Human Brain HiC

Awais Choudhry

25/10/2017

### Dump the fasta Files from the .SRA file

#!/bin/bash  
#SBATCH -p batch # partition (this is the queue your job will be added to)   
#SBATCH -N 1 # number of nodes (use a single node)  
#SBATCH -n 8 # number of cores (sequential job uses 1 core)  
#SBATCH --time=05:00:00 # time allocation, which has the format (D-HH:MM:SS), here set to 1 hour  
#SBATCH --mem=52GB # memory pool for all cores (here set to 4 GB)  
  
# Executing script (Example here is sequential script)  
module load SRA-Toolkit/2.7.0-centos\_linux64  
  
  
fastq-dump --split-files /fast/users/a1649239/BrainHiCHomoSapien/SRR2106512.sra  
fastq-dump --split-files /fast/users/a1649239/BrainHiCHomoSapien/SRR2106513.sra

## Build the bowtie 2 index for the Hg19 genome

This script is downloaded of the offical bowtie2 webpage and run on phoenix web server.

#!/bin/bash  
#SBATCH -p batch # partition (this is the queue your job will be added to)   
#SBATCH -N 1 # number of nodes (use a single node)  
#SBATCH -n 8 # number of cores (sequential job uses 1 core)  
#SBATCH --time=05:00:00 # time allocation, which has the format (D-HH:MM:SS), here set to 1 hour  
#SBATCH --mem=50GB # memory pool for all cores (here set to 4 GB)  
  
# Executing script (Example here is sequential script)  
module load SRA-Toolkit/2.7.0-centos\_linux64  
module load Bowtie2/2.2.9-foss-2016b  
  
  
#  
# Downloads sequence for the hg19 version of H. spiens (human) from  
# UCSC.  
#  
# Note that UCSC's hg19 build has three categories of compressed fasta  
# files:  
#  
# 1. The base files, named chr??.fa.gz  
# 2. The unplaced-sequence files, named chr??\_gl??????\_random.fa.gz  
# 3. The alternative-haplotype files, named chr??\_?????\_hap?.fa.gz  
#  
# By default, this script builds and index for just the base files,  
# since alignments to those sequences are the most useful. To change  
# which categories are built by this script, edit the CHRS\_TO\_INDEX  
# variable below.  
#  
  
  
BASE\_CHRS="\  
chr1 \  
chr2 \  
chr3 \  
chr4 \  
chr5 \  
chr6 \  
chr7 \  
chr8 \  
chr9 \  
chr10 \  
chr11 \  
chr12 \  
chr13 \  
chr14 \  
chr15 \  
chr16 \  
chr17 \  
chr18 \  
chr19 \  
chr20 \  
chr21 \  
chr22 \  
chrX \  
chrY \  
chrM"  
  
RANDOM\_CHRS="\  
chr1\_gl000191\_random \  
chr1\_gl000192\_random \  
chr4\_gl000193\_random \  
chr4\_gl000194\_random \  
chr7\_gl000195\_random \  
chr8\_gl000196\_random \  
chr8\_gl000197\_random \  
chr9\_gl000198\_random \  
chr9\_gl000199\_random \  
chr9\_gl000200\_random \  
chr9\_gl000201\_random \  
chr11\_gl000202\_random \  
chr17\_gl000203\_random \  
chr17\_gl000204\_random \  
chr17\_gl000205\_random \  
chr17\_gl000206\_random \  
chr18\_gl000207\_random \  
chr19\_gl000208\_random \  
chr19\_gl000209\_random \  
chr21\_gl000210\_random \  
chrUn\_gl000211 \  
chrUn\_gl000212 \  
chrUn\_gl000213 \  
chrUn\_gl000214 \  
chrUn\_gl000215 \  
chrUn\_gl000216 \  
chrUn\_gl000217 \  
chrUn\_gl000218 \  
chrUn\_gl000219 \  
chrUn\_gl000220 \  
chrUn\_gl000221 \  
chrUn\_gl000222 \  
chrUn\_gl000223 \  
chrUn\_gl000224 \  
chrUn\_gl000225 \  
chrUn\_gl000226 \  
chrUn\_gl000227 \  
chrUn\_gl000228 \  
chrUn\_gl000229 \  
chrUn\_gl000230 \  
chrUn\_gl000231 \  
chrUn\_gl000232 \  
chrUn\_gl000233 \  
chrUn\_gl000234 \  
chrUn\_gl000235 \  
chrUn\_gl000236 \  
chrUn\_gl000237 \  
chrUn\_gl000238 \  
chrUn\_gl000239 \  
chrUn\_gl000240 \  
chrUn\_gl000241 \  
chrUn\_gl000242 \  
chrUn\_gl000243 \  
chrUn\_gl000244 \  
chrUn\_gl000245 \  
chrUn\_gl000246 \  
chrUn\_gl000247 \  
chrUn\_gl000248 \  
chrUn\_gl000249"  
  
