

Sushi: An R/Bioconductor package for visualizing genomic data

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

March 31, 2014

Contents

1	Introduction	2
2	Data	2
2.1	Data types	2
2.2	Example datasets	2
3	Functions	3
3.1	Functions overview	3
3.2	Non-Sushi Functions	4
3.3	plotBedgraph	4
3.4	plotHic	10
3.5	plotBedpe	12
3.6	plotBed	14
3.7	plotManhattan	18
3.8	plotGenes	19
3.9	Zoom functions	22
3.10	Color functions	25
3.10.1	SushiColors	25
3.10.2	opaque	26
3.10.3	maptocolors	27
3.11	labeling functions	28
3.11.1	labelgenome	28
3.11.2	labelplot	30
4	Appendix	31
5	Bibliography	42

*Corresponding contact: doug.phanstiel@gmail.com

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 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 can be represented as 1 or -1 or "+" 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:

Sushi_5C.bedpe	Sanyal et al. [7]
Sushi_ChIAPET_pol2.bedpe	Li et al. [4]
Sushi_ChIPExo_CTCF.bedgraph	Rhee and Pugh [6]
Sushi_ChIPSeq_CTCF.bedgraph	The ENCODE Project Consortium [8]
Sushi_ChIPSeq_pol2.bed	The ENCODE Project Consortium [8]
Sushi_ChIPSeq_pol2.bedgraph	The ENCODE Project Consortium [8]
Sushi_ChIPSeq_severalfactors.bed	The ENCODE Project Consortium [8]
Sushi_DNaseI.bedgraph	Neph et al. [5]
Sushi_GWAS.bed	International Consortium for Blood Pressure [3]
Sushi_HiC.matrix	Dixon et al. [2]
Sushi_RNASeq_K562.bedgraph	The ENCODE Project Consortium [8]
Sushi_genes.bed	Biomart [1]
Sushi_hg18_genome	Biomart [1]
Sushi_transcripts.bed	The ENCODE Project Consortium [8]

These data sets 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_ChIPExo_CTCF.bedgraph"          "Sushi_ChIPSeq_CTCF.bedgraph"
[5] "Sushi_ChIPSeq_pol2.bed"                "Sushi_ChIPSeq_pol2.bedgraph"
[7] "Sushi_ChIPSeq_severalfactors.bed"      "Sushi_DNaseI.bedgraph"
[9] "Sushi_GWAS.bed"                       "Sushi_HiC.matrix"
[11] "Sushi_RNASeq_K562.bedgraph"           "Sushi_genes.bed"
[13] "Sushi_hg18_genome"                   "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()`, `mttext()`, 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.

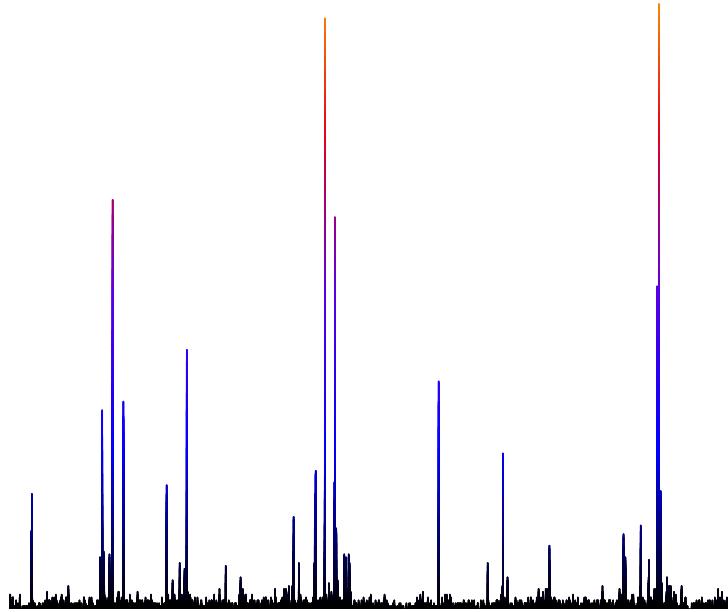
3.3 plotBedgraph

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
```

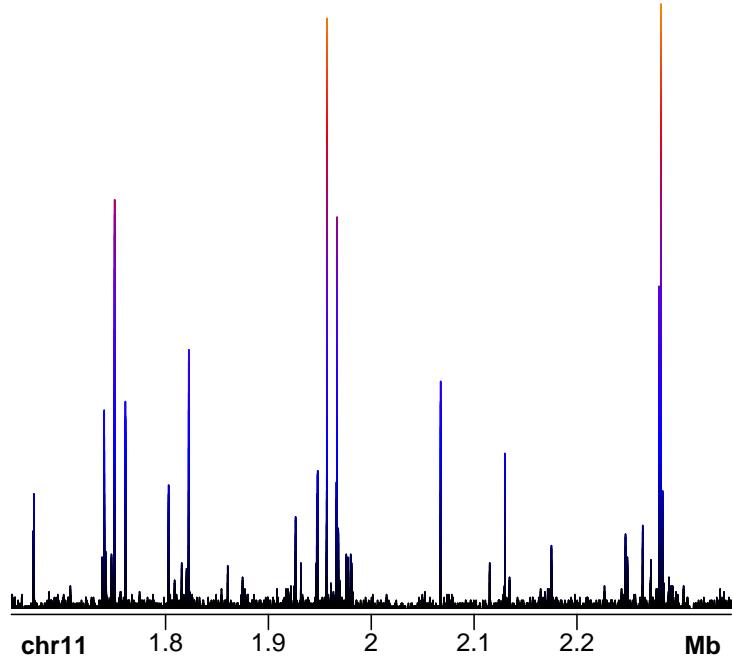
The `plotBedgraph()` function is used to plot the data. As with most Sushi functions the basic required arguments include the data to be plotted, the chromosome, and a start and stop position.

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



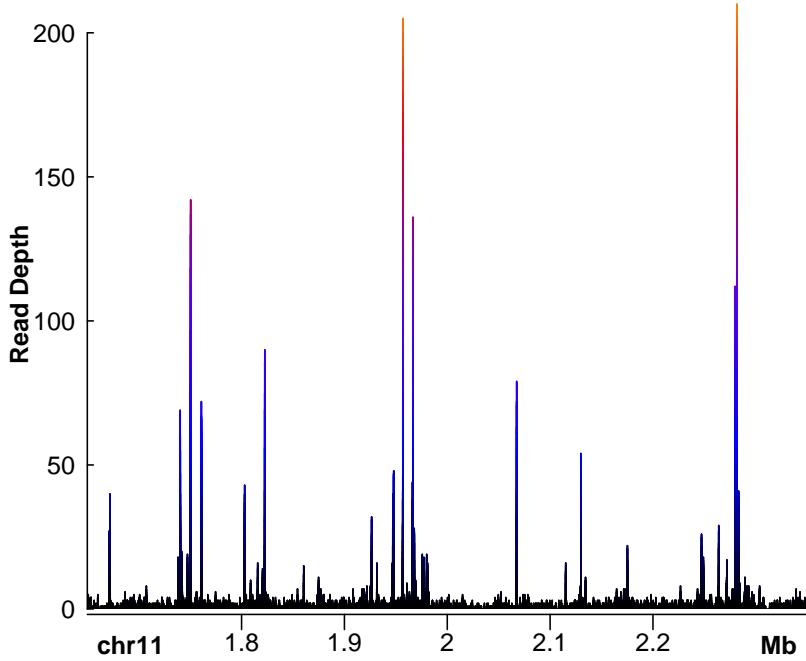
To annotate the genome position we use the `labelgenome()` function. We us `n = 4` to specify the desired number of tickmarks. The scale is set to Mb (other options are Kb or bp).

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



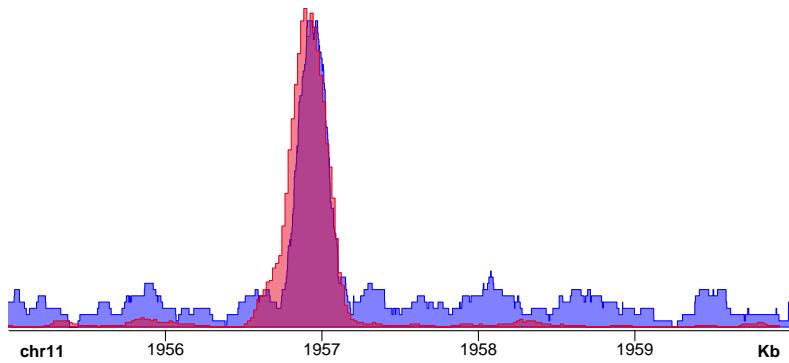
The y-axis can be added using basic R functions `mtext()` and `axis()`.

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



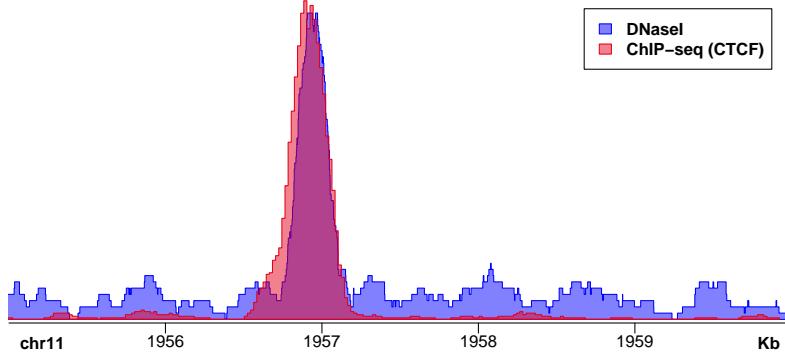
Multiple bedgraph tracks can be plotted on the same plot by setting `overlay=TRUE`. Transparency can be added for easier viewing by adjusting the transparency value. The second plot can be rescaled to the maximum of the first plot by setting `rescaleoverlay=TRUE`.

