```
#This file contains small scripts required to run SNPscript.R  I have found many of these functions are also useful for other analyses.
#This script was written in the R programming language to determine intervals of significant SNP density based on a bootstrap method.
#Required input for the script is a list of SNPs with chromosomal positions.  Output of the script is similar to Figure 2 and Figure 3.
#Created on 04/06/10 by Andrew Severin andrewseverin@gmail.com
#Iowa State University


############################################################
#This function will give the index number in a matrix given the rowname.
IndexFromGeneCall<-function(inputmatrix,rowname){
      match(rowname,rownames(inputmatrix),nomatch=0)
}


############################################################
#plotting function
#this function will generate intervals of significant clustering of SNPs on soybean chromosomes scaled to the longest chromosome.

ChromosomePlot<-function(chromosomelength,intervalstart,intervalend,intervalheight,maxNumGenesInCluster,maxchromosomelength,binScales,currentBinScale,SNPcoords){

            #The axis labels require resizing depending on the width of the plot
            #the following is an estimate of the resizing required
                  if (maxchromosomelength<750000){
                        XcexVar<-1
                  }
                  if (maxchromosomelength>750000 & maxchromosomelength<12000000){
                        XcexVar<-(-0.3125)*log(maxchromosomelength/1000000)+0.91
                  }
                  if (maxchromosomelength>12000000){
                        XcexVar<-.1
                  }



            #I make use of the rect function that has input as (xleft, ybottom, xright, ytop)
            #keep in mind everything is scaled to Gm18, the largest chromosome
            xleft=(1/maxchromosomelength)*intervalstart
            ybottom=0
            width=(intervalend-intervalstart)/maxchromosomelength
            averagepos<-(intervalend+intervalstart)/(2*maxchromosomelength)

            average<-(intervalend+intervalstart)/(2)
            xright=xleft+width
            ytop=intervalheight/maxNumGenesInCluster
            rect(0,0,chromosomelength/maxchromosomelength,-0.04)                                                    #this draws the chromosome on the bottom
            rect(SNPcoords/maxchromosomelength,0,SNPcoords/maxchromosomelength ,-0.04,lwd=.1)                       #this draws the location of each SNP
            rect(xleft,ybottom,xright,ytop,border="black",col=rainbow(length(binscales))[which(binscales==currentBinScale)])   #Significant Intervals of SNPs
            text(chromosomelength/(2*maxchromosomelength),-.02,labels=paste("chromosome",i),cex=.5)                 #chromosome name
            text(SNPcoords/maxchromosomelength-2000/maxchromosomelength,-.02,labels=rownames(SNPcoords),cex=XcexVar,srt=90)   #location of each Interval
            axis(1,tick=T,at=intervalstart/maxchromosomelength,labels=intervalstart,cex.axis=XcexVar,las = 2,lwd=.5)  #axis 1
            axis(2,tick=T,at=seq(0,1,1/maxNumGenesInCluster),labels=0:maxNumGenesInCluster,cex.axis=.8)             #axis 2
```

```
        ablineMulti<-function(i){abline(i,0,col="darkgrey",lwd=.1)}                    #Creates a grid at 2 SNP intervals
        sapply(seq(0,1,2/maxNumGenesInCluster),ablineMulti)                            #sapply to create the grid
        }

##########################################################

#this function is not used in the SNPscript but is a handy little function.
#Used in a similar script for clustering genes.
