trackViewer guide

Jianhong Ou[*], Lihua Julie Zhu[†‡]

April 16, 2015

Contents

1 Introduction

There are two packages available in Bioconductor for visualizing genomic data: *rtracklayer* and *Gviz*. *rtracklayer* provides an interface to genome browsers and associated annotation tracks. *Gviz* plots data and annotation information along genomic coordinates. *TrackViewer* is a light-weighted visualization tool for generating neat figures for publication. It utilizes *Gviz*, is easy to use, and has a low memory and cpu consumption.

[*]jianhong.ou@umassmed.edu
[†]julie.zhu@umassmed.edu
[‡]The authors would like to acknowledge Paul Shannon for technical advice during development.

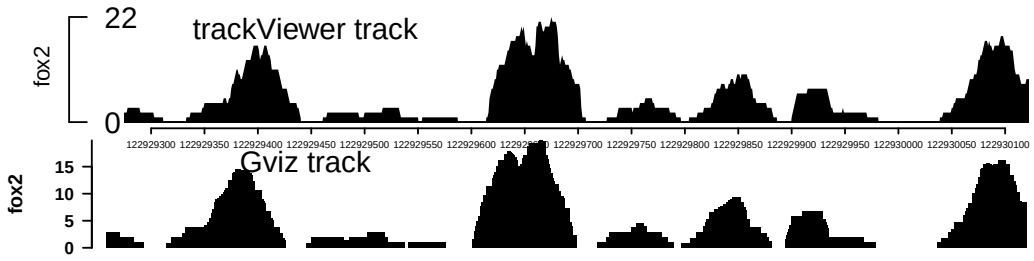

Figure 1: Plot data with *Gviz* and *trackViewer*. Note that *trackViewer* can generate similar figure as *Gviz* with several lines of simple codes.

```
> library(Gviz)
> library(rtracklayer)
> library(trackViewer)
> extdata <- system.file("extdata", package="trackViewer",
+                        mustWork=TRUE)
> gr <- GRanges("chr11", IRanges(122929275, 122930122), strand="-")
> fox2 <- importScore(file.path(extdata, "fox2.bed"), format="BED",
+                     ranges=gr)
> fox2$dat <- coverageGR(fox2$dat)
> viewTracks(trackList(fox2), gr=gr, autoOptimizeStyle=TRUE, newpage=FALSE)
> dt <- DataTrack(range=fox2$dat[strand(fox2$dat)=="-"] ,
+                 genome="hg19", type="hist", name="fox2",
+                 window=-1, chromosome="chr11",
+                 fill.histogram="black", col.histogram="NA",
+                 background.title="white",
+                 col.frame="white", col.axis="black",
+                 col="black", col.title="black")
> plotTracks(dt, from=122929275, to=122930122, strand="-")
```

TrackViewer not only has the functionalities to plot the figures generated by *Gviz*, as shown in Figure 1, but also provides additional plotting styles as shown in Figure 2. The mimimalist design requires minimum input from users while retaining the flexibility to change output style easily. In addition, *trackViewer* provides interactive browser for genomic data. For more informations, please type `?interactiveViewer` in a R session.

```
> gr <- GRanges("chr1", IRanges(c(1, 6, 10), c(3, 6, 12)), score=c(3, 4, 1))
> dt <- DataTrack(range=gr, data="score", type="hist")
> plotTracks(dt, from=2, to=11)
> tr <- new("track", dat=gr, type="data", format="BED")
> viewTracks(trackList(tr), chromosome="chr1", start=2, end=11)
```

It requires huge memory space to handle big wig files. To solve this problem, *trackViewer* rewrote the import function to import whole file first and parse it later when plot. *trackViewer* provides higher import speed (21 min vs. over 180 min) and acceptable memory cost (5.32G vs. over 10G) for a half giga wig file (GSM917672) comparing to *Gviz*.

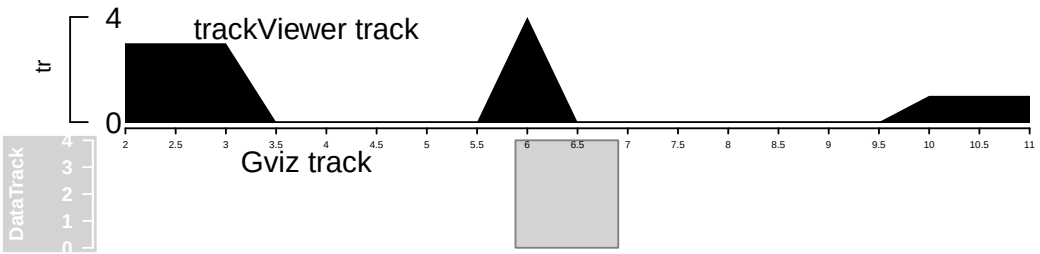

Figure 2: Plot data with *Gviz* and *trackViewer*. Note that *trackViewer* is not only including more details but also showing all the data involved in the given range.

