PIPELINE FOR DATA RETRIEVAL, FILTERING AND PROCESSING

```
################################################################################
## Pipeline of PiRNA cluster extraction from GRCh37 genome version of 1000GP ##
################################################################################

(0) Preliminary steps:

If you do not have administrator privileges, do the following:

mkdir ~/local

Go to your .bashrc profile and add the following lines:

nano ~/.bashrc

PATH=~/Scripts:$PATH # Only to run our personalized scripts

PATH=~/local/bin:$PATH

export X # We can add any environment variables required, such as BCFTOOLS_PLUGINS.

(1) Installation of key features:

|A| Bcftools: This pipeline uses the experimental version because it has a feature to
replace ./. with 0/0 in merge.

# Install the regular version [Not necessary]

wget https://github.com/samtools/htslib/releases/download/1.3/bcftools-1.3.tar.bz2
tar -xvf bcftools-1.3.tar.bz2

cd bcftools-1.3

make

make prefix=~/local install

# Install the experimental version:

git clone --branch=develop --recursive git://github.com/pd3/bcftools.git

cd bcftools; make



################################################################
## PIRNA CLUSTER EXTRACTIONS FROM GENOTYPES OF 1KGP, GRCh37 ##
################################################################

#(1) Extract all fragments from each chromosome and index them with TABIX, storing in
the corresponding to each chromosome directory

CHRS=`seq 1 22`

for CHR in $CHRS; do

  mkdir -p chr${CHR}
```

```
  POSITIONS=`cat cluster_positions_GRCh37/Human_clusters_hg19_CHR${CHR}.bed | sed
"s/^chr\(\S\+\)\t\(\S\+\)\t\(\S\+\)/\1:\2-\3/g" | tr "\n" " "`

  tabix -f -h
ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/ALL.chr${CHR}.phase3_shapeit2_
mvncall_integrated_v5a.20130502.genotypes.vcf.gz ${POSITIONS} >chr${CHR}/chr${CHR}.vcf

  bcftools view -O -S ^InbredIndividuals.txt chr${CHR}/chr${CHR}.vcf >
chr${CHR}/chr${CHR}_filtered.vcf

  vcf-sort chr${CHR}/chr${CHR}_filtered.vcf > chr${CHR}/chr${CHR}_sort.vcf

  bgzip chr${CHR}/chr${CHR}_sort.vcf

  tabix -p vcf chr${CHR}/chr${CHR}_sort.vcf.gz

done


############################################################
## PIRNA CLUSTER EXTRACTIONS FROM CHIMPANZEE ALIGNMENT ##
############################################################

#(1) Download MFA files from Vista Browser, fasta reference files from UCSC browser,
convert human-chimp alingment into vcf file with MFAtoVCF.py script

mkdir -p Outgroup_GRCh37

CHRS=`seq 1 22`

for CHR in $CHRS; do

    mkdir -p chr${CHR}

    wget http://pipeline.lbl.gov/data/hg19_panTro4/chr${CHR}.mfa.gz >
chr${CHR}/chr${CHR}.mfa.gz

       gunzip chr${CHR}/chr${CHR}.mfa.gz

    wget http://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/chr${CHR}.fa.gz >
chr${CHR}/chr${CHR}.fa.gz

    gunzip chr${CHR}/chr${CHR}.fa.gz

       grep -v "score" chr${CHR}/chr${CHR}.mfa > chr${CHR}/chr${CHR}.wscore.mfa

    /home/path/to/soft/Python-3.5.0/python MFAtoVCF.py -s chr${CHR}/chr${CHR}.fa -q
Chimp -c ${CHR} chr${CHR}/chr${CHR}.wscore.mfa

done

############################
#the same for chromosome X##

mkdir -p chrX

    wget http://pipeline.lbl.gov/data/hg19_panTro4/chrX.mfa.gz > chrX/chrX.mfa.gz

       gunzip chrX/chrX.mfa.gz

    wget http://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/chrX.fa.gz >
chrX/chrX.fa.gz

    gunzip chrX/chrX.fa.gz
```

```
      grep -v "score" chrX/chrX.mfa > chrX/chrX.wscore.mfa

    /home/path/to/soft/Python-3.5.0/python chrX/MFAtoVCF.py -s chrX/chrX.fa -q Chimp -c
X chrX/chrX.wscore.mfa


#(2) Extract all fragments from each chromosome and index them with TABIX, storing in
the corresponding to each chromosome directory

CHRS=`seq 1 22`

for CHR in $CHRS; do

  mkdir -p chr${CHR}

  POSITIONS=`cat cluster_positions_GRCh37/Human_clusters_hg19_CHR${CHR}.bed | sed
"s/^chr\(\S\+\)\t\(\S\+\)\t\(\S\+\)/\1:\2-\3/g" | tr "\n" " "`

  tabix -f -h chr${CHR}/chr${CHR}.vcf.gz ${POSITIONS} > chr${CHR}/chr${CHR}_chimp.vcf

  vcf-sort chr${CHR}/chr${CHR}_chimp.vcf > chr${CHR}/chr${CHR}_chimp_sort.vcf

  bgzip chr${CHR}/chr${CHR}_chimp_sort.vcf

  tabix -p vcf chr${CHR}/chr${CHR}_chimp_sort.vcf.gz

done

#Test on chromosome X

  POSITIONS=`cat cluster_positions_GRCh37/Human_clusters_hg19_CHRX.bed | sed
"s/^chr\(\S\+\)\t\(\S\+\)\t\(\S\+\)/\1:\2-\3/g" | tr "\n" " "`

  tabix -f -h chrX/chrX.vcf.gz ${POSITIONS} > chrX/chrX_chimp.vcf

  vcf-sort chrX/chrX_chimp.vcf > chrX/chrX_chimp_sort.vcf

  bgzip chrX/chrX_chimp_sort.vcf

  tabix -p vcf chrX/chrX_chimp_sort.vcf.gz


#########################################################
## MERGE VCF.GZ FILES OF HUMAN GENOMES WITH OUTGROUP ##
#########################################################

mkdir -p Merged_chromosomes

CHRS=`seq 1 22`

for CHR in $CHRS; do

$bcf/bcftools merge  -Oz --missing-to-ref path/to/human/chr${CHR}_sort.vcf.gz
path/to/chimp/chr${CHR}_chimp_sort.vcf.gz >
path/to/Merged_chromosomes/chr${CHR}_merged.vcf.gz

tabix -p vcf path/to/Merged_chromosomes/chr${CHR}_merged.vcf.gz;

done

#Merge chrX for test
```

```
$bcf/bcftools merge  -Oz --missing-to-ref path/to/human/chrX_sort.vcf.gz
path/to/chimp/chrX_chimp.vcf.gz > path/to/Merged_chromosomes/chrX_merged.vcf.gz

tabix -p vcf path/to/Merged_chromosomes/chrX_merged.vcf.gz

################################################
## EXTRACTION OF INTERGENIC REGIONS FOR  MKT ##
################################################

#(1) Obtaining the coordinates of intergenic regions

#Download the GENCODE annotations for GRCh37/hg19 genome reference version

wget
ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_human/release_25/GRCh37_mapping/gencode.v25li
ft37.annotation.gtf.gz

#To define intronic regions, we need to define the gene i.e. obtain the gene coordinates

zcat gencode.v25lift37.annotation.gtf.gz | awk 'BEGIN{OFS="\t";} $3=="gene" {print
$1,$4,$5}' > gencode_GRCh37_gene.bed

/home/path/to/software/bedtools/bin/sortBed -i gencode_GRCh37_gene.bed >
gencode_GRCh37_gene_temp.bed

mv -f gencode_GRCh37_gene_temp.bed gencode_GRCh37_gene.bed

#And finally to define intergenic regions, we use complementBed to find regions not
covered by genes.

