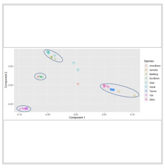

Dec 15, 2020

Investigation of ancestral alleles in the Bovinae subfamily

Maulana Naji[1], Yuri Utsunomiya[2], Johann Sölkner[1], Benjamin Rosen[3], Gábor Mészáros[1]

[1]University of Natural Resources and Life Sciences Vienna, Austria; [2]Sao Paulo State University, Brazil;

[3]Agricultural Research Service USDA, USA

Maulana Naji: Main contact;

1 Works for me dx.doi.org/10.17504/protocols.io.bh99j996

Maulana Naji

ABSTRACT

In evolutionary theory, divergence and speciation can arise from long periods of reproductive isolation, genetic mutation, selection and environmental adaptation. After divergence, alleles can either persist in their initial state (ancestral allele - AA), co-exist or be replaced by a mutated state (derived alleles -DA). Defining AA and DA at polymorphic sites is useful to test hypotheses regarding molecular evolutionary processes, including estimation of allelic age, formation of linkage disequilibrium patterns and selection signatures. In this study, we aligned whole genome sequences of individuals from the Bovinae subfamily (gaur, yak, bison, wisent, gayal, and cattle) to the cattle reference genome (ARS.UCD-1.2). We then identified sequence variants in three outgroups, i.e. yak, bison and gayal-gaur-banteng. Accommodating independent divergent of each lineage from the initial ancestral state, AA were defined based on fixed alleles on at least two groups resulting in ~32.4 million variants. Using non-overlapping scanning windows of 10 Kb, we counted the AA observed within taurine and zebu cattle. We focused on the extreme points, regions with top 0.1% (high count) and regions without any occurrence of AA (null count). As mutation occurs across autosome independently, high count regions preserved gene functions from ancestral states that are still beneficial in the current condition, while null counts regions were linked to mutated ones. Gene ontology terms were inferred for functionality of referred regions. For both cattle, high count regions were associated to basal lipid metabolism which is essential on surviving of various condition pressures. Mutated regions were associated to productive traits in taurine such as higher metabolism, cell development and behaviors. While for zebu, mutation appeared for adaptation to marginal environment showed mainly by function in immune response domain. Our findings suggest that retaining and losing AA in some genes or regions are varied and made it species-specific with possibility of overlapping as it depends on the selective pressure they had to experience.

DOI

dx.doi.org/10.17504/protocols.io.bh99j996

PROTOCOL CITATION

Maulana Naji, Yuri Utsunomiya, Johann Sölkner, Benjamin Rosen, Gábor Mészáros 2020. Investigation of ancestral alleles in the Bovinae subfamily. **protocols.io**
https://dx.doi.org/10.17504/protocols.io.bh99j996

KEYWORDS

Ancestral Allele, WGS, Cattle

CREATED

Jul 07, 2020

LAST MODIFIED

Dec 15, 2020

PROTOCOL INTEGER ID

38945

GUIDELINES

This protocol is prepared for ancestral allele investigation of cattle using outgroup species of yak, bison, and gayal-gaur-banteng. Input files are in format of paired end reads FASTQ. Software used are mentioned in each step where necessary. For installation of software, please look at each respective attached weblinks. Custom R functions https://github.com/mas-agis/ances-al were created for subsequent steps in calling ancestral allele and comparison to focal species of taurine cattle. End of the pipeline is GO analysis for regions with high and without ancestral alleles.