2 Steps of using trackViewer

2.1 step1 import data

Function `importScore` is used to import BED, WIG, bedGraph or BigWig files. Function `importBam` is employed to import bam file. Here is the example.

```
> library(trackViewer)
> extdata <- system.file("extdata", package="trackViewer",
+                        mustWork=TRUE)
> repA <- importScore(file.path(extdata, "cpsf160.repA_-.wig"),
+                     file.path(extdata, "cpsf160.repA_+.wig"),
+                     format="WIG")
> ## because the wig file does not contain strand info,
> ## we need to set it manually
> strand(repA$dat) <- "-"
> strand(repA$dat2) <- "+"
```

Function `coverageGR` could be used to calculate coverage after importing if needed.

```
> fox2 <- importScore(file.path(extdata, "fox2.bed"), format="BED",
+                     ranges=GRanges("chr11", IRanges(122929000, 122931000)))
> dat <- coverageGR(fox2$dat)
> ## we can split the data by strand into two different track channels
> ## here we set the dat2 slot to save the negative strand info,
> ## reverse order as previous.
> fox2$dat <- dat[strand(dat)=="+"]
> fox2$dat2 <- dat[strand(dat)=="-"]
```

2.2 step2 build gene model

The gene model can be built for a given genomic range using `geneModelFromTxdb` function which uses `TranscriptDb` object as input.

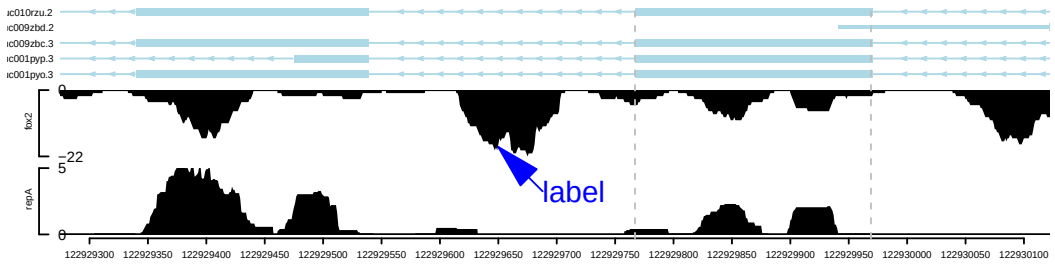

Figure 3: plot data and annotation information along genomic coordinates

```
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> trs <- geneModelFromTxdb(TxDb.Hsapiens.UCSC.hg19.knownGene,
+                          "chr11", 122929275, 122930122, "-")
```

2.3 step3 view the tracks

Use `viewTracks` function to plot data and annotation information along genomic coordinates. `addGuideLine` or `addArrowMark` can be used to highlight the peaks. (Figure 3).

```
> gr <- GRanges("chr11", IRanges(122929275, 122930122), strand="-")
> viewerStyle <- trackViewerStyle()
> setTrackViewerStyleParam(viewerStyle, "margin", c(.1, .05, .02, .02))
> vp <- viewTracks(trackList(repA, fox2, trs),
+                  gr=gr, viewerStyle=viewerStyle,
+                  autoOptimizeStyle=TRUE)
> addGuideLine(c(122929767, 122929969), vp=vp)
> addArrowMark(list(x=122929650,
+                   y=unit(.39, "npc")),
+              label="label",
+              col="blue",
+              vp=vp)
```

3 Adjust the styles

3.1 adjust x axis or x-scale

In most cases, researchers are interested in the relative position of peaks in the gene. Sometimes, margin needs to be adjusted to be able to show the entire gene model. Figure 4 shows how to add an x-scale and remove x-axis using `addGuideLine` Function .

```
> optSty <- optimizeStyle(trackList(repA, fox2, trs))
> trackList <- optSty$tracks
> viewerStyle <- optSty$style
```

