R Assignment

Data Inspection

Attributes of `fang_et_al_genotypes`

I ran the following code to analyze the fang_et_al_genotypes.txt file.

I first loaded the needed libraries and cleared the workspace.

```
rm(list = ls())
library(tidyr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     1.0.4
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.1     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

I loaded the fang_et_al_genotypes.txt for data inspection.

```
file_path <- "fang_et_al_genotypes.txt"
raw_data <- read_delim(file_path, delim="\t")
```

```
## Rows: 2782 Columns: 986
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (986): Sample_ID, JG_OTU, Group, abph1.20, abph1.22, ae1.3, ae1.4, ae1.5...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

I checked how large the file size is in MB

```
file_size <- file.info(file_path)[1,1]/1000000
print(file_size)
```

```
## [1] 11.05194
```

I inspected the data frame dimensions (returns rows and columns)

```
dim(raw_data)
```

```
## [1] 2782  986
```

I checked the first few and last few lines of the file to check for metadata

```
head(raw_data)
```

```
## # A tibble: 6 x 986
##   Sample_ID JG_OTU   Group abph1.20 abph1.22 ae1.3 ae1.4 ae1.5 an1.4 ba1.6 ba1.9
##   <chr>     <chr>    <chr> <chr>    <chr>    <chr> <chr> <chr> <chr> <chr> <chr>
## 1 SL-15     T-aust-1 TRIPS ?/?      ?/?      T/T   G/G   T/T   C/C   ?/?   G/G
## 2 SL-16     T-aust-2 TRIPS ?/?      ?/?      T/T   ?/?   T/T   C/C   A/G   G/G
## 3 SL-11     T-brav-1 TRIPS ?/?      ?/?      T/T   G/G   T/T   ?/?   G/G   G/G
## 4 SL-12     T-brav-2 TRIPS ?/?      ?/?      T/T   G/G   T/T   C/C   G/G   G/G
```

```
## 5 SL-18     T-cund   TRIPS ?/?      ?/?      T/T   G/G   T/T   C/C   ?/?   G/G
## 6 SL-2      T-dact-1 TRIPS ?/?      ?/?      T/T   G/G   T/T   C/C   A/G   G/G
## # i 975 more variables: bt2.5 <chr>, bt2.7 <chr>, bt2.8 <chr>, Fea2.1 <chr>,
## #   Fea2.5 <chr>, id1.3 <chr>, lg2.11 <chr>, lg2.2 <chr>, pbf1.1 <chr>,
## #   pbf1.2 <chr>, pbf1.3 <chr>, pbf1.5 <chr>, pbf1.6 <chr>, pbf1.7 <chr>,
## #   pbf1.8 <chr>, PZA00003.11 <chr>, PZA00004.2 <chr>, PZA00005.8 <chr>,
## #   PZA00005.9 <chr>, PZA00006.13 <chr>, PZA00006.14 <chr>, PZA00008.1 <chr>,
## #   PZA00010.5 <chr>, PZA00013.10 <chr>, PZA00013.11 <chr>, PZA00013.9 <chr>,
## #   PZA00015.4 <chr>, PZA00017.1 <chr>, PZA00018.5 <chr>, ...
```

```
tail(raw_data)
```

```
## # A tibble: 6 x 986
##   Sample_ID JG_OTU   Group abph1.20 abph1.22 ae1.3 ae1.4 ae1.5 an1.4 ba1.6 ba1.9
##   <chr>     <chr>    <chr> <chr>    <chr>    <chr> <chr> <chr> <chr> <chr> <chr>
## 1 SYN262    Zmm-IL-~ ZMMIL C/C      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 2 S0398     Zmm-IL-~ ZMMIL G/G      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 3 S1636     Zmm-IL-~ ZMMIL G/G      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 4 CU0201    Zmm-IL-~ ZMMIL C/C      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 5 S0215     Zmm-IL-~ ZMMIL G/G      A/A      T/T   ?/?   C/C   C/C   G/G   G/G
## 6 CU0202    Zmm-IL-~ ZMMIL C/C      A/A      T/T   G/G   C/C   C/C   ?/?   G/G
## # i 975 more variables: bt2.5 <chr>, bt2.7 <chr>, bt2.8 <chr>, Fea2.1 <chr>,
## #   Fea2.5 <chr>, id1.3 <chr>, lg2.11 <chr>, lg2.2 <chr>, pbf1.1 <chr>,
## #   pbf1.2 <chr>, pbf1.3 <chr>, pbf1.5 <chr>, pbf1.6 <chr>, pbf1.7 <chr>,
## #   pbf1.8 <chr>, PZA00003.11 <chr>, PZA00004.2 <chr>, PZA00005.8 <chr>,
## #   PZA00005.9 <chr>, PZA00006.13 <chr>, PZA00006.14 <chr>, PZA00008.1 <chr>,
## #   PZA00010.5 <chr>, PZA00013.10 <chr>, PZA00013.11 <chr>, PZA00013.9 <chr>,
## #   PZA00015.4 <chr>, PZA00017.1 <chr>, PZA00018.5 <chr>, ...
```

I checked the file encoding using the unix command

```
command <- sprintf("file %s",file_path)
command_output <- system(command, intern = TRUE)
print(command_output)
```

```
## [1] "fang_et_al_genotypes.txt: ASCII text, with very long lines (10846)"
```

I also checked the head and tail data for the last columns since it was cutoff previously.

```
get_last_columns <- function(data, columns_to_return = 9) {
  last_column <- ncol(data)
  last_n_columns <- data.frame(data[,(last_column-columns_to_return+1):last_column])
  return(last_n_columns)
}
last_columns_data <- get_last_columns(raw_data)

head(last_columns_data)
```

```
##   te1.4 zagl1.1 zagl1.6 zap1.2 zen1.1 zen1.2 zen1.4 zfl2.6 zmm3.4
## 1   G/G     C/C     C/C    A/A    ?/?    ?/?    ?/?    G/G    T/T
## 2   G/G     C/C     C/C    A/A    ?/?    ?/?    ?/?    G/G    T/T
## 3   G/G     C/C     C/C    A/A    ?/?    A/A    ?/?    G/G    T/T
## 4   G/G     A/C     C/C    A/A    ?/?    A/A    ?/?    G/G    T/T
## 5   G/G     ?/?     C/C    A/A    ?/?    A/G    ?/?    ?/?    T/T
## 6   G/G     A/C     C/C    A/A    ?/?    A/A    ?/?    G/G    T/T
```

```
tail(last_columns_data)
```

```
##      te1.4 zagl1.1 zagl1.6 zap1.2 zen1.1 zen1.2 zen1.4 zfl2.6 zmm3.4
## 2777   G/G     A/A     T/T    G/G    G/G    A/A    T/T    G/G    C/C
## 2778   T/T     A/A     T/T    G/G    C/C    G/G    C/C    G/G    C/C
## 2779   ?/?     A/A     T/T    G/G    C/C    G/G    C/C    G/G    C/C
## 2780   T/T     A/A     T/T    A/A    C/C    G/G    C/C    C/C    C/C
## 2781   T/T     A/A     T/T    A/A    G/G    A/A    T/T    G/G    C/C
## 2782   G/G     ?/?     T/T    A/A    C/C    G/G    C/C    G/G    C/C
```

I checked to see if any of the SNPs had no missing data and I also checked for maximums and minimums

```
count_missing <- function(data) {
  length(grep("\\?",data))
}
SNP_missing_count <- sapply(raw_data[,4:ncol(raw_data)],count_missing)
tabulated_SNP<- sort(SNP_missing_count)
head(tabulated_SNP, n = 3)
```

```
## PZA00538.8 PZB00175.5 PZA00731.7
##          6          7          9
```

```
tail(tabulated_SNP, n = 3)
```

```
##  PZD00073.1 PZA00307.17  PZB01223.4
##         449         487         509
```

I checked if there were any samples which did not have missing data as well as finding the samples with the most and least missing data.

```
sample_missing_count <- apply(raw_data,1,count_missing)
row_names <- unlist(raw_data[,1],use.names = FALSE)
names(sample_missing_count) <- row_names
head(sample_missing_count, n=10)
```

```
## SL-15 SL-16 SL-11 SL-12 SL-18  SL-2  SL-4  SL-6  SL-7  SL-5
##   332   345   241   268   317   324   326   322   317   319
```

```
sample_missing_count[sample_missing_count == 0]
```

```
## named integer(0)
```

```
sample_missing_count[sample_missing_count == max(sample_missing_count)]
```

```
## TAMex0516.1
##         398
```

```
sample_missing_count[sample_missing_count == min(sample_missing_count)]
```

```
## DOE5H-1556
##          5
```

I checked what the most common gene groups were:

```
groups <- unlist(raw_data[,3],use.names = FALSE)
tabulated_groups<- sort(table(groups))
head(tabulated_groups, n = 3)
```

```
## groups
## ZMXNT ZMXIL ZMXNO
##     4     6     7
```

```
tail(tabulated_groups, n = 3)
```

```
## groups
## ZMMIL ZMPBA ZMMLR
##   290   900  1256
```

By inspecting this file I learned that:

- There are 2,783 lines in the file with a file size of 11.06 MB. Thus, there are not many lines, but there appear to be a lot of words and characters relatively speaking. This is due to there being a large number of columns on each line, one for each single nucleotide polymorphisms (SNP) in the sample along with some metadata.
- Since only the first line of the file contained metadata with the head camp, and since the tail doesn't show any metadata, it appears there is only metadata for the first row. This implies there are 2,782 samples in the SNP dataset.
- Inspecting the number of columns, I got 986. Since the first three columns are metadata (sample ID, JG_OTU, and group), this appears to imply each sample has info on 983 SNPs.
- There is missing data for some of the SNPs with "?/?" appearing. I couldn't find any sample which did not have "?/?" for at least one SNP. Thus, I assume this is pretty common.
  - The most common SNP with missing data is PZB01223.4 at 509 samples with missing data at that location while the PZA00538.8 SNP has the least missing data at only 6.
  - The most common sample with missing data is TAMex0516.1 at 398 SNPs with missing data while DOE5H-1556 has the lowest at 5.
- The file is ASCII text, with very long lines.
- There were no duplicate gene types, but there were duplicate groups with the largest being the ZMMLR group at 1256 members. The smallest was the ZMXNT group at 4 members.