ABSTRACT

In evolutionary theory, divergence and speciation can arise from long periods of reproductive isolation, genetic mutation, selection and environmental adaptation. After divergence, alleles can either persist in their initial state (ancestral allele - AA), co-exist or be replaced by a mutated state (derived alleles -DA). Defining AA and DA at polymorphic sites is useful to test hypotheses regarding molecular evolutionary processes, including estimation of allelic age, formation of linkage disequilibrium patterns and selection signatures. In this study, we aligned whole genome sequences of individuals from the Bovinae subfamily (gaur, yak, bison, wisent, gayal, and cattle) to the cattle reference genome (ARS.UCD-1.2). We then identified sequence variants in three outgroups, i.e. yak, bison and gayal-gaur-banteng. Accommodating independent divergent of each lineage from the initial ancestral state, AA were defined based on fixed alleles on at least two groups resulting in ~32.4 million variants. Using non-overlapping scanning windows of 10 Kb, we counted the AA observed within taurine and zebu cattle. We focused on the extreme points, regions with top 0.1% (high count) and regions without any occurrence of AA (null count). As mutation occurs across autosome independently, high count regions preserved gene functions from ancestral states that are still beneficial in the current condition, while null counts regions were linked to mutated ones. Gene ontology terms were inferred for functionality of referred regions. For both cattle, high count regions were associated to basal lipid metabolism which is essential on surviving of various condition pressures. Mutated regions were associated to productive traits in taurine such as higher metabolism, cell development and behaviors. While for zebu, mutation appeared for adaptation to marginal environment showed mainly by function in immune response domain. Our findings suggest that retaining and losing AA in some genes or regions are varied and made it species-specific with possibility of overlapping as it depends on the selective pressure they had to experience.

BEFORE STARTING

Example scripts in this protocol are demonstration on how we run the analysis.
For running the analysis, one has to adapt path to folders containing input, output files, and executable programs respectively in their cluster/local computation unit.
Moreover, parameters on the second step needs to be adapt in respect to available quality of service (qos) in one's respective high performance cluster.

Reads alignment and variant calling on individuals separately

1 Download Read Sequences in format of SRA FASTQ

Linux - wget

```
wget
ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR201/006/SRR2016766/SRR2016766_1.fastq.gz
wget
ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR201/006/SRR2016766/SRR2016766_2.fastq.gz
```

This script using wget for fetching sequence reads from NCBI.
Further details on downloading dataset refers to https://www.ncbi.nlm.nih.gov/home/tools/
Please refer to table 2 and additional file for list of SRA reads used in our study.
Example for retrieve SRR2016766, paste the command below:

2 Mapping with BWA-Mem

Linux - HPC_Slurm

```
#!/bin/bash
#
## usage:  sbatch  ./BWA_Gen.sbmt.scrpt
#
#SBATCH -J BWA$1
#SBATCH -N 1
#SBATCH --partition mem_0064
#SBATCH --qos normal_0064
#SBATCH --ntasks-per-node 16

module purge
module load gcc/5.3 R/3.4.0 java/1.8.0_121 python/3.6

~masagis/bwa-0.7.17/bwa mem -t 8 ~masagis/REF/ARS-UCD1.2_Btau5.0.1Y.fa
~masagis/FASTQ/SRR$1_1.fastq.gz ~masagis/FASTQ/SRR$1_2.fastq.gz |
~masagis/samtools-1.10/samtools sort -@8 -o ~masagis/SORT/SRR$1.bam
wait
~masagis/gatk-4.1.0.0/gatk MarkDuplicates --INPUT ~masagis/SORT/SRR$1.bam --
METRICS_FILE ~masagis/MARKDUP/MF_SRR$1 --OUTPUT
~masagis/MARKDUP/MD_SRR$1.bam --CREATE_INDEX true
wait
~masagis/gatk-4.1.0.0/gatk AddOrReplaceReadGroups --INPUT
~masagis/MARKDUP/MD_SRR$1.bam --OUTPUT
~masagis/MARKDUP/RG_MD_SRR$1.bam  --RGID $1 -LB lib$1 -PL illumina -PU unit$1
-SM SRR$1
wait
~masagis/gatk-4.1.0.0/gatk BaseRecalibrator --reference
~masagis/ResourceBundle/Galaxy47-[ARS-UCD1.2_Btau5.0.1Y.fa.gz].fasta --input
~masagis/MARKDUP/RG_MD_SRR$1.bam --known-sites
~masagis/ResourceBundle/ARS1.2PlusY_BQSR.vcf --output
~masagis/BQSR/BQSR_$1.table
wait
~masagis/gatk-4.1.0.0/gatk ApplyBQSR -I ~masagis/MARKDUP/RG_MD_SRR$1.bam -
bqsr ~masagis/BQSR/BQSR_$1.table -O ~masagis/BQSR/$1.bam
```

