiColors()` provides default color palettes for the Sushi package.

To see a list of available color palettes:

```
> SushiColors(palette='list')
```

```
[1] 2 3 4 5 6 7
```

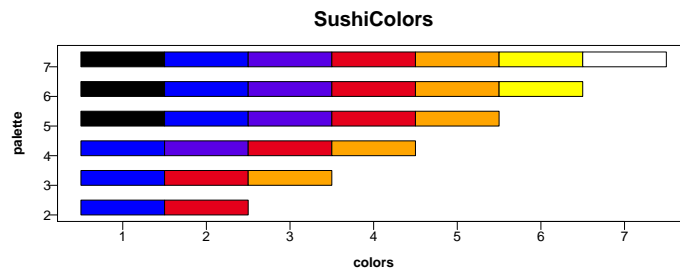
To view the color palettes:

```
> plot(1,xlab='',xaxt='n',ylab='',yaxt='n',xlim=c(0.5,7.5),
      ylim=c(2,7.5),type='n')
> for (i in (2:7))
{
  for (j in (1:i))
  {
```

```

      rect(j-.5,i,j+.5,i+.5,col=SushiColors(i)(i)[j])
    }
  }
> axis(side=2,at=(2:7),labels=(2:7),las=2)
> axis(side=1,at=(1:7),labels=(1:7))
> mtext("SushiColors",side=3,font=2, line=1, cex=1.5)
> mtext("colors",side=1,font=2, line=2)
> mtext("palette",side=2,font=2, line=2)

```



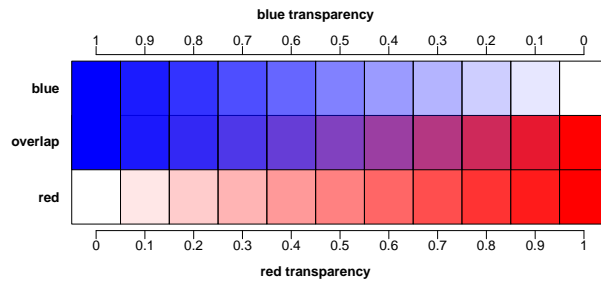
3.9.2 opaque

`opaque()` takes any color or vector of colors and makes them opaque. The degree of transparency is determined by the argument `transparency` which is a value between 0 and 1.

```

> plot(1,xlab='',xaxt='n',ylab='',yaxt='n',bty='n',type='n',
      xlim=c(-.15,1.05),ylim=c(-1,2))
> for (i in seq(0,1,by=0.1))
  {
    rect(i-.05,-1,i+.05,1,col=opaque("red",transparency=i))
    rect(i-.05,0,i+.05,2,col=opaque("blue",transparency=1-i))
  }
> axis(side=1,at=seq(0,1,by=0.1),labels=seq(0,1,by=0.1))
> mtext("red transparency",side=1,font=2, line=2)
> axis(side=3,at=seq(0,1,by=0.1),labels=seq(1,0,by=-0.1))
> mtext("blue transparency",side=3,font=2, line=2)
> text(-0.075,1.5,labels="blue",font=2,adj=1)
> text(-0.075,0.5,labels="overlap",font=2,adj=1)
> text(-0.075,-.5,labels="red",font=2,adj=1)

```



3.9.3 maptocolors

`maptocolors()` takes a vector of values and maps them to a color palette which can be used for plotting.

```
> set.seed(3)
> values = rnorm((1:10))
> colorpalette = SushiColors(5)
> plot(x=(1:10),y=values,col=maptocolors(values,colorpalette),
      pch=19,cex=4,xlab="data points",yaxt='n',ylim=range(values)*1.2)
> addlegend(range(values),title="key",palette=colorpalette,
      side='left',xoffset = -0.125,width=0.03,bottominset = 0.5, topinset = 0.025)
> axis(side=2,las=2)
```