```
> 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 out 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)
```

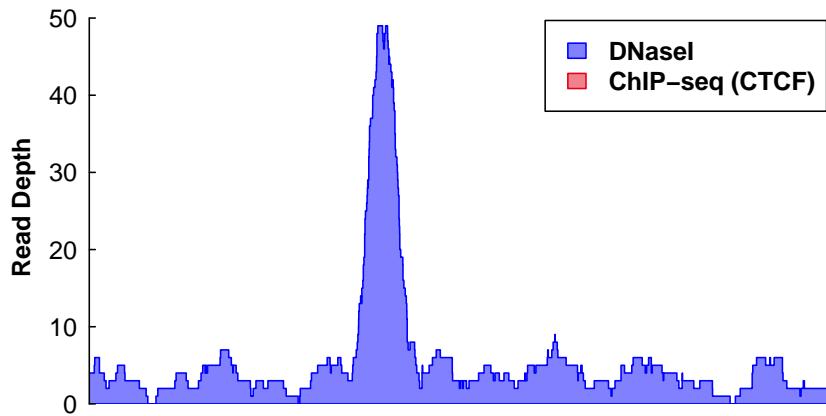


Setting `flip=TRUE` is another method that can be used to compare tracks. First, we will use `mfrow` to divided the plotting device into two vertically stacked regions.

```
> 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)
```



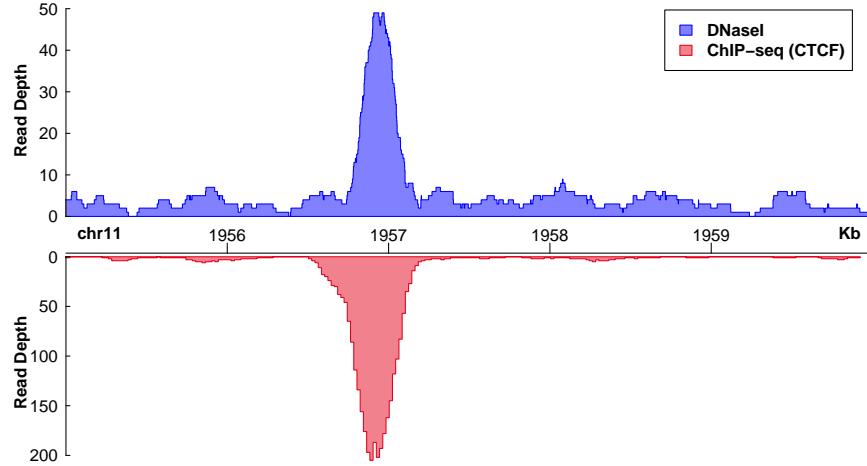
Finally, we add the second plot with `flip=TRUE`. We will also label the x-axis using `labelgenome()` and label the y-axis using `mtext()` and `axis()`.

```
> 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.

```

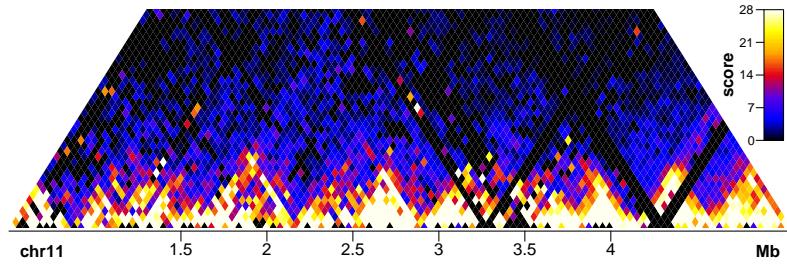
> chrom          = "chr11"
> chromstart     = 500000
> chromend       = 5050000
> phic = plotHic(Sushi_HiC.matrix, chrom,chromstart, chromend, max_y = 20,
                  zrange=c(0,28), palette=SushiColors(7))

```

```

> addlegend(phiic[[1]], palette=phiic[[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.

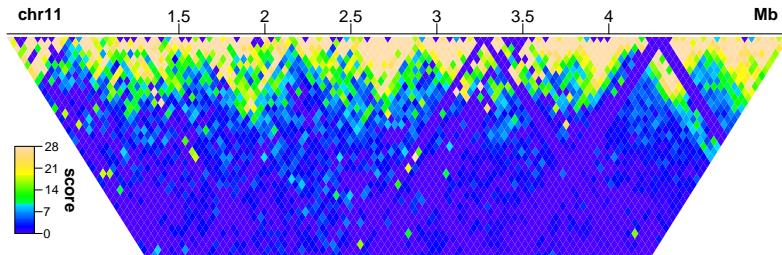
`addlegend()` also has customizable features. The legend can be moved to the left side of the plot by setting `side = "left"` and the labeling can be moved to the right side of the legend by setting `labelside = "right"`. The vertical position of the legend can be adjusted by changing the `topinset` 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
> phiic = plotHic(Sushi_HiC.matrix,chrom,chromstart,chromend,max_y = 20,
                    zrange=c(0,28),flip=TRUE,palette=topo.colors)
> addlegend(phiic[[1]],palette=phiic[[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          .
2          .
3          .
4          .
5          .
6          .
```

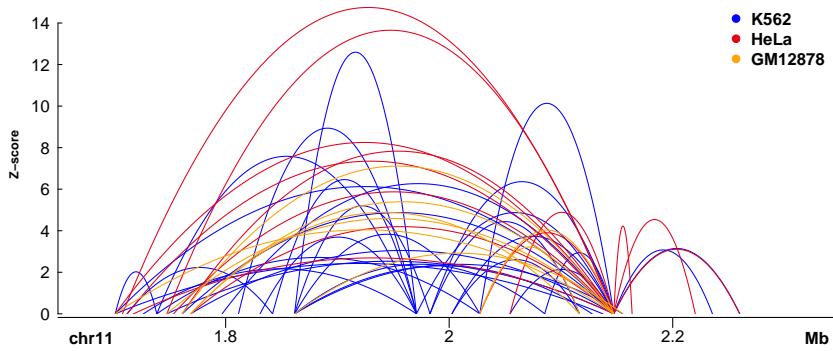
`plotBedpe()` can plot bedpe as arches. The height, linewidth, and color of each arch can be scaled to represent different aspects of the data. Here the height of the arches represents the Z-score of the 5C interaction, the color represents the cell line each interaction was detected in, and the line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

```

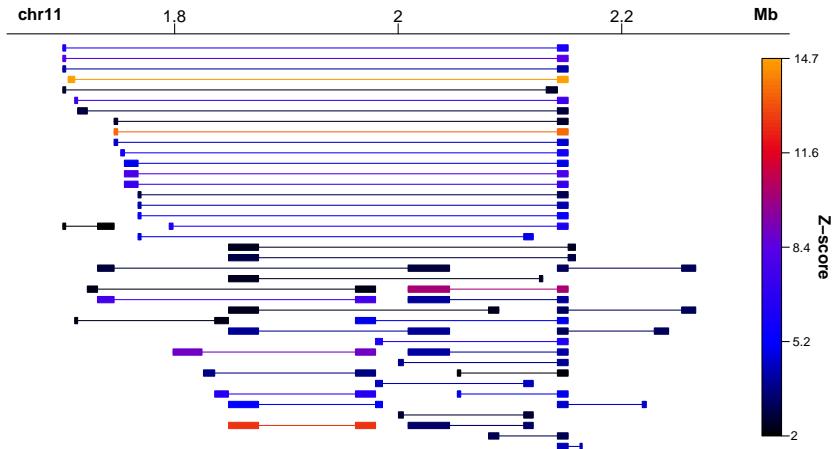


The plot can be flipped over the x-axis by setting `flip = TRUE`, Bedpe elements can be represented by boxes and straight lines by setting `plottype = "lines"`. And colors can be used to represent Z-scores by setting `colorby = "Sushi_5C.bedpe$score"`.