We do subsequent process of mapping paired reads by using BWA Mem http://bio-bwa.sourceforge.net/bwa.shtml and piped it to samtools http://www.htslib.org/doc/samtools.html for sorting by chromosome and then pipe it for MarkDuplicates, AddOrReplaceGroups, BaseRecalibrator and ApplyBQSR using GATK https://gatk.broadinstitute.org/hc/en-us.

Above script is saved as single file named "script.sh".

The script is then submitted as slurm in HPC for each individual dataset using command "sbatch ./script.sh 2016766" for respective SRR2016766 individuals

3 Individual Haplotype Calling

Linux - GATK_HaplotypeCaller

```
#!/bin/bash
#
## usage:  sbatch  ./HapCalGen.scrpt
#
#SBATCH -J Brahman$1
#SBATCH -N 1
#SBATCH --partition mem_0064
#SBATCH --qos normal_0064
#SBATCH --ntasks-per-node 16

module purge
module load gcc/5.3 R/3.4.0 java/1.8.0_121 python/3.6

~masagis/gatk-4.1.0.0/gatk HaplotypeCaller -R
~masagis/ResourceBundle/Galaxy47-[ARS-UCD1.2_Btau5.0.1Y.fa.gz].fasta -I
~masagis/BQSR/$1.bam -O ~masagis/BQSR/$1.g.vcf.gz -ERC GVCF --native-pair-
hmm-threads 28
```

Calling variants for each individual BAM file after subsequent previous steps. The script is submitted as slurm in HPC using command "sbatch ./HapCal.sh 2016766" outputting file in format of 2016766.g.vcf.gz

Joint analysis of individuals

4 Combine GVCFs

Linux - GATK_CombineGVCFs

```
./gatk CombineGVCFs -R ~/data/Cattle/ResourceBundle/Galaxy47-[ARS-
UCD1.2_Btau5.0.1Y.fa.gz].fasta --variant ~/data/Cattle/BQSR/2016766.g.vcf.gz --
variant .......
-O ~/data/Cattle/Large/cohort4.g.vcf.gz
```

We combine GVCFs from each individual .g.vcf.gz with output cohort4.g.vcf.gz.
In the command above, each individual with .g.vcf.gz file name is specified after "–variant " represented by empty dots.

5 Genotype GVCFs

Linux - GATK_GenotypeGVCFs

```
./gatk GenotypeGVCFs -R ~/data/Cattle/ResourceBundle/Galaxy47-[ARS-
UCD1.2_Btau5.0.1Y.fa.gz].fasta -V ~/data/Cattle/Large/cohort4.g.vcf.gz   -O
~/data/Cattle/Large/cohort4.vcf.gz
```

Calling variants in single file for all individuals

6 Split VCF

Linux - GATK_SplitVcfs

```
./gatk SplitVcfs -I ~/data/Cattle/Large/cohort4.vcf.gz --SNP_OUTPUT
~/data/Cattle/Large/SNP_cohort4.vcf --INDEL_OUTPUT
~/data/Cattle/Large/Indel_cohort4.vcf.gz --STRICT false
```

Split VCF file to indel vcf and SNP vcf

7 Getting name of individuals in vcf

Linux - bcftools

```
bcftools query -l  ~/data/Cattle/Large/SNP_cohort4.vcf >
~/data/Cattle/Large/list_samples.txt
```

Above script generates list of individuals in SNP_cohort4.vcf and stored it in list_samples.txt

We then manually changed to initial name as intended using the correct order and save it into list_samples_new.txt

For example:

```
$ cat list_samples.txt
SRR2016766
SRR2016789
$ cat list_samples_new.txt
Hols1
Hols2
```

Bcftools is provided in http://samtools.github.io/bcftools/bcftools.html

8 Reheader VCF files with new individuals name

Linux - bcftools

```
bcftools reheader -s ~/data/Cattle/Large/list_samples_new.txt
~/data/Cattle/Large/SNP_cohort4.vcf -o
~/data/Cattle/Large2/reheader_SNP_cohort4.vcf
```

Above script will rename each individuals in input file SNP_cohort4.vcf.

list_samples_new.txt is simple txt file containing intended new name for each individual corresponding to previously generated list_samples.txt in step 7.

