Visualizing Exon Junction Reads

Jennifer Chang

November 16, 2011

Introduction

Dependencies

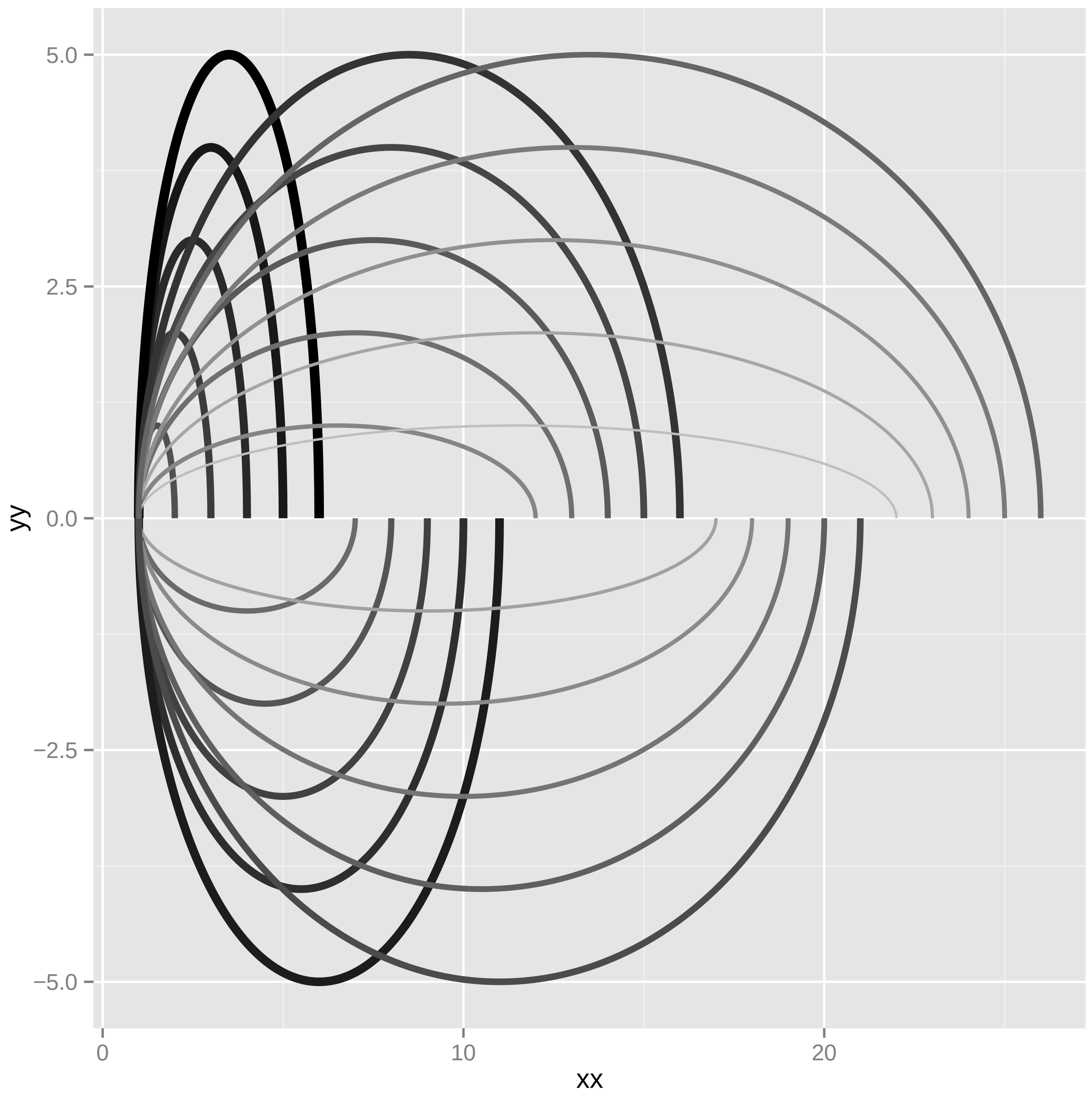
Open R or Rstudio and create a blank script. Visualizing the exon junction reads will require the ggplot2[2] and GenomicRanges[1] packages. Therefore the first step is to install the packages, if you do not have them already.

```
install.packages("ggplot2")
source("http://bioconductor.org/biocLite.R")
biocLite("GenomicRanges")
```

