BCB546-R\_Assignment

Md Shazid Hasan

2023-03-12

## part I

### Data inspection

#### Loading the 2 files to be the genotype and pos data frames

#Load the Library and import data  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.0 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.1 ✔ tibble 3.1.8  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors

genotype <- as.data.frame(read.table("fang\_et\_al\_genotypes.txt", sep="\t",header=TRUE))  
pos <- as.data.frame(read.table("snp\_position.txt", sep="\t",header=TRUE))

#### SNP genotypes data

* fang\_et\_al\_genotypes.txt is assigned to be the genotype data frame, which is large dimension data set, so use dim, str, glimpse and etc functions to know the number of rows and columns, column names and their variable types.

dim(genotype)

## [1] 2782 986

sapply(genotype, class)[1:6]

## Sample\_ID JG\_OTU Group abph1.20 abph1.22 ae1.3   
## "character" "character" "character" "character" "character" "character"

#str(genotype)  
#glimpse(genotype)  
colnames(genotype)[1:6]

## [1] "Sample\_ID" "JG\_OTU" "Group" "abph1.20" "abph1.22" "ae1.3"

genotype[1:6,1:6]

## Sample\_ID JG\_OTU Group abph1.20 abph1.22 ae1.3  
## 1 SL-15 T-aust-1 TRIPS ?/? ?/? T/T  
## 2 SL-16 T-aust-2 TRIPS ?/? ?/? T/T  
## 3 SL-11 T-brav-1 TRIPS ?/? ?/? T/T  
## 4 SL-12 T-brav-2 TRIPS ?/? ?/? T/T  
## 5 SL-18 T-cund TRIPS ?/? ?/? T/T  
## 6 SL-2 T-dact-1 TRIPS ?/? ?/? T/T

genotype %>%   
 group\_by(Group) %>%   
 count()

## # A tibble: 16 × 2  
## # Groups: Group [16]  
## Group n  
## <chr> <int>  
## 1 TRIPS 22  
## 2 ZDIPL 15  
## 3 ZLUXR 17  
## 4 ZMHUE 10  
## 5 ZMMIL 290  
## 6 ZMMLR 1256  
## 7 ZMMMR 27  
## 8 ZMPBA 900  
## 9 ZMPIL 41  
## 10 ZMPJA 34  
## 11 ZMXCH 75  
## 12 ZMXCP 69  
## 13 ZMXIL 6  
## 14 ZMXNO 7  
## 15 ZMXNT 4  
## 16 ZPERR 9

# About fang\_et\_al\_genotypes.txtfile:

* It has 2782 rows and 986 columns.
* It has 16 different groups.

#### SNP markers information

* snp\_position.txt is assigned to be the pos data frame.
* Using the same function to know the data strcuture of the pos data frame.
* Replacing the unknown and multiple in Position column to be NA and to know how many numbers of SNP markers, and their maximum and minimum position value in each of chromosome.

dim(pos)

## [1] 983 15

sapply(pos, class)[1:6]

## SNP\_ID cdv\_marker\_id Chromosome Position alt\_pos   
## "character" "integer" "character" "character" "character"   
## mult\_positions   
## "character"

str(pos)

## 'data.frame': 983 obs. of 15 variables:  
## $ SNP\_ID : chr "abph1.20" "abph1.22" "ae1.3" "ae1.4" ...  
## $ cdv\_marker\_id : int 5976 5978 6605 6606 6607 5982 3463 3466 5983 5985 ...  
## $ Chromosome : chr "2" "2" "5" "5" ...  
## $ Position : chr "27403404" "27403892" "167889790" "167889682" ...  
## $ alt\_pos : chr "" "" "" "" ...  
## $ mult\_positions : chr "" "" "" "" ...  
## $ amplicon : chr "abph1" "abph1" "ae1" "ae1" ...  
## $ cdv\_map\_feature.name: chr "AB042260" "AB042260" "ae1" "ae1" ...  
## $ gene : chr "abph1" "abph1" "ae1" "ae1" ...  
## $ candidate.random : chr "candidate" "candidate" "candidate" "candidate" ...  
## $ Genaissance\_daa\_id : int 8393 8394 8395 8396 8397 8398 8399 8400 8401 8402 ...  
## $ Sequenom\_daa\_id : int 10474 10475 10477 10478 10479 10481 10482 10483 10486 10487 ...  
## $ count\_amplicons : int 1 0 1 0 0 1 1 0 1 0 ...  
## $ count\_cmf : int 1 0 1 0 0 1 0 0 1 0 ...  
## $ count\_gene : int 1 0 1 0 0 1 1 0 1 0 ...