Output file is reheader_SNP_cohort4.vcf

9 Variant Filtration

Linux - GATK_VariantFiltration

```
./gatk VariantFiltration -R ~/data/Cattle/ResourceBundle/Galaxy47-[ARS-UCD1.2_Btau5.0.1Y.fa.gz].fasta -V ~/data/Cattle/Large2/reheader_SNP_cohort4.vcf -O ~/data/Cattle/Large2/filteredSNP.vcf -filter "MQ < 40.0" --filter-name "MQfil" -filter "QD < 3.0" --filter-name "QDfil" -filter "QUAL < 30.0" --filter-name "QUAfil"
```

Filter SNP file with intended parameters. Here, we applied the default as found in GATK webpage.

Principal Component Analysis

10 Creating PCA Plot using PLINK

10.1 Convert VCF file into plink format using VCFTools

Linux - vcftools

```
vcftools  --vcf ~/data/Cattle/Large2/filteredSNP.vcf --keep ~/data/Cattle/Large/list_samples_new.txt --plink --out ~/data/Cattle/Large2/Plink_cohort5
```

Script above is converting filteredSNP.vcf file into Plink_cohort5.ped format by keeping all individuals listed in list_samples_new.txt

10.2 Pruning Plink file and create mds plot with four components

Linux - Plink

```
plink --file Plink_Cohort5 --chr 1:29 --make-bed --out B_cohort5

plink --bfile B_cohort5 --indep-pairwise 50 5 0.7

plink --bfile B_cohort5 --extract plink.prune.in --make-bed --out pruneddata5

plink --bfile pruneddata5 --genome --out Cohort5

plink --bfile pruneddata5 --read-genome Cohort5.genome --cluster --mds-plot 4
```

We use plink command http://zzz.bwh.harvard.edu/plink/ to prune .ped file generated from step 10.1 and to create .mds file with four components.
Each line in the above script must be executed as a single command. Next line, should be executed after the current command is finished.
These commands are executed in linux terminal

10.3 Plotting the output of plink mds us

R

```
setwd("D:/maulana/Cattle/Large2")
library(ggplot2)

df<-read.table("plink.mds", header=TRUE) #data with 4 principal components
head(df)

df$FID <- as.character(df$FID)
df$FID[c(36,37,54:58,72)] <- ("Zebu")
df$FID[c(17:28,67)] <- ("Yak")
df$FID[c(2,38:40)] <- ("Gayal")
df$FID[c(41,42,69:71)] <- ("Eur.Bison")
df$FID[c(43:47)] <- ("Banteng")
df$FID[c(48:51)] <- ("Gaur")
df$FID[c(31,32,52,53)] <- ("Ame.Bison")
df$FID[c(29,30)] <- ("Aurochs")
df$FID[c(1,4:5,10:15,33:35,7:9,16,6,62,63:66,68)] <- ("Taurus")
df <- df[-c(3,60,61,59),]

colnames(df)[1] <- "Species"
df$Species <- as.factor(df$Species)

ggplot(df, aes(x = C1, y = C2)) + geom_point(aes(colour = Species, shape=
Species), size=2) +
  labs(title = "Principle Component Analysis", x = "Component 1", y=
"Component 2")
```

This script utilizing plink.mds file, output of plink in step 10.2.
After reading the data, we add a new column of group name in df$FID manually according to individuals set. For example, we add label "taurus" for individuals of holstein, jersey, angus and simmental.
Then, we change the column name to 'species' and plot using ggplot.