Attributes of `snp_position.txt`

I ran the following code to analyze the snp_position.txt file. I first saved the file data as a dataframe

```
file_path <- "snp_position.txt"
raw_data <- read_delim(file_path, delim="\t")
```

```
## Rows: 983 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (9): SNP_ID, Chromosome, Position, alt_pos, mult_positions, amplicon, cd...
## dbl (6): cdv_marker_id, Genaissance_daa_id, Sequenom_daa_id, count_amplicons...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

I checked how large the file size is in MB.

```
file_size <- file.info(file_path)[1,1]/1000000
print(file_size)
```

```
## [1] 0.082763
```

I inspected the data frame dimensions (returns rows and columns)

```
dim(raw_data)
```

```
## [1] 983  15
```

I checked the first few and last few lines of the file to check for metadata

```r
head(raw_data)
```

```
## # A tibble: 6 x 15
##   SNP_ID   cdv_marker_id Chromosome Position  alt_pos mult_positions amplicon
##   <chr>            <dbl> <chr>      <chr>     <chr>   <chr>          <chr>
## 1 abph1.20          5976 2          27403404  <NA>    <NA>           abph1
## 2 abph1.22          5978 2          27403892  <NA>    <NA>           abph1
## 3 ae1.3             6605 5          167889790 <NA>    <NA>           ae1
## 4 ae1.4             6606 5          167889682 <NA>    <NA>           ae1
## 5 ae1.5             6607 5          167889821 <NA>    <NA>           ae1
## 6 an1.4             5982 1          240498509 <NA>    <NA>           an1
## # i 8 more variables: cdv_map_feature.name <chr>, gene <chr>,
## #   `candidate/random` <chr>, Genaissance_daa_id <dbl>, Sequenom_daa_id <dbl>,
## #   count_amplicons <dbl>, count_cmf <dbl>, count_gene <dbl>
```

```r
tail(raw_data)
```

```
## # A tibble: 6 x 15
##   SNP_ID cdv_marker_id Chromosome Position  alt_pos mult_positions amplicon
##   <chr>          <dbl> <chr>      <chr>     <chr>   <chr>          <chr>
## 1 zap1.2          3514 2          233128584 <NA>    <NA>           zap1
## 2 zen1.1          3519 unknown    unknown   <NA>    <NA>           zen1
## 3 zen1.2          3520 unknown    unknown   <NA>    <NA>           zen1
## 4 zen1.4          3521 unknown    unknown   <NA>    <NA>           zen1
## 5 zfl2.6          6463 2          12543294  <NA>    <NA>           zfl2
## 6 zmm3.4          3527 9          16966348  <NA>    <NA>           zmm3
## # i 8 more variables: cdv_map_feature.name <chr>, gene <chr>,
## #   `candidate/random` <chr>, Genaissance_daa_id <dbl>, Sequenom_daa_id <dbl>,
## #   count_amplicons <dbl>, count_cmf <dbl>, count_gene <dbl>
```

I checked the file encoding using the unix command.

```r
command <- sprintf("file %s",file_path)
command_output <- system(command, intern = TRUE)
print(command_output)
```

```
## [1] "snp_position.txt: ASCII text"
```

I also checked the head and tail data for the last columns since it was cutoff previously.

```r
get_last_columns <- function(data, columns_to_return = 9) {
  last_column <- ncol(data)
  last_n_columns <- data.frame(data[,(last_column-columns_to_return+1):last_column])
  return(last_n_columns)
}
last_columns_data <- get_last_columns(raw_data)

head(last_columns_data)
```

```
##   amplicon cdv_map_feature.name  gene candidate.random Genaissance_daa_id
## 1    abph1             AB042260 abph1        candidate               8393
## 2    abph1             AB042260 abph1        candidate               8394
## 3      ae1                  ae1   ae1        candidate               8395
## 4      ae1                  ae1   ae1        candidate               8396
## 5      ae1                  ae1   ae1        candidate               8397
## 6      an1                  an1   an1        candidate               8398
##   Sequenom_daa_id count_amplicons count_cmf count_gene
```

```
## 1           10474               1         1          1
## 2           10475               0         0          0
## 3           10477               1         1          1
## 4           10478               0         0          0
## 5           10479               0         0          0
## 6           10481               1         1          1
```

```
tail(last_columns_data)
```

```
##     amplicon cdv_map_feature.name gene candidate.random Genaissance_daa_id
## 978     zap1               L46400 zap1        candidate               8434
## 979     zen1             CF649098 zen1        candidate               8435
## 980     zen1             CF649098 zen1        candidate               8436
## 981     zen1             CF649098 zen1        candidate               8437
## 982     zfl2                 zfl2 zfl2        candidate               8438
## 983     zmm3               Y09301 zmm3        candidate              10104
##     Sequenom_daa_id count_amplicons count_cmf count_gene
## 978           11823               1         0          1
## 979           11824               1         1          1
## 980           11826               0         0          0
## 981           11827               0         0          0
## 982           11828               1         1          1
## 983           11829               1         0          1
```

I checked what the most common chromosomes were:

```
chromosomes <- unlist(raw_data[,3],use.names = FALSE)
tabulated_chromosomes<- sort(table(chromosomes))
head(tabulated_chromosomes, n = 3)
```

```
## chromosomes
## multiple  unknown       10
##        6       27       53
```

```
tail(tabulated_chromosomes, n = 3)
```

```
## chromosomes
##   5   2   1
## 122 127 155
```

I checked what the most common genes were:

```
genes <- unlist(raw_data[,9],use.names = FALSE)
tabulated_genes<- sort(table(genes))
head(tabulated_genes, n = 3)
```

```
## genes
##   AI622483        an1 ath-miR167
##          1          1          1
```

```
tail(tabulated_genes, n = 3)
```

```
## genes
##     zag1 PZA03450    zmm28
##        8        9       11
```

By inspecting this file I learned that:

- There are 984 lines in the file with a file size of 83 KB. Thus, the file is considerably smaller than the fang_et_al_genotypes.txt file as expected.

- The file doesn't appear to have any metadata with the only the first line providing the column names, with there appearing to be 15 columns. Thus, there appear to be 983 individual SNPs documented in the file.
  - From looking at the first 15 lines in the file (only the first 9 columns) it did not appear that there was any missing data. I would search for it, but since it is not obvious how it would be documented, I can't easily grep for it.
- The file is ASCII text, without very long lines.
- The most common chromosomes containing a SNP were chromosomes 1 (155 SNPs), 2 (127 SNPs) and 5 (122 SNPs). The lowest was chromosome 10 at 53 SNPs.
- The most common genes containing SNPs were zmm28 (11 SNPs), PZA03450 (9 SNPs), and zag1 (8 SNPs).

Data Processing

Maize Data Processing

I processed the maize data into the requested files using the following code. The output files are stored under the "maize" subfolder in the repository.

I first removed the output folders and files if they already exist. This is because when I'm making the chromosome subfiles, I append data to the end of them. Thus, if the files already exist, it will duplicate the data in the file.

```
main_dir <- getwd()
sub_dir <- "maize"
output_dir <- paste(main_dir,sub_dir, sep="/")
unlink(output_dir, recursive = TRUE)
```

I next create the output folders and the increasing/decreasing subfolders for the desired output files.

```
main_dir <- paste(getwd(),"maize", sep="/")
sub_dir <- c("increasing","decreasing")
list_of_paths <- paste(main_dir,sub_dir, sep="/")
sapply(list_of_paths,dir.create, recursive = TRUE)
```

```
## /Users/ryangodin/Downloads/bcb5460_R_assignment/maize/increasing
##                                                            TRUE
## /Users/ryangodin/Downloads/bcb5460_R_assignment/maize/decreasing
##                                                            TRUE
```

I next prepare the raw data from fang_et_al_genotypes.txt by extracting it the desired columns, arranging them, and transposing them.