```

> 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)

```



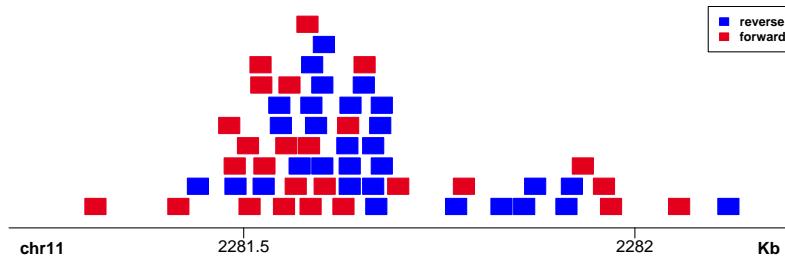
3.6 plotBed

`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 GGGCTCTCTCCGGCTTCCCTGTCCCCGT 63   -1
2 chr11 2288946 2288973 CCTTCCCATCCGCAGGGCACCACATG 1000  -1
3 chr11 2272471 2272498 TGGGCATCAGTCAGGCTCCTCCCCAG 1000  -1
4 chr11 2288939 2288966 ATCCGCAGGGGCACCACATGAGTCACC 1000  -1
5 chr11 2281534 2281561 TGTCCCTAGTGACAAGTGGCCGGACTTG 250   -1
6 chr11 2286805 2286832 GGTGAGGCCAGCAGCTCCCTGGGGGG 250   1
```

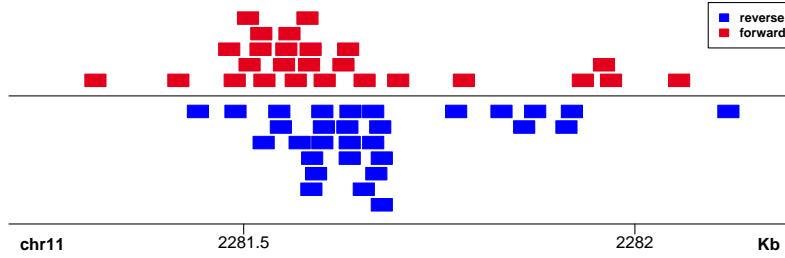
Leaving `row` set to `auto` provides a pile-sup style plot. Here the `colorby` argument is used to color the bed elements by the strand.

```
> 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.

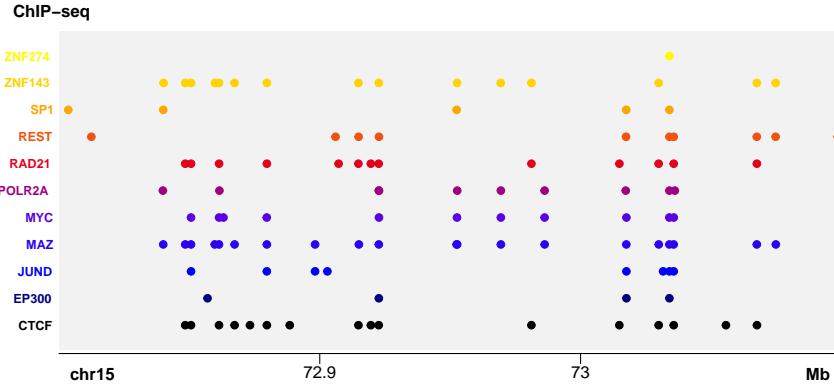
```
> chrom          = "chr11"
> chromstart    = 2281200
> chromend      = 2282200
> plotBed(beddata = Sushi_ChIPSeq_pol2.bed, chrom = chrom, chromstart = chromstart,
           chromend = chromend, colorby = Sushi_ChIPSeq_pol2.bed$strand,
           colorbycol = SushiColors(2), row = "auto", wiggle=0.001, 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$color =
  maptocolors(Sushi_ChIPSeq_severalfactors.bed$row,
  col=SushiColors(6))
```

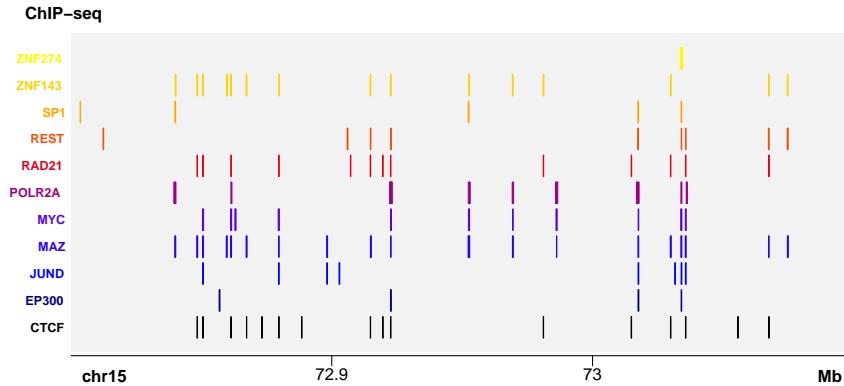
By providing row and color information `plotBed()` can be used to compare bed elements from different samples by plotting them on different rows.



```
> 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)

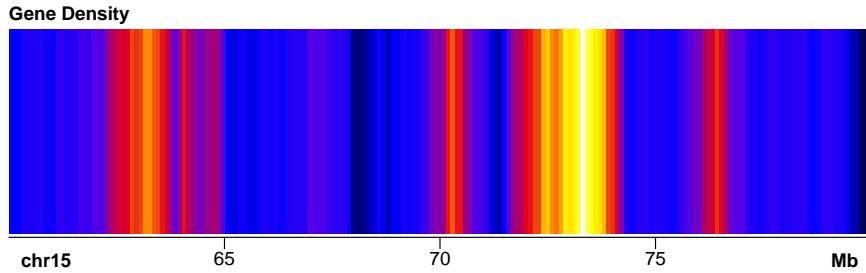
  chromosome_name start_position end_position
1         chr15        73372069     73372334
2         chr15        64580642     64580710
3         chr15        63375442     63375557
4         chr15        72570353     72570422
5         chr15        60903209     60903293
6         chr15        70130646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

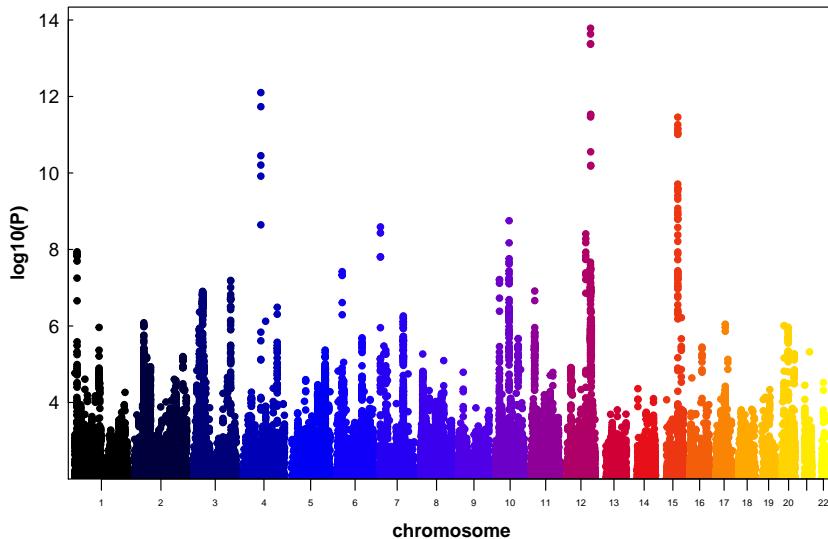
```

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 plotGenes

`plotGenes()` can be used to plot gene structures that are stored in bed format.

```

> head(Sushi_genes.bed)

  chrom      start      stop  gene score strand
1 chr15 73017309 73017438 COX5A      .     -1

```

```

2 chr15 72999672 72999836 COX5A      .      -1
3 chr15 73003042 73003164 COX5A      .      -1
4 chr15 73006160 73006281 COX5A      .      -1
5 chr15 73008510 73008626 COX5A      .      -1

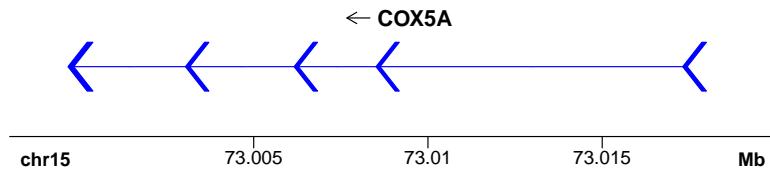
```

Using `plotGenes()` with arguments `bentline=FALSE` and `plotgenetype="arrow"` produces arrow and line gene structures.