glimpse(pos)

## Rows: 983  
## Columns: 15  
## $ SNP\_ID <chr> "abph1.20", "abph1.22", "ae1.3", "ae1.4", "ae1.5"…  
## $ cdv\_marker\_id <int> 5976, 5978, 6605, 6606, 6607, 5982, 3463, 3466, 5…  
## $ Chromosome <chr> "2", "2", "5", "5", "5", "1", "3", "3", "4", "4",…  
## $ Position <chr> "27403404", "27403892", "167889790", "167889682",…  
## $ alt\_pos <chr> "", "", "", "", "", "", "", "", "", "", "", "", "…  
## $ mult\_positions <chr> "", "", "", "", "", "", "", "", "", "", "", "", "…  
## $ amplicon <chr> "abph1", "abph1", "ae1", "ae1", "ae1", "an1", "ba…  
## $ cdv\_map\_feature.name <chr> "AB042260", "AB042260", "ae1", "ae1", "ae1", "an1…  
## $ gene <chr> "abph1", "abph1", "ae1", "ae1", "ae1", "an1", "ba…  
## $ candidate.random <chr> "candidate", "candidate", "candidate", "candidate…  
## $ Genaissance\_daa\_id <int> 8393, 8394, 8395, 8396, 8397, 8398, 8399, 8400, 8…  
## $ Sequenom\_daa\_id <int> 10474, 10475, 10477, 10478, 10479, 10481, 10482, …  
## $ count\_amplicons <int> 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1…  
## $ count\_cmf <int> 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1…  
## $ count\_gene <int> 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1…

colnames(pos)[1:6]

## [1] "SNP\_ID" "cdv\_marker\_id" "Chromosome" "Position"   
## [5] "alt\_pos" "mult\_positions"

pos[1:6,1:6]

## SNP\_ID cdv\_marker\_id Chromosome Position alt\_pos mult\_positions  
## 1 abph1.20 5976 2 27403404   
## 2 abph1.22 5978 2 27403892   
## 3 ae1.3 6605 5 167889790   
## 4 ae1.4 6606 5 167889682   
## 5 ae1.5 6607 5 167889821   
## 6 an1.4 5982 1 240498509

pos[pos == "unknown"] <- NA  
pos[pos == "multiple"] <- NA  
pos %>%   
 group\_by(Chromosome) %>%   
 summarise(Max=max(Position, na.rm = T), Min=min(Position, na.rm = T), Number=length(Position))

## # A tibble: 11 × 4  
## Chromosome Max Min Number  
## <chr> <chr> <chr> <int>  
## 1 1 "95897171" "10069039" 155  
## 2 10 "96216463" "10432605" 53  
## 3 2 "69623323" "10429605" 127  
## 4 3 "95541392" "106631676" 107  
## 5 4 "78946482" "103665461" 91  
## 6 5 "945545" "100227859" 122  
## 7 6 "98507715" "113705211" 76  
## 8 7 "43948320" "104898448" 97  
## 9 8 "83913342" "115257234" 62  
## 10 9 "94285743" "104237516" 60  
## 11 <NA> "" "" 33

## About snp\_position.txt file:

* It has 983 rows and 15 column.

### Data processing

#### Subset the pos data frame to keep the SNP\_ID, Chr, and Pos columns for the merging purpose

* Adjust the variable type of chromosome and Position to be numeric and remove the unknown and multiple which are regarded as NAs out to be posred dataframe.
* This following code is performing several operations on the data frame pos using the dplyr package and assigning the resulting data frame to a new object called posred. The str() function is then used to display the structure of posred.

