MayW2

Awais Choudhry

10/05/2017

10/5/2017

Today i revisted my shiny app that i developed approximately a month ago to see which componments i can improve upon, optomize and remove to make the app more readable and faster to run. Overall i have noticed serval short commings. One is that i convert GRanges to dataframes in order to subset however, there needs to be more efficent methds of doing this.

#  
# This is the server logic of a Shiny web application. You can run the   
# application by clicking 'Run App' above.  
#  
# Find out more about building applications with Shiny here:  
#   
# http://shiny.rstudio.com/  
#  
  
library(shiny)  
library(BSgenome.Mmusculus.UCSC.mm9)  
library(Gviz)  
library(magrittr)  
library(dplyr)  
library(magrittr)  
library(dplyr)  
library(GenomicRanges)  
library(magrittr)  
library(parallel)  
library(biomaRt)  
library(MotifDb)  
library(TxDb.Mmusculus.UCSC.mm9.knownGene)  
library(org.Mm.eg.db)  
# Define server logic required to draw a histogram  
shinyServer(function(input, output) {  
 output$gvizPlot <- renderPlot({   
 if(!exists("chrM")){  
   
 chrM<- "chrX"  
 toM<- 20000000  
 fromM<- 5000000  
 percent<-"100%"  
 setwd("~/Research Proposal/")  
   
 ##Ideogram  
 Itrack<-IdeogramTrack(chromosome = chrM, genome = "mm9")  
   
 ##Genes  
  
 txdbMm9 <- TxDb.Mmusculus.UCSC.mm9.knownGene  
 knownGenesMouse <- GeneRegionTrack(txdbMm9, genome="mm9", chromosome=chrM, showId=TRUE, geneSymbol=TRUE, name="UCSC", stacking = "pack")  
 symbols <- unlist(mapIds(org.Mm.eg.db, gene(knownGenesMouse), "SYMBOL", "ENTREZID", multiVals = "first"))  
 symbol(knownGenesMouse) <- symbols[gene(knownGenesMouse)]  
   
   
 ## Mouse Arx Motifs  
   
 mouseArxPWM <-rbind( A=c(0,1,1,0,0,1,0.25,0.25,0,1,1,0,0,1),  
 C=c(0,0,0,0,0,0,0.25,0.25,0),  
 G=c(0,0,0,0,0,0,0.25,0.25,0) ,  
 T=c(1,0,0,1,1,0,0.25,0.25,1,0,0,1,1,0))  
 grangesMouseArx<- matchPWM(mouseArxPWM, BSgenome.Mmusculus.UCSC.mm9[[chrM]], percent)%>%ranges()%>%as.data.frame()%>%cbind("chromosome"=chrM)  
   
   
 MouseARXmotifs<-AnnotationTrack( makeGRangesFromDataFrame(grangesMouseArx),  
 genome= "mm9",  
 chromosome = chrM,  
 name= "ARX binding sites",  
 stacking = "dense",  
 colour = "red")  
   
 ##cluster Track  
 #clustergrange<-distanceToNearest(grangesMouseArx)  
 #clustergrange<-subset(clustergrange, distance<=200)%>%countRnodeHits()  
 #clustergrange<-subset(grangesMouseArx, clustergrange)  
 #clusterDataTrack<-AnnotationTrack(clustergrange, chromosome = chrM)  
   
   
   
   
 ##DNase seq track  
   
 dnaseSeqMouse <- import(con = "~/DataFiles/DNase/Mouse/cerebrummousednaseseq.bigWig", which =  
 GRanges(chrM, IRanges(1, end = .Machine$integer.max - 1)))%>%   
 DataTrack(type= "smooth", name ="DNase 14.5", color= "green")  
   
 ##Histone modifications  
 H3K27acTrack<- import(con= "~/DataFiles/ChIPseq/Mouse/HistoneModifications/first5col/H3K27ac2.bed")%>%DataTrack(type= "histogram", name= "H3k27ac")  
 H3k36meTrack<- import(con= "~/DataFiles/ChIPseq/Mouse/HistoneModifications/first5col/H3K36me2.bed")%>%DataTrack(type= "histogram", name= "H3k36me")  
 H3K9me32Track<- import(con= "~/DataFiles/ChIPseq/Mouse/HistoneModifications/first5col/H3K9me32.bed")%>%DataTrack(type= "histogram", name= "H3K9me32")  
  