```

> chrom          = "chr15"
> chromstart     = 72998000
> chromend       = 73020000
> pg = plotGenes(Sushi_genes.bed,chrom,chromstart,chromend ,
                  types=Sushi_genes.bed$type,maxrows=1,bheight=0.2,
                  plotgenetype="arrow",bentline=FALSE,
                  labeloffset=.4,fontsize=1.2,arrowlength = 0.025,
                  labeltext=TRUE)
> labelgenome( chrom, chromstart,chromend,n=3,scale="Mb")

```



This function can also be used to plot transcript structures. The first 20 lines of a data frame describing RNA seq data are shown below.

```

> Sushi_transcripts.bed[1:20,]

  chrom    start     stop      gene   score strand type
1 chr15 73062668 73062770 ENST00000362710 0.000000 -1 exon
2 chr15 73097788 73097929 ENST00000361900 0.000000  1 exon
3 chr15 73097264 73097365 ENST00000361900 0.000000  1 exon
4 chr15 73095987 73096143 ENST00000361900 0.000000  1 exon
5 chr15 73092071 73092199 ENST00000361900 0.000000  1 exon
6 chr15 73091234 73091240 ENST00000361900 0.000000  1 exon
7 chr15 73017309 73017408 ENST00000322347 31.488695 -1 exon
8 chr15 73006160 73006281 ENST00000322347 31.488695 -1 exon
9 chr15 73008510 73008626 ENST00000322347 31.488695 -1 exon
10 chr15 72984058 72984106 ENST00000357635 7.473977 -1 exon
11 chr15 72984548 72984625 ENST00000357635 7.473977 -1 exon
12 chr15 72985672 72985759 ENST00000357635 7.473977 -1 exon

```

```

13 chr15 72985981 72986194 ENST00000357635 7.473977 -1 exon
14 chr15 72975546 72975719 ENST00000379693 2.422616 1 exon
15 chr15 72972532 72972714 ENST00000379693 2.422616 1 exon
16 chr15 72972055 72972196 ENST00000379693 2.422616 1 exon
17 chr15 72970773 72970973 ENST00000379693 2.422616 1 exon
18 chr15 72969965 72970048 ENST00000379693 2.422616 1 exon
19 chr15 72972532 72972714 ENST00000352410 0.917141 1 exon
20 chr15 72972055 72972196 ENST00000352410 0.917141 1 exon

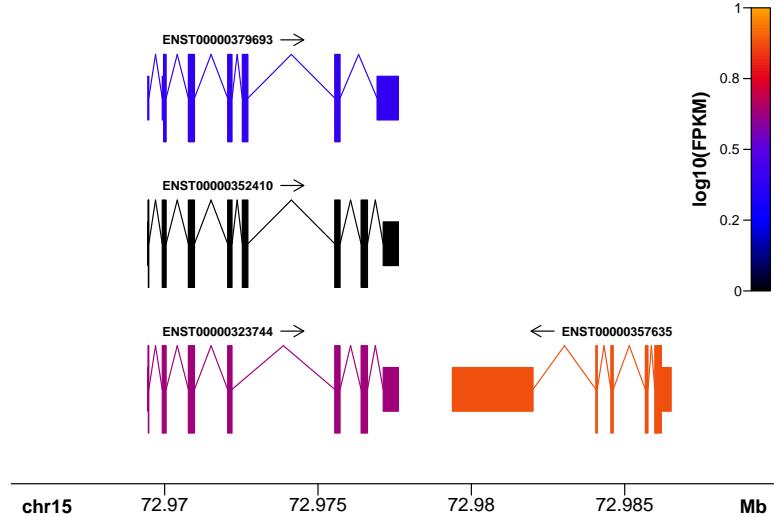
```

A vector type can be used to specify if each region is an 'exon' or 'utr' while `plotgenetype="box"` plots regions as a boxes rather than arrows. The data can be plotted using `plotGenes()`. The `colorby` argument is used to color the transcripts by $\log_{10}(\text{FPKM})$. UTR regions are drawn as shorter boxes than exons.

```

> chrom      = "chr15"
> chromstart = 72965000
> chromend   = 72990000
> pg = plotGenes(Sushi_transcripts.bed,chrom,chromstart,chromend ,
                  types = Sushi_transcripts.bed$type,
                  colorby=log10(Sushi_transcripts.bed$score+0.001),
                  colorbycol= SushiColors(5),colorbyrange=c(0,1.0),
                  labeltext=TRUE,maxrows=50,height=0.4,plotgenetype="box")
> labelgenome(chrom, chromstart,chromend,n=3,scale="Mb")
> addlegend(pg[[1]],palette=pg[[2]],title="log10(FPKM)",side="right",
            bottominset=0.4,topinset=0,xoffset=-.035,labelside="left",
            width=0.025,title.offset=0.055)

```



3.9 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 bedgrphah 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 accoridng 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 with 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))
```

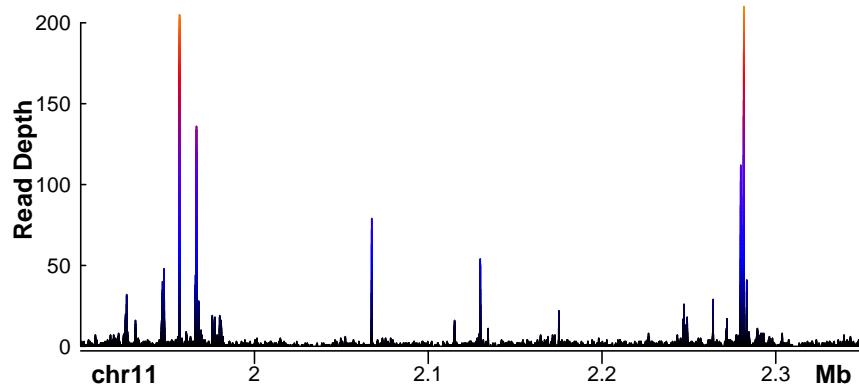
Next we will add the first plot

```
> chrom          = "chr11"
> chromstart     = 1900000
> chromend       = 2350000
```

```

> plotBedgraph(Sushi_DNaseI.bedgraph,chrom,chromstart=chromstart,
               chromend=chromend,colorbycol= SushiColors(5))
> 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)