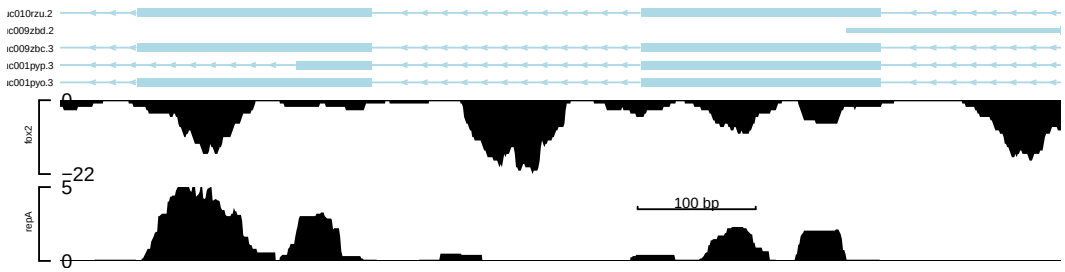

Figure 4: plot data with x-scale

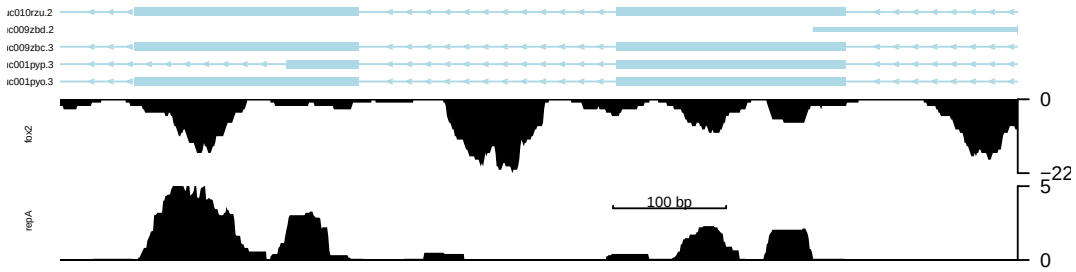

Figure 5: plot data with y-axis in right side

```
> setTrackViewerStyleParam(viewerStyle, "xaxis", FALSE)
> setTrackViewerStyleParam(viewerStyle, "margin", c(.01, .05, .01, .01))
> setTrackXscaleParam(trackList[[1]], "draw", TRUE)
> setTrackXscaleParam(trackList[[1]], "gp", list(cex=.5))
> viewTracks(trackList, gr=gr, viewerStyle=viewerStyle)
```

3.2 adjust y axis

y-axis can be put to right side of the track by setting main slot to FALSE in y-axis slot of each track (Figure 5).

```
> setTrackViewerStyleParam(viewerStyle, "margin", c(.01, .05, .01, .05))
> setTrackYaxisParam(trackList[[1]], "main", FALSE)
> setTrackYaxisParam(trackList[[2]], "main", FALSE)
> viewTracks(trackList, gr=gr, viewerStyle=viewerStyle)
```

3.3 adjust y label

Y label style can be changed by setting the ylabgp slot in style of each track (Figure 6).

```
> setTrackStyleParam(trackList[[1]], "ylabgp", list(cex=.8, col="green"))
> ## set cex to avoid automatic adjust
> setTrackStyleParam(trackList[[2]], "ylabgp", list(cex=.8, col="blue"))
```

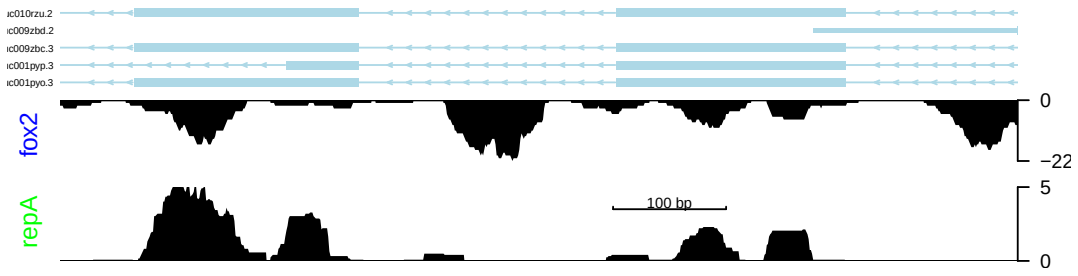

Figure 6: plot data with adjusted color and size of y label

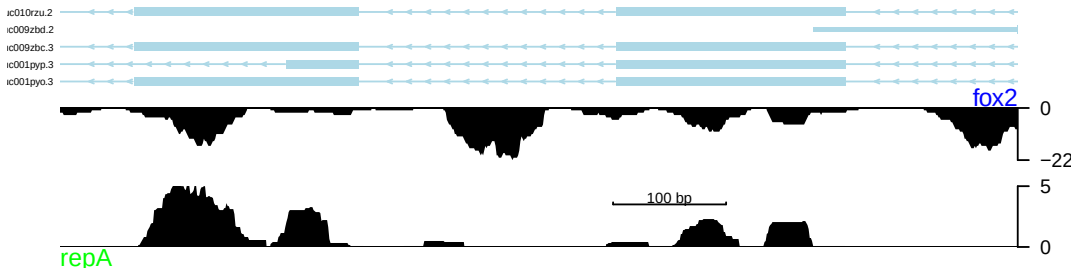

Figure 7: plot data with adjusted y label position

```
> setTrackStyleParam(trackList[[2]], "marginBottom", .2)
> viewTracks(trackList, gr=gr, viewerStyle=viewerStyle)
```