ALT\_HAP\_CHRS="\  
chr4\_ctg9\_hap1 \  
chr6\_apd\_hap1 \  
chr6\_cox\_hap2 \  
chr6\_dbb\_hap3 \  
chr6\_mann\_hap4 \  
chr6\_mcf\_hap5 \  
chr6\_qbl\_hap6 \  
chr6\_ssto\_hap7 \  
chr17\_ctg5\_hap1"  
  
CHRS\_TO\_INDEX="$BASE\_CHRS $RANDOM\_CHRS"  
  
UCSC\_HG19\_BASE=ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes  
  
get() {  
 file=$1  
 if ! wget --version >/dev/null 2>/dev/null ; then  
 if ! curl --version >/dev/null 2>/dev/null ; then  
 echo "Please install wget or curl somewhere in your PATH"  
 exit 1  
 fi  
 curl -o `basename $1` $1  
 return $?  
 else  
 wget $1  
 return $?  
 fi  
}  
  
BOWTIE\_BUILD\_EXE=./bowtie2-build  
if [ ! -x "$BOWTIE\_BUILD\_EXE" ] ; then  
 if ! which bowtie2-build ; then  
 echo "Could not find bowtie2-build in current directory or in PATH"  
 exit 1  
 else  
 BOWTIE\_BUILD\_EXE=`which bowtie2-build`  
 fi  
fi  
  
INPUTS=  
for c in $CHRS\_TO\_INDEX ; do  
 if [ ! -f ${c}.fa ] ; then  
 F=${c}.fa.gz  
 get ${UCSC\_HG19\_BASE}/$F || (echo "Error getting $F" && exit 1)  
 gunzip $F || (echo "Error unzipping $F" && exit 1)  
 fi  
 [ -n "$INPUTS" ] && INPUTS=$INPUTS,${c}.fa  
 [ -z "$INPUTS" ] && INPUTS=${c}.fa  
done  
  
CMD="${BOWTIE\_BUILD\_EXE} $\* ${INPUTS} hg19"  
echo Running $CMD  
if $CMD ; then  
 echo "hg19 index built; you may remove fasta files"  
else  
 echo "Index building failed; see error message"  
fi

## Run Hi-C Pro pipleline

To run the Hi-C pro pipeline we ran the following script linking to our bowtie 2 index in the config file

#!/bin/bash  
#SBATCH -p batch # partition (this is the queue your job will be added to)  
#SBATCH -N 1 # number of nodes (use a single node)  
#SBATCH -n 16 # number of cores (sequential job uses 1 core)  
#SBATCH --time=18:00:00 # time allocation, which has the format (D-HH:MM:SS), here set to 1 hour  
#SBATCH --mem=64GB # memory pool for all cores (here set to 4 GB)  
  
# Executing script (Example here is sequential script)  
module load HiC-Pro/2.8.1\_devel-foss-2016uofa  
module load Python/2.7.11-foss-2016uofa  
  
HiC-Pro -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt -i /home/a1649239/fastdir/BrainHiCHomoSapien/Input2 -o /home/a1649239/fastdir/BrainHiCHomoSapien/OutputRepilcate2

# Hi-C Pro Human Config file

the 2 noteable changes is changing the bin size to 5kb

#########################################################################  
## Paths and Settings - Do not edit !  
#########################################################################  
  