#To create a hg19_chrom_info.txt file, use the fetchChromSizes executable available at
http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/fetchChromSizes

#to create the hg19_chrom_info.txt file for hg19.

wget http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/fetchChromSizes

chmod +x fetchChromSizes

./fetchChromSizes hg19 > hg19_chrom_info.txt

/home/rpath/to/software/bedtools/bin/complementBed -i gencode_GRCh37_gene.bed -g
hg19_chrom_info.txt > gencode_GRCh37_intergenic.bed

#############################################################################################
#To extract defined manually intergenic regions to clusters,
follow the same steps as for extraction of piRNA clusters
```

```
#####################################################################
##Defining overlapping regions with genes, pseudogenes and polyAs##
#####################################################################

/home/path/to/software/bedtools/bin/intersectBed -a Clusters_GRCh37.bed -b
gencode_GRCh37_gene.bed -wo > clusters_overlap_genes.bed

zcat gencode.v25.2wayconspseudos.gtf.gz | awk 'BEGIN{OFS="\t";} $3=="transcript" {print
$1,$4,$5}' > gencode_GRCh37_pseudogenes.bed

/home/path/to/software/bedtools/bin/intersectBed -a Clusters_GRCh37.bed -b
gencode_GRCh37_pseudogenes.bed -wo > clusters_overlap_pseudogenes.bed

olga@olga-HP-Compaq-8200-Elite-MT-PC:~/Downloads$ zcat gencode.v25.polyAs.gtf.gz | awk
'BEGIN{OFS="\t";} {print $1,$4,$5}' > gencode_GRCh37_polyAs.bed
```

```
odolgova@andromeda:~/Files$ /home/path/to/software/bedtools/bin/intersectBed -a
Clusters_GRCh37.bed -b gencode_GRCh37_polyAs.bed -wo > clusters_overlap_polyAs.bed

#Concatenate the intergenic regions in BED file#

odolgova@andromeda:~/Files/Closest_intergenic_regions/Human_intergenic/Intergenic_region
s_hg19$ cat Intergenic_hg19_CHR*.bed | sort -k 1,1 -k2,2n > Intergenic_hg19_all.bed


###############Definining overlapped with intergenic regions#######################

/home/raquel/pj15011/software/bedtools/bin/intersectBed -a Intergenic_hg19_all.bed -b
gencode_GRCh37_pseudogenes.bed -wo > intergenic_overlap_pseudogenes.bed

/home/raquel/pj15011/software/bedtools/bin/intersectBed -a Intergenic_hg19_all.bed -b
gencode_GRCh37_polyAs.bed -wo > intergenic_overlap_polyAs.bed

zcat gencode.v25.tRNAs.gtf.gz | awk 'BEGIN{OFS="\t";} {print $1,$4,$5}' >
gencode_GRCh37_tRNAs.bed

/home/raquel/pj15011/software/bedtools/bin/intersectBed -a Intergenic_hg19_all.bed -b
gencode_GRCh37_tRNAs.bed -wo > intergenic_overlap_tRNAs.bed

#############################################################################################
##Discarding the regions overlapping with genes, pseudogenes and polyAs from BED files#
#############################################################################################

odolgova@andromeda:~/Files/Merged_GRCh37/overlapping$ cat intergenic_overlap_*.bed |
sort  -k 1,1 -k2,2n > intergenic_overlap_all.bed

odolgova@andromeda:~/Files/Merged_GRCh37/overlapping$ cat clusters_overlap_*.bed | sort
-k 1,1 -k2,2n > clusters_overlap_all.bed

#################################################################
##Substraction of the gene/pseudogene regions from BED files##
#################################################################

/home/path/to/software/bedtools/bin/subtractBed -a Clusters_GRCh37.bed -b
clusters_overlap.bed SV_map.bed > new_clusters.bed

/home/path/to/software/bedtools/bin/subtractBed -a Intergenic_hg19_all.bed -b
intergenic_overlap.bed SV_map.bed > new_intergenics.bed

#################################################################
##Extraction of superpopulations from chromosomic vcf files##
#################################################################

#For clusters

CHRS=`seq 1 22`

SETS=`seq 1 5`

for CHR in $CHRS; do

echo "starting chr${CHR}"

 rm chr${CHR}.vcf.gz.tbi

 gunzip chr${CHR}.vcf.gz

for SET in $SETS; do
```

```
echo "starting population ${SET}"
 vcf-subset -c ${SET}_individuals.txt chr${CHR}.vcf > chr${CHR}_${SET}.vcf
bgzip chr${CHR}_${SET}.vcf
tabix -p vcf chr${CHR}_${SET}.vcf.gz
done
bgzip chr${CHR}.vcf
tabix -p vcf chr${CHR}.vcf.gz
done
#test on X chromosome
SETS=`seq 1 5`
rm chrX.vcf.gz.tbi
gunzip chrX.vcf.gz
for SET in $SETS; do
vcf-subset -c ${SET}_individuals.txt chrX.vcf > chrX_${SET}.vcf
bgzip chrX_${SET}.vcf
tabix -p vcf chrX_${SET}.vcf.gz
done
bgzip chrX.vcf
tabix -p vcf chrX.vcf.gz
vcf-subset -c 5_individuals.txt chrX.vcf > chrX_5.vcf
bgzip chrX_5.vcf
bgzip chrX.vcf
tabix -p vcf chrX_5.vcf.gz
tabix -p vcf chrX.vcf.gz

#For intergenic regions
CHRS=`seq 1 22`
SETS=`seq 1 5`
for CHR in $CHRS; do
echo "starting chr${CHR}"
rm chr${CHR}_intergenic.vcf.gz.tbi
gunzip chr${CHR}_intergenic.vcf.gz
for SET in $SETS; do
```

```
echo "starting population ${SET}"

vcf-subset -c ${SET}_individuals.txt chr${CHR}_intergenic.vcf >