Phylogenetic analysis

11 Creating Phylogenetic Trees

11.1 Splitting VCF by chromosome

Linux - vcftools

```
for i in {1..29};
do vcftools  --vcf ~/data/Cattle/Large2/filteredSNP.vcf --keep
~/data/Cattle/Large/list_samples_new.txt --chr $i  --recode --recode-INFO-all --
out ~/data/Cattle/Large2/VCF5_$i;
done
```

We utilized vcftools to split filteredSNP.vcf by chromosome one to twenty-nine using for loop function.

11.2 Reducing redundant variants by LD

Linux - SNPhylo

```
for i in {1..29};
do /home/masagis/data/snphylo/SNPhylo/snphylo.sh -v VCF5_$i.recode.vcf -p 10 -c 2 -P chr$i -a 30 -l 0.9 -m 0.1 -M 0.1 ;
done
```

We run SNPhylo from http://chibba.pgml.uga.edu/snphylo/ for reducing number of variants based on its LD using parameters specified above.
We do for loop for each chromosome one to twenty-nine.

11.3 Visualize the tree per chromosome wise

Manually visualize the trees using MEGAX https://www.megasoftware.net/ using SNPhylo outputs with intial tree built using maximum parsimony method and bootstrap 200 using maximum likelihood of Jukes Cantor Model

Inferring Ancestral Alleles

12 Defining ancestral alleles

We defined ancestral alleles of cattle as alleles fixed in at least two outgroup of bison, yak and gayal-gaur-banteng. In the steps of 12, 13, and 14, R-scripts used several in-house functions written in https://github.com/mas-agis/ances-al . Simply copy and run the functions in the R-environment before running the intended script, we will put description on which function to be loaded on the description.

12.1 Getting allele frequency for chromosome 1 to 29

Linux - vcftools

```
for i in {1..29};
do vcftools --vcf ~/data/Cattle/Large2/filteredSNP.vcf --chr $i --freq --keep ~/data/Cattle/Large/Yak_List.txt --out ~/data/Cattle/Large3/yak_Chr_$i;
done
```

Command above is for outputting allele frequency of yak individuals.
For bison and gayal-gaur-banteng, one just need to change the list of individual belong to particular group in '–keep' parameter and output name in '–out' parameter for each respective group.
Additionally, "all" group is combination of all individuals within these three groups.

12.2 Filtering sites with allele frequency of 1 for each group and then combined between bison, yak and gayal-gaur-banteng.

R

```
setwd("D:/maulana/Cattle/Large3/")
#getting sites with frequency 1
bis <- ancealls("Bison",n=9,cr=1:29)
yk <- ancealls("yak",n=13,cr=1:29)
ggb <- ancealls("gagaba",n=13,cr=1:29)
al <- ancealls("all",n=35,cr=1:29)
bison <- bis$Ances_Allele
yak <- yk$Ances_Allele
gagaba <- ggb$Ances_Allele
All <- al$Ances_Allele

#label fixation in new column
Bison$Fixed_In <- "Bison"
Yak$Fixed_In <- "Yak"
Gagaba$Fixed_In <- "Gagaba"
All$Fixed_In <- "Bison,Gagaba,Yak"

#Joining steps
Yagaba <- inner_join(Yak,Gagaba, by=c("Chr","Pos","AA"))
Bigaba <- inner_join(Bison,Gagaba, by=c("Chr","Pos","AA"))
Biyak <- inner_join(Bison,Yak, by=c("Chr","Pos","AA"))
Alles <- rbind(Yagaba,Bigaba)
Alles <- rbind(Alles,Biyak)

Allesa <- Alles
Allesa$Alleles_n <- Allesa$Alleles_n.x
Allesa$Fixed_In <- paste0(Allesa$Fixed_In.x,",",Allesa$Fixed_In.y)
Allesa$Freq <- Allesa$Freq.x
Allesa <- Allesa[c(1,2,10,4,12,11)]
Allesa <- arrange(Allesa)
All$Pos <- as.factor(All$Pos)
final <- union(All,Allesa)
final <- arrange(final, Chr, Pos)
Finalis <- final
Finalis$sort <- paste0(Finalis$Chr,"+",Finalis$Pos)
Finalis <- Finalis[!duplicated(Finalis$sort),]
Finalis$Pos <- as.integer(Finalis$Pos)
Finalis <- arrange(Finalis, Chr, Pos)
Finalis$Group <- as.factor(Finalis$Fixed_In)

#Writing output files
setwd("D:/maulana/Cattle/Large3/")
write.table(Finalis[1:6],"Additional file 2.txt",quote = F)
```