```

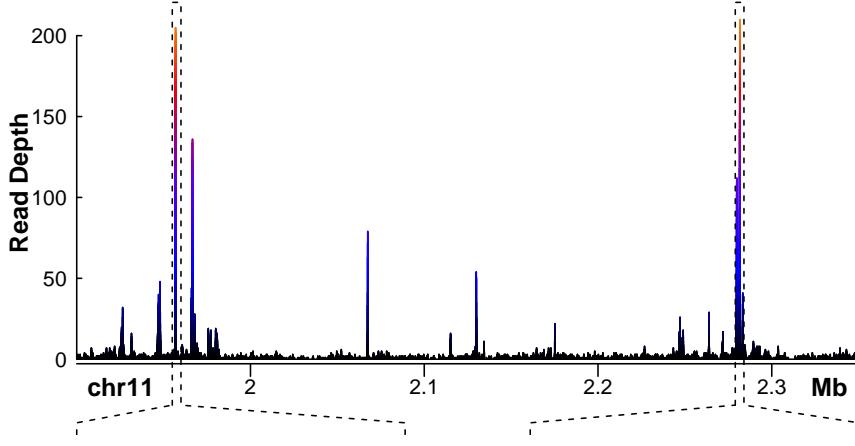


Next we will add the zoom regions using the function `zoomsregion()`. The argument `offsets` is used to precisely position the left and right edges of the widest part of the zoom.

```

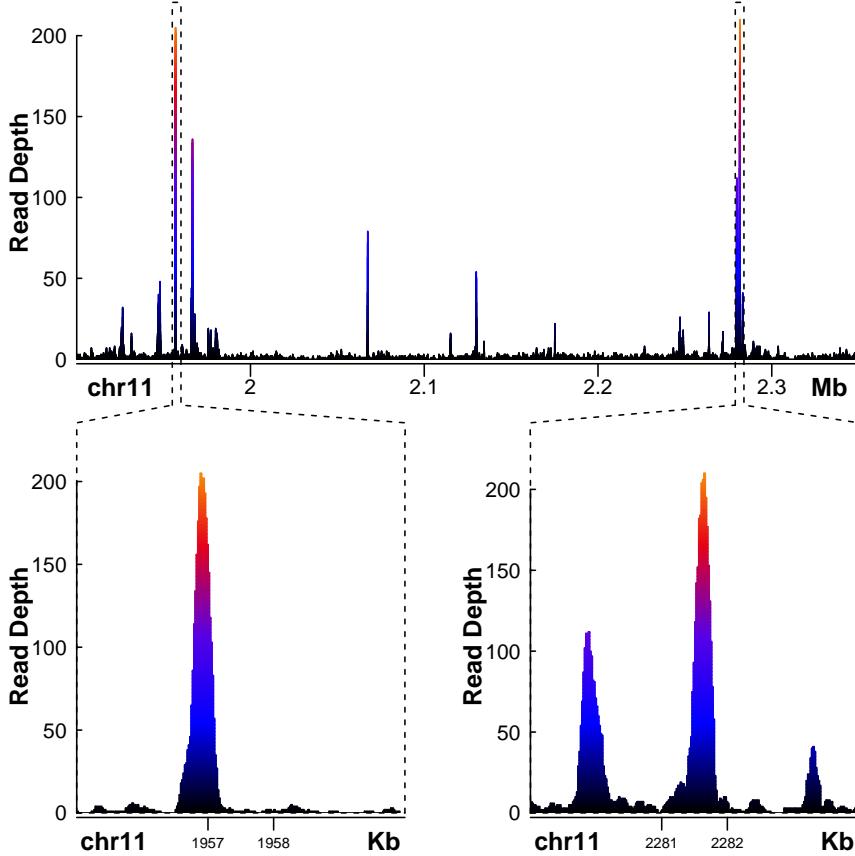
> zoomregion1      = c(1955000,1960000)
> zoomregion2      = c(2279000,2284000)
> zoomsregion(zoomregion1,extend=c(0.01,0.13),wideextend=0.05,
              offsets=c(0,0.580))
> zoomsregion(zoomregion2,extend=c(0.01,0.13),wideextend=0.05,
              offsets=c(0.580,0))

```



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)
```



3.10 Color functions

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

3.10.1 SushiColors

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

To see a list of available color palettes:

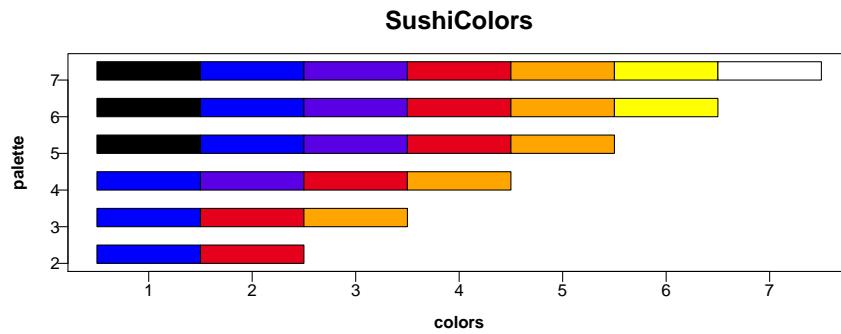
```
> SushiColors(palette='list')
[1] 2 3 4 5 6 7
```

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)

```



3.10.2 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.

```

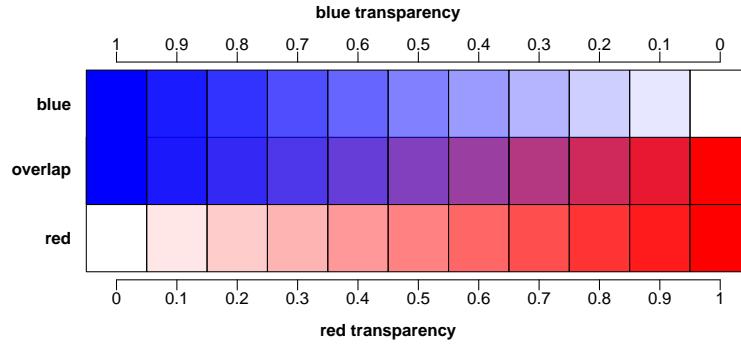
> plot(1,xlab='',xaxt='n',ylab='',yaxt='n',bty='n',type='n',
      xlim=c(-.15,1.05),ylim=c(-1,2))
> for (i in seq(0,1,by=0.1))
{
  rect(i-.05,-1,i+.05,1,col=opaque("red",transparency=i))
  rect(i-.05,0,i+.05,2,col=opaque("blue",transparency=1-i))
}
> axis(side=1,at=seq(0,1,by=0.1),labels=seq(0,1,by=0.1))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

```



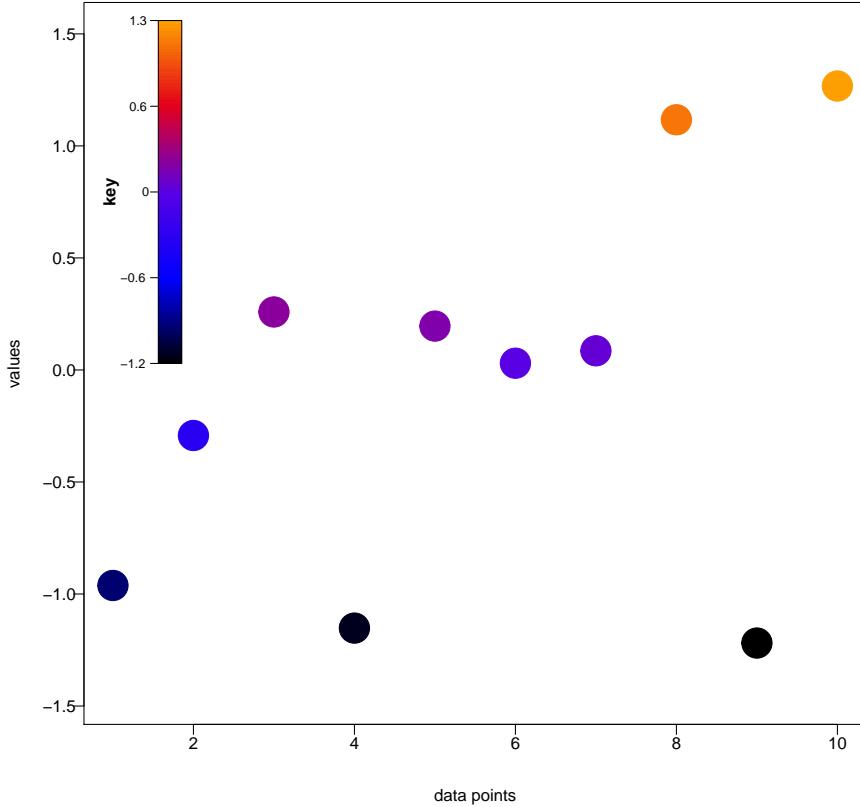
3.10.3 maptocolors

`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)

```

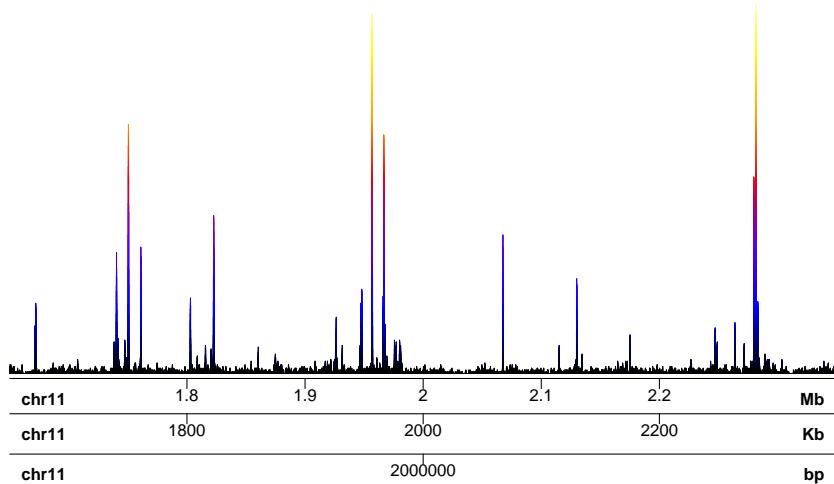


3.11 labeling functions

3.11.1 labelgenome

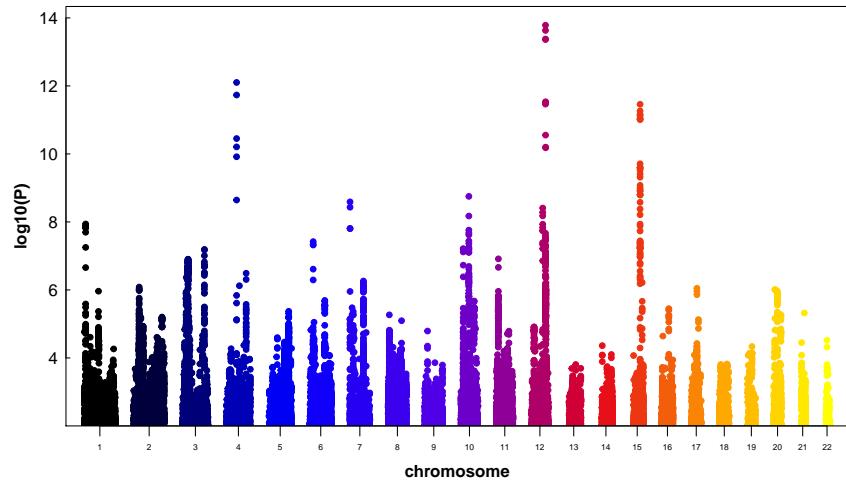
`labelgenome()` Add genome coordinates to the x-axis of a plot. The `line` argument can be used to offset the axis and `n` can be used to determine the desired number of tick marks.