chr${CHR}_${SET}_intergenic.vcf

bgzip chr${CHR}_${SET}_intergenic.vcf

tabix -p vcf chr${CHR}_${SET}_intergenic.vcf.gz

done

bgzip chr${CHR}_intergenic.vcf

tabix -p vcf chr${CHR}_intergenic.vcf.gz

done

#test on X chromosome

SETS=`seq 1 5`

rm chrX_intergenic.vcf.gz.tbi

gunzip chrX_intergenic.vcf.gz

for SET in $SETS; do

vcf-subset -c ${SET}_individuals.txt chrX_intergenic.vcf > chrX_${SET}_intergenic.vcf

bgzip chrX_${SET}_intergenic.vcf

tabix -p vcf chrX_${SET}_intergenic.vcf.gz

done

bgzip chrX_intergenic.vcf

tabix -p vcf chrX_intergenic.vcf.gz


######################################################################
##Substraction of the gene/pseudogene regions for each population##
######################################################################

/home/path/to/software/bedtools/bin/subtractBed -a Clusters_GRCh37.bed -b
clusters_overlap.bed SV_map.bed > new_clusters.bed

/home/path/to/software/bedtools/bin/subtractBed -a Intergenic_hg19_all.bed -b
intergenic_overlap.bed SV_map.bed > new_intergenics.bed

/home/path/to/software/bedtools/bin/subtractBed -a Clusters_GRCh37.bed -b
clusters_overlap.bed SV_map.bed -wao > old_new_clusters.bed

/home/path/to/software/bedtools/bin/subtractBed -a Intergenic_hg19_all.bed -b
intergenic_overlap.bed SV_map.bed -wao > old_new_intergenics.bed

#For clusters

CHRS=`seq 1 22`

SETS=`seq 1 5`

for CHR in $CHRS; do
```

```
echo "starting chr${CHR}"

 rm chr${CHR}.vcf.gz.tbi

 gunzip chr${CHR}.vcf.gz

/home/path/to/software/bedtools/bin/subtractBed -header -a chr${CHR}.vcf -b
clusters_overlap.bed SV_map.bed > new_chr${CHR}.vcf

bgzip new_chr${CHR}.vcf

tabix -p vcf new_chr${CHR}.vcf.gz

for SET in $SETS; do

echo "starting chr${CHR} population ${SET}"

rm chr${CHR}_${SET}.vcf.gz.tbi

gunzip chr${CHR}_${SET}.vcf.gz

/home/path/to/software/bedtools/bin/subtractBed -header -a chr${CHR}_${SET}.vcf -b
clusters_overlap.bed SV_map.bed > new_chr${CHR}_${SET}.vcf

bgzip new_chr${CHR}_${SET}.vcf

tabix -p vcf new_chr${CHR}_${SET}.vcf.gz

done

done

#test on X chromosome

SETS=`seq 1 5`

rm chrX.vcf.gz.tbi

gunzip chrX.vcf.gz

/home/path/to/software/bedtools/bin/subtractBed -header -a chrX.vcf -b
clusters_overlap.bed SV_map.bed > new_chrX.vcf

bgzip new_chrX.vcf

tabix -p vcf new_chrX.vcf.gz

for SET in $SETS; do

echo "starting population ${SET}"

rm chrX_${SET}.vcf.gz.tbi

gunzip chrX_${SET}.vcf.gz

/home/path/to/software/bedtools/bin/subtractBed -header -a chrX_${SET}.vcf -b
clusters_overlap.bed SV_map.bed > new_chrX_${SET}.vcf

bgzip new_chrX_${SET}.vcf

tabix -p vcf new_chrX_${SET}.vcf.gz

done
```

```
#For intergenic regions

CHRS=`seq 1 22`

SETS=`seq 1 5`

for CHR in $CHRS; do

echo "starting chr${CHR}"

rm chr${CHR}_intergenic.vcf.gz.tbi

gunzip chr${CHR}_intergenic.vcf.gz

/home/path/to/software/bedtools/bin/subtractBed -header -a chr${CHR}_intergenic.vcf -b
intergenic_overlap.bed SV_map.bed > new_chr${CHR}_intergenic.vcf

bgzip new_chr${CHR}_intergenic.vcf

tabix -p vcf new_chr${CHR}_intergenic.vcf.gz

for SET in $SETS; do

echo "starting chr${CHR} population ${SET}"

rm chr${CHR}_${SET}_intergenic.vcf.gz.tbi

gunzip chr${CHR}_${SET}_intergenic.vcf.gz

/home/path/to/software/bedtools/bin/subtractBed -header -a
chr${CHR}_${SET}_intergenic.vcf -b intergenic_overlap.bed SV_map.bed >
new_chr${CHR}_${SET}_intergenic.vcf

bgzip new_chr${CHR}_${SET}_intergenic.vcf

tabix -p vcf new_chr${CHR}_${SET}_intergenic.vcf.gz

done

done

#test on X chromosome

rm chrX_intergenic.vcf.gz.tbi

gunzip chrX_intergenic.vcf.gz

/home/path/to/software/bedtools/bin/subtractBed -header -a chrX_intergenic.vcf -b
intergenic_overlap.bed SV_map.bed > new_chrX_intergenic.vcf

bgzip new_chrX_intergenic.vcf

tabix -p vcf new_chrX_intergenic.vcf.gz

SETS=`seq 1 5`

for SET in $SETS; do

echo "starting population ${SET}"

rm chrX_${SET}.vcf_intergenic.gz.tbi

gunzip chrX_${SET}_intergenic.vcf.gz
```

```
/home/path/to/software/bedtools/bin/subtractBed -header -a chrX_${SET}_intergenic.vcf -b
intergenic_overlap.bed SV_map.bed > new_chrX_${SET}_intergenic.vcf

bgzip new_chrX_${SET}_intergenic.vcf

tabix -p vcf new_chrX_${SET}_intergenic.vcf.gz

done
```

EXAMPLE OF AN R SCRIPT FOR EACH CHROMOSOME (HERE: CHROMOSOME 18)

```
setwd("~/Path/to/vcf.gz.files/clusters")

library (PopGenome)

Clusters.all <- readVCF("new_chr18.vcf.gz", numcols=10000, tid="18",
 from=11633196, to=11698981, include.unknown = TRUE)
Clusters.all <- set.outgroup(Clusters.all,c("Chimp"), diploid=TRUE)
get.sum.data(Clusters.all)

#Neutrality statistics
Clusters.all <- neutrality.stats(Clusters.all, do.R2 = TRUE)
get.neutrality(Clusters.all, theta=TRUE) [[1]]
Mu <- (Clusters.all@theta_Watterson/40357)/40000
Mu

#Diversities
Clusters.all <- diversity.stats(Clusters.all, pi=TRUE)
get.diversity(Clusters.all) [[1]]

#Divergence calculation