Let’s break down the individual operations:

dplyr::select(SNP\_ID, Chromosome, Position): This selects only the columns named SNP\_ID, Chromosome, and Position from the pos data frame.

mutate(Chromosome=as.numeric(Chromosome), Position=as.numeric(Position)): This converts the Chromosome and Position columns to numeric data types using the as.numeric() function. The resulting numeric values are assigned back to the same columns.

filter\_all(all\_vars(. != "NA")): This filters out any rows where any column has a missing value (“NA”). The filter\_all() function applies the same condition to all columns, and all\_vars(. != “NA”) specifies the condition that all values in each column must not be equal to “NA”. The . represents the current column being evaluated.

Overall, this code is creating a new data frame posred that contains only the selected columns from pos, with Chromosome and Position converted to numeric data types, and any rows containing missing values are removed. The str() function is then used to display the structure of the resulting data frame.

posred <- pos %>%   
 dplyr::select(SNP\_ID, Chromosome, Position) %>%   
 mutate(Chromosome=as.numeric(Chromosome),  
 Position=as.numeric(Position))%>%   
 filter\_all(all\_vars(. != "NA"))   
str(posred)

## 'data.frame': 939 obs. of 3 variables:  
## $ SNP\_ID : chr "abph1.20" "abph1.22" "ae1.3" "ae1.4" ...  
## $ Chromosome: num 2 2 5 5 5 1 3 3 4 4 ...  
## $ Position : num 2.74e+07 2.74e+07 1.68e+08 1.68e+08 1.68e+08 ...

#### Subset the genotype data into maize and teosinte datasets

* Subset the maize and teosinte genotypes by the Group column. Either [ which()] or %>% filter() works out.

maize <- genotype[which(genotype$Group=="ZMMIL" | genotype$Group =="ZMMLR" | genotype$Group == "ZMMMR"),]  
teosinte <-genotype[which(genotype$Group=="ZMPBA" | genotype$Group =="ZMPIL" | genotype$Group == "ZMPJA"),]

* Another method:

maize <- genotype %>% filter(Group=="ZMMIL" | Group =="ZMMLR" | Group == "ZMMMR")  
teosinte <-genotype %>% filter(Group=="ZMPBA" | Group =="ZMPIL" | Group == "ZMPJA")

#### Formatting the maize genotype with SNP information by merging the posred and maize data

* Transform the maize data for merging with posred by SNP\_ID column, and descend the Chromosome and Position.

maize <- maize[,c(-2,-3)]  
maize[1:6,1:6] ## have a look

## Sample\_ID abph1.20 abph1.22 ae1.3 ae1.4 ae1.5  
## 1210 ZDP\_0752a C/G A/A T/T G/G C/C  
## 1211 ZDP\_0793a C/G A/A T/T G/G C/T  
## 1212 ZDP\_0612a C/C A/A T/T G/G C/C  
## 1213 ZDP\_0602a C/G A/A G/T A/G C/T  
## 1214 ZDP\_0581a C/C A/A T/T G/G C/T  
## 1215 ZDP\_0552a C/G A/A T/T G/G C/T

maize <- t(maize)  
maize <- cbind(rownames(maize),maize)  
rownames(maize) <- NULL  
colnames(maize) <- maize[1,]  
maize <- maize[-1,]  
maize <- as.data.frame(maize)  
colnames(maize)[1] <- "SNP\_ID"   
maizewp <- merge(posred, maize, by = "SNP\_ID")  
maizewp <- maizewp %>% arrange(Chromosome,Position)   
## maize genotypes with SNP position information

* This above code is performing several operations on the maize data frame and the posred data frame, and assigning the resulting data frame to a new object maizewp.