```
> par(mar=c(8,3,3,1),mgp=c(3, .3, 0))
> plotBedgraph(Sushi_DNaseI.bedgraph,chrom="chr11",chromstart=1650000,
+                chromend=2350000,colorbycol=SushiColors(7))
> labelgenome(chrom="chr11",chromstart=1650000,chromend=2350000,
+              side=1,n=4,scale="Mb",line=.25)
> labelgenome(chrom="chr11",chromstart=1650000,chromend=2350000,
+              side=1,n=3,scale="Kb",line=2)
> labelgenome(chrom="chr11",chromstart=1650000,chromend=2350000,
+              side=1,n=1,scale="bp",line=4)
```



Manhattan plots include multiple genomes and labeling the axes of Manhattan plots requires the same `genome` object and value of `space` that were used to in `plotManhattan()`

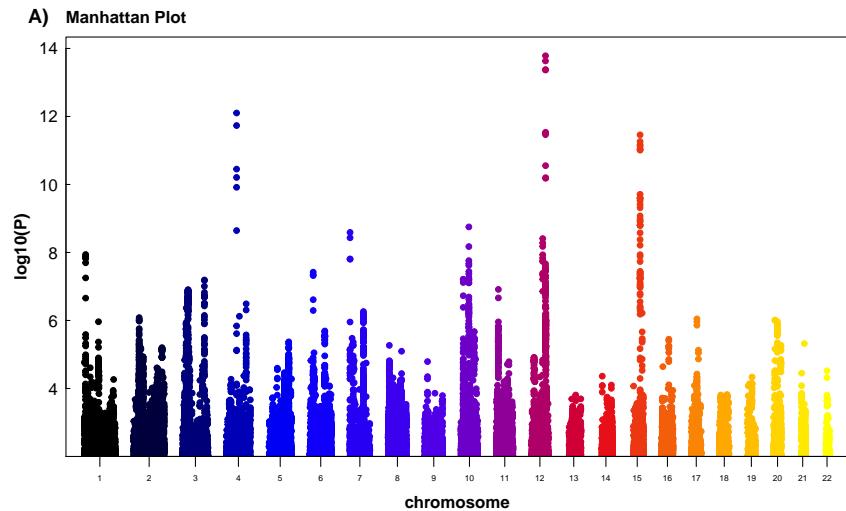
```
> plotManhattan(bedfile=Sushi_GWAS.bed,pvalues=Sushi_GWAS.bed[,5],
+                 col=SushiColors(6),genome=Sushi_hg18_genome,
+                 cex=0.75,space=0.05)
> labelgenome(genome=Sushi_hg18_genome,n=4,scale="Mb",
+              edgeblankfraction=0.20,cex.axis=.5,space=0.05)
> axis(side=2,las=2,tcl=.2)
> mtext("log10(P)",side=2,line=1.75,cex=1,font=2)
> mtext("chromosome",side=1,line=1.75,cex=1,font=2)
```



3.11.2 labelplot

Plot labels and titles can be added with the `labelplot()` function.

```
> labelplot("A", "Manhattan Plot")
```



4 Appendix

For illustrative purposes we include a complex figure as published in the accompanying manuscript (Phanstiel, et al.).

```
1 library('Sushi')
2 pdfname = "inst/doc/Figure_dhp_new.pdf"
3 Sushi_data = data(package = 'Sushi')
4 data(list = Sushi_data$results[,3])
5 makepdf = TRUE
6
7 #####
8 #### CODE
9 ####
10
11 if (makepdf == TRUE)
12 {
13   pdf(pdfname, height=10, width=12)
14 }
15
16 # make a layout for all of the plots
17 layout(matrix(c(1,1,1,1,
18               1,1,1,1,
19               2,2,8,8,
20               2,2,9,9,
21               3,3,10,10,
22               3,3,10,10,
23               4,4,11,11,
24               4,4,11,11,
25               5,5,12,12,
26               5,5,12,12,
27               6,7,13,13,
28               6,7,14,14
29 ), 12, 4, byrow=TRUE))
30 par(mgp=c(3,.3,0))
31
32
33 #####
34 #### (A) manhattan plot
35 ####
36
37 # set the margins
38 par(mar=c(3,4,3,2))
39
40 # set the genomic regions
41 chrom1      = "chr11"
42 chromstart1 = 500000
43 chromend1   = 5050000
44
45 chrom2      = "chr15"
```

```

46 chromstart2      = 73000000
47 chromend2       = 89500000
48
49 # make the manhattan plot
50 plotManhattan(bedfile=Sushi_GWAS.bed, pvalues=Sushi_GWAS.bed
51   [,5], genome=Sushi_hg18_genome, cex=0.75)
52
53 # add zoom 1
54 zoomsregion(region=c(chromstart1,chromend1), chrom=chrom1,
55   genome=Sushi_hg18_genome, extend=c(0.07,0.2), wideextend
56   =0.2, offsets=c(0,.535))
57
58 # add labels
59 labelgenome(genome=Sushi_hg18_genome, n=4, scale="Mb",
60   edgeblankfraction=0.20)
61
62 # add y-axis
63 axis(side=2, las=2, tcl=.2)
64 mtext("log10(P)", side=2, line=1.75, cex=.75, font=2)
65
66 # Add plot label
67 labelplot("A", "GWAS", letteradj=-.025)
68
69 #####
70 #### (B) Hi-C
71 ####
72
73 # set the margins
74 par(mar=c(3,4,2,2))
75
76 # set the genomic regions
77 chrom          = "chr11"
78 chromstart     = 500000
79 chromend       = 5050000
80 zoomregion     = c(1700000,2350000)
81
82 # plot the HiC data
83 phic = plotHiC(Sushi_HiC.matrix, chrom, chromstart, chromend,
84   max_y = 20, zrange=c(0,28))
85
86 # add labels
87 labelgenome(chrom, chromstart, chromend, n=4, scale="Mb",
   edgeblankfraction=0.20)

```

```

88  # add the legend
89  addlegend(phic[[1]], palette=phic[[2]], title="score", side="right",
90    bottominset=0.4, topinset=0, xoffset=-.035,
91    labelside="left", width=0.025, title.offset=0.035)
92
93
94  # add zoom
95  zoomsregion(region=zoomregion, extend=c(0.05,0.25))
96
97  # add zoombox
98  zoombox(zoomregion=zoomregion)
99
100
101 #####
102 #### (C) 5C
103 ####
104
105  # set the margins
106  par(mar=c(3,4,2,2))
107
108  # set the genomic regions
109  chrom           = "chr11"
110  chromstart      = 1650000
111  chromend        = 2350000
112
113  # plot the loops
114  pbpe = plotBedpe(Sushi_5C.bedpe, chrom, chromstart, chromend,
115    heights=Sushi_5C.bedpe$score, offset=0, flip=FALSE, bty='n',
116    lwd=1, plottype="loops", colorby=Sushi_5C.bedpe$samplenumber,
117    colorbycol=SushiColors(3))
118
119  # add zoombox
120  zoombox(passthrough=TRUE)
121
122  # add the genome labels
123  labelgenome(chrom, chromstart, chromend, n=3, scale="Mb")
124
125  # add the legend
126  legend("topright", inset=0.01, legend=c("K562", "HeLa", "GM12878"),
127    col=SushiColors(3)(3), pch=19, bty='n', text.font=2)
128
129  # add y-axis
130  axis(side=2, las=2, tcl=.2)
131  mtext("Z-score", side=2, line=1.75, cex=.75, font=2)
132
133  # Add plot label
134  labelplot("C", "5C")
135

```