identifyGenesOnChromosomeForSoybean<-function(GeneList){
        #this loop identifies all gene model names (glymas) on a specified chromosome
        if (i<10){
        glymas<-GeneList[grep(paste("0",i,"g",sep=""),GeneList)]
        }else{
        glymas<-GeneList[grep(paste(i,"g",sep=""),GeneList)]
        }
        return(glymas)
}

identifySNPSOnChromosomeForSoybean<-function(snpList,chromosomeNumber){
                #this loop identifies all SNPs on a specified chromosome
                if (chromosomeNumber<10){
                snps<-snpList[grep(paste("Gm0",chromosomeNumber,sep=""),rownames(snpList)),]
                }else{
                snps<-snpList[grep(paste("Gm",chromosomeNumber,sep=""),rownames(snpList)),]
                }
                return(snps) #this will return the snp matrix that corresponds only to the chromosome of interest
    }


##########################################################


#this section will take the same number of genes and simulate how the genes will fall into the bins based on the 1000(or numofsims) random collections of genes
    simulateData<-function(numofsims,AllGeneCalls,geneCoordinates,SNPS,chromosomeNumber){
                #matrix for storing simulations
                    genesAll<-identifyGenesOnChromosomeForSoybean(AllGeneCalls)
                    genesSample<-sample(genesAll,dim(SNPS)[1]*numofsims,replace=T)
                    AllindexCoords<-function(i){IndexFromGeneCall(geneCoordinates,genesSample[i])}
                    sampleIndex<-sapply(1:length(genesSample),AllindexCoords)
                    genesSample<-matrix(genesSample,ncol=dim(SNPS)[1])
                    sampleIndex<-matrix(sampleIndex,ncol=dim(SNPS)[1])
                    return(list(genesSample=genesSample,sampleIndex=sampleIndex))
    }

##########################################################
#Boostrap function for clustering on a chromosome

#generation of the bin sizes across the chromosome
clusterByBoostrap<-function(chromosomelength,binsize,geneIndexValues,SNPCoordinates,AllGeneCalls,numofsims,bootData){
    print(SNPCoordinates)
    numBins<-floor(chromosomelength/binsize)
```

```
#this section will calcluate how many of of the SNPs we are interested in fall into each bin
	breaks1<-seq(0,chromosomelength,binsize)
	chromBinsFind<-findInterval(SNPCoordinates,breaks1, rightmost.closed=T)

	chromBins<-hist(chromBinsFind,breaks=seq(0,length(breaks1),1),plot=F)$counts
	print(chromBins)
	sampleIndex<-bootData$sampleIndex                                        #actual data

	chromBinsSample<-matrix(0,numofsims,length(breaks1))                     #simulated data
		for (j in 1:numofsims){
		#generation of the bin sizes across the chromosome for the simulated data
			breaks1<-seq(0,chromosomelength,binsize)
			chromBinsSam<-findInterval(geneCoordinatesAve[sampleIndex[j,]],breaks1, rightmost.closed=T)
			chromBinsSample[j,]<-hist(chromBinsSam,breaks=seq(0,length(breaks1),1),plot=F)$counts
		}
		#Average and standard deviation of the simulated data
		chrombinsAve<-colMeans(chromBinsSample)
		chrombinsSD<-sd(chromBinsSample)


#this section is the determination of the bins that are significant
		over3stdev<-which((chromBins-(chrombinsAve+3*chrombinsSD))>0)
		over3stdevBy<-(chromBins-(chrombinsAve+3*chrombinsSD))[over3stdev]
		over3stdevZscore<-round(((chromBins-(chrombinsAve))[over3stdev])/chrombinsSD[over3stdev],2)

		#this if statement is required in case no intervals are found to be significant

		if (length(over3stdev)==0){
			print("No significant intervals found")
			return(0)
		}else{
		significantIntervals<-matrix(c(breaks1[over3stdev],breaks1[over3stdev]+binsize),length(over3stdev),2)
		}

		#append number of genes in the bin and the zscore to significantIntervals
		significantIntervals<-matrix(cbind(significantIntervals,chromBins[over3stdev],over3stdevZscore),ncol=4)
		significantIntervals<-matrix(significantIntervals[sort(significantIntervals[,1],index.return=T)$ix,],ncol=4)
		significantIntervals<-matrix(significantIntervals[which(significantIntervals[,3]>1),],ncol=4)

		print(significantIntervals)
		if(dim(significantIntervals)[1]==0){
		return(0)}else{
		return(significantIntervals)
		}
}
```

```
#This script is used to cluster SNPs onto Soybean chromosomes.  Requires SNPsource.R and .RDataSNP
#This script was written in the R programming language to determine intervals of significant SNP density based on a bootstrap method.