Here’s a breakdown of the individual operations:

* maize <- maize[,c(-2,-3)]: This selects all columns of the maize data frame except for the second and third columns, and reassigns the resulting data frame to maize.
* maize <- t(maize): This transposes the maize data frame.
* maize <- cbind(rownames(maize),maize): This adds a new column to the maize data frame containing the row names of the original data frame.
* rownames(maize) <- NULL: This removes the row names from the maize data frame.
* colnames(maize) <- maize[1,]: This sets the column names of the maize data frame to the values in the first row.
* maize <- maize[-1,]: This removes the first row of the maize data frame.
* maize <- as.data.frame(maize): This converts maize to a data frame.
* colnames(maize)[1] <- "SNP\_ID": This renames the first column of maize to “SNP\_ID”.
* maizewp <- merge(posred, maize, by = "SNP\_ID"): This merges the posred data frame with the modified maize data frame using the common column “SNP\_ID”. The resulting data frame is assigned to maizewp.
* maizewp <- maizewp %>% arrange(Chromosome,Position): This arranges the rows of maizewp in ascending order of Chromosome and Position using the arrange() function from the dplyr package.

Overall, this code is modifying the maize data frame by removing certain columns, transposing the data frame, and adding and removing row and column names. It then merges the modified maize data frame with the posred data frame and sorts the resulting data frame by Chromosome and Position. The resulting data frame is assigned to maizewp.

#### Formatting the teosinte genotype with SNP information by merging the posred and teosinte data

* The same methods as with maize for merging data frame.

teosinte <- teosinte[,c(-2,-3)]  
teosinte <- t(teosinte)  
teosinte <- cbind(rownames(teosinte),teosinte)  
rownames(teosinte) <- NULL  
colnames(teosinte) <- teosinte[1,]  
teosinte <- teosinte[-1,]  
teosinte <- as.data.frame(teosinte)  
colnames(teosinte)[1] <- "SNP\_ID"   
teosintewp <- merge(posred, teosinte, by = "SNP\_ID")  
teosintewp <- teosintewp %>% arrange(Chromosome,Position)

#### Splitting the maize data into different files by the chromosomes and SNP positions.

* The followings are using loop to separate the maizewp and teosintewp data frames to 10, 10, 10, and 10 files, respectively, by the chromosome and SNP positions in total 40 files. Also, change the missing genotype to be ? or -.

chr <- 1:10  
for (i in chr) {  
 files\_inc <- maizewp[maizewp$Chromosome == i,]  
 files\_inc[files\_inc == "?/?"] <- "?"  
 if (i < 10) { write.table(files\_inc, file = paste("Maize\_Chr0",i,"\_increase.txt",sep=""),row.names = FALSE,sep = "\t",quote = FALSE) }  
 else {write.table(files\_inc, file = paste("Maize\_Chr",i,"\_increase.txt",sep=""),row.names = FALSE, sep = "\t",quote = FALSE)}  
   
 files\_dec <- maizewp[maizewp$Chromosome == i,]  
 files\_dec[files\_dec == "?/?"] <- "-"  
 files\_dec <- files\_dec %>% arrange(desc(Chromosome),desc(Position))  
 if (i < 10) { write.table(files\_dec, file = paste("Maize\_Chr0",i,"\_decrease.txt",sep=""),row.names = FALSE,sep = "\t",quote = FALSE) }  
 else {write.table(files\_dec, file = paste("Maize\_Chr",i,"\_decrease.txt",sep=""),row.names = FALSE, sep = "\t",quote = FALSE)}  
}