Y label can be also put to top or bottom of each track (Figure 7).

```
> setTrackStyleParam(trackList[[1]], "ylabpos", "bottomleft")
> setTrackStyleParam(trackList[[1]], "marginBottom", .2)
> ## set cex to avoid automatic adjust
> setTrackStyleParam(trackList[[2]], "ylabpos", "topright")
> setTrackStyleParam(trackList[[2]], "marginTop", .2)
> viewTracks(trackList, gr=gr, viewerStyle=viewerStyle)
```

3.4 adjust track color

The track color can be changed by setting the color slot in style of each track (Figure 8). The first color is for dat slot of `track` and seconde color is for dat2 slot.

```
> setTrackStyleParam(trackList[[1]], "color", c("green", "black"))
> setTrackStyleParam(trackList[[2]], "color", c("black", "blue"))
> for(i in 3:length(trackList))
+     setTrackStyleParam(trackList[[i]], "color", "black")
> viewTracks(trackList, gr=gr, viewerStyle=viewerStyle)
```

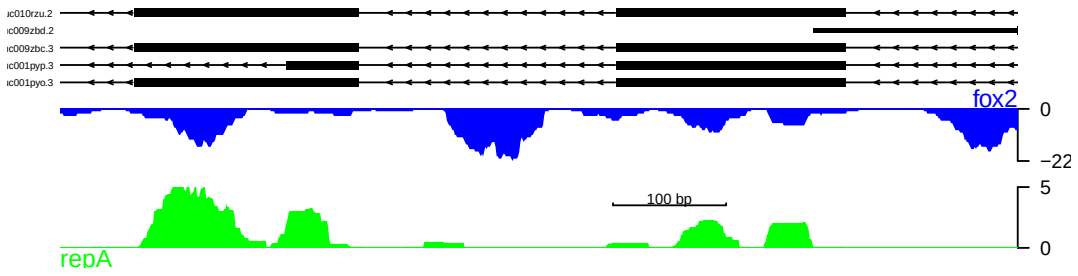

Figure 8: plot data with adjusted track color

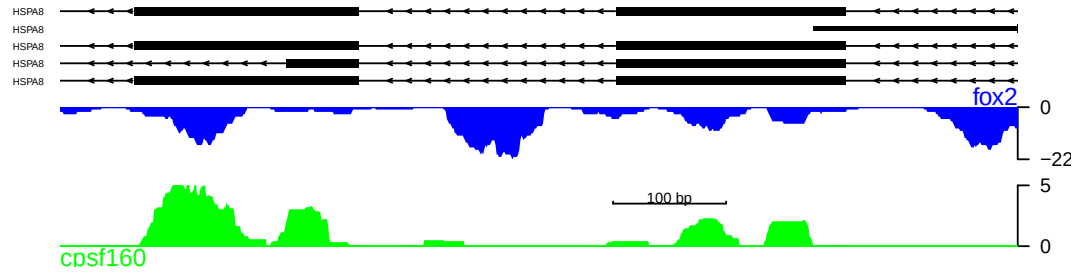

Figure 9: change the track names

3.5 change track names

The track names such as gene model names can also be edited easily by changing the names of `trackList` (Figure 9).

```
> names(trackList) <- c("cpsf160", "fox2", rep("HSPA8", 5))
> viewTracks(trackList, gr=gr, viewerStyle=viewerStyle)
```

3.6 show paired data in the same track

trackViewer can be used to show to-be-compared data in the same track side by side (Figure 10).

```
> cpsf160 <- importScore(file.path(extdata, "cpsf160.repA_-.wig"),
+                        file.path(extdata, "cpsf160.repB_-.wig"),
+                        format="WIG")
> strand(cpsf160$dat) <- strand(cpsf160$dat2) <- "-"
> setTrackStyleParam(cpsf160, "color", c("black", "red"))
> viewTracks(trackList(trs, cpsf160), gr=gr, viewerStyle=viewerStyle)
```

3.7 flip the x-axis

The x-axis can be horizotally flipped for the genes in negative strand (Figure 11).

```
> setTrackViewerStyleParam(viewerStyle, "flip", TRUE)
> setTrackViewerStyleParam(viewerStyle, "xaxis", TRUE)
```