   
 ##enhancer Track  
 enhancerTrack <- import( con = "~/DataFiles/Enhancer Tracks/Mouse/Enhanceresmm9.bed", which =  
 GRanges(chrM, IRanges(1, end = .Machine$integer.max - 1))) %>%  
 AnnotationTrack(name = "Enhancers", stacking= "dense")  
   
   
 ##Conservations!  
   
   
 session <- browserSession()  
 genome(session) <- "mm9"  
 trackNames(session) ## list the track names  
 ## choose the Conservation track for a portion of mm9 chr1  
 query <- ucscTableQuery(session, "Conservation",  
 GRangesForUCSCGenome("mm9", chrM,  
 IRanges(1, .Machine$integer.max - 1)))  
 ## list the table names  
 tableNames(query)  
 ## get the phastCons30way track  
 tableName(query) <- "phastConsElements30way" ## retrieve the track data  
 conservationTrack<-track(query) # a GRanges object  
   
 tableName(query) <- "multiz30way"## get a data.frame summarizing the multiple alignment  
 object<-getTable(query)  
 dataFrameGrange<- as.data.frame(cbind(object[2:4], object[7]))  
 GrangeConservation<- makeGRangesFromDataFrame(dataFrameGrange,  
 keep.extra.columns=TRUE,  
 ignore.strand=FALSE,  
 seqinfo=NULL,  
 seqnames.field=c("seqnames", "seqname",  
 "chromosome", "chrom",  
 "chr", "chromosome\_name",  
 "seqid"),  
 start.field="start",  
 end.field=c("end", "stop"),  
 strand.field="strand",  
 starts.in.df.are.0based=FALSE)%>%  
 DataTrack(type = "heatmap", name = "Conservation Track")  
   
   
   
   
 gtfUCSCexonscoding<-import("~/Scripts/March/FullMm9genome.GTF")  
 gtfUCSCgenes<- import("~/Scripts/March/mm9.bed")  
 methylationGrange<- import(con = "~/DataFiles/Methylation Tracks/Mouse/CpGIslands.bed")  
 promoterGrange<- promoters(gtfUCSCgenes)  
   
   
 fromM <- input$fromM  
 toM<- input$toM  
 chrM<- input$chrM  
   
 # draw the histogram with the specified number of bins  
 plotTracks(list( Itrack, knownGenesMouse, enhancerTrack, MouseARXmotifs, GrangeConservation,  
 dnaseSeqMouse, dnaseSeqMouse2, H3K27acTrack,H3k36meTrack,H3K9me32Track ),  
 from = fromM,   
 to = toM,  
 chromosome= chrM)  
   
 }else if(!(chrM==input$chrM))  
 {  
 ## Mouse Tracks because they're likely necessary as i'll be ultising mouse models  
 fromM <- input$fromM  
 toM<- input$toM  
 chrM<- input$chrM  
 percent<-"100%"  
 setwd("~/Research Proposal/")  
   
 ##Ideogram  
 Itrack<-IdeogramTrack(chromosome = chrM, genome = "mm9")  
   
 ##Genes  
   
 txdbMm9 <- TxDb.Mmusculus.UCSC.mm9.knownGene  
 knownGenesMouse <- GeneRegionTrack(txdbMm9, genome="mm9", chromosome=chrM, showId=TRUE, geneSymbol=TRUE, name="UCSC", stacking = "pack")  
 symbols <- unlist(mapIds(org.Mm.eg.db, gene(knownGenesMouse), "SYMBOL", "ENTREZID", multiVals = "first"))  
 symbol(knownGenesMouse) <- symbols[gene(knownGenesMouse)]  
   
   
 ## Mouse Arx Motifs  
   
 mouseArxPWM <-rbind( A=c(0,1,1,0,0,1,0.25,0.25,0,1,1,0,0,1),  
 C=c(0,0,0,0,0,0,0.25,0.25,0),  
 G=c(0,0,0,0,0,0,0.25,0.25,0) ,  
 T=c(1,0,0,1,1,0,0.25,0.25,1,0,0,1,1,0))  
 grangesMouseArx<- matchPWM(mouseArxPWM, BSgenome.Mmusculus.UCSC.mm9[[chrM]], percent)%>%ranges()%>%as.data.frame()%>%cbind("chromosome"=chrM)  
   
   
 MouseARXmotifs<-AnnotationTrack( makeGRangesFromDataFrame(grangesMouseArx),  
 genome= "mm9",  
 chromosome = chrM,  
 name= "ARX binding sites",  
 stacking = "dense",  
 colour = "red")  
   