#Required input for the script is a list of SNPs with chromosomal positions.  Output of the script is similar to Figure 2 and Figure 3.
#Created on 04/06/10 by Andrew Severin andrewseverin@gmail.com
#Iowa State University


#required libraries
	library(gplots)
	source('SNPsource.R')
	load(".RDataSNP")

#starting parameters
	dir<-"./"
	numofsims<-3
	SNPsofInterest<-read.table('./exampleSNPsFile.txt')		#list with SNPs of interest
	numberOfChromosome<-20 #this variable will allow you to loop through the first X chromosomes (see for loop below)

#this section is optional if you would like to have multiple bin sizes uncomment
	#StartingBinsize<-6000000 #important the the vector in this forloop results in binsizes that include the binsizes before it
	#binscales<-c(1,2,6,12,60,120) #for binsize 6M 3M 1M 500K 100K 50k
#For multiple bin sizes comment out this block of code
	StartingBinsize<-500000 #Here I chose just one binsize
	binscales<-c(1)


#variables calculated from the input parameters
	geneCoordinatesAve<-matrix(round((geneCoordinates[,3]+geneCoordinates[,4])/2),ncol=1)
	rownames(geneCoordinatesAve)<-rownames(geneCoordinates)
	chromosomelengthAll<-chrom[,4]
	maxchromosomelength<-max(chrom[,4])
	significantIntervalsOrig<-0


#this for loop will cycle through each chromosomes.
	for (i in 1:numberOfChromosome){
	#for (i in numberOfChromosome:numberOfChromosome){		#This line can be uncommented if you want to run it on a specific chromosome
		dir.create(paste("./",i,sep=""))		#create directory to export outfiles
		chromosomelength<-chromosomelengthAll[i]

	maxNumGenesInCluster<-0 #initiate a variable that will be needed later for plotting
	SNPs<-identifySNPSOnChromosomeForSoybean(SNPsofInterest,i)	#this function will identify the SNPs on each chromosome as it goes through the loop

	if (dim(SNPs)[1]<3){ #No need to look at chromosomes that do not have at least 3 SNPs
	print(SNPs)
```

```
        next()
        }

        bootData<-simulateData(numofsims,AllGeneCalls,geneCoordinates,SNPs,i)              #generate the simulated data  (See SNPsource for code)



    #binSize (for loop) will cycle through the binsizes determined above
        for (b in binscales){

            binsize<-StartingBinsize/b
            print(binsize)
            appendtofilename<-paste("_",binsize/binscales,sep="")                          #this variable is used for the outputfiles to distinguish between bins
            SNPcoords<-SNPs[,1]                                                                    #for retrieval of the coordinates of the SNPs of interest

    #function to do bootstrap method
            significantIntervals<-clusterByBoostrap(chromosomelength,binsize,geneCoordinatesAve,SNPcoords,AllGeneCalls,numofsims,bootData)
            print(significantIntervals)

            if (b==binscales[1]){
            #open a pdf file
            pdf(file=paste(i,"/chrom",i,"ALL",".pdf",sep=""),paper="special",height=7,width=100)
            plot(0:1, 0:1, type="n", axes=FALSE, ann=FALSE)
            }
    #if there are no significant Intervals go to the next binsize in the loop

            if(significantIntervals==0){
                next
            }
    #This block estimates the required Y dimension for plotting and works for most cases
                if (maxNumGenesInCluster==0){
                    maxNumGenesInCluster<-max(significantIntervals[,3])+1
                }


    #required input variables for the plotting function.  ChromosomePlot can be found in SNPsource.