TMP\_DIR = tmp  
LOGS\_DIR = logs  
BOWTIE2\_OUTPUT\_DIR = bowtie\_results  
MAPC\_OUTPUT = hic\_results  
RAW\_DIR = rawdata  
  
#######################################################################  
## SYSTEM AND SCHEDULER - Start Editing Here !!  
#######################################################################  
N\_CPU = 16  
LOGFILE = hicpro.log  
  
JOB\_NAME =HumanBrainHiCPro   
JOB\_MEM = 64GB  
JOB\_WALLTIME =   
JOB\_QUEUE =   
JOB\_MAIL =a1649239@student.adleaide.edu.au   
  
#########################################################################  
## Data  
#########################################################################  
  
PAIR1\_EXT = \_1  
PAIR2\_EXT = \_2  
  
#######################################################################  
## Alignment options  
#######################################################################  
  
FORMAT = phred33  
MIN\_MAPQ = 0  
  
BOWTIE2\_IDX\_PATH = /fast/users/a1649239/BowTieIndex/Hg19  
  
BOWTIE2\_GLOBAL\_OPTIONS = --very-sensitive -L 30 --score-min L,-0.6,-0.2 --end-to-end --reorder  
BOWTIE2\_LOCAL\_OPTIONS = --very-sensitive -L 20 --score-min L,-0.6,-0.2 --end-to-end --reorder  
  
#######################################################################  
## Annotation files  
#######################################################################  
  
REFERENCE\_GENOME = hg19  
GENOME\_SIZE = /data/biohub/Refs/human/hg19\_GRCh37d5/hg19.sizes  
  
#######################################################################  
## Allele specific analysis  
#######################################################################  
  
#######################################################################  
## Digestion Hi-C  
#######################################################################  
  
GENOME\_FRAGMENT =/fast/users/a1649239/BrainHiCHomoSapien/HindIII\_resfrag\_hg19.bed  
LIGATION\_SITE = AAGCTAGCTT  
MIN\_FRAG\_SIZE =100   
MAX\_FRAG\_SIZE =1000000  
MIN\_INSERT\_SIZE =100  
MAX\_INSERT\_SIZE =600  
  
#######################################################################  
## Hi-C processing  
#######################################################################  
  
MIN\_CIS\_DIST =  
GET\_ALL\_INTERACTION\_CLASSES = 1  
GET\_PROCESS\_SAM = 0  
RM\_SINGLETON = 1  
RM\_MULTI = 1  
RM\_DUP = 1  
  
#######################################################################  
## Contact Maps  
#######################################################################  
  
BIN\_SIZE = 20000 40000 150000 1000000 5000  
MATRIX\_FORMAT = complete  
  
#######################################################################  
## Normalization  
#######################################################################  
MAX\_ITER = 100  
FILTER\_LOW\_COUNT\_PERC = 0.02  
FILTER\_HIGH\_COUNT\_PERC = 0  
EPS = 0.1