```
file_path <- "fang_et_al_genotypes.txt"
raw_data <- read_delim(file_path, delim="\t")
```

```
## Rows: 2782 Columns: 986
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (986): Sample_ID, JG_OTU, Group, abph1.20, abph1.22, ae1.3, ae1.4, ae1.5...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
extracted_data <- filter(raw_data, `Group` %in% c("ZMMIL", "ZMMLR", "ZMMMR"))
extracted_data <- arrange(extracted_data, `Sample_ID`)
transposed_data <- as_tibble(t(extracted_data[,4:ncol(extracted_data)]),rownames = "SNP_ID")
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
```

```
## `.name_repair` is omitted as of tibble 2.0.0.
## i Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
colnames(transposed_data) <- c("SNP_ID",unlist(extracted_data[1:nrow(extracted_data),1]), use.names = F
head(extracted_data)
```

```
## # A tibble: 6 x 986
##   Sample_ID JG_OTU   Group abph1.20 abph1.22 ae1.3 ae1.4 ae1.5 an1.4 ba1.6 ba1.9
##   <chr>     <chr>    <chr> <chr>    <chr>    <chr> <chr> <chr> <chr> <chr> <chr>
## 1 BKN_001   Zmm-LR-~ ZMMLR C/C      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 2 BKN_002   Zmm-LR-~ ZMMLR C/C      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 3 BKN_003   Zmm-LR-~ ZMMLR C/C      A/A      T/T   G/G   C/C   C/C   ?/?   G/G
## 4 BKN_004   Zmm-LR-~ ZMMLR C/C      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 5 BKN_009   Zmm-MR2~ ZMMMR C/C      A/A      T/T   G/G   T/T   C/C   G/G   G/G
## 6 BKN_010   Zmm-MR2~ ZMMMR C/C      A/A      T/T   G/G   C/C   G/G   G/G   G/G
## # i 975 more variables: bt2.5 <chr>, bt2.7 <chr>, bt2.8 <chr>, Fea2.1 <chr>,
## #   Fea2.5 <chr>, id1.3 <chr>, lg2.11 <chr>, lg2.2 <chr>, pbf1.1 <chr>,
## #   pbf1.2 <chr>, pbf1.3 <chr>, pbf1.5 <chr>, pbf1.6 <chr>, pbf1.7 <chr>,
## #   pbf1.8 <chr>, PZA00003.11 <chr>, PZA00004.2 <chr>, PZA00005.8 <chr>,
## #   PZA00005.9 <chr>, PZA00006.13 <chr>, PZA00006.14 <chr>, PZA00008.1 <chr>,
## #   PZA00010.5 <chr>, PZA00013.10 <chr>, PZA00013.11 <chr>, PZA00013.9 <chr>,
## #   PZA00015.4 <chr>, PZA00017.1 <chr>, PZA00018.5 <chr>, ...
```

```
head(transposed_data)
```

```
## # A tibble: 6 x 1,574
##   SNP_ID BKN_001 BKN_002 BKN_003 BKN_004 BKN_009 BKN_010 BKN_011 BKN_012 BKN_013
##   <chr>  <chr>   <chr>   <chr>   <chr>   <chr>   <chr>   <chr>   <chr>   <chr>
## 1 abph1~ C/C     C/C     C/C     C/C     C/C     C/C     C/C     ?/?     C/G
## 2 abph1~ A/A     A/A     A/A     A/A     A/A     A/A     A/A     A/A     A/A
## 3 ae1.3  T/T     T/T     T/T     T/T     T/T     T/T     T/T     G/G     T/T
## 4 ae1.4  G/G     G/G     G/G     G/G     G/G     G/G     G/G     A/A     G/G
## 5 ae1.5  C/C     C/C     C/C     C/C     T/T     C/C     C/C     T/T     C/C
## 6 an1.4  C/C     C/C     C/C     C/C     C/C     G/G     C/C     C/C     C/C
## # i 1,564 more variables: BKN_014 <chr>, BKN_015 <chr>, BKN_016 <chr>,
## #   BKN_017 <chr>, BKN_018 <chr>, BKN_019 <chr>, BKN_020 <chr>, BKN_021 <chr>,
## #   BKN_022 <chr>, BKN_023 <chr>, BKN_024 <chr>, BKN_025 <chr>, BKN_026 <chr>,
## #   BKN_027 <chr>, BKN_028 <chr>, BKN_029 <chr>, BKN_030 <chr>, BKN_031 <chr>,
## #   BKN_032 <chr>, BKN_033 <chr>, BKN_034 <chr>, BKN_035 <chr>, CU0001 <chr>,
## #   CU0002 <chr>, CU0003 <chr>, CU0004 <chr>, CU0005 <chr>, CU0006 <chr>,
## #   CU0007 <chr>, CU0008 <chr>, CU0009 <chr>, CU0010 <chr>, CU0011 <chr>, ...
```

I next load the data from the snp_position.tx file and merge the desired columns with the extracted fang_et_al_genotypes.txt data.

```
snp_data = read_delim("snp_position.txt", "\t")
```

```
## Rows: 983 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (9): SNP_ID, Chromosome, Position, alt_pos, mult_positions, amplicon, cd...
## dbl (6): cdv_marker_id, Genaissance_daa_id, Sequenom_daa_id, count_amplicons...
##
## i Use `spec()` to retrieve the full column specification for this data.
```

```
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
merged_data <- merge(snp_data[,c(1,3,4)], transposed_data, by="SNP_ID", all = TRUE)
merged_data <- as_tibble(merged_data)
head(merged_data)
```

```
## # A tibble: 6 x 1,576
##   SNP_ID   Chromosome Position  BKN_001 BKN_002 BKN_003 BKN_004 BKN_009 BKN_010
##   <chr>    <chr>      <chr>     <chr>   <chr>   <chr>   <chr>   <chr>   <chr>
## 1 abph1.20 2          27403404  C/C     C/C     C/C     C/C     C/C     C/C
## 2 abph1.22 2          27403892  A/A     A/A     A/A     A/A     A/A     A/A
## 3 ae1.3    5          167889790 T/T     T/T     T/T     T/T     T/T     T/T
## 4 ae1.4    5          167889682 G/G     G/G     G/G     G/G     G/G     G/G
## 5 ae1.5    5          167889821 C/C     C/C     C/C     C/C     T/T     C/C
## 6 an1.4    1          240498509 C/C     C/C     C/C     C/C     C/C     G/G
## # i 1,567 more variables: BKN_011 <chr>, BKN_012 <chr>, BKN_013 <chr>,
## #   BKN_014 <chr>, BKN_015 <chr>, BKN_016 <chr>, BKN_017 <chr>, BKN_018 <chr>,
## #   BKN_019 <chr>, BKN_020 <chr>, BKN_021 <chr>, BKN_022 <chr>, BKN_023 <chr>,
## #   BKN_024 <chr>, BKN_025 <chr>, BKN_026 <chr>, BKN_027 <chr>, BKN_028 <chr>,
## #   BKN_029 <chr>, BKN_030 <chr>, BKN_031 <chr>, BKN_032 <chr>, BKN_033 <chr>,
## #   BKN_034 <chr>, BKN_035 <chr>, CU0001 <chr>, CU0002 <chr>, CU0003 <chr>,
## #   CU0004 <chr>, CU0005 <chr>, CU0006 <chr>, CU0007 <chr>, CU0008 <chr>, ...
```

I sorted the data based on position, making sure to convert position to numeric before doing so.

```
merged_data$Position <- as.numeric(merged_data$Position)
```

```
## Warning: NAs introduced by coercion
```

```
merged_data <- arrange(merged_data, `Position`)
```

I next generated the chromosome files for chromsomes 1-10 for the increasing position order, omitting the NA entries that correspond to multiple or unknown chromosome positions.

```
make_chromosome_file <- function(dataset_row, species, order, missing_marker, dataset){
  dataset_row <- gsub("\\?", missing_marker, dataset_row)
  dataset_row <- as.data.frame(t(dataset_row))
  chromosome <- dataset_row["Chromosome"]
  output_file <- sprintf("%s/%s/%s/chromosome_%s.txt",getwd(),species,order,chromosome)
  if (file.exists(output_file) == FALSE) {
            write.table(
              t(colnames(dataset)), output_file, sep="\t", row.names = FALSE,
              col.names = FALSE, append = TRUE, quote = FALSE)
  }
  write.table(
    dataset_row, output_file, sep="\t", row.names = FALSE,
    col.names = FALSE, append = TRUE, quote = FALSE)
}

merged_data_filtered <- filter(merged_data, !is.na(Position))
apply(merged_data_filtered,1,make_chromosome_file,species="maize",order="increasing", missing_marker="\
```

```
## NULL
```

I next prepared the chromsome files for decreasing position order, making sure to specify the missing marker symbol as "-" not the "?" symbol.

```
merged_data_filtered <- arrange(merged_data_filtered, desc(`Position`))
apply(merged_data_filtered,1,make_chromosome_file,species="maize",order="decreasing", missing_marker="\
```

```
## NULL
```

I next prepared the chromosome files for the multiple and unknown positions.

```
make_misc_file <- function(dataset_row, species,dataset){
  dataset_row <- as.data.frame(t(dataset_row))
  chromosome <- dataset_row["Chromosome"]
  position <- dataset_row["Position"]
  output_file <- sprintf("%s/%s/chromosome_%s.txt",getwd(),species,chromosome)
  if (is.na(position) == TRUE){
    if ((chromosome %in% c("multiple", "unknown")) == FALSE){
      chromosome <- "multiple"
      output_file <- sprintf("%s/%s/chromosome_%s.txt",getwd(),species,chromosome)
    }
    if (file.exists(output_file) == FALSE) {
            write.table(
              t(colnames(dataset)), output_file, sep="\t", row.names = FALSE,
              col.names = FALSE, append = TRUE, quote = FALSE)
    }
            write.table(
              dataset_row, output_file, sep="\t", row.names = FALSE,
              col.names = FALSE, append = TRUE, quote = FALSE)
  }
}
apply(merged_data,1, make_misc_file, species="maize",dataset=merged_data)
```

```
## NULL
```

Teosinte Data Processing

I processed the teosinte data into the requested files using the following code. The output files are stored under the "teosinte" subfolder in the repository.

I first removed the output folders and files if they already exist. This is because when I'm making the chromosome subfiles, I append data to the end of them. Thus, if the files already exist, it will duplicate the data in the file.

```
main_dir <- getwd()
sub_dir <- "teosinte"
output_dir <- paste(main_dir,sub_dir, sep="/")
unlink(output_dir, recursive = TRUE)
```

I next create the output folders and the increasing/decreasing subfolders for the desired output files.