                intervalstart<-significantIntervals[,1]
                intervalend<-significantIntervals[,2]
                intervalheight<-significantIntervals[,3]
                currentBinScale<-b
                ChromosomePlot(chromosomelength,intervalstart,intervalend,intervalheight,maxNumGenesInCluster,maxchromosomelength, binScales, currentBinScale,SNPcoords)

                if(b==binscales[length(binscales)]){
                #now that the plotting is finished, close the pdf file
                dev.off()
                }
```

```
            #write to file gene lists with intervals that are significant
            colnames(significantIntervals)<-c('intervalstart','intervalend','numberInInterval','ZscoreaboveBootstrap')
            write.table(significantIntervals,file=paste(i,"/clusterTable",i,appendtofilename,".txt",sep=""),append=T,quote=F,col.names=T)

    }

    #this commands save the R sessions for each chromosome into each chromosome folder respectively.
    save.image(file = paste(i,"/.RData",i,sep=""))

}
```

#the two columns are identical required to read in as a table in the script
position position
Gm01_53617124 53617124 53617124
Gm02_5687687 5687687 5687687
Gm02_42350182 42350182 42350182
Gm03_36460374 36460374 36460374
Gm03_36554101 36554101 36554101
Gm03_36559857 36559857 36559857
Gm03_36559926 36559926 36559926
Gm03_36560002 36560002 36560002
Gm03_36952394 36952394 36952394
Gm03_36959955 36959955 36959955
Gm03_36996777 36996777 36996777
Gm03_36997184 36997184 36997184
Gm03_36997185 36997185 36997185
Gm03_37024958 37024958 37024958
Gm03_37027198 37027198 37027198
Gm03_37144398 37144398 37144398
Gm03_37165409 37165409 37165409
Gm03_37827399 37827399 37827399
Gm03_37828684 37828684 37828684
Gm03_37828791 37828791 37828791
Gm03_37832228 37832228 37832228
Gm03_37863675 37863675 37863675
Gm03_38065066 38065066 38065066
Gm03_38083417 38083417 38083417
Gm03_38117453 38117453 38117453
Gm03_38117485 38117485 38117485
Gm03_38132996 38132996 38132996
Gm03_38136913 38136913 38136913
Gm03_38170431 38170431 38170431
Gm03_38173719 38173719 38173719
Gm03_38173815 38173815 38173815
Gm03_38174150 38174150 38174150
Gm03_38174157 38174157 38174157
Gm03_38718231 38718231 38718231
Gm03_38718672 38718672 38718672
Gm03_38942920 38942920 38942920
Gm03_38942927 38942927 38942927
Gm03_39788794 39788794 39788794
Gm03_39788799 39788799 39788799
Gm03_39790874 39790874 39790874

Gm03_39795164 39795164 39795164
Gm03_39795203 39795203 39795203
Gm03_39964048 39964048 39964048
Gm03_39966061 39966061 39966061
Gm03_39967974 39967974 39967974
Gm03_39986148 39986148 39986148
Gm03_39993121 39993121 39993121
Gm03_40055679 40055679 40055679
Gm03_40056070 40056070 40056070
Gm03_40131229 40131229 40131229
Gm03_40131350 40131350 40131350
Gm03_40132046 40132046 40132046
Gm03_40154304 40154304 40154304
Gm03_40154315 40154315 40154315
Gm03_40160427 40160427 40160427
Gm03_40172340 40172340 40172340
Gm03_40179701 40179701 40179701
Gm03_40211426 40211426 40211426
Gm03_40371846 40371846 40371846
Gm03_40459321 40459321 40459321
Gm03_40462434 40462434 40462434
Gm03_40462640 40462640 40462640
Gm03_40585266 40585266 40585266
Gm03_40585499 40585499 40585499
Gm03_40586108 40586108 40586108
Gm03_40587775 40587775 40587775
Gm03_40600203 40600203 40600203
Gm03_40600256 40600256 40600256