Example Run

```
require(ggplot2,quietly=TRUE)
require(GenomicRanges, quietly=TRUE,warn.conflicts=FALSE)

## Loading required package:  parallel
##
## Attaching package:  'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##    clusterExport, clusterMap, parApply, parCapply, parLapply,
##    parLapplyLB, parRapply, parSapply, parSapplyLB
##
## The following object is masked from 'package:stats':
##
##    xtabs
##
## The following objects are masked from 'package:base':
##
##    anyDuplicated, append, as.data.frame, as.vector, cbind,
##    colnames, do.call, duplicated, eval, evalq, Filter, Find, get,
##    intersect, is.unsorted, lapply, Map, mapply, match, mget,
##    order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##    rbind, Reduce, rep.int, rownames, sapply, setdiff, sort,
##    table, tapply, union, unique, unlist, unsplit
##
## Loading required package:  stats4
```

```r
gr<-GRanges("chr1", IRanges(c(1,20,50,60), width=c(5,10,6,7)))
combs<-combn(1:4,2)
grl<-do.call("GRangesList", apply(combs,2,function(x){
  res<-gr[x]
        values(res)<-data.frame(pvalue=runif(2))
        res
}))

values(grl)<-data.frame(counts=sample(1:100,size=6), score=rnorm(6))

# Function
geom_arch<-function(data,...,startX, endX, y, h){
  xx<-c()
        yy<-c()

        #CREATES UNIT ARCH
        #only calculate points to draw a quarter of the curve, reduces time spent in for loop
        n=500 #number of points to draw quarter of curve, just has to be sufficiently high so curve loo
        for(i in 1:n){
                ang<-i*pi/(2*n)
                xx[i]<-cos(ang)
                yy[i]<-sin(ang)
        }
        #takes the quarter of the curve calculated, flips a copy over the y axis
        #reduces time spent in for loop
        xx<-c(1,xx,rev(-xx),-1)
        yy<-c(0,yy,rev(yy), 0)

        #SETS UP DATAFRAME TO KEEP TRACK OF ALL POINTS TO DRAW ALL ARCHES
        apoint<-data.frame()
        jump<-abs(endX-startX)
        jumpAdj=max(jump)/max(abs(h))
        for(i in 1:length(startX)){
                temp<-data.frame(xx=xx*(abs(startX[i]-endX[i])/2)+(startX[i]+endX[i])/2,
                                                                 yy=yy*h[i]+y,
                                                                 junc=i,
                                                                 s=(abs(h[i])-jump[i]/jumpAdj))
                apoint<-rbind(apoint,temp)
        }
        geom_line(data=apoint, aes(xx,yy,group=junc,size=s,colour=s))
}

#TESTING CODE
start<-c(rep(1, 25)); end<-c(2:26); height<-c(rep(c(1:5, 1:5*(-1)), 2), c(1:5)); y=0
p<-ggplot()+geom_arch(startX=start,endX=end,y=y,h=height)
p+scale_size(range=c(0.5,2))+scale_colour_gradient(low="grey", high="black")+
  theme(legend.position="none")
```



References

[1] Michael Lawrence, Wolfgang Huber, Hervé Pagès, Patrick Aboyoun, Marc Carlson, Robert Gentleman, Martin Morgan, and Vincent Carey. Software for computing and annotating genomic ranges. *PLoS Computational Biology*, 9, 2013.

[2] Hadley Wickham. *ggplot2: elegant graphics for data analysis.* Springer New York, 2009.