bial <- get.biallelic.matrix(Clusters.all,1) # Biallelic matrix
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0 # Sites polymorphic in humans:
 non REF (0) alleles
div <- bial[n+1,] == 1
divsites <- sum (div & !polym,na.rm=T)
divsites
D <- divsites/40357
D

#3. Establishing populations (only females)
AFR <- c("HG01880", "HG01883", "HG01886", "HG01889", "HG01894",
 "HG01896", "HG01956", "HG01958", "HG01985", "HG01989", "HG02010",
 "HG02012", "HG02052", "HG02054", "HG02095", "HG02108", "HG02111",
 "HG02144", "HG02256", "HG02282", "HG02308", "HG02309", "HG02315",
 "HG02318", "HG02322", "HG02325", "HG02337", "HG02339", "HG02419",
 "HG02427", "HG02450", "HG02462", "HG02465", "HG02476", "HG02477",
 "HG02479", "HG02485", "HG02497", "HG02502", "HG02505", "HG02508",
 "HG02511", "HG02537", "HG02546", "HG02549", "HG02555", "HG02558",
 "HG02562", "HG02568", "HG02571", "HG02574", "HG02577", "HG02580",
 "HG02583", "HG02586", "HG02589", "HG02595", "HG02611", "HG02614",
 "HG02621", "HG02629", "HG02635", "HG02646", "HG02667", "HG02676",
 "HG02679", "HG02703", "HG02716", "HG02722", "HG02757", "HG02760",
 "HG02763", "HG02769", "HG02772", "HG02799", "HG02808", "HG02811",
 "HG02814", "HG02817", "HG02820", "HG02837", "HG02840", "HG02855",
 "HG02861", "HG02870", "HG02879", "HG02882", "HG02885", "HG02888",
 "HG02891", "HG02922", "HG02943", "HG02946", "HG02952", "HG02970",
 "HG02974", "HG02976", "HG02979", "HG02983", "HG03025", "HG03028",
 "HG03040", "HG03049", "HG03052", "HG03055", "HG03058", "HG03061",
 "HG03064", "HG03073", "HG03079", "HG03082", "HG03086", "HG03088",
 "HG03091", "HG03095", "HG03099", "HG03105", "HG03108", "HG03111",
 "HG03117", "HG03123", "HG03126", "HG03129", "HG03132", "HG03135",
 "HG03159", "HG03162", "HG03168", "HG03189", "HG03195", "HG03198",
 "HG03212", "HG03267", "HG03270", "HG03279", "HG03291", "HG03294",
 "HG03297", "HG03303", "HG03342", "HG03351", "HG03363", "HG03366",
 "HG03369", "HG03372", "HG03378", "HG03401", "HG03410", "HG03419",
 "HG03437", "HG03446", "HG03449", "HG03452", "HG03455", "HG03461",
 "HG03470", "HG03473", "HG03476", "HG03479", "HG03485", "HG03499",
```

```
 "HG03511", "HG03514", "HG03517", "HG03520", "HG03539", "HG03548",
 "HG03558", "HG03563", "HG03567", "HG03572", "HG03575", "HG03578",
 "HG03583", "NA18488", "NA18489", "NA18499", "NA18502", "NA18505",
 "NA18508", "NA18511", "NA18517", "NA18520", "NA18523", "NA18858",
 "NA18861", "NA18864", "NA18867", "NA18870", "NA18873", "NA18876",
 "NA18878", "NA18881", "NA18907", "NA18909", "NA18912", "NA18916",
 "NA18924", "NA18933", "NA19017", "NA19019", "NA19023", "NA19024",
 "NA19030", "NA19036", "NA19037", "NA19038", "NA19093", "NA19095",
 "NA19099", "NA19102", "NA19108", "NA19114", "NA19116", "NA19118",
 "NA19129", "NA19131", "NA19137", "NA19143", "NA19147", "NA19149",
 "NA19152", "NA19159", "NA19172", "NA19185", "NA19190", "NA19197",
 "NA19201", "NA19204", "NA19206", "NA19209", "NA19214", "NA19222",
 "NA19225", "NA19235", "NA19238", "NA19247", "NA19310", "NA19314",
 "NA19315", "NA19321", "NA19323", "NA19324", "NA19327", "NA19328",
 "NA19332", "NA19338", "NA19351", "NA19355", "NA19377", "NA19378",
 "NA19390", "NA19391", "NA19399", "NA19401", "NA19403", "NA19404",
 "NA19431", "NA19435", "NA19436", "NA19437", "NA19438", "NA19439",
 "NA19440", "NA19445", "NA19446", "NA19449", "NA19456", "NA19457",
 "NA19462", "NA19463", "NA19467", "NA19468", "NA19472", "NA19473",
 "NA19474", "NA19475", "NA19701", "NA19707", "NA19712", "NA19819",
 "NA19835", "NA19901", "NA19909", "NA19914", "NA19917", "NA19921",
 "NA19923", "NA20127", "NA20274", "NA20276", "NA20282", "NA20287",
 "NA20289", "NA20294", "NA20317", "NA20321", "NA20332", "NA20334",
 "NA20339", "NA20357", "NA20359", "NA20412")
AMR <- c("HG00551", "HG00554", "HG00638", "HG00641", "HG00732",
 "HG00734", "HG00737", "HG00740", "HG00743", "HG01049", "HG01052",
 "HG01055", "HG01058", "HG01064", "HG01067", "HG01070", "HG01073",
 "HG01077", "HG01080", "HG01086", "HG01089", "HG01092", "HG01095",
 "HG01098", "HG01102", "HG01105", "HG01108", "HG01111", "HG01113",
 "HG01119", "HG01122", "HG01125", "HG01131", "HG01137", "HG01140",
 "HG01149", "HG01162", "HG01168", "HG01171", "HG01174", "HG01177",
 "HG01183", "HG01188", "HG01191", "HG01205", "HG01242", "HG01248",
 "HG01251", "HG01254", "HG01257", "HG01269", "HG01272", "HG01281",
 "HG01284", "HG01303", "HG01312", "HG01323", "HG01326", "HG01342",
 "HG01345", "HG01351", "HG01357", "HG01360", "HG01363", "HG01366",
 "HG01369", "HG01372", "HG01375", "HG01378", "HG01384", "HG01390",
 "HG01393", "HG01396", "HG01403", "HG01414", "HG01432", "HG01435",
 "HG01441", "HG01444", "HG01447", "HG01456", "HG01459", "HG01462",
 "HG01474", "HG01486", "HG01489", "HG01492", "HG01495", "HG01551",
 "HG01566", "HG01572", "HG01578", "HG01893", "HG01918", "HG01921",
 "HG01924", "HG01927", "HG01933", "HG01936", "HG01939", "HG01942",