# Slurm output from running the Hi-C pro script

Run HiC-Pro 2.8.1\_devel  
--------------------------------------------  
Wed Aug 16 16:20:05 ACST 2017  
Bowtie2 alignment step1 ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/bowtie\_wrap.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt -u >> hicpro.log  
--------------------------------------------  
Wed Aug 16 19:14:03 ACST 2017  
Bowtie2 alignment step2 ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/bowtie\_wrap.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt -l >> hicpro.log  
--------------------------------------------  
Wed Aug 16 20:05:27 ACST 2017  
Combine both alignment ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/bowtie\_combine.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt >> hicpro.log  
[bam\_sort\_core] merging from 80 files...  
[bam\_sort\_core] merging from 80 files...  
--------------------------------------------  
Wed Aug 16 20:28:19 ACST 2017  
Bowtie2 mapping statistics for R1 and R2 tags ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/mapping\_stat.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt >> hicpro.log  
--------------------------------------------  
Wed Aug 16 20:36:59 ACST 2017  
Pairing of R1 and R2 tags ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/bowtie\_pairing.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt >> hicpro.log  
--------------------------------------------  
Wed Aug 16 21:45:01 ACST 2017  
Assign alignments to restriction fragments ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/mapped\_2hic\_fragments.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt >> hicpro.log  
--------------------------------------------  
Wed Aug 16 22:32:31 ACST 2017  
Merge multiple files from the same sample ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/merge\_valid\_interactions.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt >> hicpro.log  
--------------------------------------------  
Wed Aug 16 22:37:09 ACST 2017  
Merge stat files per sample ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/merge\_stats.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt >> hicpro.log  
--------------------------------------------  
Wed Aug 16 22:37:10 ACST 2017  
Run quality checks for all samples ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/make\_plots.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt -p "all" >> hicpro.log  
--------------------------------------------  
Wed Aug 16 22:37:20 ACST 2017  
Generate binned matrix files ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/build\_raw\_maps.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt  
--------------------------------------------  
Wed Aug 16 22:49:20 ACST 2017  
Run ICE Normalization ...  
/apps/software/HiC-Pro/2.8.1\_devel-foss-2016uofa/scripts/ice\_norm.sh -c /home/a1649239/fastdir/BrainHiCHomoSapien/config-hicpro.txt >> hicpro.log   
  
===========================================================================  
Phoenix Job Utilisation Reporting  
===========================================================================  
Job Name : HicProTAD.sh  
Job ID : 3660946  
User : a1649239  
Account : biosci  
Cluster : phoenix  
Partition : cpu  
Nodes (List) : 1 (r1n05)  
Cores : 16  
GPUs : 0  
State : COMPLETED  
Submit : 2017-08-16T15:35:13  
Start : 2017-08-16T16:20:03  
End : 2017-08-16T23:04:00  
Walltime reserved : 18:00:00  
Walltime elapsed (%): 06:43:57 (37.4% \* reserved)  
CPU-time elapsed : 107.72 core-hours  
% CPU used (Total) : 60.88% (65.58 core-hours)  
% User CPU (Compute): 55.71% (60.01 core-hours)  
% System CPU (I/O) : 5.17% (5.57 core-hours)  
Mem reserved : 64G/node  
% Mem used (Max) : 45.84% (29.33G/node)   
Max Disk Write : 677.01G (r1n05)  
Max Disk Read : 946.87G (r1n05)  
===========================================================================

## Converting the Hi-C Pro matrices to FitHiC format

We downloaded the hicproToFitHiC.py script off of the Hi-CPro github page owned by nicholar servant.

#! /usr/bin/env python  
  
import argparse  
import math  
import os  
import gzip  
  
# Created by Arya Kaul - 1/12/2017  
# Modified by Ferhat Ay - 1/13/2017  
# Modified by Nicolas Servant - 1/23/2017  
# Modified by Ferhat Ay - 6/5/2017 - added resolution (-r <res>) argument to avoid some problems with inferring it from the first entry of bedFile  
# Modified by Ferhat Ay - 6/7/2017 - hitCount to ints, and gzip output   
  
  
def outputfithicform(bedPath, matrixPath, intCPath, fragMapPath, biasVectorPath=None, biasVectorOutput=None,res=0):  
 print "Loading matrix file..."  
 fragDic = {}  
 # resolution of data to be determined if res=0 at this point  
 with open(bedPath, 'r') as bedFile:  
 for lines in bedFile:  
 line = lines.rstrip().split()  
 chrNum = line[0]  
 start = line[1]  
 en = line[2]  
 if res==0: res=int(en)-int(start) # edge case but if first fragment in file is smaller than res, there is a problem  
 mid = int(start)+ int(res/2)  
 index = int(line[3])  
 fragDic[index] = [chrNum, start, mid, 0] # last field is total contact count: tcc  
  
 lineCount=0  
 with open(matrixPath, 'r') as matrixFile:  
 with gzip.open(intCPath, 'w') as interactionCountsFile:  
 for lines in matrixFile:  
 line = lines.rstrip().split()  
 i = int(line[0])  
 j = int(line[1])  
 cc = float(line[2]) # this can be float or int  
 fragDic[i][3] += cc  
 fragDic[j][3] += cc  
 if cc==int(cc): cc=int(cc) # make sure to convert to integer if it ends with ".0"  
 interactionCountsFile.write(str(fragDic[i][0])+'\t'+str(fragDic[i][2])+'\t'+\  
 str(fragDic[j][0])+'\t'+str(fragDic[j][2])+'\t'+str(cc)+"\n")  
 lineCount+=1  
 if lineCount%1000000==0: print "%d million lines read" % int(lineCount/1000000)  
  