Load R function of "ancealls" from https://github.com/mas-agis/ances-al . Then run the above script.
This ancealls function will calling putative ancestral alleles from frequency files generated in step 12.1 by its group name (such as "Bison", "yak", etc.) independently, add new label of defined group, join the defined ancestral allele from each two group combination as stated in our paper, removing duplicate positions and arrange columns order.
Final list of cattle ancestral allele is saved as "Additional file 2.txt"

Comparisons to cattle group

13 Comparisons ancestral allele in subset of taurus and indicus group

We compared the frequency file of cattle to the ancestral allele file named as "Additional file 2.txt" generated in step 12.2.
Example below is only demonstration script for taurine cattle comprising of 23 individuals.
For zebu cattle, should be done in similar manner with changing respective parameters.

13.1 Getting alele frequency for taurine cattle

Linux - vcftools

```
for i in {1..29};
do vcftools --vcf ~/data/Cattle/Large2/filteredSNP.vcf --chr $i --freq --keep
~/data/Cattle/Large/Taurus_List.txt --out ~/data/Cattle/Large3/taurus_Chr_$i;
done
```

Taurus_List.txt is list of individual names of the 23 taurine cattle.
This script is run in the linux terminal.

13.2 Persisting ancestral allele in taurine

R

```
setwd("D:/maulana/Cattle/Large3/")
ancestor<- read.table("Additional file 2.txt")
#calling persisting ancestral allele
taurus <- persist.ancesA(ancestor, gr="taurus",n=23,cr=1:29)

#Writing persisting ancestral alleles
write.table(taurus$AACounts, "Taurus_AACounts_3Groups.txt", col.names =
T, row.names = F, quote = F)
write.table(taurus$treshold, "TaurusTreshold_Null_Changes_3Groups.txt",
col.names = T, row.names = F, quote = F)
```

Ancestor is file of cattle ancestral alleles loaded from "additional file 2.txt" defined in previous step 12.2.
We use "persist.ancesA" function to count sum of ancestral alleles from frequency file of taurus generated in step 13.1 within fixed size of scanning window. Here, we use default parameter of persist.ancesA.
Results is written in "taurus" list.
Writing results to local directory is optional.

13.3 Creating manhattan plot based on ancestral allele

R

```
for (i in 1:29) {
  plt.ances(taurus$AACounts,taurus$treshold,cr=i)
  ggsave(paste0("Manhattan_",i,"taurus.png"), width = 7.57, height = 3.99)
}
```

We do loop over chromosome 1 to 29 to generate manhattan plots such in our paper.
We use "plt.ances" function with two dataframe from step 13.2 as input files, and then run with default parameters.
Images will be saved in the current directory named as specified in ggsave part.

13.4 Preparing for annovar input

R

```
#Using AAcounts and threshold dataset of ancestral allele from previous step
taurine <- annv.ances(taurus$AACounts, taurus$treshold)
#writing for annovar input
#High AA regions
write.table(taurine$Above, "Taurus_High_Region.txt", col.names = F, row.names = F, quote = F)
#Regions without AA
write.table(taurine$Zero, "Taurus_Null_Region.txt", col.names = F, row.names = F, quote = F)
#Ratio of regions without AA to all scanning windows
write.table(taurine$Ratio, "Taurus_ratio.txt", col.names = T, row.names = F, quote = F)
```

Using "annv.ances" function, we separate scanning windows from outputs of function in step 13.2 based on its ancestral alleles count.
This step will produces three dataframes, i.e. Above, Zero, and Ratio.
Dataframe Above and Zero are complied as input for annovar gene annotation, for regions of interest due to high or without ancestral allele counts. While, Ratio dataframe is ratio of windows without ancestral alleles to total scanning windows per chromosome.
Then, we write each dataframe in local directory.