 "HG01945", "HG01948", "HG01954", "HG01965", "HG01968", "HG01971",
 "HG01973", "HG01976", "HG01980", "HG01992", "HG01997", "HG02003",
 "HG02006", "HG02089", "HG02102", "HG02105", "HG02147", "HG02252",
 "HG02260", "HG02266", "HG02272", "HG02275", "HG02278", "HG02286",
 "HG02292", "HG02298", "HG02301", "HG02312", "HG02345", "HG02348",
 "HG02425", "NA19651", "NA19654", "NA19663", "NA19669", "NA19678",
 "NA19681", "NA19684", "NA19716", "NA19719", "NA19722", "NA19725",
 "NA19728", "NA19731", "NA19734", "NA19740", "NA19746", "NA19749",
 "NA19752", "NA19755", "NA19758", "NA19761", "NA19764", "NA19770",
 "NA19773", "NA19776", "NA19779", "NA19782", "NA19785", "NA19788",
 "NA19794")
EAS <- c("HG00404", "HG00407", "HG00410", "HG00419", "HG00422",
 "HG00428", "HG00437", "HG00443", "HG00446", "HG00449", "HG00452",
 "HG00458", "HG00464", "HG00473", "HG00476", "HG00479", "HG00513",
 "HG00525", "HG00531", "HG00534", "HG00537", "HG00543", "HG00557",
 "HG00560", "HG00566", "HG00584", "HG00590", "HG00593", "HG00596",
 "HG00599", "HG00608", "HG00611", "HG00614", "HG00620", "HG00623",
 "HG00626", "HG00629", "HG00632", "HG00651", "HG00654", "HG00657",
 "HG00663", "HG00672", "HG00675", "HG00684", "HG00690", "HG00693",
```

```
 "HG00699", "HG00705", "HG00717", "HG00729", "HG00759", "HG00851",
 "HG00864", "HG00867", "HG00879", "HG00956", "HG00978", "HG01029",
 "HG01046", "HG01595", "HG01597", "HG01598", "HG01599", "HG01600",
 "HG01794", "HG01796", "HG01797", "HG01798", "HG01799", "HG01800",
 "HG01801", "HG01802", "HG01804", "HG01805", "HG01806", "HG01807",
 "HG01808", "HG01809", "HG01812", "HG01813", "HG01815", "HG01817",
 "HG01841", "HG01843", "HG01845", "HG01847", "HG01848", "HG01850",
 "HG01851", "HG01853", "HG01855", "HG01857", "HG01858", "HG01859",
 "HG01862", "HG01863", "HG01868", "HG01869", "HG01870", "HG01871",
 "HG01874", "HG01878", "HG02016", "HG02019", "HG02025", "HG02028",
 "HG02031", "HG02048", "HG02049", "HG02057", "HG02069", "HG02072",
 "HG02075", "HG02078", "HG02081", "HG02084", "HG02086", "HG02087",
 "HG02113", "HG02121", "HG02127", "HG02130", "HG02133", "HG02136",
 "HG02139", "HG02140", "HG02142", "HG02151", "HG02152", "HG02153",
 "HG02154", "HG02155", "HG02156", "HG02164", "HG02165", "HG02178",
 "HG02179", "HG02180", "HG02181", "HG02182", "HG02184", "HG02185",
 "HG02186", "HG02187", "HG02190", "HG02513", "HG02522", "NA18525",
 "NA18526", "NA18528", "NA18531", "NA18532", "NA18533", "NA18535",
 "NA18537", "NA18538", "NA18539", "NA18541", "NA18542", "NA18545",
 "NA18547", "NA18550", "NA18552", "NA18553", "NA18555", "NA18560",
 "NA18564", "NA18565", "NA18566", "NA18567", "NA18570", "NA18571",
 "NA18573", "NA18574", "NA18577", "NA18579", "NA18582", "NA18591",
 "NA18592", "NA18593", "NA18595", "NA18596", "NA18597", "NA18599",
 "NA18602", "NA18610", "NA18614", "NA18615", "NA18616", "NA18617",
 "NA18618", "NA18619", "NA18625", "NA18626", "NA18627", "NA18628",
 "NA18630", "NA18631", "NA18634", "NA18640", "NA18641", "NA18642",
 "NA18644", "NA18646", "NA18939", "NA18941", "NA18942", "NA18946",
 "NA18947", "NA18949", "NA18950", "NA18951", "NA18954", "NA18956",
 "NA18957", "NA18963", "NA18964", "NA18968", "NA18969", "NA18972",
 "NA18973", "NA18975", "NA18976", "NA18978", "NA18979", "NA18980",
 "NA18981", "NA18991", "NA18993", "NA18997", "NA18998", "NA18999",
 "NA19001", "NA19002", "NA19003", "NA19010", "NA19011", "NA19054",
 "NA19057", "NA19059", "NA19064", "NA19065", "NA19074", "NA19077",
 "NA19078", "NA19080", "NA19081", "NA19084", "NA19087", "NA19090")
EUR <- c("HG00097", "HG00099", "HG00102", "HG00110", "HG00111",
 "HG00118", "HG00120", "HG00121", "HG00122", "HG00125", "HG00127",
 "HG00128", "HG00130", "HG00132", "HG00133", "HG00137", "HG00146",
 "HG00150", "HG00154", "HG00158", "HG00171", "HG00173", "HG00174",
 "HG00176", "HG00177", "HG00178", "HG00179", "HG00180", "HG00231",
 "HG00232", "HG00233", "HG00235", "HG00236", "HG00237", "HG00238",
 "HG00239", "HG00245", "HG00250", "HG00253", "HG00254", "HG00255",
 "HG00257", "HG00258", "HG00259", "HG00261", "HG00262", "HG00263",
 "HG00266", "HG00268", "HG00269", "HG00272", "HG00274", "HG00275",
 "HG00276", "HG00281", "HG00282", "HG00285", "HG00288", "HG00304",
 "HG00306", "HG00309", "HG00313", "HG00315", "HG00318", "HG00319",
 "HG00320", "HG00323", "HG00324", "HG00326", "HG00327", "HG00328",
 "HG00330", "HG00331", "HG00332", "HG00334", "HG00337", "HG00339",
 "HG00343", "HG00344", "HG00346", "HG00349", "HG00350", "HG00353",
 "HG00355", "HG00356", "HG00357", "HG00361", "HG00362", "HG00364",
 "HG00365", "HG00367", "HG00368", "HG00373", "HG00376", "HG00378",
 "HG00379", "HG00380", "HG00381", "HG00383", "HG00384", "HG01501",
 "HG01504", "HG01507", "HG01510", "HG01513", "HG01516", "HG01519",
 "HG01522", "HG01525", "HG01528", "HG01531", "HG01537", "HG01602",
 "HG01605", "HG01607", "HG01612", "HG01613", "HG01618", "HG01620",