#### Splitting the teosinte data into different files by chromosomes and SNP positions.

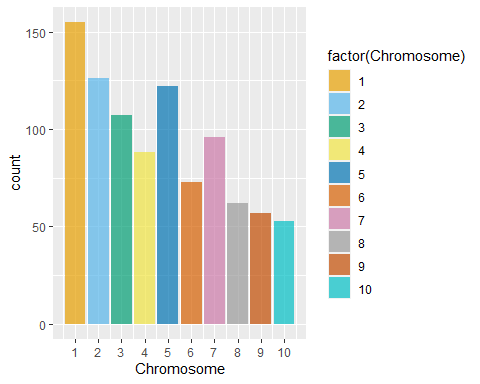
chr <- 1:10  
for (i in chr) {  
 files\_inc <- teosintewp[teosintewp$Chromosome == i,]  
 files\_inc[files\_inc == "?/?"] <- "?"  
 if (i < 10) { write.table(files\_inc, file = paste("Teosinte\_Chr0",i,"\_increase.txt",sep=""),row.names = FALSE,sep = "\t",quote = FALSE) }  
 else {write.table(files\_inc, file = paste("Teosinte\_Chr",i,"\_increase.txt",sep=""),row.names = FALSE, sep = "\t",quote = FALSE)}  
   
 files\_dec <- teosintewp[teosintewp$Chromosome == i,]  
 files\_dec[files\_dec == "?/?"] <- "-"  
 files\_dec <- files\_dec %>% arrange(desc(Chromosome),desc(Position))  
 if (i < 10) { write.table(files\_dec, file = paste("Teosinte\_Chr0",i,"\_decrease.txt",sep=""),row.names = FALSE,sep = "\t",quote = FALSE) }  
 else {write.table(files\_dec, file = paste("Teosinte\_Chr",i,"\_decrease.txt",sep=""),row.names = FALSE, sep = "\t",quote = FALSE)}  
}

## Part II

### Plotting

#### SNPs per chromosome

pos %>%   
 dplyr::select(SNP\_ID, Chromosome, Position) %>%  
 drop\_na() %>%  
 mutate(Chromosome = as.double(Chromosome)) %>%   
 ggplot() +  
 geom\_bar(aes(x = Chromosome, fill = factor(Chromosome)), alpha = 0.7) +  
 scale\_x\_continuous(breaks = 1:10) +  
 scale\_fill\_manual(values = c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7", "#999999", "#C14B00", "#00BFC4"))



# Explanation of the above script:

This is a series of operations using the *dplyr* and *ggplot2* packages to create a bar plot of the number of SNPs on each chromosome in a data frame called pos. Here’s an explanation of each operation:

* pos %>%: %>% is the pipe operator, which takes the output of the left-hand side of the pipe and passes it as the first argument to the function on the right-hand side of the pipe. In this case, it takes the pos data frame and passes it as the input to the next function.
* dplyr::select(SNP\_ID, Chromosome, Position) %>%: The select() function from dplyr is used to select only the columns “SNP\_ID”, “Chromosome”, and “Position” from the data frame. The dplyr:: syntax specifies that we want to use the select() function from the dplyr package. The %>% operator passes the output of this function to the next function.
* drop\_na() %>%: The drop\_na() function from dplyr is used to remove any rows that contain missing values (“NA”) in any of the columns. The %>% operator passes the output of this function to the next function.
* mutate(Chromosome=as.double(Chromosome)) %>%: The mutate() function from dplyr is used to create a new column called “Chromosome” that converts the “Chromosome” column to a numeric data type using the as.double() function. The %>% operator passes the output of this function to the next function.
* ggplot()+: The ggplot() function from ggplot2 is used to initialize a new plot. The + operator is used to add layers to the plot.
* geom\_bar(aes(x=Chromosome))+: The geom\_bar() function from ggplot2 is used to create a bar plot of the number of SNPs on each chromosome. The aes() function is used to map the “Chromosome” column to the x-axis. The + operator is used to add the geom\_bar() layer to the plot.
* scale\_x\_continuous(breaks = 1:10): The scale\_x\_continuous() function from ggplot2 is used to set the breaks (i.e., tick marks) on the x-axis to the integers 1 through 10. This function specifies that the x-axis should be a continuous scale.

#### Missing data and amount of heterozygosity

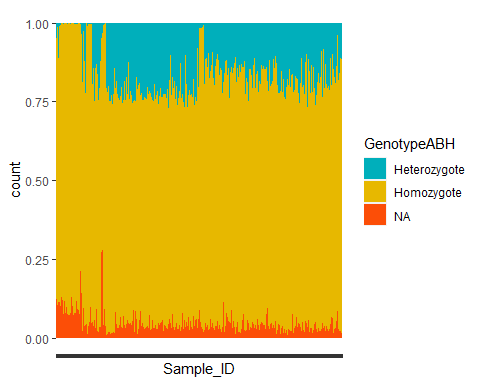
* This chunk is the first version code for substituting the genotypes.