13.5 Getting average of ancestral allele

R

```
AverageTaurus <- AverageAA(taurus$AACounts,taurus$treshold)
write.table(AverageTaurus, "TaurusAverageAAcount.txt", col.names = T, row.names = F, quote = F)
```

We use "averageAA" function in this step. As its name, we want to get mean of ancestral allele counts per chromosome.
Then, we write it to the local directory.

13.6 Checking whether regions without AA due to non-existing defined AA or reckoned mutation

R

```
CheckTaurusnull <- CheckNullRegion(ancestor,taurine$Zero,cr=1:29)
write.table(CheckTaurusnull, "CheckTaurusnull.txt", col.names = T,
row.names = F, quote = F)
```

In this step we use "CheckNullRegion" function. It is checking whether scanning windows without ancestral allele are due to change of ancestral allele or there were no ancestral alleles defined within the scanning windows. The fifth column generated from this function called "Actual_AA_Sites". It contains actual number of sites with defined ancestral allele. Supposed it contains any number than zero, than non-existing ancestral allele in scanning window is due to mutation.

Gene annotation using annovar

14 Gene annotation using annovar https://doc-openbio.readthedocs.io/projects/annovar/en/latest/ with input from previous step 13.4

14.1 Moving input files to annovar directory

Linux

```
mv ~data/Cattle/Large3/*Region.txt ~/data/annovar/Region/
```

Moving files generated in step 13.4 containing regions information with high and no ancestral alleles for annovar analysis.

14.2 Gene annotation with respective regions with high and without ancestral allele

Linux - annovar

```
#Taurus high AA
annotate_variation.pl -out Sites19 -build bosTau9
Region/Taurus_High_Region.txt cattledb/
#Taurus without AA
annotate_variation.pl -out Sites21 -build bosTau9
Region/Taurus_Null_Region.txt cattledb/
```

This script respectively does gene annotation for regions with high and null counts of ancestral alleles

14.3 Listing genes without indel and duplication

Linux

```
#genes from regions high ancestral allele
grep -v 'intergenic' Sites19.variant_function | grep -v 'upstream' | grep -v
'downstream' | cut -f2 | sed 's/\,/\n/g' | grep -v 'ins' | grep -v 'del' >
Sites19Gene.txt

#genes from regions without ancestral allele
grep -v 'intergenic' Sites21.variant_function | grep -v 'upstream' | grep -v
'downstream' | cut -f2 | sed 's/\,/\n/g' | grep -v 'ins' | grep -v 'del' >
Sites21Gene.txt
```

In this script respectively we output only list of genes without duplication.
We save it with Sites19Gene.txt and Sites21Gene.txt for genes found within regions with high and null counts of ancestral allele, respectively.

14.4 Filtered out genes found in regions without AA from regions with high AA

R

```
setwd("D:/maulana/annovar/")
library(dplyr)
sites19 <- read.table("Sites19Gene.txt")
sites21 <- read.table("Sites21Gene.txt")
write.table(setdiff(sites19,sites21),"Sites19GeneEdit.txt",quote = F,
row.names = F,col.names = F)
```

As scanning window might cut off gene transcripts, sites19GeneEdit is a list of genes with high ancestral alleles after being filtered out from genes that also found in regions without ancestral allele.

Gene Ontology analysis

15 Gene Ontology analysis using https://david.ncifcrf.gov/summary.jsp

Running GO analysis by pasting gene names from Sites21Gene.txt in step 14.3 for regions without ancestral allele and Sites19GeneEdit.txt in step 14.4 for regions with high count of ancestral alleles. Our interests were Biological Process (BP) related to GO term.