 ##cluster Track  
 #clustergrange<-distanceToNearest(grangesMouseArx)  
 #clustergrange<-subset(clustergrange, distance<=200)%>%countRnodeHits()  
 #clustergrange<-subset(grangesMouseArx, clustergrange)  
 #clusterDataTrack<-AnnotationTrack(clustergrange, chromosome = chrM)  
   
   
   
   
 ##DNase seq track  
   
 dnaseSeqMouse <- import(con = "~/DataFiles/DNase/Mouse/cerebrummousednaseseq.bigWig", which =  
 GRanges(chrM, IRanges(1, end = .Machine$integer.max - 1)))%>%   
 DataTrack(type= "smooth", name ="DNase 14.5", color= "green")  
   
 ##Histone modifications  
 H3K27acTrack<- import(con= "~/DataFiles/ChIPseq/Mouse/HistoneModifications/first5col/H3K27ac2.bed")%>%DataTrack(type= "histogram", name= "H3k27ac")  
 H3k36meTrack<- import(con= "~/DataFiles/ChIPseq/Mouse/HistoneModifications/first5col/H3K36me2.bed")%>%DataTrack(type= "histogram", name= "H3k36me")  
 H3K9me32Track<- import(con= "~/DataFiles/ChIPseq/Mouse/HistoneModifications/first5col/H3K9me32.bed")%>%DataTrack(type= "histogram", name= "H3K9me32")  
   
   
 ##enhancer Track  
 enhancerTrack <- import( con = "~/DataFiles/Enhancer Tracks/Mouse/Enhanceresmm9.bed", which =  
 GRanges(chrM, IRanges(1, end = .Machine$integer.max - 1))) %>%  
 AnnotationTrack(name = "Enhancers", stacking= "dense")  
   
   
 ##Conservations!  
   
   
 session <- browserSession()  
 genome(session) <- "mm9"  
 trackNames(session) ## list the track names  
 ## choose the Conservation track for a portion of mm9 chr1  
 query <- ucscTableQuery(session, "Conservation",  
 GRangesForUCSCGenome("mm9", chrM,  
 IRanges(1, .Machine$integer.max - 1)))  
 ## list the table names  
 tableNames(query)  
 ## get the phastCons30way track  
 tableName(query) <- "phastConsElements30way" ## retrieve the track data  
 conservationTrack<-track(query) # a GRanges object  
   
 tableName(query) <- "multiz30way"## get a data.frame summarizing the multiple alignment  
 object<-getTable(query)  
 dataFrameGrange<- as.data.frame(cbind(object[2:4], object[7]))  
 GrangeConservation<- makeGRangesFromDataFrame(dataFrameGrange,  
 keep.extra.columns=TRUE,  
 ignore.strand=FALSE,  
 seqinfo=NULL,  
 seqnames.field=c("seqnames", "seqname",  
 "chromosome", "chrom",  
 "chr", "chromosome\_name",  
 "seqid"),  
 start.field="start",  
 end.field=c("end", "stop"),  
 strand.field="strand",  
 starts.in.df.are.0based=FALSE)%>%  
 DataTrack(type = "heatmap", name = "Conservation Track")  
   
   
   
   
 gtfUCSCexonscoding<-import("~/Scripts/March/FullMm9genome.GTF")  
 gtfUCSCgenes<- import("~/Scripts/March/mm9.bed")  
 methylationGrange<- import(con = "~/DataFiles/Methylation Tracks/Mouse/CpGIslands.bed")  
 promoterGrange<- promoters(gtfUCSCgenes)  
   
   
 # draw the histogram with the specified number of bins  
 plotTracks(list( Itrack, knownGenesMouse, enhancerTrack, MouseARXmotifs, GrangeConservation, dnaseSeqMouse, dnaseSeqMouse2, H3K27acTrack,H3k36meTrack,H3K9me32Track ),  
 from = fromM,   
 to = toM,  
 chromosome= chrM)  
   
 } else if(input$cluster==TRUE)  
 {  
 fromM <- input$fromM  
 toM<- input$toM  
 chrM<- input$chrM  
   
 # draw the histogram with the specified number of bins  
 plotTracks(list( Itrack, knownGenesMouse, enhancerTrack, MouseARXmotifs, GrangeConservation, dnaseSeqMouse, dnaseSeqMouse2, H3K27acTrack,H3k36meTrack,H3K9me32Track ),  
 from = fromM,   
 to = toM,  
 chromosome= chrM)  
 }  
 else{  
 fromM <- input$fromM  
 toM<- input$toM  
 chrM<- input$chrM  
   