```
main_dir <- paste(getwd(),"teosinte", sep="/")
sub_dir <- c("increasing","decreasing")
list_of_paths <- paste(main_dir,sub_dir, sep="/")
sapply(list_of_paths,dir.create, recursive = TRUE)
```

```
## /Users/ryangodin/Downloads/bcb5460_R_assignment/teosinte/increasing
##                                                                TRUE
## /Users/ryangodin/Downloads/bcb5460_R_assignment/teosinte/decreasing
##                                                                TRUE
```

I next prepare the raw data from fang_et_al_genotypes.txt by extracting it the desired columns, arranging them, and transposing them.

```
file_path <- "fang_et_al_genotypes.txt"
raw_data <- read_delim(file_path, delim="\t")
```

```
## Rows: 2782 Columns: 986
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (986): Sample_ID, JG_OTU, Group, abph1.20, abph1.22, ae1.3, ae1.4, ae1.5...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
extracted_data <- filter(raw_data, `Group` %in% c("ZMPBA", "ZMPIL", "ZMPJA"))
extracted_data <- arrange(extracted_data, `Sample_ID`)
transposed_data <- as_tibble(t(extracted_data[,4:ncol(extracted_data)]),rownames = "SNP_ID")
colnames(transposed_data) <- c("SNP_ID",unlist(extracted_data[1:nrow(extracted_data),1]), use.names = FA
head(extracted_data)
```

```
## # A tibble: 6 x 986
##   Sample_ID JG_OTU   Group abph1.20 abph1.22 ae1.3 ae1.4 ae1.5 an1.4 ba1.6 ba1.9
##   <chr>     <chr>    <chr> <chr>    <chr>    <chr> <chr> <chr> <chr> <chr> <chr>
## 1 S0728     Zmp-J-J~ ZMPJA G/G      A/A      T/T   G/G   T/T   C/C   A/A   G/G
## 2 S0768     Zmp-B-T~ ZMPBA G/G      A/A      T/T   A/G   T/T   C/C   A/A   G/G
## 3 S0881     Zmp-B-A~ ZMPBA C/G      A/A      T/T   G/G   T/T   C/C   A/G   G/G
## 4 S0887     Zmp-B-C~ ZMPBA C/G      A/A      T/T   G/G   T/T   C/C   A/A   G/G
## 5 S1057     Zmp-B-A~ ZMPBA C/G      A/A      G/T   A/G   T/T   C/C   A/A   G/G
## 6 S1087     Zmp-B-A~ ZMPBA G/G      A/A      T/T   G/G   T/T   C/C   A/A   G/G
## # i 975 more variables: bt2.5 <chr>, bt2.7 <chr>, bt2.8 <chr>, Fea2.1 <chr>,
## #   Fea2.5 <chr>, id1.3 <chr>, lg2.11 <chr>, lg2.2 <chr>, pbf1.1 <chr>,
## #   pbf1.2 <chr>, pbf1.3 <chr>, pbf1.5 <chr>, pbf1.6 <chr>, pbf1.7 <chr>,
## #   pbf1.8 <chr>, PZA00003.11 <chr>, PZA00004.2 <chr>, PZA00005.8 <chr>,
## #   PZA00005.9 <chr>, PZA00006.13 <chr>, PZA00006.14 <chr>, PZA00008.1 <chr>,
## #   PZA00010.5 <chr>, PZA00013.10 <chr>, PZA00013.11 <chr>, PZA00013.9 <chr>,
## #   PZA00015.4 <chr>, PZA00017.1 <chr>, PZA00018.5 <chr>, ...
```

```
head(transposed_data)
```

```
## # A tibble: 6 x 976
##   SNP_ID S0728 S0768 S0881 S0887 S1057 S1087 S1106 S1688 S1689 S1690 S1691 S1692
##   <chr>  <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 abph1~ G/G   G/G   C/G   C/G   C/G   G/G   C/C   C/G   G/G   G/G   C/G   G/G
## 2 abph1~ A/A   A/A   A/A   A/A   A/A   A/A   A/A   A/A   A/A   A/A   A/A   A/A
## 3 ae1.3  T/T   T/T   T/T   T/T   G/T   T/T   T/T   T/T   T/T   G/T   T/T   T/T
## 4 ae1.4  G/G   A/G   G/G   G/G   A/G   G/G   G/G   G/G   G/G   A/G   G/G   G/G
## 5 ae1.5  T/T   T/T   T/T   T/T   T/T   T/T   C/T   T/T   T/T   T/T   C/C   T/T
## 6 an1.4  C/C   C/C   C/C   C/C   C/C   C/C   C/C   C/C   C/C   C/G   C/C   C/C
## # i 963 more variables: S1693 <chr>, S1694 <chr>, S1695 <chr>, S1696 <chr>,
## #   S1697 <chr>, S1698 <chr>, S1699 <chr>, S1700 <chr>, S1701 <chr>,
## #   S1702 <chr>, S1703 <chr>, S1704 <chr>, S1705 <chr>, S1706 <chr>,
## #   S1707 <chr>, S1708 <chr>, TA002 <chr>, TA005 <chr>, TA013 <chr>,
## #   TA014 <chr>, TA023 <chr>, TA024 <chr>, TA025 <chr>, TA026 <chr>,
## #   TA029 <chr>, TA037 <chr>, TA038 <chr>, TA043 <chr>, TA044 <chr>,
## #   TA045 <chr>, TA046 <chr>, TA049 <chr>, TA050 <chr>, TA057 <chr>, ...
```

I next load the data from the snp_position.tx file and merge the desired columns with the extracted

fang_et_al_genotypes.txt data.

```
snp_data = read_delim("snp_position.txt", "\t")
```

```
## Rows: 983 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (9): SNP_ID, Chromosome, Position, alt_pos, mult_positions, amplicon, cd...
## dbl (6): cdv_marker_id, Genaissance_daa_id, Sequenom_daa_id, count_amplicons...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
merged_data <- merge(snp_data[,c(1,3,4)], transposed_data, by="SNP_ID", all = TRUE)
merged_data <- as_tibble(merged_data)
head(merged_data)
```

```
## # A tibble: 6 x 978
##   SNP_ID   Chromosome Position  S0728 S0768 S0881 S0887 S1057 S1087 S1106 S1688
##   <chr>    <chr>      <chr>     <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 abph1.20 2          27403404  G/G   G/G   C/G   C/G   C/G   G/G   C/C   C/G
## 2 abph1.22 2          27403892  A/A   A/A   A/A   A/A   A/A   A/A   A/A   A/A
## 3 ae1.3    5          167889790 T/T   T/T   T/T   T/T   G/T   T/T   T/T   T/T
## 4 ae1.4    5          167889682 G/G   A/G   G/G   G/G   A/G   G/G   G/G   G/G
## 5 ae1.5    5          167889821 T/T   T/T   T/T   T/T   T/T   T/T   C/T   T/T
## 6 an1.4    1          240498509 C/C   C/C   C/C   C/C   C/C   C/C   C/C   C/C
## # i 967 more variables: S1689 <chr>, S1690 <chr>, S1691 <chr>, S1692 <chr>,
## #   S1693 <chr>, S1694 <chr>, S1695 <chr>, S1696 <chr>, S1697 <chr>,
## #   S1698 <chr>, S1699 <chr>, S1700 <chr>, S1701 <chr>, S1702 <chr>,
## #   S1703 <chr>, S1704 <chr>, S1705 <chr>, S1706 <chr>, S1707 <chr>,
## #   S1708 <chr>, TA002 <chr>, TA005 <chr>, TA013 <chr>, TA014 <chr>,
## #   TA023 <chr>, TA024 <chr>, TA025 <chr>, TA026 <chr>, TA029 <chr>,
## #   TA037 <chr>, TA038 <chr>, TA043 <chr>, TA044 <chr>, TA045 <chr>, ...
```

I sorted the data based on position, making sure to convert position to numeric before doing so.

```
merged_data$Position <- as.numeric(merged_data$Position)
```

```
## Warning: NAs introduced by coercion
```

```
merged_data <- arrange(merged_data, `Position`)
```

I next generated the chromosome files for chromsomes 1-10 for the increasing position order, omitting the NA entries that correspond to multiple or unknown chromosome positions.

```
make_chromosome_file <- function(dataset_row, species, order, missing_marker, dataset){
  dataset_row <- gsub("\\?", missing_marker, dataset_row)
  dataset_row <- as.data.frame(t(dataset_row))
  chromosome <- dataset_row["Chromosome"]
  output_file <- sprintf("%s/%s/%s/chromosome_%s.txt",getwd(),species,order,chromosome)
  if (file.exists(output_file) == FALSE) {
            write.table(
              t(colnames(dataset)), output_file, sep="\t", row.names = FALSE,
              col.names = FALSE, append = TRUE, quote = FALSE)
  }
  write.table(
    dataset_row, output_file, sep="\t", row.names = FALSE,
    col.names = FALSE, append = TRUE, quote = FALSE)
```

```
}

merged_data_filtered <- filter(merged_data, !is.na(Position))
apply(merged_data_filtered,1,make_chromosome_file,species="teosinte",order="increasing",
missing_marker="\\?",dataset=merged_data_filtered)
```

```
## NULL
```

I next prepared the chromsome files for decreasing position order, making sure to specify the missing marker symbol as "-" not the "?" symbol.

```
merged_data_filtered <- arrange(merged_data_filtered, desc(`Position`))
apply(merged_data_filtered,1,make_chromosome_file,species="teosinte",order="decreasing",
missing_marker="\\-",dataset=merged_data_filtered)
```

```
## NULL
```

I next prepared the chromosome files for the multiple and unknown positions.