Gm03_40603941 40603941 40603941
Gm03_40628097 40628097 40628097
Gm03_40656449 40656449 40656449
Gm03_40676583 40676583 40676583
Gm03_40676856 40676856 40676856
Gm03_40678154 40678154 40678154
Gm03_40683015 40683015 40683015
Gm03_40685721 40685721 40685721
Gm03_40785291 40785291 40785291
Gm03_40785299 40785299 40785299
Gm03_40823593 40823593 40823593
Gm03_40874888 40874888 40874888
Gm03_40886333 40886333 40886333
Gm03_40889026 40889026 40889026
Gm03_40906651 40906651 40906651

Gm03_41007691 41007691 41007691
Gm03_41007937 41007937 41007937
Gm03_41008255 41008255 41008255
Gm03_41008442 41008442 41008442
Gm03_41008459 41008459 41008459
Gm03_41171468 41171468 41171468
Gm03_41185505 41185505 41185505
Gm03_41223681 41223681 41223681
Gm03_41228284 41228284 41228284
Gm03_41228321 41228321 41228321
Gm03_41234338 41234338 41234338
Gm03_41274006 41274006 41274006
Gm03_41275788 41275788 41275788
Gm03_41982791 41982791 41982791
Gm03_42142835 42142835 42142835
Gm03_42179147 42179147 42179147
Gm03_42224757 42224757 42224757
Gm03_42224778 42224778 42224778
Gm03_42243466 42243466 42243466
Gm03_42243699 42243699 42243699
Gm03_42672131 42672131 42672131
Gm03_45026222 45026222 45026222
Gm03_45416367 45416367 45416367
Gm03_45503654 45503654 45503654
Gm03_45516926 45516926 45516926
Gm04_44787569 44787569 44787569
Gm04_45061509 45061509 45061509
Gm04_45090419 45090419 45090419
Gm04_45152573 45152573 45152573
Gm04_45157974 45157974 45157974
Gm04_45380790 45380790 45380790
Gm04_45594453 45594453 45594453
Gm05_38251772 38251772 38251772
Gm05_38278658 38278658 38278658
Gm05_38279366 38279366 38279366
Gm05_38280387 38280387 38280387
Gm05_38337295 38337295 38337295
Gm05_38402327 38402327 38402327
Gm05_38432746 38432746 38432746
Gm05_38433427 38433427 38433427
Gm05_38433580 38433580 38433580
Gm05_38589019 38589019 38589019
Gm05_38596037 38596037 38596037

Gm05_38913395 38913395 38913395
Gm05_38913666 38913666 38913666
Gm05_39082457 39082457 39082457
Gm05_39082469 39082469 39082469
Gm06_5721969 5721969 5721969
Gm06_5721970 5721970 5721970
Gm08_3461787 3461787 3461787
Gm09_6112681 6112681 6112681
Gm10_38131569 38131569 38131569
Gm10_38131571 38131571 38131571
Gm10_47783606 47783606 47783606
Gm12_3862466 3862466 3862466
Gm12_38459068 38459068 38459068
Gm13_35524268 35524268 35524268
Gm13_35617464 35617464 35617464
Gm13_35823484 35823484 35823484
Gm13_35823512 35823512 35823512
Gm13_35823533 35823533 35823533
Gm13_35823811 35823811 35823811
Gm13_35835159 35835159 35835159
Gm13_35862124 35862124 35862124
Gm13_35862205 35862205 35862205
Gm13_39517326 39517326 39517326
Gm14_28088505 28088505 28088505
Gm15_3100509 3100509 3100509
Gm16_30464934 30464934 30464934
Gm16_30479527 30479527 30479527
Gm16_30539483 30539483 30539483
Gm16_30539518 30539518 30539518
Gm16_30539519 30539519 30539519
Gm16_30626524 30626524 30626524
Gm16_31191863 31191863 31191863
Gm16_31204160 31204160 31204160
Gm16_31204451 31204451 31204451
Gm16_31225684 31225684 31225684
Gm16_31461554 31461554 31461554
Gm16_31475163 31475163 31475163
Gm16_31476359 31476359 31476359
Gm16_31827645 31827645 31827645
Gm16_31827884 31827884 31827884
Gm16_31827991 31827991 31827991
Gm16_31828137 31828137 31828137
Gm16_31829738 31829738 31829738

Gm16_31840753 31840753 31840753
Gm16_31840819 31840819 31840819
Gm16_31842815 31842815 31842815