 # draw the histogram with the specified number of bins  
 plotTracks(list( Itrack, knownGenesMouse, enhancerTrack, MouseARXmotifs, GrangeConservation, dnaseSeqMouse, dnaseSeqMouse2, H3K27acTrack,H3k36meTrack,H3K9me32Track ),  
 from = fromM,   
 to = toM,  
 chromosome= chrM)  
   
 }  
   
   
 }  
  
)  
})

The first change i made was i removed the variable "match percentage" in the function "matchPWM". This allows me to change accuracy of the Arx motifs in the shiny app, for example if i set the match percentage to 90% i would see all arx motifs and 1 mismatch.

## ChIP-chip

The second thing i have been working on is calling the peaks from the RAW ChIP-chip data to examine the peak hieght etc.

We have downloaded the ChIP-chip peaks from Quille et al (2011)'s Arx experiement and analysized it using Ringo.

I am having a lot of difficulting reading this into R as the header is causing the program to miss align the columns. Furthermore, Ringo does not appear to enjoy using this file. And i have yet to identify why, i need to talkt o jimmy about this.

## ChIP-chip Data workflow thingo magico  
library(Ringo)  
library(magrittr)  
library(limma)  
library(mclust)  
library(ggplot2)  
  
  
  
  
  
arrayfiles <- list.files(path="/home/a1649239/DataFiles/ChIPseq/Mouse/",  
 pattern="txt.gz")  
RG <- read.maimages(arrayfiles[[1]],  
 source="agilent",  
 path="/home/a1649239/DataFiles/ChIPseq/Mouse/")  
  
#par(mar=c(0.01,0.01,0.01,0.01), bg="black")  
#image(RG, 1, channel="red", dim1="Col", dim2="Row",  
# mycols=c("sienna","darkred","orangered"))  
  
#image(RG,arrayno,channel=c("red","green","logratio"),  
# mycols=NULL, mybreaks=NULL, dim1="PositionX", dim2="PositionY",  
# ppch=20, pcex=0.3, verbose=TRUE)  
  
pA <- extractProbeAnno(RG, "agilent", genome="mouse",  
 microarray="Agilent Tiling N2a")  
X <- preprocess(RG[RG$genes$ControlType==0,], method="nimblegen",  
 idColumn="ProbeName")  
  
probeDists <- diff(pA["Y.start"])  
br <- c(0, 100, 200, 300, 500, 1000, 10000, max(probeDists))  
table(cut(probeDists, br))  
  
##Working with my data up to here  
  
smoothX <- computeRunningMedians(X, modColumn="FileName",  
 winHalfSize=500, min.probes=3, probeAnno=pA)  
sampleNames(smoothX) <- paste(sampleNames(X),"smooth",sep=".")  
  
combX <- combine(X, smoothX)  
  
frameData<-as.data.frame(combX@featureData@data$SystematicName)  
  
chromsomeSplit<-data.frame(do.call('rbind',   
 strsplit(as.character(frameData$`combX@featureData@data$SystematicName`),  
 ':',fixed=TRUE)))   
startSplit<-data.frame(do.call('rbind',   
 strsplit(as.character(chromsomeSplit$X2),  
 '-',fixed=TRUE)))   
genes<-RG$genes  
gff<-cbind("name",chromsomeSplit$X1, as.character(startSplit$X1)%>%as.numeric, as.character(startSplit$X2)%>%as.numeric,"\*")%>%as.data.frame  
colnames(gff)<- c("name", "chr", "start", "end", "strand")  
  
ggplot()  
#plot(combX, pA, chr="X",  
 # gff=gff,  
 # maxInterDistance=450, paletteName="Paired")  
  
  
  
y0 <- upperBoundNull(exprs(smoothX))  
y0G <- twoGaussiansNull(exprs(smoothX), max.adj.p=0.01)  
##baseplot  
# hist(exprs(smoothX), n=100000, main=NA,  
# xlab="GSM742106\_US45103054\_251471711563\_S01\_ChIP.smooth")  
# abline(v=y0, col="red", lwd=2)  
# abline(v=y0G, col="blue", lwd=2)  
# legend(x="topright", lwd=2, col=c("red","blue"),  
# legend=c("Non-parametric symmetric Null", "Gaussian Null"))  
#  
  