```
make_misc_file <- function(dataset_row, species,dataset){
  dataset_row <- as.data.frame(t(dataset_row))
  chromosome <- dataset_row["Chromosome"]
  position <- dataset_row["Position"]
  output_file <- sprintf("%s/%s/chromosome_%s.txt",getwd(),species,chromosome)
  if (is.na(position) == TRUE){
    if ((chromosome %in% c("multiple", "unknown")) == FALSE){
      chromosome <- "multiple"
      output_file <- sprintf("%s/%s/chromosome_%s.txt",getwd(),species,chromosome)
    }
    if (file.exists(output_file) == FALSE) {
            write.table(
              t(colnames(dataset)), output_file, sep="\t", row.names = FALSE,
              col.names = FALSE, append = TRUE, quote = FALSE)
    }
            write.table(
              dataset_row, output_file, sep="\t", row.names = FALSE,
              col.names = FALSE, append = TRUE, quote = FALSE)
  }
}
apply(merged_data,1, make_misc_file, species="teosinte",dataset=merged_data)
```

```
## NULL
```

Data Visualization

SNPs per chromosome

I first investigated the distribution of SNPs between chromosomes by plotting number of SNPs per chromosome for both maize and teosinte.

```
# Load maize data.
file_path <- "fang_et_al_genotypes.txt"
raw_data <- read_delim(file_path, delim="\t")
```

```
## Rows: 2782 Columns: 986
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (986): Sample_ID, JG_OTU, Group, abph1.20, abph1.22, ae1.3, ae1.4, ae1.5...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

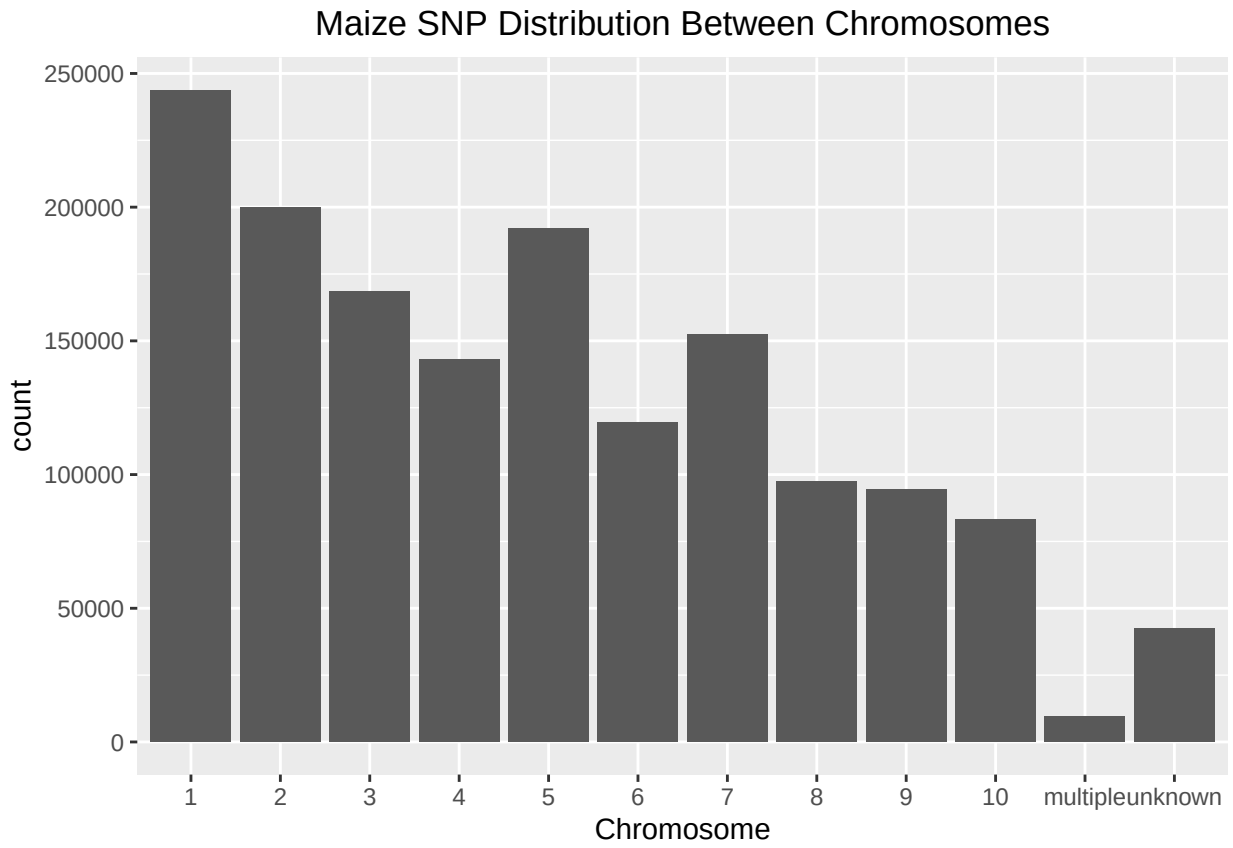
```
extracted_data <- filter(raw_data, `Group` %in% c("ZMMIL", "ZMMLR", "ZMMMR"))
extracted_data <- arrange(extracted_data, `Sample_ID`)
transposed_data <- as_tibble(t(extracted_data[,4:ncol(extracted_data)]),rownames = "SNP_ID")
colnames(transposed_data) <- c("SNP_ID",unlist(extracted_data[1:nrow(extracted_data),1]), use.names = FA
snp_data = read_delim("snp_position.txt", "\t")
```

```
## Rows: 983 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (9): SNP_ID, Chromosome, Position, alt_pos, mult_positions, amplicon, cd...
## dbl (6): cdv_marker_id, Genaissance_daa_id, Sequenom_daa_id, count_amplicons...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
merged_data <- merge(snp_data[,c(1,3,4)], transposed_data, by="SNP_ID", all = TRUE)
merged_data <- as_tibble(merged_data)
merged_data$Position <- as.numeric(merged_data$Position)
```

```
## Warning: NAs introduced by coercion
```

```
merged_data <- arrange(merged_data, `Position`)
maize_merged_data <- pivot_longer(
  merged_data,
  cols = 4:ncol(merged_data),
  values_to = "SNP"
)

proper_chromosome_order <- c(
  "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "multiple", "unknown"
)
ggplot(data = maize_merged_data) + geom_bar(mapping = aes(x=Chromosome)) +
scale_x_discrete(limits = proper_chromosome_order)+
ggtitle("Maize SNP Distribution Between Chromosomes")+
theme(plot.title = element_text(hjust = 0.5))
```




```
# Load teosinte data.
file_path <- "fang_et_al_genotypes.txt"
raw_data <- read_delim(file_path, delim="\t")
```

```
## Rows: 2782 Columns: 986
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (986): Sample_ID, JG_OTU, Group, abph1.20, abph1.22, ae1.3, ae1.4, ae1.5...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

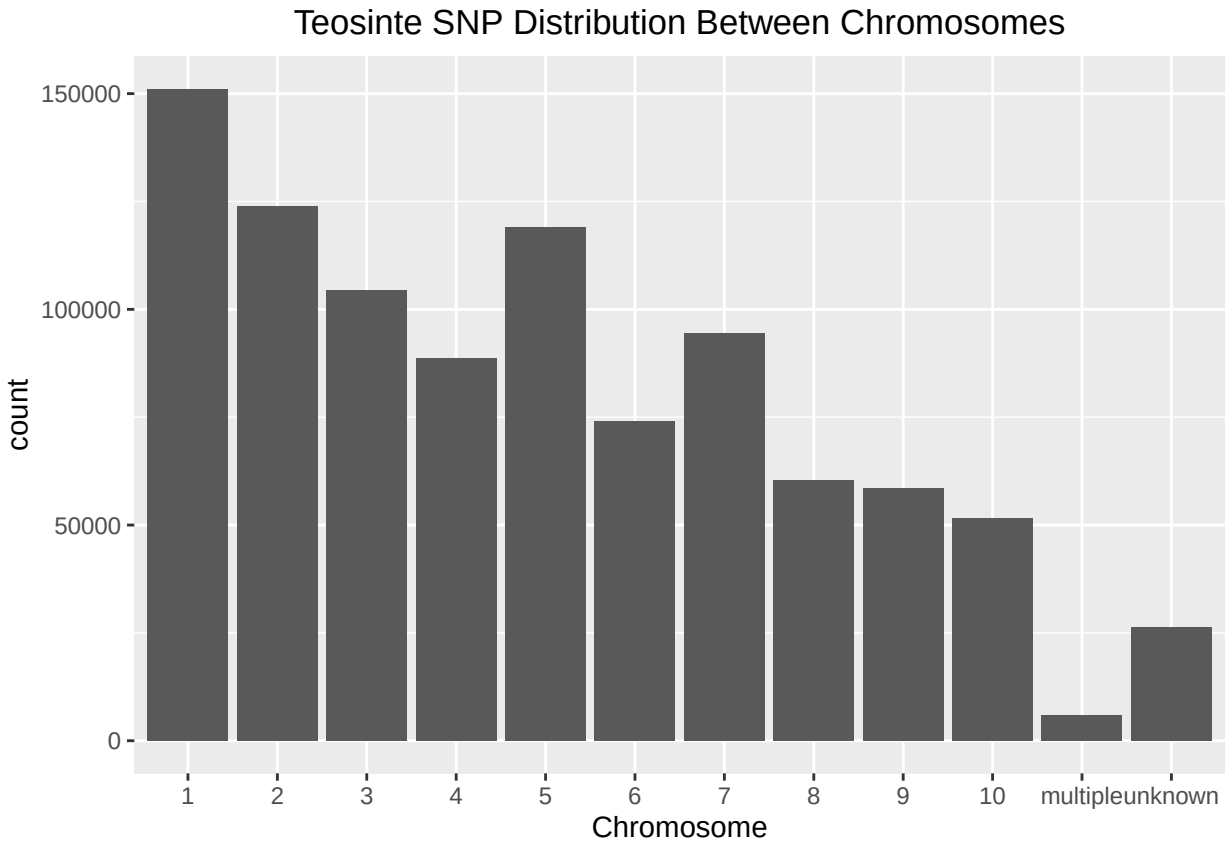
```
extracted_data <- filter(raw_data, `Group` %in% c("ZMPBA", "ZMPIL", "ZMPJA"))
extracted_data <- arrange(extracted_data, `Sample_ID`)
transposed_data <- as_tibble(t(extracted_data[,4:ncol(extracted_data)]),rownames = "SNP_ID")
colnames(transposed_data) <- c("SNP_ID",unlist(extracted_data[1:nrow(extracted_data),1]), use.names = F
snp_data = read_delim("snp_position.txt", "\t")
```