 "HG01623", "HG01626", "HG01628", "HG01632", "HG01668", "HG01670",
 "HG01673", "HG01676", "HG01679", "HG01684", "HG01685", "HG01695",
 "HG01697", "HG01702", "HG01704", "HG01707", "HG01710", "HG01746",
 "HG01757", "HG01762", "HG01766", "HG01768", "HG01770", "HG01773",
 "HG01776", "HG01779", "HG01784", "HG01786", "HG01790", "HG02215",
 "HG02220", "HG02223", "HG02230", "HG02232", "HG02235", "HG02239",
```

```
 "NA06985", "NA06989", "NA07000", "NA07037", "NA07056", "NA10847",
 "NA11830", "NA11832", "NA11840", "NA11892", "NA11894", "NA11918",
 "NA11920", "NA11933", "NA11995", "NA12004", "NA12006", "NA12044",
 "NA12046", "NA12058", "NA12156", "NA12234", "NA12249", "NA12273",
 "NA12275", "NA12283", "NA12287", "NA12341", "NA12348", "NA12400",
 "NA12414", "NA12489", "NA12717", "NA12718", "NA12749", "NA12751",
 "NA12761", "NA12763", "NA12776", "NA12778", "NA12813", "NA12815",
 "NA12828", "NA12830", "NA12843", "NA12873", "NA12878", "NA12890",
 "NA20502", "NA20503", "NA20504", "NA20505", "NA20506", "NA20507",
 "NA20508", "NA20514", "NA20517", "NA20522", "NA20529", "NA20530",
 "NA20531", "NA20533", "NA20535", "NA20540", "NA20541", "NA20542",
 "NA20582", "NA20585", "NA20587", "NA20589", "NA20753", "NA20756",
 "NA20757", "NA20760", "NA20761", "NA20764", "NA20766", "NA20768",
 "NA20769", "NA20771", "NA20772", "NA20773", "NA20774", "NA20775",
 "NA20790", "NA20795", "NA20797", "NA20799", "NA20800", "NA20802",
 "NA20804", "NA20807", "NA20808", "NA20813", "NA20818", "NA20819",
 "NA20821", "NA20822", "NA20826", "NA20828", "NA20832")
SAS <- c("HG02491", "HG02494", "HG02601", "HG02652", "HG02661",
 "HG02688", "HG02697", "HG02725", "HG02728", "HG02731", "HG02737",
 "HG02775", "HG02778", "HG02784", "HG02787", "HG02790", "HG02793",
 "HG03007", "HG03016", "HG03235", "HG03238", "HG03488", "HG03491",
 "HG03589", "HG03595", "HG03598", "HG03604", "HG03607", "HG03611",
 "HG03616", "HG03631", "HG03634", "HG03640", "HG03642", "HG03643",
 "HG03645", "HG03653", "HG03668", "HG03673", "HG03684", "HG03689",
 "HG03692", "HG03698", "HG03703", "HG03706", "HG03709", "HG03714",
 "HG03717", "HG03730", "HG03731", "HG03736", "HG03757", "HG03760",
 "HG03762", "HG03765", "HG03770", "HG03772", "HG03774", "HG03780",
 "HG03781", "HG03782", "HG03787", "HG03789", "HG03793", "HG03796",
 "HG03802", "HG03805", "HG03808", "HG03814", "HG03817", "HG03823",
 "HG03826", "HG03829", "HG03832", "HG03849", "HG03857", "HG03861",
 "HG03863", "HG03868", "HG03874", "HG03882", "HG03886", "HG03888",
 "HG03897", "HG03907", "HG03910", "HG03913", "HG03916", "HG03919",
 "HG03922", "HG03925", "HG03928", "HG03931", "HG03934", "HG03937",
 "HG03940", "HG03945", "HG03947", "HG03949", "HG03951", "HG03968",
 "HG03973", "HG03977", "HG03986", "HG03989", "HG04001", "HG04014",
 "HG04018", "HG04025", "HG04026", "HG04042", "HG04047", "HG04059",
 "HG04062", "HG04063", "HG04075", "HG04076", "HG04090", "HG04099",
 "HG04118", "HG04141", "HG04144", "HG04153", "HG04156", "HG04159",
 "HG04171", "HG04177", "HG04180", "HG04183", "HG04186", "HG04189",
 "HG04195", "HG04200", "HG04202", "HG04212", "HG04214", "HG04216",
 "HG04227", "NA20847", "NA20849", "NA20851", "NA20853", "NA20854",
 "NA20856", "NA20859", "NA20862", "NA20868", "NA20869", "NA20872",
 "NA20874", "NA20875", "NA20876", "NA20877", "NA20878", "NA20881",
 "NA20882", "NA20888", "NA20892", "NA20894", "NA20896", "NA20899",
 "NA20902", "NA20906", "NA20908", "NA20910", "NA21086", "NA21088",
 "NA21089", "NA21097", "NA21101", "NA21102", "NA21103", "NA21106",
 "NA21108", "NA21110", "NA21120", "NA21122", "NA21125", "NA21137",
 "NA21141", "NA21143", "NA21144")
Clusters.all <- set.populations(Clusters.all,list(AFR,AMR,EAS,EUR,SAS),
 diploid = TRUE)
Clusters.all <- set.outgroup(Clusters.all, c("Chimp"), diploid=TRUE)

#F_ST
Clusters.all <- F_ST.stats(Clusters.all, mode="nucleotide")
get.F_ST(Clusters.all, mode="nucleotide", pairwise=TRUE) [[1]]
Clusters.all@nuc.F_ST.vs.all [[1]]
Clusters.all@nuc.F_ST.vs.all [[2]]
Clusters.all@nuc.F_ST.vs.all [[3]]
Clusters.all@nuc.F_ST.vs.all [[4]]
Clusters.all@nuc.F_ST.vs.all [[5]]
```

```
#AFR
Clusters.all_AFR <- readVCF("new_chr18_1.vcf.gz", numcols=10000,
 tid="18", from=11633196, to=11698981, include.unknown = TRUE)
Clusters.all_AFR <- set.outgroup(Clusters.all_AFR,c("Chimp"),
 diploid=TRUE)
get.sum.data(Clusters.all_AFR)

#Divergence calculation
bial <- get.biallelic.matrix(Clusters.all_AFR,1) # Biallelic matrix
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0 # Sites polymorphic in humans:
 non REF (0) alleles
div <- bial[n+1,] == 1
divsites_AFR <- sum (div & !polym,na.rm=T)
divsites_AFR
D=divsites_AFR/40357
D

#Neutrality statistics
Clusters.all_AFR <- neutrality.stats(Clusters.all_AFR, do.R2 = TRUE)
get.neutrality(Clusters.all_AFR, theta=TRUE) [[1]]
Mu <- (Clusters.all_AFR@theta_Watterson/40357)/40000
Mu

#Diversities