 with gzip.open(fragMapPath, 'w') as fragmentMappabilityFile:  
 for indices in sorted(fragDic): # sorted so that we start with the smallest index  
 toWrite = 0  
 if fragDic[indices][3]> 0: toWrite=1   
 fragmentMappabilityFile.write(str(fragDic[indices][0])+'\t'+str(fragDic[indices][1])+'\t'+\  
 str(fragDic[indices][2])+'\t'+str(int(fragDic[indices][3]))+'\t'+str(toWrite)+'\n')  
  
 if biasVectorPath is not None and biasVectorOutput is not None:  
 print "Converting bias file..."  
  
 biasVec=[0,0] # bias sum and biasCount  
 biasDic={} # bias for each index  
 i=1 # one-based indices  
 with open(biasVectorPath, 'r') as biasVectorFile:  
 for lines in biasVectorFile:  
 value = float(lines.rstrip()) #just one entry that can be nan or a float  
 index = int(i)  
 i+=1  
 biasDic[index]=value  
 if not math.isnan(value):  
 biasVec[0]+=value #sum  
 biasVec[1]+=1 # count  
 #  
  
 # Centering the bias values on 1.  
 biasAvg=biasVec[0]/biasVec[1]  
  
 with gzip.open(biasVectorOutput, 'w') as biasVectorOutputFile:  
 for index in sorted(biasDic):  
 value=biasDic[index]  
 if not math.isnan(value):  
 value=value/biasAvg  
 else:   
 value=-1  
 biasVectorOutputFile.write(str(fragDic[index][0])+'\t'+str(fragDic[index][2])+'\t'+str(value)+'\n')  
 print "Conversion from HiC-Pro to Fit-Hi-C format completed"  
  
#outputfithicform(args.bedPath, args.matrixPath, args.intCPath, args.fragMapPath, args.biasVectorPathandOutput[0], args.biasVectorPathandOutput[1])  
  
def main():  
 # Example without bias files  
 outputfithicform('raw/1000000/hIMR90\_HindIII\_r1\_1000000\_abs.bed', 'raw/1000000/hIMR90\_HindIII\_r1\_1000000.matrix', 'fithic.interactionCounts', 'fithic.fragmentMappability')  
 # Example with bias files  
 outputfithicform('raw/1000000/hIMR90\_HindIII\_r1\_1000000\_abs.bed', 'raw/1000000/hIMR90\_HindIII\_r1\_1000000.matrix', 'fithic.interactionCounts', 'fithic.fragmentMappability','hicpro.biases','fithic.biases')  
  
if \_\_name\_\_=="\_\_main\_\_":  
  
 parser = argparse.ArgumentParser()  
 parser.add\_argument("-i", "--matrix", help="Input matrix file with raw contact frequencies.", required=True)  
 parser.add\_argument("-b", "--bed", help="BED file with bins coordinates.", required=True)  
 parser.add\_argument("-s", "--bias", help="The bias file provided after IC normalization.", default=None)  
 parser.add\_argument("-o", "--output", help="Output path", default="./")  
 parser.add\_argument("-r", "--resolution", help="Resolution of the matrix", type=int, default=0) # 0 means it is inferred from fragments file's first entry  
  
 args = parser.parse\_args()  
  
 icounts\_output = os.path.join(args.output + "/fithic.interactionCounts.gz")  
 fragmap\_output = os.path.join(args.output + "/fithic.fragmentMappability.gz")  
 bias\_output = None  
  
 if args.bias is not None:  
 bias\_output = os.path.join(args.output + "/fithic.biases.gz")  
  
 outputfithicform(args.bed, args.matrix, icounts\_output, fragmap\_output, args.bias, bias\_output, args.resolution)