```
## Rows: 983 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (9): SNP_ID, Chromosome, Position, alt_pos, mult_positions, amplicon, cd...
## dbl (6): cdv_marker_id, Genaissance_daa_id, Sequenom_daa_id, count_amplicons...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
merged_data <- merge(snp_data[,c(1,3,4)], transposed_data, by="SNP_ID", all = TRUE)
merged_data <- as_tibble(merged_data)
merged_data$Position <- as.numeric(merged_data$Position)
```

```
## Warning: NAs introduced by coercion
```

```
merged_data <- arrange(merged_data, `Position`)
teosinte_merged_data <- pivot_longer(
  merged_data,
  cols = 4:ncol(merged_data),
  values_to = "SNP"
)

proper_chromosome_order <- c(
  "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "multiple", "unknown"
)
ggplot(data = teosinte_merged_data) + geom_bar(mapping = aes(x=Chromosome)) +
scale_x_discrete(limits = proper_chromosome_order) +
ggtitle("Teosinte SNP Distribution Between Chromosomes")+
theme(plot.title = element_text(hjust = 0.5))
```

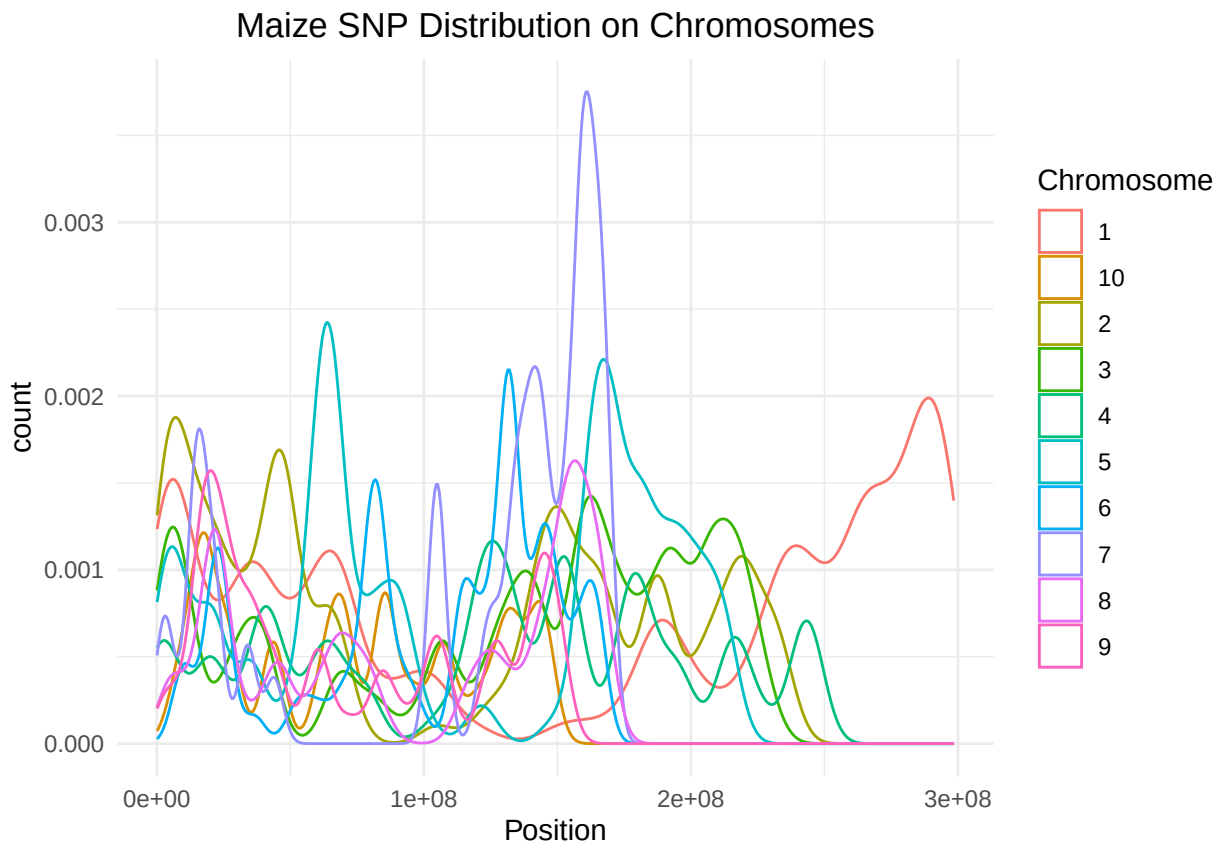


From the data, it looks like they have the same distribution of SNPs across chromosomes. However, maize has more SNPs per position as the number of columns in the dataframe is larger (1576) compared to teosinte vs. (978).

I next investigated the distribution of SNPs on the chromsomes by plotting the number of SNPs per position on each chromosome for both maize and teosinte.

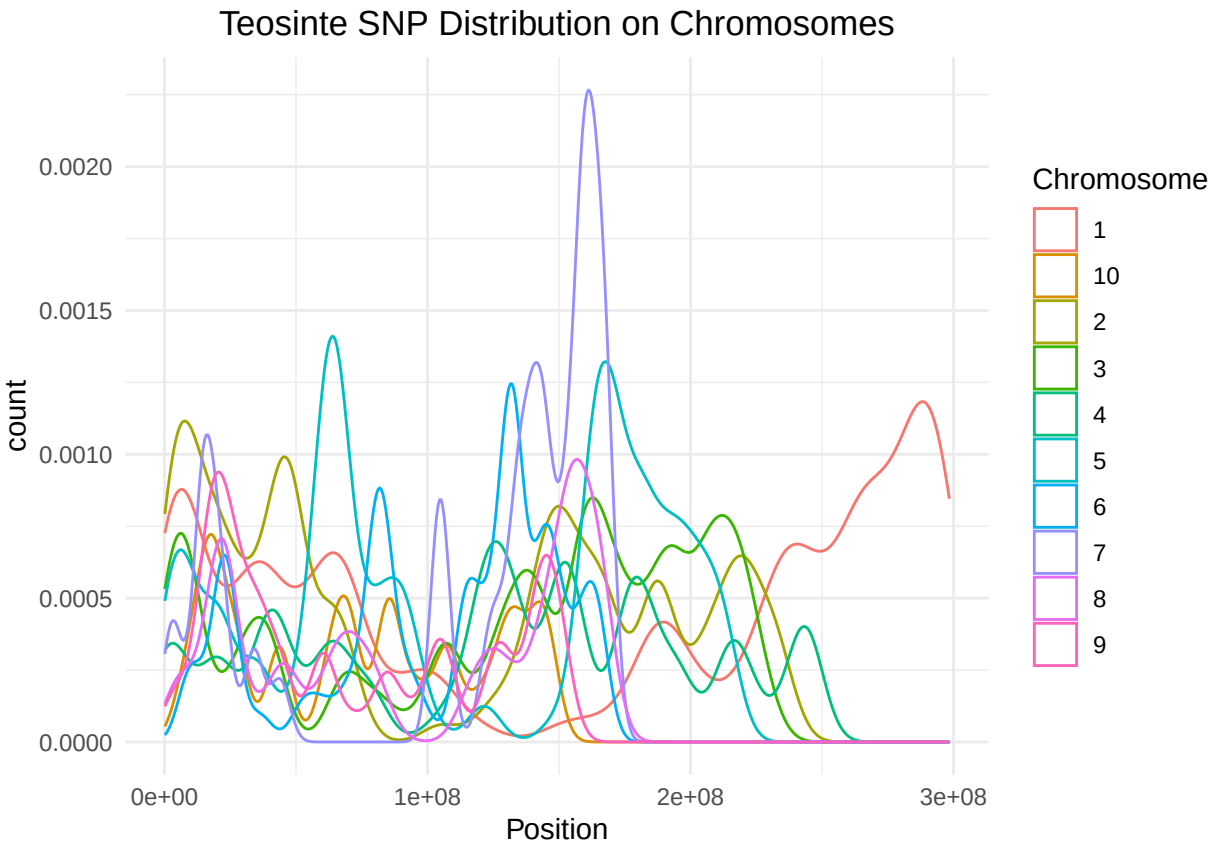
```
ggplot(data = maize_merged_data) +
  geom_density(aes(x=Position, color=Chromosome,y = after_stat(count))) +
  theme_minimal()+
  ggtitle("Maize SNP Distribution on Chromosomes")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 69212 rows containing non-finite outside the scale range
## (`stat_density()`).
```