Clusters.all_AFR <- diversity.stats(Clusters.all_AFR, pi=TRUE)
get.diversity(Clusters.all_AFR) [[1]]

#AMR
Clusters.all_AMR <- readVCF("new_chr18_2.vcf.gz", numcols=10000,
 tid="18", from=11633196, to=11698981, include.unknown = TRUE)
Clusters.all_AMR <- set.outgroup(Clusters.all_AMR,c("Chimp"),
 diploid=TRUE)
get.sum.data(Clusters.all_AMR)

#Divergence calculation
bial <- get.biallelic.matrix(Clusters.all_AMR,1) # Biallelic matrix
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0 # Sites polymorphic in humans:
 non REF (0) alleles
div <- bial[n+1,] == 1
divsites_AMR <- sum (div & !polym,na.rm=T)
divsites_AMR
D=divsites_AMR/40357
D

#Neutrality statistics
Clusters.all_AMR <- neutrality.stats(Clusters.all_AMR, do.R2 = TRUE)
get.neutrality(Clusters.all_AMR, theta=TRUE) [[1]]
Mu <- (Clusters.all_AMR@theta_Watterson/40357)/40000
Mu

#Diversities
Clusters.all_AMR <- diversity.stats(Clusters.all_AMR, pi=TRUE)
get.diversity(Clusters.all_AMR) [[1]]

#EAS
Clusters.all_EAS <- readVCF("new_chr18_3.vcf.gz", numcols=10000,
 tid="18", from=11633196, to=11698981, include.unknown = TRUE)
```

```
Clusters.all_EAS <- set.outgroup(Clusters.all_EAS,c("Chimp"),
 diploid=TRUE)
get.sum.data(Clusters.all_EAS)

#Divergence calculation
bial <- get.biallelic.matrix(Clusters.all_EAS,1) # Biallelic matrix
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0 # Sites polymorphic in humans:
 non REF (0) alleles
div <- bial[n+1,] == 1
divsites_EAS <- sum (div & !polym,na.rm=T)
divsites_EAS
D=divsites_EAS/40357
D

#Neutrality statistics
Clusters.all_EAS <- neutrality.stats(Clusters.all_EAS, do.R2 = TRUE)
get.neutrality(Clusters.all_EAS, theta=TRUE) [[1]]
Mu <- (Clusters.all_EAS@theta_Watterson/40357)/40000
Mu

#Diversities
Clusters.all_EAS <- diversity.stats(Clusters.all_EAS, pi=TRUE)
get.diversity(Clusters.all_EAS) [[1]]

#EUR
Clusters.all_EUR <- readVCF("new_chr18_4.vcf.gz", numcols=10000,
 tid="18", from=11633196, to=11698981, include.unknown = TRUE)
Clusters.all_EUR <- set.outgroup(Clusters.all_EUR,c("Chimp"),
 diploid=TRUE)
get.sum.data(Clusters.all_EUR)

#Divergence calculation
bial <- get.biallelic.matrix(Clusters.all_EUR,1) # Biallelic matrix
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0 # Sites polymorphic in humans:
 non REF (0) alleles
div <- bial[n+1,] == 1
divsites_EUR <- sum (div & !polym,na.rm=T)
divsites_EUR
D=divsites_EUR/40357
D

#Neutrality statistics
Clusters.all_EUR <- neutrality.stats(Clusters.all_EUR, do.R2 = TRUE)
get.neutrality(Clusters.all_EUR, theta=TRUE) [[1]]
Mu <- (Clusters.all_EUR@theta_Watterson/40357)/40000
Mu

#Diversities
Clusters.all_EUR <- diversity.stats(Clusters.all_EUR, pi=TRUE)
get.diversity(Clusters.all_EUR) [[1]]

#SAS
Clusters.all_SAS <- readVCF("new_chr18_5.vcf.gz", numcols=10000,
 tid="18", from=11633196, to=11698981, include.unknown = TRUE)
Clusters.all_SAS <- set.outgroup(Clusters.all_SAS,c("Chimp"),
 diploid=TRUE)
get.sum.data(Clusters.all_SAS)
```

```
#Divergence calculation
bial <- get.biallelic.matrix(Clusters.all_SAS,1) # Biallelic matrix
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0 # Sites polymorphic in humans:
 non REF (0) alleles
div <- bial[n+1,] == 1
divsites_SAS <- sum (div & !polym,na.rm=T)
divsites_SAS
D=divsites_SAS/40357
D

#Neutrality statistics
Clusters.all_SAS <- neutrality.stats(Clusters.all_SAS, do.R2 = TRUE)
get.neutrality(Clusters.all_SAS, theta=TRUE) [[1]]
Mu <- (Clusters.all_SAS@theta_Watterson/40357)/40000
Mu

#Diversities
Clusters.all_SAS <- diversity.stats(Clusters.all_SAS, pi=TRUE)
get.diversity(Clusters.all_SAS) [[1]]

######################INTERGENIC REGIONS#################################

setwd("~/Path/to/vcf.gz.files/intergenic_regions")

Intergenic.all <- readVCF("new_chr18_intergenic.vcf.gz", numcols=10000,
 tid="18", from=11552846, to=11609474, include.unknown = TRUE)
Intergenic.all <- set.outgroup(Intergenic.all,c("Chimp"), diploid=TRUE)
get.sum.data(Intergenic.all)

#Divergence calculation
bial <- get.biallelic.matrix(Intergenic.all,1)
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0
div <- bial[n+1,] == 1
divsites_inter <- sum (div & !polym,na.rm=T)
divsites_inter
D = divsites_inter/56627
D

Intergenic.all <- neutrality.stats(Intergenic.all, do.R2 = TRUE)
get.neutrality(Intergenic.all, theta=TRUE) [[1]]

Mu <- (Intergenic.all@theta_Watterson/56627)/(4*10000)
Mu

Intergenic.all <- diversity.stats(Intergenic.all, pi=TRUE)
get.diversity(Intergenic.all) [[1]]

#########################POPULATIONS#################################

#AFR
Intergenic.all_AFR <- readVCF("new_chr18_1_intergenic.vcf.gz",
 numcols=10000, tid="18", from=11552846, to=11609474, include.unknown =
 TRUE)
Intergenic.all_AFR <- set.outgroup(Intergenic.all_AFR, c("Chimp"),
 diploid=TRUE)
get.sum.data(Intergenic.all_AFR)

#Divergence calculation
```

```
bial <- get.biallelic.matrix(Intergenic.all_AFR,1)