## Converting it to our page

This script will 1. read in the interactions as a datatable from the fitHiC format 2. subset the table for all significant interactions. 3. Convert to bedpe format and write it to a table 4. Import the Bedpe Table to make an Genomic Interaction Object 5. Import enhancer information for the human genome and indentify which enhancers have accessible/Active ARX motifs in them. 6. Identify which of these are interactings with promoter regions using our Genomic INteraction object 7. Adjust the seqinfo and seqlevels for each GRange to be the same so that the ggbio package can align across different levels correctly 8. Make Circos plot of these intearctions

Additonally, it will identify all significant interactions occuring between ALL human enhancers and promoter regions.

##  
library(readr)  
library(rtracklayer)  
library(dplyr)  
library(Biostrings)  
library(ggbio)  
library(GenomicInteractions)  
  
  
  
### HiC Circular Plot Generation   
  
#Read in significant itneractions from Spline 2  
 spline\_pass2\_significances <- read\_delim("/media/awais/NewDrivewho/Phoenix/HiC/Human Brain/OutPut/.spline\_pass2.significances.txt.gz",   
 "\t", escape\_double = FALSE, trim\_ws = TRUE)  
  
#Subset Significant Interactions  
SignificantValues<-filter(spline\_pass2\_significances, q\_value<=0.05)  
  
  
## Convert back to bedpe format   
cbind.data.frame(SignificantValues$chr1,   
 SignificantValues$fragmentMid1-2500,  
 SignificantValues$fragmentMid1+2500,  
 SignificantValues$chr2,  
 SignificantValues$fragmentMid2-2500,  
 SignificantValues$fragmentMid2+2500,  
 "Interaction",  
 SignificantValues$contactCount,  
 1)%>%write.table(file = "/media/awais/NewDrivewho/Phoenix/HiC/Human Brain/OutPut/SignificantHumanInteractions.bedpe",append = FALSE,   
 quote=FALSE, col.names=FALSE,  
 row.names= FALSE, sep= "\t")  
  
  
#Import BedBPe file  
SignificantInteractions<-makeGenomicInteractionsFromFile(fn = "/media/awais/NewDrivewho/Phoenix/HiC/Human Brain/OutPut/SignificantHumanInteractions.bedpe",  
 type= "bedpe", experiment\_name = "NL CL Repilicate 2")  
  
  
#Add metadata column: Q value  
mcols(SignificantInteractions)<-cbind.data.frame("counts"=SignificantInteractions$counts  
 , SignificantValues$q\_value)  
  
  
# Data imports  
humanEnhancers<-import("/media/awais/NewDrivewho/Downloads/human\_permissive\_enhancers\_phase\_1\_and\_2.bed.gz")  
humangenes<-import("/media/awais/NewDrivewho/Downloads/hg.bed.gz")  
humanPromoters<-promoters(humangenes)  
  
  
## Identify actie motifs  
arx6merTFBS<-readRDS("/media/awais/NewDrivewho/Downloads/ARX6merHg19Sites")  
brainMale<-import("/media/awais/NewDrivewho/Downloads/E081\_15\_coreMarks\_mnemonics.bed.gz")  
  
  
activeMotifs<-subset(brainMale, name %in% c( "5\_TxWk",   
 "4\_Tx",  
 "1\_TssA",   
 "3\_TxFlnk",  
 "2\_TssAFlnk",   
 "7\_Enh",  
 "6\_EnhG",  
 "12\_EnhBiv",   
 "11\_BivFlnk",   
 "10\_TssBiv" ))%>%subsetByOverlaps(arx6merTFBS, .)  
  