##Ggplot  
ggplot(as.data.frame(exprs(smoothX)), aes(x=`GSM742100\_US45103054\_251471611506\_S01\_ChIP.txt.smooth`, fill=`GSM742100\_US45103054\_251471611506\_S01\_ChIP.txt.smooth`))+  
 geom\_histogram(bins = 1000)+  
 geom\_vline(xintercept=y0G,show.legend = TRUE)+  
 geom\_vline(xintercept=y0,show.legend = TRUE)+  
 theme\_bw()  
chersX <- findChersOnSmoothed(smoothX, probeAnno=pA, threshold=y0)  
gff$start<- gff$start%>%as.character%>%as.numeric%>%as.data.frame()  
gff$end<- gff$end%>%as.character%>%as.numeric%>%as.data.frame()  
gff$chr<-gff$chr%>%as.character()%>%as.data.frame()  
gff<-gff[gff$end-gff$start>0,]  
  
gff<-na.omit(gff)  
  
  
##converting it to a Grange  
library(GenomicRanges)  
  
arxChiPPeaks<-chersX%>%as.data.frame  
arxChiPPeaks<-cbind(arxChiPPeaks$chr, arxChiPPeaks$start, arxChiPPeaks$end, arxChiPPeaks$maxLevel, arxChiPPeaks$score)%>%as.data.frame   
colnames(arxChiPPeaks)<-c("chromosome", "start", "end", "max level", "score")  
  
  
  
peaks<-arxChiPPeaks%>%na.omit()%>%makeGRangesFromDataFrame(  
 keep.extra.columns=TRUE,  
 ignore.strand=FALSE,  
 seqinfo=NULL,  
 seqnames.field=c("seqnames", "seqname",  
 "chromosome", "chrom",  
 "chr", "chromosome\_name",  
 "seqid"),  
 start.field="start",  
 end.field=c("end", "stop"),  
 strand.field="strand",  
 starts.in.df.are.0based=FALSE)  
seqlevelsStyle(peaks)<-"UCSC"  
##Seeing the overlap of sites containing a 6mer vs not containing a 6mer  
library(Biostrings)  
library(BiocInstaller)  
library(BSgenome.Mmusculus.UCSC.mm9)  
library(pander)  
library(GenomicFeatures)  
library(GenomicRanges)  
arx6Mer <-  
 rbind(  
 A = c(0, 1, 1, 0, 0, 1),  
 C = c(0, 0, 0, 0, 0, 0),  
 G = c(0, 0, 0, 0, 0, 0) ,  
 T = c(1, 0, 0, 1, 1, 0)  
 )  
  
arx6MerTFBS<-matchPWM(arx6Mer, BSgenome.Mmusculus.UCSC.mm9, "100%")  
  
length(subset(x = peaks,findOverlaps(arx6MerTFBS, peaks)%>%countRnodeHits()))

## Conducting denovo motif searching on these ChIP-chip peaks

Here we have read in the DNA-sequences from our ChIP-chip peaks and added a ">" to ensure that they can be read by weeder to allow for easier analysis of these regions.

Edit 30/10/2017: we noticed we did not increase the size of the probe so we likely missed a lot of important genomic information

library(readr)  
  
read.csvFunction<- function(x){ read\_delim(x, "\t", escape\_double = FALSE, na = "NA", trim\_ws = TRUE, skip = 1,col\_names = FALSE)}  
  
files <- list.files(path="~/WEEDER/",  
 pattern="file.txt")  
PWM<-lapply(files, read.csvFunction)  
asNumericFunction<-function(x){apply(x, 2, as.numeric)}  
PWM<-lapply(PWM, asNumericFunction)  
roundFunction<-function(x){round(100\*x)}  
PFM<-lapply(PWM, roundFunction)  
  
  
for(i in 1:9){  
 names(PFM)[[i]] <- paste0(">",i)  
 i<-i+1  
}  
##Removve the Column of NAs  
subsetter<-function(x){x[,1:(dim(x)[2]-1)]}  
  
PFM<-lapply(PFM, subsetter)  
  
PFM<-lapply(PFM, t)  
PFM<- lapply(PFM, as.data.frame)  
i<-1  
for(i in 1:9){  
 colnames(PFM[[i]])<-NULL   
i<-i+1  
}  
  
  
i<-1  
for(i in 1:9){  
 rownames(PFM[[i]])<-NULL  
 i<-i+1  
}  
  
  
## exporting the data  
  
sink("my-example.txt")  
PFM  
sink()

```