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0
div <- bial[n+1,] == 1
divsites_inter_AFR <- sum (div & !polym,na.rm=T)
divsites_inter_AFR
D <- divsites_inter_AFR/56627
D

Intergenic.all_AFR <- neutrality.stats(Intergenic.all_AFR, do.R2 = TRUE)
get.neutrality(Intergenic.all_AFR, theta=TRUE) [[1]]
Mu <- (Intergenic.all_AFR@theta_Watterson/56627)/(4*10000)
Mu

Intergenic.all_AFR <- diversity.stats(Intergenic.all_AFR, pi=TRUE)
get.diversity(Intergenic.all_AFR) [[1]]

#AMR
Intergenic.all_AMR <- readVCF("new_chr18_2_intergenic.vcf.gz",
 numcols=10000, tid="18", from=11552846, to=11609474, include.unknown =
 TRUE)
Intergenic.all_AMR <- set.outgroup(Intergenic.all_AMR, c("Chimp"),
 diploid=TRUE)
get.sum.data(Intergenic.all_AMR)

#Divergence calculation
bial <- get.biallelic.matrix(Intergenic.all_AMR,1)
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0
div <- bial[n+1,] == 1
divsites_inter_AMR <- sum (div & !polym,na.rm=T)
divsites_inter_AMR
D <- divsites_inter_AMR/56627
D

Intergenic.all_AMR <- neutrality.stats(Intergenic.all_AMR, do.R2 = TRUE)
get.neutrality(Intergenic.all_AMR, theta=TRUE) [[1]]
Mu <- (Intergenic.all_AMR@theta_Watterson/56627)/(4*10000)
Mu

Intergenic.all_AMR <- diversity.stats(Intergenic.all_AMR, pi=TRUE)
get.diversity(Intergenic.all_AMR) [[1]]

#EAS
Intergenic.all_EAS <- readVCF("new_chr18_3_intergenic.vcf.gz",
 numcols=10000, tid="18", from=11552846, to=11609474, include.unknown =
 TRUE)
Intergenic.all_EAS <- set.outgroup(Intergenic.all_EAS, c("Chimp"),
 diploid=TRUE)
get.sum.data(Intergenic.all_EAS)

#Divergence calculation
bial <- get.biallelic.matrix(Intergenic.all_EAS,1)
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0
div <- bial[n+1,] == 1
divsites_inter_EAS <- sum (div & !polym,na.rm=T)
divsites_inter_EAS
D <- divsites_inter_EAS/56627
D
```

```
Intergenic.all_EAS <- neutrality.stats(Intergenic.all_EAS, do.R2 = TRUE)
get.neutrality(Intergenic.all_EAS, theta=TRUE) [[1]]
Mu <- (Intergenic.all_EAS@theta_Watterson/56627)/(4*10000)
Mu

Intergenic.all_EAS <- diversity.stats(Intergenic.all_EAS, pi=TRUE)
get.diversity(Intergenic.all_EAS) [[1]]

#EUR
Intergenic.all_EUR <- readVCF("new_chr18_4_intergenic.vcf.gz",
 numcols=10000, tid="18", from=11552846, to=11609474, include.unknown =
 TRUE)
Intergenic.all_EUR <- set.outgroup(Intergenic.all_EUR, c("Chimp"),
 diploid=TRUE)
get.sum.data(Intergenic.all_EUR)

#Divergence calculation
bial <- get.biallelic.matrix(Intergenic.all_EUR,1)
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0
div <- bial[n+1,] == 1
divsites_inter_EUR <- sum (div & !polym,na.rm=T)
divsites_inter_EUR
D <- divsites_inter_EUR/56627
D

Intergenic.all_EUR <- neutrality.stats(Intergenic.all_EUR, do.R2 = TRUE)
get.neutrality(Intergenic.all_EUR, theta=TRUE) [[1]]
Mu <- (Intergenic.all_EUR@theta_Watterson/56627)/(4*10000)
Mu

Intergenic.all_EUR <- diversity.stats(Intergenic.all_EUR, pi=TRUE)
get.diversity(Intergenic.all_EUR) [[1]]

#SAS
Intergenic.all_SAS <- readVCF("new_chr18_5_intergenic.vcf.gz",
 numcols=10000, tid="18", from=11552846, to=11609474, include.unknown =
 TRUE)
Intergenic.all_SAS <- set.outgroup(Intergenic.all_SAS, c("Chimp"),
 diploid=TRUE)
get.sum.data(Intergenic.all_SAS)

#Divergence calculation
bial <- get.biallelic.matrix(Intergenic.all_SAS,1)
n <- nrow(bial)-2
polym <- apply(bial[1:n,,drop=F],2,sum)>0
div <- bial[n+1,] == 1
divsites_inter_SAS <- sum (div & !polym,na.rm=T)
divsites_inter_SAS
D <- divsites_inter_SAS/56627
D

Intergenic.all_SAS <- neutrality.stats(Intergenic.all_SAS, do.R2 = TRUE)
get.neutrality(Intergenic.all_SAS, theta=TRUE) [[1]]
Mu <- (Intergenic.all_SAS@theta_Watterson/56627)/(4*10000)
Mu

Intergenic.all_SAS <- diversity.stats(Intergenic.all_SAS, pi=TRUE)
get.diversity(Intergenic.all_SAS) [[1]]
```

```
#MKT
#chi-square
MKT =rbind(c(Intergenic.all@n.biallelic.sites, divsites_inter),
 c(Clusters.all@n.biallelic.sites, divsites))
MKT
chisq.test(MKT, simulate.p.value=TRUE)

MKT_AFR=rbind(c(Intergenic.all_AFR@n.biallelic.sites,
 divsites_inter_AFR), c(Clusters.all_AFR@n.biallelic.sites,
 divsites_AFR))
MKT_AFR
chisq.test(MKT_AFR, simulate.p.value=TRUE)

MKT_AMR=rbind(c(Intergenic.all_AMR@n.biallelic.sites,
 divsites_inter_AMR), c(Clusters.all_AMR@n.biallelic.sites,
 divsites_AMR))
MKT_AMR
chisq.test(MKT_AMR, simulate.p.value=TRUE)

MKT_EAS=rbind(c(Intergenic.all_EAS@n.biallelic.sites,
 divsites_inter_EAS), c(Clusters.all_EAS@n.biallelic.sites,
 divsites_EAS))
MKT_EAS
chisq.test(MKT_EAS, simulate.p.value=TRUE)

MKT_EUR=rbind(c(Intergenic.all_EUR@n.biallelic.sites,
 divsites_inter_EUR), c(Clusters.all_EUR@n.biallelic.sites,
 divsites_EUR))
MKT_EUR
chisq.test(MKT_EUR, simulate.p.value=TRUE)

MKT_SAS=rbind(c(Intergenic.all_SAS@n.biallelic.sites,
 divsites_inter_SAS), c(Clusters.all_SAS@n.biallelic.sites,
 divsites_SAS))
MKT_SAS
chisq.test(MKT_SAS, simulate.p.value=TRUE)
```