#Now identify which active motifs are in enhancer regions  
ArxContainingEhancers<-subsetByOverlaps(humanEnhancers, activeMotifs)  
  
  
  
## Rename a col to id for Annotation  
colnames(mcols(ArxContainingEhancers))<-c("id", "score", "itemRgB", "thick", "blocks")  
colnames(mcols(humanPromoters))<-c("id", "score", "itemRgb", "thick", "blocks" )  
  
##Annotate the BedPe File  
AnnotatingList<-c(promoters=GRanges(humanPromoters),  
 enhancers=ArxContainingEhancers)  
annotateInteractions(SignificantInteractions,AnnotatingList )  
  
## Subset for interactioms between enhancer promoter  
PromoterEnhancerInteractions<-SignificantInteractions[isInteractionType(SignificantInteractions, x = "promoters", "enhancers")]  
  
### Save for use future Use on VM  
saveRDS(PromoterEnhancerInteractions, "/media/awais/NewDrivewho/Phoenix/HiC/Human Brain/StasticallySignificantHg19BetweenPromoterEnhancer")  
  
  
  
  
  
  
##################################333  
## Make Circos Plots On Computer  
###################################  
  
#Function to convert bedpe to Granges that are linked (therefore Anchor 1= GRange 1, anchor 2 = meta data for Grange 1 as a Grange)  
InteractionToGenomicRanges<-function(x){  
 Test<-x%>%as.data.frame()  
 Test1<-cbind(Test$seqnames1%>%as.character(), Test$start1, Test$end1, Test$strand1%>%as.character)%>%as.data.frame()  
 Test2<-cbind(Test$seqnames2%>%as.character(), Test$start2, Test$end2, Test$strand2%>%as.character)%>%as.data.frame()  
 colnames(Test1)<- c("chromosome",   
 "start",  
 "end",  
 "strand")  
 colnames(Test2)<- c("chromosome",   
 "start",  
 "end",  
 "strand")  
   
 Grange1<-makeGRangesFromDataFrame(Test1)  
 Grange2<-makeGRangesFromDataFrame(Test2)  
   
 seqlevelsStyle(Grange1)<-"ucsc"  
 seqlevelsStyle(Grange2)<-"ucsc"  
 mcols(Grange2)<-Test$counts  
 mcols(Grange1)<-Grange2  
 return(Grange1)  
}  
  
  
# Use the function   
PromoterEnhancerGrange<-InteractionToGenomicRanges(PromoterEnhancerInteractions)  
  
  
## Remove meta data from these guys because ggbio doesn't like it for some reason ?  
InteractingPromoters<-reduce(subsetByOverlaps(humanPromoters, PromoterEnhancerInteractions))  
InteractingEnhancers<-reduce(subsetByOverlaps(humanEnhancers, PromoterEnhancerInteractions))  
  
# We need to normalize all seq levels  
library(BSgenome.Hsapiens.UCSC.hg19)  
  
  
#gET sEQINFO OBJECT  
seqinfoObject<-seqinfo(BSgenome.Hsapiens.UCSC.hg19)  
  
  
#Subset Seqinfo Object  
seqlevels(seqinfoObject)<-seqlevels(PromoterEnhancerGrange)  
  
  
##Apply Seq info to all other Granges including meta data of the linked Grange  
seqinfo(PromoterEnhancerGrange)<-seqinfoObject  
seqinfo(PromoterEnhancerGrange$X)<-seqinfoObject  
  
seqlevels(InteractingPromoters)<-seqlevels(seqinfoObject)  
seqinfo(InteractingPromoters)<-seqinfoObject  
  
seqlevels(InteractingEnhancers, force=TRUE)<-seqlevels(seqinfoObject)  
seqinfo(InteractingEnhancers)<-seqinfoObject  
  
## Make le plot  
ggbio()+circle(PromoterEnhancerGrange, geom="link", linked.to="X", aes(color=seqnames), radius=45)+  
 circle(reduce(InteractingEnhancers), geom="rect", aes(color=seqnames), color="#F78DC9", radius = 50)+  
 circle(reduce(InteractingPromoters), geom="rect", aes(color=seqnames), color="steelblue", radius = 55)+  
 circle(PromoterEnhancerGrange, geom= "ideogram", aes(fill=seqnames))+  
 circle(PromoterEnhancerGrange, geom= "scale", aes(color=seqnames))+  
 guides(fill=guide\_legend(ncol = 2), color=FALSE)  
  
  
  
## Rename a col to id for Annotation  
colnames(mcols(humanEnhancers))<-c("id", "score", "itemRgB", "thick", "blocks")  
colnames(mcols(humanPromoters))<-c("id", "score", "itemRgb", "thick", "blocks" )  
  