```

132
133 #####
134 #### (D) ChIA PET (PolII)
135 ####
136
137 # set the margins
138 par(mar=c(3,4,2,2))
139
140 # set the genomic regions
141 chrom = "chr11"
142 chromstart = 1650000
143 chromend = 2350000
144
145 # plot the loops
146 pbpe = plotBedpe(Sushi_ChIAPET_pol2.bedpe, chrom, chromstart,
147   chromend, flip=TRUE, bty='n', lwd=1, plottype="lines",
148   colorby=abs(Sushi_ChIAPET_pol2.bedpe$start1-Sushi_ChIAPET-
149   pol2.bedpe$start2), colorbycol=SushiColors(5))
150
151 # add the genome labels
152 labelgenome(chrom, chromstart, chromend, n=4, scale="Mb")
153
154 # add the legend
155 addlegend(pbpe[[1]], palette=pbpe[[2]], title="distance_(bp)",
156   side="right", bottominset=0.05, topinset=0.35, xoffset
157   =-.035, labelside="left", width=0.025, title.offset=0.08,
158   labels.digits=0)
159
160
161 #####
162 #### (E) DNaseI
163 ####
164
165 # set the margins
166 par(mar=c(3,4,2,2))
167
168 # set the genomic regions
169 chrom = "chr11"
170 chromstart = 1650000
171 chromend = 2350000
172 zoomregion1 = c(1860000,1861000)
173 zoomregion2 = c(2281000,2282400)
174
175 # overlapping, transparent, and rescaled

```

```

176 plotBedgraph(Sushi_DNaseI.bedgraph, chrom, chromstart,
                 chromend, colorbycol=SushiColors(5))
177
178 # add zoom 1
179 zoomsregion(zoomregion1, extend=c(-0.8,0.18), wideextend=0.10,
               offsets=c(0,.577))
180
181 # add the genome labels
182 labelgenome(chrom, chromstart, chromend, n=4, scale="Mb")
183
184 # add zoombox
185 zoombox(zoomregion=zoomregion)
186
187 # add zoom 2
188 zoomsregion(zoomregion2, extend=c(0.01,0.18), wideextend=0.10,
               offsets=c(.577,0))
189
190 # add y-axis
191 axis(side=2, las=2, tcl=.2)
192 mtext("Read_Depth", side=2, line=1.75, cex=.75, font=2)
193
194 # Add plot label
195 labelplot("E", "DnaseI")
196
197
198 #####
199 ##### (F) ChIP-Seq ChIP Exo
200 #####
201
202 # set the genomic regions
203 chrom = "chr11"
204 chromstart = 1650000
205 chromend = 2350000
206 zoomregion1 = c(1860000,1861000)
207 zoomregion2 = c(2281000,2282400)
208
209 # plot chip-seq data
210 plotBedgraph(Sushi_ChIPSeq_CTCF.bedgraph, chrom, zoomregion1
                [1], zoomregion1[2], transparency=.50, color=SushiColors
                (2)(2)[1])
211
212 # plot chip-seq data
213 plotBedgraph(Sushi_ChIPExo_CTCF.bedgraph, chrom, zoomregion1
                [1], zoomregion1[2], transparency=.50, color=SushiColors
                (2)(2)[2], overlay=TRUE, rescaleoverlay=TRUE)
214
215 # Add plot label
216 labelplot("F", "ChIP-Sequ/_ChIP-Exo", letteradj=-.125)
217
218 # add the genome labels

```

```

219 labelgenome(chrom, zoomregion1[1], zoomregion1[2], n=3, line
220 =.5, scale="Mb", edgeblankfraction=0.2)
221 # add zoombox
222 zoombox()
223
224 # add legend
225 legend("topright", inset=0.025, legend=c("ChIP-seq_(CTCF)", "
226 ChIP-exo_(CTCF)", fill=opaque(SushiColors(2)(2),0.5),
227 border=SushiColors(2)(2), text.font=2, cex=0.75)
228 #####
229 #### (G) Bed Pile up
230 ####
231
232 # set the genomic regions
233 chrom = "chr11"
234 chromstart = 1650000
235 chromend = 2350000
236 zoomregion1 = c(1955000,1965000)
237 zoomregion2 = c(2281000,2282400)
238
239 # plt the chip-seq data as a pile-up
240 plotBed(beddata=Sushi_ChIPSeq_pol2.bed, chrom=chrom,
241 chromstart=zoomregion2[1], chromend=zoomregion2[2],
242 colorby=Sushi_ChIPSeq_pol2.bed$strand, colorbycol=
243 SushiColors(2), wiggle=0.001, height=0.25)
244
245 # add the genome labels
246 labelgenome(chrom, zoomregion2[1], zoomregion2[2], n=2, scale=
247 "Mb")
248
249 # add zoombox
250 zoombox()
251
252 # add legend
253 legend("topright", inset=0.025, legend=c("reverse","forward"),
254 fill=SushiColors(2)(2), border=SushiColors(2)(2), text.
255 font=2, cex=0.75)
256
257 #####
258 #### (H) manhattan plot zoomed
259 ####
260
261 # set the margins

```

```

260  par(mar=c(0.1,4,2,2))
261
262 # set the genomic regions
263 chrom           = "chr15"
264 chromstart      = 60000000
265 chromend        = 80000000
266 chromstart2     = 72000000
267 chromend2       = 74000000
268
269 # make the manhattan plot
270 plotManhattan(bedfile=Sushi_GWAS.bed, chrom=chrom2, chromstart=
271   =chromstart, chromend=chromend, pvalues=Sushi_GWAS.bed$`pval.GC.DBP`,
272   col=SushiColors(6)(nrow(Sushi_hg18_genome)))[15], cex=0.75)
273
274 # add zoom in
275 zoomsregion(region=c(chromstart2,chromend2), chrom=chrom2,
276   genome=NULL, extend=c(0.075,1), offsets=c(0.0,0))
277
278 # add zoom box
279 zoombox(passthrough=TRUE, topextend=5)
280
281 # add y-axis
282 axis(side=2, las=2, tcl=.2)
283 mtext("Z-score", side=2, line=1.75, cex=.75, font=2)
284
285
286 #####
287 #### (I) Gene density
288 ####
289
290 # set the margins
291 par(mar=c(3,4,1.8,2))
292
293 # set the genomic regions
294 chrom           = "chr15"
295 chromstart      = 60000000
296 chromend        = 80000000
297 chrom_biomart   = gsub("chr","",chrom)
298
299 # set the mart (since we want hg18 coordinates)
300 mart=useMart(host='may2009.archive.ensembl.org', biomart='ENSEMBL_MART_ENSEMBL',
301   dataset='hsapiens_gene_ensembl')
302
303 # get just gene info
304 geneinfobed = getBM(attributes=c("chromosome_name","start_
305   position","end_position"), filters=c("chromosome_name",

```

```

            start","end") , values=list(chrom_biomart , chromstart ,
chromend) , mart=mart)
304
305 # add "chr" to the chrom column
306 geneinfobed [,1] = paste("chr",geneinfobed [,1] , sep="")
307
308 # plot gene density
309 plotBed(beddata=geneinfobed [ !duplicated(geneinfobed ) , ] , chrom=
chrom , chromstart=chromstart , row='supplied' , chromend=
chromend , palettes=list(SushiColors(7)) , type="density")
310
311 #label genome
312 labelgenome(chrom=chrom , chromstart , chromend , n=4, scale="Mb"
, edgeblankfraction =0.10)
313
314 # add zoom in
315 zoomregion(region=c(chromstart2 , chromend2) , chrom=chrom2 ,
genome=NULL, extend=c(2 ,1.0) , wideextend=.75 , offsets=c
(0.0 ,0))
316
317 # add zoombox
318 zoombox(zoomregion=c(chromstart2 , chromend2) , topextend=5)
319
320 # Add plot label
321 labelplot("I)" , "Gene_Density")
322
323
324 #####
325 #### (J) RNA seq
326 ####
327
328 # set the margins
329 par(mar=c(3 ,4 ,2 ,2))
330
331 # set the genomic regions
332 chrom2 = "chr15"
333 chromstart2 = 72800000
334 chromend2 = 73100000
335 zoomregion = c(72998000 ,73020000)
336 chrom2_biomart = 15
337
338 # plot transcripts
339 pg = plotGenes(Sushi_transcripts .bed , chrom2 , chromstart2 ,
chromend2 , types=Sushi_transcripts .bed$type , colorby=log10
(Sushi_transcripts .bed$score +0.001) , colorbycol=
SushiColors(5) , labeltext=FALSE, maxrows=50, height=0.4,
plotgenetype="box")
340
341 # label genome
342 labelgenome(chrom2 , chromstart2 , chromend2 , n=3, scale="Mb")

```