```
ggplot(data = teosinte_merged_data) +
  geom_density(aes(x=Position, color=Chromosome,y = after_stat(count))) +
  theme_minimal()+
  ggtitle("Teosinte SNP Distribution on Chromosomes")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 42900 rows containing non-finite outside the scale range
## (`stat_density()`).
```



From the data, it looks like the distributions are the same between maize and teosinte. This is somewhat expected since the SNP positions are the same between them. Some notable observations are that chromosome 7 appears to have its SNP positions more heavily concentrated in the middl while chromosome 1 appears to have a bimodal distribution with the SNPs being more highly concentrated at the ends.

Missing data and amount of heterozygosity

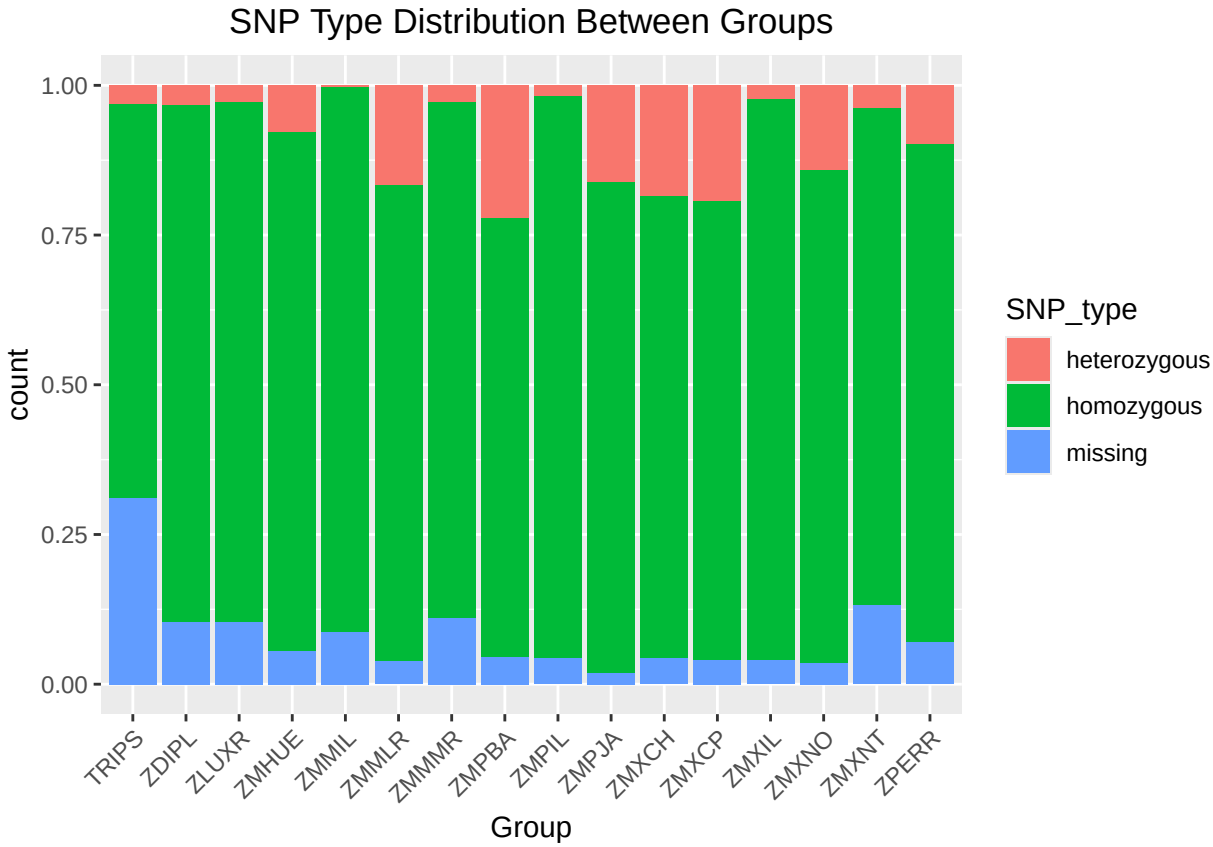
I next investigated the proportion of homozygous, heterozygous, and missing data sites in each sample and each group.

```
# Load data
file_path <- "fang_et_al_genotypes.txt"
raw_data <- read_delim(file_path, delim="\t")
```

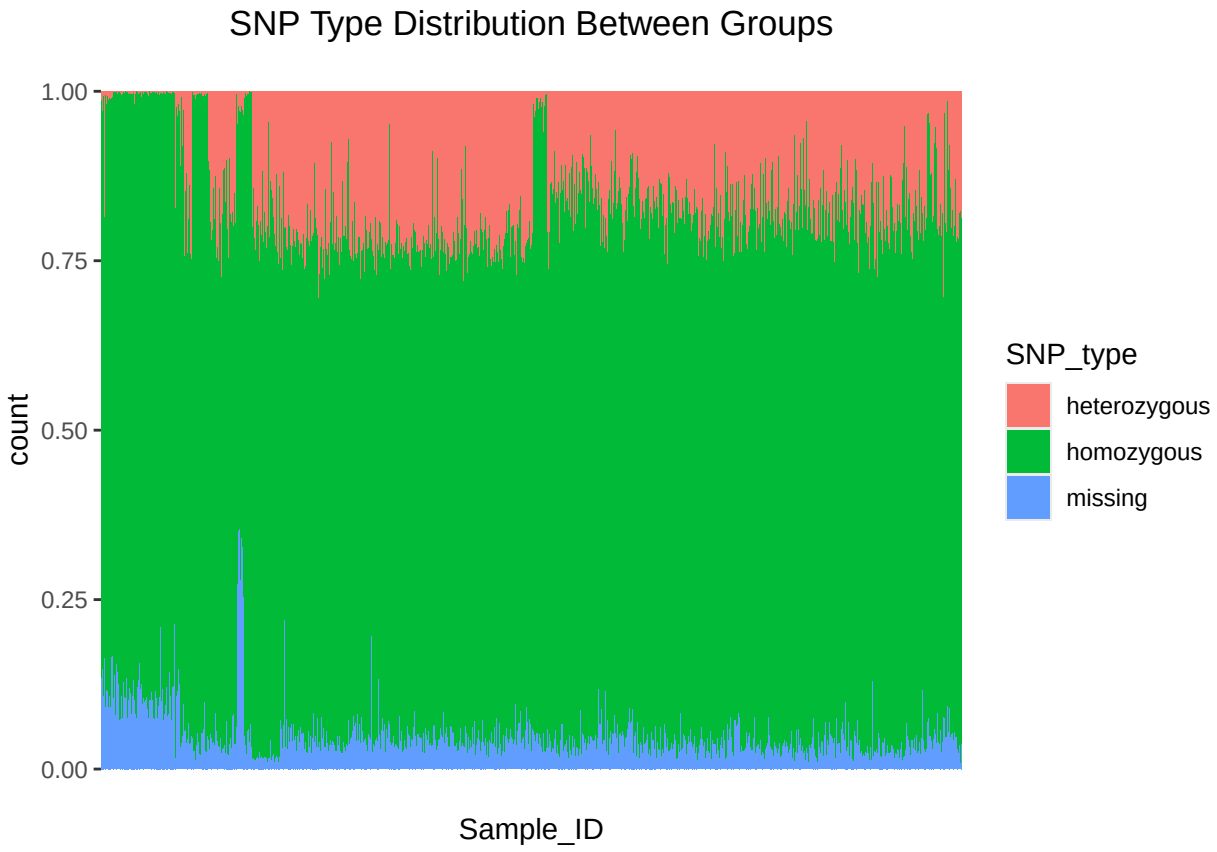
```
## Rows: 2782 Columns: 986
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (986): Sample_ID, JG_OTU, Group, abph1.20, abph1.22, ae1.3, ae1.4, ae1.5...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
raw_data_long <- pivot_longer(
  raw_data,
  cols = 4:ncol(raw_data),
  values_to = "SNP"
)

# Find SNP type
```

```
find_snp_type <- function(SNP){
  SNP_string <- toString(SNP)
  SNP_positions <- strsplit(SNP_string,split="")
  position_1 <- SNP_positions[[1]][1]
  position_2 <- SNP_positions[[1]][3]
  if (position_1 == "?" | position_2 == "?"){
    return("missing")
  }
  else if (position_1 == position_2){
    return("homozygous")
  }
  else if (position_1 != position_2){
    return("heterozygous")
  }
}
vectorized_find <- Vectorize(find_snp_type) # Needed to properly extract point mutations for each SNP.
raw_data_long <- mutate(
  raw_data_long,
  SNP_type = vectorized_find(SNP)
)

# Plot data
ggplot(data = raw_data_long) +
  geom_bar(mapping = aes(x = Group, fill = SNP_type), position = "fill")+
  ggtitle("SNP Type Distribution Between Groups")+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(axis.text.x=element_text(angle=45,hjust=1))
```