##Annotate the BedPe File  
AnnotatingList<-c(promoters=GRanges(humanPromoters),  
 enhancers=humanEnhancers)  
annotateInteractions(SignificantInteractions,AnnotatingList )  
  
## Subset for interactioms between enhancer promoter  
PromoterEnhancerInteractions<-SignificantInteractions[isInteractionType(SignificantInteractions, x = "promoters", "enhancers")]  
  
### Save for use future Use on VM  
saveRDS(PromoterEnhancerInteractions, "/media/awais/NewDrivewho/Phoenix/HiC/Human Brain/StasticallySignificantHg19BetweenPromoterEnhancer")  
  
  
  
  
  
  
#####################################  
## Make Circos Plots On Computer for ALl Interactions  
#####################################  
  
#Function to convert bedpe to Granges that are linked (therefore Anchor 1= GRange 1, anchor 2 = meta data for Grange 1 as a Grange)  
InteractionToGenomicRanges<-function(x){  
 Test<-x%>%as.data.frame()  
 Test1<-cbind(Test$seqnames1%>%as.character(), Test$start1, Test$end1, Test$strand1%>%as.character)%>%as.data.frame()  
 Test2<-cbind(Test$seqnames2%>%as.character(), Test$start2, Test$end2, Test$strand2%>%as.character)%>%as.data.frame()  
 colnames(Test1)<- c("chromosome",   
 "start",  
 "end",  
 "strand")  
 colnames(Test2)<- c("chromosome",   
 "start",  
 "end",  
 "strand")  
   
 Grange1<-makeGRangesFromDataFrame(Test1)  
 Grange2<-makeGRangesFromDataFrame(Test2)  
   
 seqlevelsStyle(Grange1)<-"ucsc"  
 seqlevelsStyle(Grange2)<-"ucsc"  
 mcols(Grange2)<-Test$counts  
 mcols(Grange1)<-Grange2  
 return(Grange1)  
}  
  
  
# Use the function   
PromoterEnhancerGrange<-InteractionToGenomicRanges(PromoterEnhancerInteractions)  
  
  
## Remove meta data from these guys because ggbio doesn't like it for some reason ?  
InteractingPromoters<-reduce(subsetByOverlaps(humanPromoters, PromoterEnhancerInteractions))  
InteractingEnhancers<-reduce(subsetByOverlaps(humanEnhancers, PromoterEnhancerInteractions))  
  
# We need to normalize all seq levels  
library(BSgenome.Hsapiens.UCSC.hg19)  
  
  
#gET sEQINFO OBJECT  
seqinfoObject<-seqinfo(BSgenome.Hsapiens.UCSC.hg19)  
  
  
#Subset Seqinfo Object  
seqlevels(seqinfoObject)<-seqlevels(PromoterEnhancerGrange)  
  
  
##Apply Seq info to all other Granges including meta data of the linked Grange  
seqinfo(PromoterEnhancerGrange)<-seqinfoObject  
seqinfo(PromoterEnhancerGrange$X)<-seqinfoObject  
  
seqlevels(InteractingPromoters)<-seqlevels(seqinfoObject)  
seqinfo(InteractingPromoters)<-seqinfoObject  
  
seqlevels(InteractingEnhancers, force=TRUE)<-seqlevels(seqinfoObject)  
seqinfo(InteractingEnhancers)<-seqinfoObject  
  
## Make le plot  
ggbio()+circle(PromoterEnhancerGrange, geom="link", linked.to="X", aes(color=seqnames), radius=45)+  
 circle(reduce(InteractingEnhancers), geom="rect", aes(color=seqnames), color="#F78DC9", radius = 50)+  
 circle(reduce(InteractingPromoters), geom="rect", aes(color=seqnames), color="steelblue", radius = 55)+  
 circle(PromoterEnhancerGrange, geom= "ideogram", aes(fill=seqnames))+  
 circle(PromoterEnhancerGrange, geom= "scale", aes(color=seqnames))+  
 guides(fill=guide\_legend(ncol = 2), color=FALSE)