```

343
344 # add the legend
345 addlegend(pg[[1]], palette=pg[[2]], title="log10(FPKM)", side=
346   "right", bottominset=0.4, topinset=0, xoffset=-.035,
347   labelside="left", width=0.025, title.offset=0.055)
348
349
350 # add zoom
351 zoomsregion(region=zoomregion, extend=c(-.025,1))
352
353 # Add plot label
354 labelplot("J", "RNA-seq")
355
356
357 #####
358 #### (K) ChIP Seq peaks
359 #####
360
361 # set the margins
362 par(mar=c(3,4,2,2))
363
364 # set the genomic regions
365 chrom = "chr15"
366 chromstart = 72800000
367 chromend = 73100000
368 zoomregion = c(72998000,73020000)
369
370 Sushi_ChIPSeq_severalfactors.bed$color = maptocolors(Sushi_
371   ChIPSeq_severalfactors.bed$row, col=SushiColors(6))
372
373 # plot it
374 plotBed(beddata=Sushi_ChIPSeq_severalfactors.bed, chrom=chrom,
375   chromstart=chromstart, chromend=chromend, rownumber=Sushi_
376   ChIPSeq_severalfactors.bed$row, type="circles", color=
377   Sushi_ChIPSeq_severalfactors.bed$color, row="given",
378   plotbg="grey95", rowlabels=unique(Sushi_ChIPSeq_
379   severalfactors.bed$name), rowlabelcol=unique(Sushi_ChIPSeq_
380   severalfactors.bed$color), rowlabelcex=0.75)
381
382 # label genome
383 labelgenome(chrom, chromstart, chromend, n=3, scale="Mb")
384
385 # add zoom
386 zoombox(zoomregion = zoomregion)
387
388 # add zoom in
389 zoomsregion(region=zoomregion, chrom=chrom, extend=c(0.5,.22),
390   wideextend=0.15, offsets=c(0.0,0))

```

```

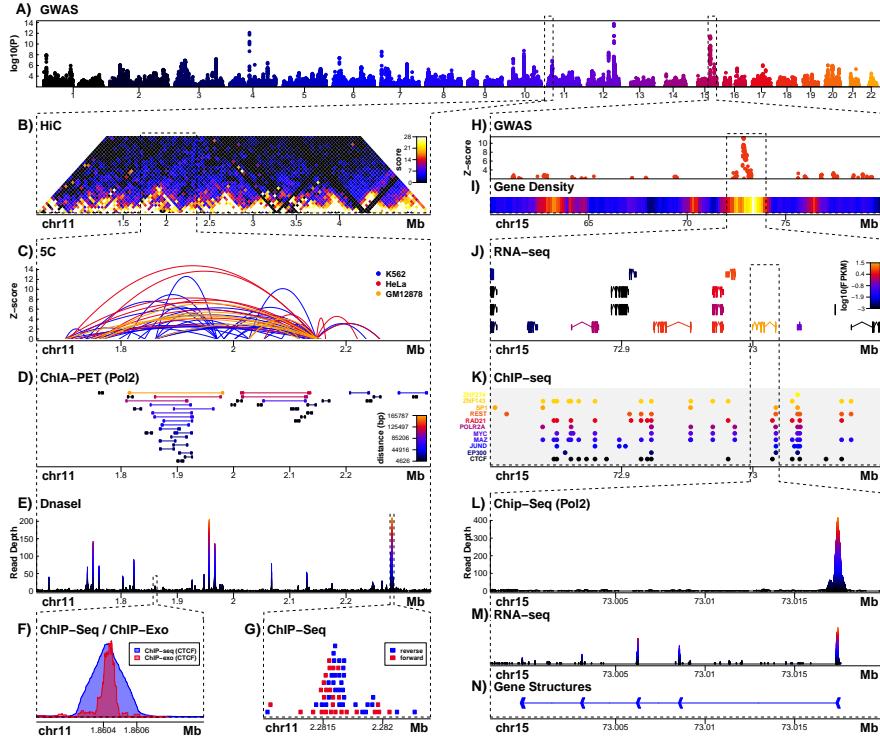
383
384 # Add plot label
385 labelplot("K", "ChIP-seq")
386
387
388 #####
389 #### (L) Pol2 bedgrph
390 #####
391
392 # set the margins
393 par(mar=c(3,4,2,2))
394
395 # set the genomic regions
396 chrom = "chr15"
397 chromstart = 72998000
398 chromend = 73020000
399
400 # plot the Pol2 bedgraph data
401 plotBedgraph(Sushi_ChIPSeq_pol2.bedgraph, chrom, chromstart,
               chromend, colorbycol=SushiColors(5))
402
403 # label genome
404 labelgenome(chrom, chromstart, chromend, n=3, scale="Mb")
405
406 # add zoombox
407 zoombox(passthrough=TRUE)
408
409 # add y-axis
410 axis(side=2, las=2, tcl=.2)
411 mtext("Read_Depth", side=2, line=1.75, cex=.75, font=2)
412
413 # Add plot label
414 labelplot("L", "Chip-Seq_(Pol2)")
415
416
417 #####
418 #### (M) RNA-seq bedgraph
419 #####
420
421 # set the margins
422 par(mar=c(2,4,.5,2))
423
424 # set the genomic regions
425 chrom = "chr15"
426 chromstart = 72998000
427 chromend = 73020000
428
429 # plot the K562 RNAseq bedgraph data
430 plotBedgraph(Sushi_RNASeq_K562.bedgraph, chrom, chromstart,
               chromend, colorbycol=SushiColors(5))

```

```

431
432 # label genome
433 labelgenome(chrom, chromstart, chromend, n=3, scale="Mb")
434
435 # add zoombox
436 zoombox(passthrough=TRUE)
437
438 # Add plot label
439 labelplot("M", "RNA-seq")
440
441
442 #####
443 #### (N) Gene Structures
444 ####
445
446 # set the margins
447 par(mar=c(3,4,.5,2))
448
449 # set the genomic region
450 chrom = "chr15"
451 chromstart = 72998000
452 chromend = 73020000
453
454 # plot gene structures
455 plotGenes(Sushi_genes.bed, chrom, chromstart, chromend,
             maxrows=1, bheight=0.2, plotgenetype="arrow", bentline=
             FALSE, labeloffset=1, fontsize=1.2)
456
457 # label genome
458 labelgenome(chrom, chromstart, chromend, n=3, scale="Mb")
459
460 # add zoombox
461 zoombox()
462
463 # Add plot label
464 labelplot("N", "Gene Structures", letterline=-0.4, titleline=
             -0.4)
465
466 if (makepdf == TRUE)
467 {
468   dev.off()
469 }

```



5 Bibliography

- [1] Biomart. URL <http://www.biomart.org/>.
- [2] JR Dixon, S Selvaraj, F Yue, A Kim, Y Li, Y Shen, M Hu, JS Liu, and B Ren. Topological domains in mammalian genomes identified by analysis of chromatin interactions. *Nature*, 485, September 2012. doi: 10.1038/nature11082. PMID: 22495300.
- [3] International Consortium for Blood Pressure. Genetic variants in novel pathways influence blood pressure and cardiovascular disease risk. *Nature*, 478, September 2011. doi: 10.1038/nature10405. PMID: 21909115.
- [4] G Li, X Ruan, RK Auerbach, KS Sandhu, M Zheng, P Wang, HM Poh, Y Goh, J Lim, J Zhang, HS Sim, SQ Peh, FH Mulawadi, CT Ong, YL Orlov, S Hong, Z Zhang, S Landt, D Raha, G Euskirchen, CL Wei, W Ge, H Wang, C Davis, KI Fisher-Aylor, A Mortazavi, M Gerstein, T Gingeras, B Wold, Y Sun, MJ Fullwood, E Cheung, E Liu, WK Sung, M Snyder, and Y Ruan. Extensive promoter-centered chromatin interactions provide a topological basis for transcription regulation. *Cell*, 148:84–98, January 2012. doi: 10.1016/j.cell.2011.12.014. PMID: 22265404.

- [5] S Neph, J Vierstra, AB Stergachis, AP Reynolds, E Haugen, B Vernot, RE Thurman, S John, R Sandstrom, AK Johnson, MT Maurano, R Humbert, E Rynes, H Wang, S Vong, K Lee, D Bates, M Diegel, V Roach, D Dunn, J Neri, A Schafer, RS Hansen, T Kutyavin, E Giste, M Weaver, T Canfield, P Sabo, M Zhang, G Balasundaram, R Byron, MJ MacCoss, JM Akey, MA Bender, M Groudine, R Kaul, and JA Stamatoyannopoulos. An expansive human regulatory lexicon encoded in transcription factor footprints. *Nature*, 489:83–90, September 2012. doi: 10.1038/nature11212. PMID: 22955618.
- [6] HS Rhee and BF Pugh. Comprehensive genome-wide protein-dna interactions detected at single-nucleotide resolution. *Cell*, 147, December 2011. doi: 10.1016/j.cell.2011.11.013. PMID: 22153082.
- [7] A Sanyal, BR Lajoie, G Jain, and J Dekker. The long-range interaction landscape of gene promoters. *Nature*, 489(7414):109–113, September 2012. doi: 10.1038/nature11279. PMID: 22955621.
- [8] The ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74, September 2012. ISSN 1476-4687. doi: 10.1038/nature11247. URL <http://www.ncbi.nlm.nih.gov/pubmed/22955616>. PMID: 22955616.