```
ggplot(data = raw_data_long) +
  geom_bar(mapping = aes(x = Sample_ID, fill = SNP_type), position = "fill")+
  ggtitle("SNP Type Distribution Between Groups")+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(axis.text.x = element_blank(),
      axis.ticks.x = element_blank(),
      axis.line.x = element_blank())
```



From the data, it looks like the proportion of heterozygosity, homozygosity, and missing data doesn't vary too significantly among groups and samples. However, there are outliers like TRIPS which has an increased proportion of missing data and ZMMIL which has almost no heterozygous SNPs.

My Own Visualization

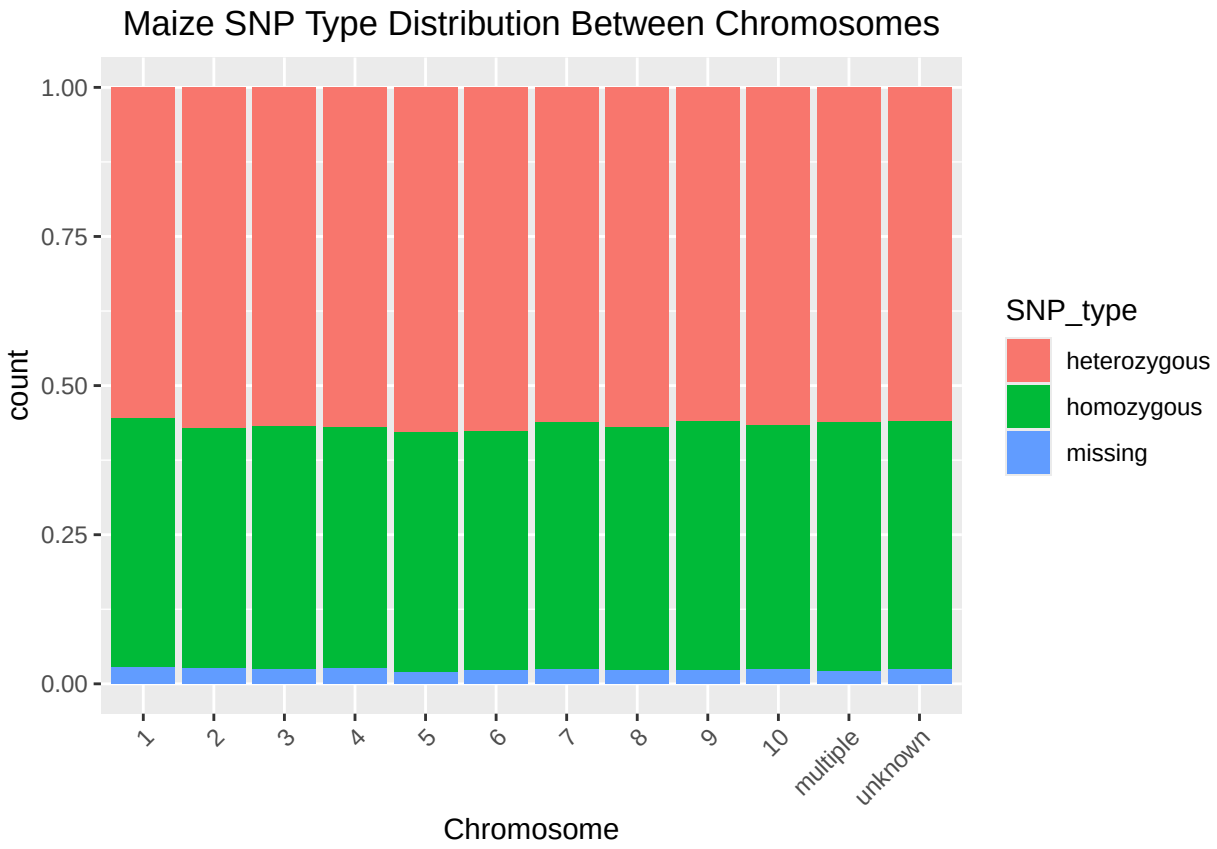
For my own visualization, I decided to investigate the proportion of homozygous, heterozygous, and missing data sites across chromosomes for both Maize and Teosinte.

```
merged_data_long <- pivot_longer(
  maize_merged_data,
  cols = 4:ncol(maize_merged_data),
  values_to = "SNP"
)
merged_data_long <- mutate(
  merged_data_long,
  SNP_type = vectorized_find(SNP)
)
head(merged_data_long)
```

```
## # A tibble: 6 x 6
##   SNP_ID     Chromosome Position name  SNP     SNP_type
##   <chr>      <chr>         <dbl> <chr> <chr>   <chr>
## 1 PZA03013.7 4            139753 name  BKN_001 heterozygous
## 2 PZA03013.7 4            139753 SNP   G/G     homozygous
## 3 PZA03013.7 4            139753 name  BKN_002 heterozygous
## 4 PZA03013.7 4            139753 SNP   G/G     homozygous
## 5 PZA03013.7 4            139753 name  BKN_003 heterozygous
```

```
## 6 PZA03013.7 4            139753 SNP   G/G     homozygous
```

```
# Plot data
ggplot(data = merged_data_long) +
  geom_bar(mapping = aes(x = Chromosome, fill = SNP_type), position = "fill") +
  scale_x_discrete(limits = proper_chromosome_order)+
  ggtitle("Maize SNP Type Distribution Between Chromosomes")+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(axis.text.x=element_text(angle=45,hjust=1))
```

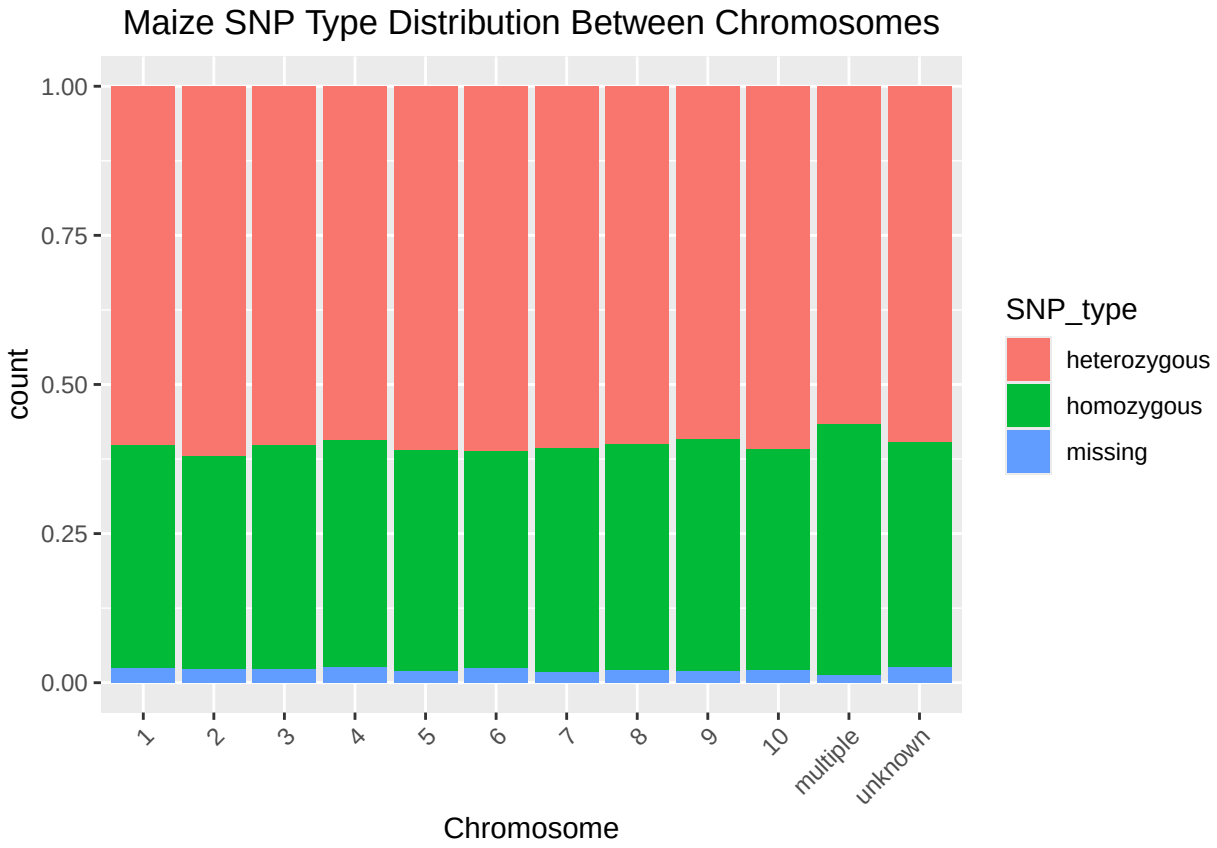



```
merged_data_long <- pivot_longer(
  teosinte_merged_data,
  cols = 4:ncol(teosinte_merged_data),
  values_to = "SNP"
)
merged_data_long <- mutate(
  merged_data_long,
  SNP_type = vectorized_find(SNP)
)
head(merged_data_long)
```

```
## # A tibble: 6 x 6
##   SNP_ID     Chromosome Position name  SNP   SNP_type
##   <chr>      <chr>         <dbl> <chr> <chr> <chr>
## 1 PZA03013.7 4            139753 name  S0728 heterozygous
## 2 PZA03013.7 4            139753 SNP   A/G   heterozygous
## 3 PZA03013.7 4            139753 name  S0768 heterozygous
## 4 PZA03013.7 4            139753 SNP   A/G   heterozygous
```

```
## 5 PZA03013.7 4            139753 name  S0881 heterozygous
## 6 PZA03013.7 4            139753 SNP   A/A   homozygous
```

```
# Plot data
ggplot(data = merged_data_long) +
  geom_bar(mapping = aes(x = Chromosome, fill = SNP_type), position = "fill") +
  scale_x_discrete(limits = proper_chromosome_order)+
  ggtitle("Maize SNP Type Distribution Between Chromosomes")+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(axis.text.x=element_text(angle=45,hjust=1))
```



From the data, it looks like there is no major difference between the SNP distribution for teosinte and maize, but it does seem like they have a low proportion of missing data and a high proportion of heterozygosity.