## create a function to detect the SNP genotypes  
ABH <- function(x) {  
 if ( x == "A/A" | x == "C/C" | x == "G/G" | x == "T/T") {  
 return("A|B")  
 }  
 else if (x == "?/?") {  
 return("NA")  
 }  
 else {return("H")}  
}  
ABH\_V <- Vectorize(ABH) ## make the function be a vectorized function   
genotype3 <- genotype2 %>%   
 pivot\_longer(3:last\_col(), names\_to = "SNP", values\_to = "Genotype") %>%   
 mutate( GenotypeABH = ABH\_V(Genotype))

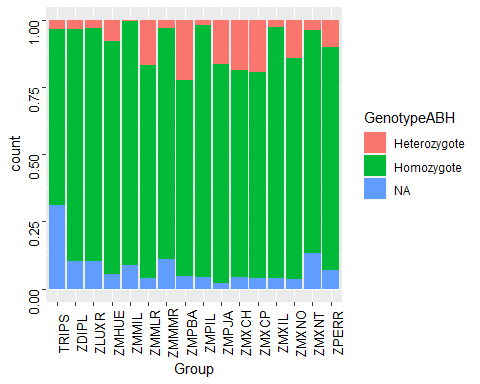
color\_Palette = c("#00AFBB", "#E7B800", "#FC4E07")  
genotype2 <- genotype[,-2]  
genotype2[1:6,1:6]

## Sample\_ID Group abph1.20 abph1.22 ae1.3 ae1.4  
## 1 SL-15 TRIPS ?/? ?/? T/T G/G  
## 2 SL-16 TRIPS ?/? ?/? T/T ?/?  
## 3 SL-11 TRIPS ?/? ?/? T/T G/G  
## 4 SL-12 TRIPS ?/? ?/? T/T G/G  
## 5 SL-18 TRIPS ?/? ?/? T/T G/G  
## 6 SL-2 TRIPS ?/? ?/? T/T G/G

genotype3 <- genotype2 %>%   
 pivot\_longer(3:last\_col(), names\_to = "SNP", values\_to = "Genotype") %>%   
 mutate( GenotypeABH = ifelse(Genotype%in% c("C/C", "G/G", "A/A", "T/T"), "Homozygote", ifelse(Genotype == "?/?", "NA", "Heterozygote")))   
ggplot(genotype3)+  
 geom\_bar(aes(x=Sample\_ID, fill=GenotypeABH), position = "fill", width=1)+  
 scale\_x\_discrete(labels=NULL)+  
 scale\_fill\_manual(values = c("#00AFBB", "#E7B800", "#FC4E07")) ## change the color palette



ggplot(genotype3)+  
 geom\_bar(aes(x=Group, fill=GenotypeABH), position = "fill")+  
 theme(axis.text = element\_text( angle =90, color="black", size=10, face=1))



#### The distribution of SNP maker postions on each of chromosomes

sample\_size = posred %>% group\_by(Chromosome) %>% summarize(num=n())  
library(viridis)

## Loading required package: viridisLite

posred %>%  
 left\_join(sample\_size) %>%  
 mutate(myaxis = paste0(Chromosome, "\n", "n=", num)) %>%  
 ggplot( aes(x=myaxis, y=Position, fill=as.character(Chromosome)))+  
 geom\_violin(width=1.4) +  
 geom\_boxplot(width=0.1, color="grey", alpha=0.2) +  
 scale\_fill\_viridis(discrete = TRUE) +  
 theme(  
 legend.position="none",  
 plot.title = element\_text(size=11)  
 ) +  
 ggtitle("The distribution of SNP postion on each chromosomes") +  
 xlab("Chromosome")

## Joining with `by = join\_by(Chromosome)`