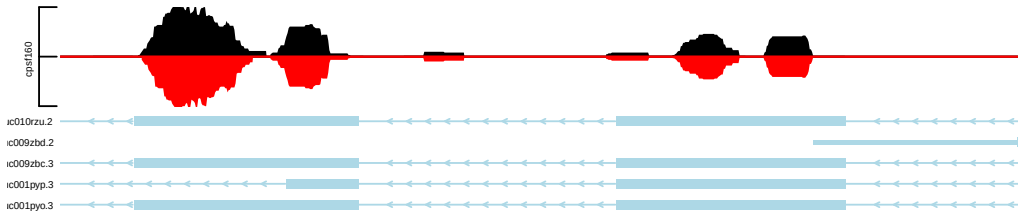

Figure 10: show two data in the same track

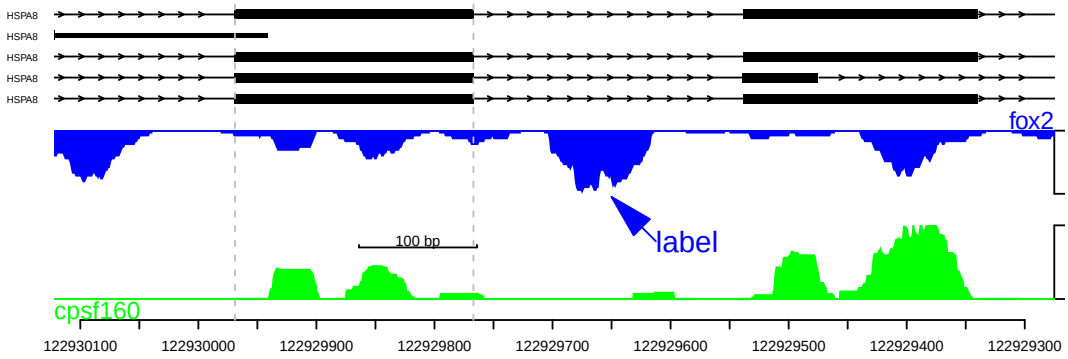

Figure 11: show data in the flipped track

```
> setTrackViewerStyleParam(viewerStyle, "margin", c(.1, .05, .01, .01))
> vp <- viewTracks(trackList, gr=gr, viewerStyle=viewerStyle)
> addGuideLine(c(122929767, 122929969), vp=vp)
> addArrowMark(list(x=122929650,
+                   y=unit(.39, "npc")),
+              label="label",
+              col="blue",
+              vp=vp)
```

4 Session Info

```
> toLatex(sessionInfo())
```

- R version 3.2.0 (2015-04-16), x86_64-unknown-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, stats4, tcltk, utils
- Other packages: AnnotationDbi 1.30.0, Biobase 2.28.0, BiocGenerics 0.14.0, GenomeInfoDb 1.4.0, GenomicFeatures 1.20.0, GenomicRanges 1.20.0, Gviz 1.12.0, IRanges 2.2.0, S4Vectors 0.6.0, TxDb.Hsapiens.UCSC.hg19.knownGene 3.1.2, XVector 0.8.0, digest 0.6.8, gWidgets 0.0-54, gWidgetstcltk 0.0-55, rtracklayer 1.28.0, trackViewer 1.4.0
- Loaded via a namespace (and not attached): BSgenome 1.36.0, BiocParallel 1.2.0, BiocStyle 1.6.0, Biostrings 2.36.0, DBI 0.3.1, Formula 1.2-1, GenomicAlignments 1.4.0, Hmisc 3.15-0, MASS 7.3-40, RColorBrewer 1.1-2, RCurl 1.95-4.5, RSQLite 1.0.0, Rcpp 0.11.5, Rsamtools 1.20.0, VariantAnnotation 1.14.0, XML 3.98-1.1, acepack 1.3-3.3, biomaRt 2.24.0, biovizBase 1.16.0, bitops 1.0-6, cluster 2.0.1, colorspace 1.2-6, dichromat 2.0-0, foreign 0.8-63, futile.logger 1.4, futile.options 1.0.0, ggplot2 1.0.1, gtable 0.1.2, lambda.r 1.1.7, lattice 0.20-31, latticeExtra 0.6-26, matrixStats 0.14.0, munsell 0.4.2, nnet 7.3-9, pbapply 1.1-1, plyr 1.8.1, proto 0.3-10, reshape2 1.4.1, rpart 4.1-9, scales 0.2.4, splines 3.2.0, stringr 0.6.2, survival 2.38-1, tools 3.2.0, zlibbioc 1.14.0